UNIVERSITÉ DU QUÉBEC À MONTRÉAL

EXTRACTION DE FLUX DE TRAVAUX ABSTRAITS À PARTIR DES TEXTES : APPLICATION À LA BIOINFORMATIQUE

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR

AHMED HALIOUI

FÉVRIER 2018

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.07-2011). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

REMERCIEMENTS

On dit que la thèse de doctorat fait souvent l'objet de métaphores : longue marche dans le désert ou traversée des océans. Cette image souvent utilisée est celle d'une aventure, un périple hasardeux. Pour ma part, je retiendrais plutôt l'image d'une aventure d'aviron qui fend l'eau dans un lac avec un kayak qui se tortille dans tous les sens. D'abord, à la veille d'une compétition, des préparatifs sont d'importance : on réfléchit, on tâte le terrain, on rassemble le matériel. Puis le lendemain, c'est le départ, avec toutes les formalités administratives qui entourent le début de chaque compétition, où un soin est apporté à la mise en place d'un camp de base. Commence alors la compétition, avec ses illusions et ses contre-temps, ses dangers et ses moments de répit. L'effort est intense et de longue durée. Il y'a toujours le risque de subir des luxations ou même des métastases osseuses qui peuvent évoquer des traumatismes de longues durées. Voilà qu'on perçoit le drapeau d'arrivée déambulant dans l'air et c'est à ce juste moment qu'on réalise l'ampleur de l'effort qu'on doit encore fournir pour y arriver et être distingué des autres.

L'accomplissement d'un grand projet comme celui-ci ne se fait pas seul ; il se fait en équipe en cordée. Dans l'équipe universitaire, je tiens tout d'abord à remercier mon directeur de recherche, le Prof. Abdoulaye Baniré Diallo, pour m'avoir accueilli au sein de son laboratoire et qui, grâce à son optimisme et son enthousiasme, j'ai pu réaliser plusieurs projets attachés à ce doctorat. Je remercie également mon co-directeur, le Prof. Petko Valtchev. Bien plus qu'un superviseur, j'ai trouvé en lui un père qui m'a appris la qualité et l'importance du travail bien fait. Aussi, merci à mes collaborateurs et amis du laboratoire de bioinformatique : Bruno, Golrokh, Wajdi, Malick, Ramzi, Sahbi et Tomas, qui m'ont montré que plusieurs façons de

voir les choses étaient possibles. C'est avec bonheur et privilège que j'ai travaillé avec Mohammed Amine, un collègue particulier qui avec les années est devenu un grand ami. Nous avons travaillé sur un projet, qui, un jour, changera le monde de l'analyse génomique. Merci à Étienne Lord, pour tous nos échanges scientifiques, tes conseils, complicité et ton soutien. Merci également à Justine, Claude et Janie. Vous avez toutes contribué à mon épanouissement social et syndicale.

Ma famille occupe une place importante pour moi. Ainsi, je veux remercier mes parents, Ahlem et Mohamed Najib, ma soeur Hiba et mon frère Sami, qui m'ont encouragé à toujours persévérer dans les projets que j'entreprends. Merci aussi à tous ceux qui m'ont inspiré : David Graeber, Youssef Seddik et Chuck Schuldiner.

À MON GRAND PÈRE.

TABLE DES MATIÈRES

LISTE DES TABLEAUX xi
LISTE DES FIGURES xiii
RÉSUMÉ xvii
ABSTRACT xix
INTRODUCTION 1
CHAPITRE I
CONTEXTE, NOTIONS DE BASE ET OBJECTIFS DE RECHERCHE 5
1.1 Préambule phylogénétique 5
1.1.1 Qu'est ce qu'une analyse phylogénétique ? 5
1.1.2 Étapes d'analyse phylogénétique 6
1.2 Flux de travaux et acquisition de connaissances de résolution de problèmes 12
1.2.1 Connaissances impliquées dans la résolution de problèmes . . 13
1.2.2 Raisonnement 16
1.2.3 Problèmes d'acquisition de connaissances processuelles 20
1.2.4 Une approche basée sur les ontologies 22
1.3 Extraction de flux de travaux à partir des textes 25
1.3.1 Extraction des entités nommées. 25
1.3.2 Extraction des relations 30
1.4 Fouille de motifs processuels basée sur une ontologie 34
1.4.1 Problème de fouille de motifs 35
1.4.2 Fouille de graphes généralisés 36
1.5 Systèmes de recommandation 39
1.6 Objectifs de la recherche 42

1.6.1 Hypothèse fondatrice 42
1.6.2 Analyse des besoins 42
CHAPITRE II
EXTRACTION DE FLUX DE TRAVAUX À PARTIR DES TEXTES . . 45
2.1 Résumé 46
2.2 Abstract 47
2.3 Introduction 48
2.4 Formalisms for enriched workflow representations 52
2.4.1 Workflow representation 52
2.4.2 Ontology representation 56
2.5 Extraction process 57
2.5.1 Morphosyntactic annotation 58
2.5.2 Semantic annotation 59
2.5.3 Word sense ambiguity recognition 62
2.5.4 Word Sense Disambiguation (WSD) 64
2.5.5 Learning module 66
2.5.6 Workflow reconstruction 70
2.5.7 Workflow similarity measure 71
2.6 Experiments and results 72
2.6.1 Dataset 72
2.6.2 Results 73
2.7 Conclusion 77
CHAPITRE III
FOUILLE GÉNÉRALISÉE DE FLUX DE TRAVAUX 79
3.1 Résumé 81
3.2 Abstract 82
3.3 Introduction 83
3.4 Related work 85

3.5 Motivating example 88
3.6 Data and pattern languages 90
3.7 Total order 94
3.8 Mining workflow patterns 98
3.8.1 Pattern space 98
3.8.2 Mining method 102
3.9 Running example 112
3.10 Experimental study 116
3.10.1 Sample mining output 117
3.10.2 Scale-up study 119
3.11 Conclusion 122
CHAPITRE IV
UN SYSTÈME D'ACQUISITION DE FLUX DE TRAVAUX ABSTRAITS 123
4.1 Résumé 124
4.2 Abstract 125
4.3 Introduction 126
4.4 T-GOWler : ontology-based workflow pattern acquisition from texts . 127
4.4.1 WfExtractor : workflow extraction from texts 129
4.4.2 WfMiner : mining generalized workflows 131
4.5 Experiments and results 133
4.5.1 Case study : phylogenetics analyses 133
4.5.2 Named entity learning results 135
4.5.3 Generalized-pattern results 136
4.6 Conclusion 139
CHAPITRE V
VERS UN SYSTÈME DE RECOMMANDATION DE FLUX DE TRAVAUX 141
5.1 Résumé 142

5.2 Abstract 143
5.3 Introduction 144
5.4 Related work 148
5.5 Advanced experiments 155
5.5.1 Dataset 155
5.5.2 Extracted workflows from texts 158
5.5.3 Generalized bioinformatics patterns 163
5.6 Discussion 168
5.7 Vers un système de recommandation 171
5.7.1 Une nouvelle approche de recommandation 171
5.7.2 Stratégie d'évaluation 177
5.7.3 Résultats préliminaires 181
5.7.4 Conclusion 185
CONCLUSION 187
BIBLIOGRAPHIE 281

LISTE DES TABLEAUX

Tableau		Page
2.1	Feature descriptions	67
2.2	PAUM Word Sense Disambiguation and relation extraction evaluation results.	74
3.1	Canonical operations.	101
3.2	Example of Workflow sequences.	113
3.3	Frequent patterns, their associated supports (Supp.) and corresponding workflow instantiation IDs (Inst.).	114
3.4	Workflow datasets.	117
4.1	Evaluation results of the extracted workflow elements based on the expert curated dataset from the PubMed PMC articles (published from January 2013 to April 2015).	136
4.2	A sample from the phylogenetic extracted patterns within the texts in the PubMed PMC database (articles published from January 2013 to April 2015).	138
5.1	Concepts Evaluation Distribution	161
5.2	Exemple de base de flux de travaux.	173
5.3	Exemple de base de motifs.	177
5.4	Mise en correspondance concept/instance.	177
5.5	Différences entre un expert et un novice.	248
5.6	Tableau comparatif des systèmes d'extraction des entités nommées dans le domaine biomédical.	250
5.7	Tableau comparatif des systèmes d'extraction des relations dans le domaine biomédical.	252

5.8 Tableau comparatif des systèmes de fouille de motifs basée sur une ontologie. 256

5.9 Tableau comparatif des systèmes de recommandation basés sur le contenu. 260

LISTE DES FIGURES

Figure		Page
1.1	Pipeline général de l'analyse phylogénétique	6
1.2	Exemple d'analyse phylogénétique des génomes mitochondriaux dans la plateforme Armadillo (Lord *et al.*, 2012)	11
1.3	Portion de l'ontologie "The Software Ontology" (SWO) (Malone *et al.*, 2014)	24
2.1	Example of a workflow excerpted from the article of Liang *et al.* (2010)	54
2.2	Example of a workflow data representation	55
2.3	A sample from the phylogenetic domain ontology PHAGE	56
2.4	Pipeline of the workflow extraction from texts.	57
2.5	An example of a chunking tree.	59
2.6	Ontology transformation of polysemic concepts	64
2.7	An example of learning features for WSD concepts and relation extraction from texts.	66
2.8	Example of a workflow reconstruction from a domain-ontology	70
2.9	Workflow similarity distributions.	76
3.1	Bioinformatics workflow : a reconstruction of phylogenies from the NKX23 proteins (Lord *et al.*, 2012)	89
3.2	Hierarchies in the Phylogenetic PHAGE Ontology (a partial)	91
3.3	(A) Workflow representation of $w_1 \in \Delta_W$. (B) Pattern representation of $p_1 \in \Gamma_W$.	93
3.4	Matching example between concepts and objects	97

3.5 Example of triple set ranking. 97

3.6 An example of canonical operations. 100

3.7 Example of adding a relation $addR(r_1, 3)$ in the matching structure of $MS'(p'_1)$ based on $MS(p_1)$ 110

3.8 Example of generated patterns. 118

3.9 Scalability experiments. 120

3.10 *Our method* vs `xPMiner` using the WFPub_3 dataset. 121

4.1 `T-Gowler` system architecture. 128

4.2 Workflow and pattern representations. 131

4.3 Pattern matching structures over levels k and $k+1$. 133

4.4 A portion from the phylogenetic `PHAGE` ontology. 134

4.5 (A) Varying support for the *PhyloFlows* and *e-TP* tourism databases. (B) Comparing the mining running time of `WfMiner` and `xPMiner`. 137

5.1 A Phylogenetic analysis example drawn from the publication of Liang *et al.* (2010) using the Armadillo platform (Lord *et al.*, 2012) . . . 146

5.2 A sample from the proposed `PHAGE` ontology published in the BioPortal repository 156

5.3 Workflow similarities results 162

5.4 Pattern coverage metrics distributions Vs. pattern ranking function. 167

5.5 Pattern raking function Vs. coverage and depth rankings. 168

5.6 Qualité de Top k meilleures recommandations. 182

5.7 Qualité des Top n meilleurs motifs en se basant sur leur rang centiles (MPR). 184

5.8 Extrait du résultat de l'extraction de flux de travaux concrets. . . 264

5.9 Extrait du résultat de la fouille de motifs généralisés (1). 266

5.10 Extrait du résultat de la fouille de motifs généralisés (2). 267

5.11 Plateforme GATE 8.1. 270

5.12 Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (1). 272

5.13 Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (2). 273

5.14 Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (3). 274

5.15 Distribution des métriques de tri de motifs. 276

5.16 Extraits des supports relatifs de concepts et relations de l'ontologie dans la base d'articles extraits depuis PubMed PMC 2013-2015. . 278

5.17 Extrait des supports relatifs des objets les plus fréquents extraits depuis PubMed PMC 2013-2015. 280

RÉSUMÉ

Dans des domaines techniques comme la didactique et la bioinformatique, l'acquisition des connaissances impliquées dans le processus de résolution de problèmes du domaine est sujet à plusieurs défis liés aux outils utilisés ainsi que son domaine d'application, *e.g.*, en analyse phylogénétique. En outre, avec le temps, la taille de données *expérientielles* (impliquées dans le processus de résolution) augmente de façon exponentielle avec une très grande diversité de méthodes et de modèles informatiques. Ceci rend la tâche de résolution de plus en plus complexe à automatiser. Nous trouvons plusieurs solutions en ligne fournissant des pipelines d'analyses semi-automatiques comme dans Phylogeny.fr ou Bioextract.org mais aucune d'elles ne définit de "meilleures pratiques génériques" basées sur les types de données utilisés et les contraintes d'applications (logiciels). Dans ce manuscrit, nous proposons un système de *fouille de motifs processuels fréquents* basé sur une *ontologie* du domaine représentant différents niveaux d'abstractions du "*savoir faire*" dans la littérature *phylogénétique* (soir par exemple des textes scientifiques). La tâche d'*extraction des connaissances processuelles* et la *désambiguïsation* contextuelle ainsi que la tâche de fouille de motifs abstraits serviront par la suite à définir une base de règles de meilleures pratiques d'analyses phylogénétiques. L'utilisation de cette base de données dans un système de recommandation assistera les chercheurs du domaine dans leurs tâches de résolution de problèmes via des flux de travaux en tenant compte leurs niveaux d'expertises.

Mots-clés : *bioinformatique, phylogénétique, flux de travaux, ontologies, extraction d'information, désambiguïsation, fouille de motifs généralisés.*

ABSTRACT

In many technical domains such as didactics and bioinformatics, the acquisition of problem-solving knowledge faces several issues related to data and software skills. In addition, the dramatic increase of the available sequenced data and the large diversity of bioinformatics software and tools make the problem solving process more complex to automate. Different solutions have been proposed in the literature offering workflow models such as in Phylogeny.fr or Bioextract.org. However, none of them defines both domain and application best practices to solve very specific problems taking into account data provenance and domain constraints. We propose in this manuscript a generalized workflow mining system to acquire problem solving knowledge from the phylogenetic literature. Our system is based on a domain ontology enriched from scientific texts to mine abstract problem solving processes. Multi-level abstract patterns have made workflows more intuitive and easy to be reused in similar situations. This could provide a stepping-stone into the identification of the domain best practices and pave the road to a recommender system to assist both novices and experts in their problem solving tasks.

keywords: *bioinformatics, phylogenetics, workflows, ontologies, information extraction, generalized pattern mining.*

INTRODUCTION

Le problème d'acquisition automatique de *connaissances liées à la résolution problèmes*, c'est-à-dire les connaissances liées à la formalisation du problème, les tâches de résolution et de leurs exécutions, est un problème fortement lié au contexte de son application. Dans les domaines techniques comme la bioinformatique, la construction automatique de chaines de traitements (*flux de travaux* - ou encore flux de tâches) (Taylor *et al.*, 2014) sont devenus des techniques de plus en plus exploitées. Ces plateformes offrent des solutions généralistes sans prendre en considération les différents niveaux d'expertises de ses utilisateurs. Ces derniers sont généralement confrontés à différentes situations (des fois complexes) d'acquisition de nouvelles connaissances liés aux différents formats de données et aux logiciels à manipuler.

Dans un domaine technique comme la phylogénie, il s'agit de résoudre un problème d'analyse de séquences nucléiques ou protéiques afin de détecter différents phénomènes évolutifs tels que les transferts de gènes horizontaux (Lemey Philippe, 2009). L'augmentation de volumes de données génomiques et la grande diversité des méthodes et des modèles informatiques utilisés dans l'inférence phylogénétique, rendent la tâche de l'analyse de plus en plus complexe (Stevens *et al.*, 2007). En outre, à travers ces tâches, une grande variabilité dans l'utilisation des outils bioinformatiques, les jeux de données et les méthodes d'analyses, rend la comparaison de différentes études portant sur les mêmes espèces difficile à réaliser (Philippe *et al.*, 2011). Bien que les conditions expérimentales et les choix des logiciels d'analyses soient dépendants des ressources informatiques disponibles,

d'autres conditions, telles que les méconnaissances des biais liées à chaque outil, sont aussi responsables de la divergence des résultats (Leigh *et al.*, 2011).

Pour décrire quelques perspectives dans cette problématique, nous prenons l'exemple suivant. Pour l'analyse phylogénétique de 9 génomes mitochondriaux mammifères, un phylogénéticien dispose de plusieurs ressources, méthodes et approches phylogénétiques. Il choisit de télécharger la plateforme Armadillo (Lord *et al.*, 2012) pour analyser ses données. Une collection de méthodes s'affiche. Son expérience en analyse phylogénétique ne lui permet pas de comprendre toutes les méthodes disponibles. Il choisit donc de se référer à un article proposé par son directeur. Un flux de travail s'affiche alors à l'aide de l'application Armadillo. En lançant le processus, des erreurs s'affichent à l'écran. Encore intrigué et perdu dans un ensemble de paramètres et contraintes logiciels, le phylogénéticien appelle son directeur pour lui demander de l'aide dans l'exécution de sa tâche.

Les *niveaux d'expertise* des phylogénéticiens varient et dépendent de l'expérience phylogénétique personnelle ainsi que l'expérience "logicielle" qui manipule des représentations symboliques de l'ordinateur. Par exemple, face à un logiciel de recherche de séquences ARN d'une espèce, l'expert doit comprendre l'interface ainsi que les paramètres de ce logiciel. Il s'agit ici de deux différents types de connaissances : connaissances du domaine vs. connaissances logicielles (voire même, des fois, matérielles). Cette attitude cognitive doit être dirigée et guidée afin d'aider l'utilisateur à mieux "manipuler" ces représentations. La multitude de connaissances à maitriser ainsi que la complexité des méthodes phylogénétiques engendrent des fois des méconnaissances liées au domaine et aux outils utilisés : ne pas bien paramétrer ces logiciels ou choisir les mauvaises versions de données peuvent "fausser" les résultats finaux.

Une *ontologie* (De Raedt, 2008) est une solution qui facilite les tâches d'*intégration*

et de *réutilisation* des données d'une façon non ambiguës en définissant un vocabulaire standard des concepts du domaine. Elle garantit ainsi un langage de partage de connaissances. Le développement des ontologies, dans les domaines, bioinformatique, et biologique, devient un pivot pour les *experts*, qui s'appuient sur le traitement de données en grande masse. Concrètement, nous comptons profiter de la représentation ontologique du domaine phylogénétique afin de faciliter la tâche d'acquisition de connaissances. Nous ne nous intéressons pas ici à résoudre un problème d'analyse phylogénétique mais plutôt à formaliser et abstraire ses concepts et pratiques processuelles. L'étude phylogénétique ne représente que le domaine d'application de notre approche ; une approche visant à suggérer un ensemble de "meilleures" pratiques.

Tout d'abord, au chapitre 1, nous présentons le contexte général de notre projet soit en (1) analyse phylogénétique, (2) acquisition de connaissances de résolution de problèmes et (3) extraction de flux de travaux concrets à partir des textes scientifiques, (4) fouille de flux de travaux abstraits et (5) recommandation basée sur le contenu. Nous présentons aussi dans ce chapitre les objectifs de recherche et les besoins de développement afin de réaliser notre projet. Au chapitre 2, nous nous attardons à présenter notre première contribution en extraction de flux de travaux concrets à partir des textes. Nous détaillons dans ce chapitre la solution proposée afin d'extraire les différents éléments des flux de travaux à savoir les procédures (techniques) et les types de données impliquées. Au chapitre 3, nous présentons la deuxième contribution de notre approche en fouille de motifs abstraits à partir des flux de travaux extraits. Ensuite, au chapitre 4 nous présentons l'architecture du système proposé dans ses deux modules d'extraction et de fouille de connaissances processuelles abstraites. Dans ce chapitre, nous reprenons les deux contributions dans leur cadre systémique. Enfin, au chapitre 5 nous étudions les résultats obtenus et leurs pertinences dans le domaine de l'analyse phylogéné-

tique. Nous présentons aussi une esquisse de solution pour le développement d'un système de recommandation de flux de travaux à partir des modèles abstraits générés.

Cette thèse inclut des textes d'articles publiés, soumis et en cours de soumission présentant les différentes contributions.

- Chapitre 2 : Halioui, A., Valtchev, P. et Diallo, A. B. (2017). Bioinformatic workflow extraction from scientific texts using word sense disambiguation, Dans IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2017. *(soumis)*
- Chapitre 3 : Halioui, A., Martin, T., Valtchev, P. et Diallo, A. B. (2017). Ontology-based Workflow Pattern Mining : Application to Bioinformatics Expertise Acquisition, Dans The 32nd ACM SIGAPP Symposium On Applied Computing (SAC), 2017. *(publié)*
- Chapitre 4 : Halioui, A., Valtchev, P. et Diallo, A. B. (2017). T-Gowler : Discovering Generalized Process Models within Texts. Dans Journal of Computational Biology (JCB), 2017. *(publié)*
  Halioui, A., Valtchev, P. et Diallo, A. B. (2015). Acquisition of generic problem solving knowledge through information extraction and pattern mining, Dans Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on (ICTAI), 2015. *(publié)*
- Chapitre 5 : Halioui, A., Valtchev, P. et Diallo, A. B. (2017). Towards an ontology-based recommender system for relevant bioinformatics workflows, Dans le journal BMC Bioinformatics, 2017. *(soumis)*

CHAPITRE I

CONTEXTE, NOTIONS DE BASE ET OBJECTIFS DE RECHERCHE

1.1 Préambule phylogénétique

1.1.1 Qu'est ce qu'une analyse phylogénétique ?

La phylogénie moléculaire est la discipline ayant pour objectif la reconstruction de l'histoire évolutive des espèces par comparaison des séquences de leurs gènes ou de leurs protéines. L'arbre phylogénétique est la forme-clé utilisée pour représenter l'évolution d'une séquence. Le cours évolutif présente un point de départ : une racine (le tout organisme) et des chemins évolutifs. Par exemple, la classification phylogénétique de l'*homo sapiens* (l'Homme), appartenant au genre "*Homo*", permet de classer la lignée humaine dans la super-famille *Hominidés* (*Hominidae* en anglais) qui englobe aussi des genres non-éteints de singes (grands-singes) ; par exemple l'homo sapiens partage plus de 95% de ses gènes (ADN) avec le chimpanzé. Aujourd'hui, de plus en plus d'arbres phylogénétiques d'espèces vérifiés sont disponibles aux chercheurs via des portails Web tel que le projet AToL (*A Tree of Life*) [1]. Les arbres de gènes individuels ou d'espèces nouvellement découvertes doivent être réalisés et vérifiés manuellement par différents experts et chercheurs

---

[1]Le projet Assembling the Tree of Life (ATOL) est un projet sponsorisé par le National Science Foundation du gouvernement américain (`http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503629`) visant à reconstruire l'arbre de toutes les espèces au fil de l'évolution

scientifiques. Des solutions en ligne fournissent des plateformes de flux de travaux, telles que la plateforme Armadillo (Lord *et al.*, 2012) et Taverna (Wolstencroft *et al.*, 2013), proposant une variété d'outils d'analyse phylogénétique dans des pipelines d'analyse semi-automatiques (Guler *et al.*, 2016). Cependant, aucun d'eux ne définit des "meilleures pratiques standards" en se basant sur des informations apriori comme les types de données, *e.g.*, protéines virales, ADN de plantes, ou les dépendances logiciels de services Web utilisés.

La reconstruction elle même de l'arbre phylogénétique ne présente pas le but de notre étude. C'est plutôt les façons dont les "experts" du domaine résolvent les problèmes phylogénétiques qui nous intéressent. Nous décrivons dans ce qui suit une description générale des processus de l'analyse phylogénétique (voir aussi Fig. 1.1) afin d'aider le lecteur à avoir une vue d'ensemble sur les concepts traités dans notre étude de cas.

1.1.2 Étapes d'analyse phylogénétique

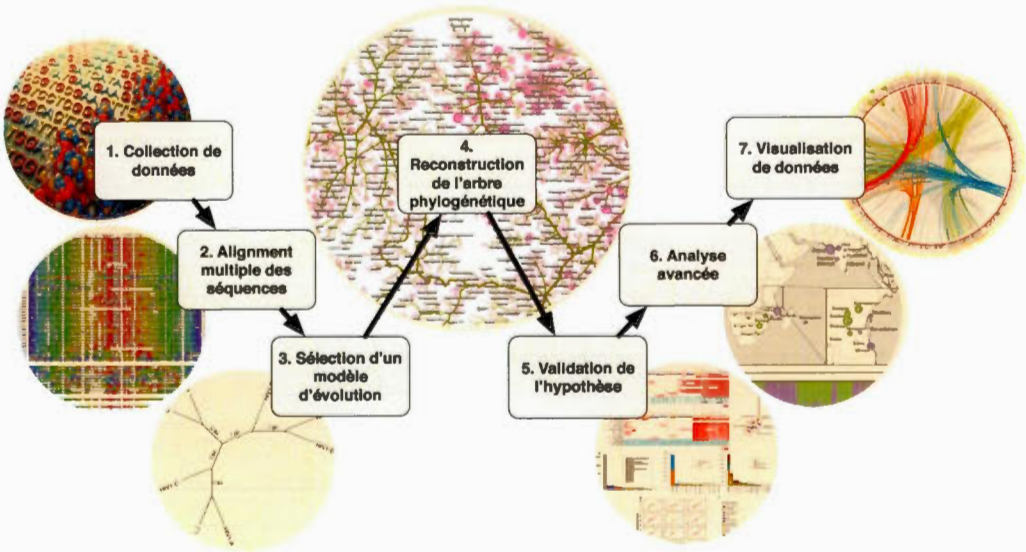


FIGURE 1.1: Pipeline général de l'analyse phylogénétique

1. Collection des données moléculaires. La collecte des séquences génomiques est la première étape dans l'analyse phylogénétique. Dans cette étape, il s'agit de chercher des séquences "similaires" dites *homologues* à un ensemble d'organismes (ou *taxa* son nom technique) et leur données moléculaires (*ADN*, *ARN*, protéine) dans des bases de séquences reconnues dans le domaine, telles que *NCBI*[2] ou *ENSEMBL*[3], afin de les "annoter" [4]. *BLAST* (Tatusova et Madden, 1999) est la méthode de recherche des homologues la plus répandue. Nous trouvons aussi d'autres outils comme *SSEARCH*[5] ou ceux basés sur des modèles markoviens cachés comme *HMMER*[6] ou *HHpred*[7]. Pour de plus amples informations, le lecteur peur se référer au travail de Finn *et al.* (2015). La qualité des séquences collectées par le séquençage et l'assemblage dans un laboratoire de biologie a un impact direct et majeur sur la reconstruction de l'arbre phylogénétique. Par exemple, une simple distinction erronée entre les paralogues[8] et les orthologues[9] peut générer des résultats incorrects. Finalement, la recherche des homologues doit répondre aux questions de l'analyse : Est ce qu'il y'a un nombre suffisant de données (de taxa

---

[2] `http://ncbi.nlm.nih.gov`

[3] `http://ensembl.org`

[4] Il s'agit d'identifier les propriétés et les composants moléculaires d'une séquence : régions codantes, structure, fonctions moléculaires, *etc.*

[5] `http://www.biology.wustl.edu/gcg/ssearch.html`

[6] `http://hmmer.org`

[7] `https://toolkit.tuebingen.mpg.de/hhpred`

[8] On dit que des gènes sont paralogues lorsqu'ils proviennent de la duplication d'un même gène mais qu'ils vont par la suite évoluer différemment suite à diverses mutations.

[9] Une fois l'homologie des gènes établie, si ces deux gènes ont un emplacement équivalent sur le génome, on dit qu'ils sont orthologues entre différentes espèces.

par exemple) ? ou Quelles séquences doit-on filtrer afin d'augmenter la fiabilité des résultats finaux.

2. Alignement multiple des séquences. L'alignement multiple représente l'hypothèse d'homologie qui réside dans les séquences à étudier. L'évolution des séquences choisies doit refléter ce que l'on veut calculer. Ainsi, chaque alignement est une sorte de test de cette hypothèse sur un descendant d'un ancêtre commun. Nous trouvons dans la littérature une multitude d'approches et techniques d'alignement multiple telles que *ClustalW* (Thompson *et al.*, 2002), *MUSCLE* (Edgar, 2004) et *ALIGN* (Cohen, 1997). Cependant, il faut bien étudier la motivation derrière l'utilisation de chaque méthode. Par exemple, si un alignement structurel est utilisé, alors il faut prendre en considération le biais qui vient avec une telle méthode : les séquences peuvent changer de structures durant l'évolution mais cela peut ne pas correspondre à une évolution historique. Les méthodes actuelles d'alignement peuvent engendrer des résultats différents en se reposant sur des hypothèses statistiques et algorithmiques différentes. Ainsi, les chercheurs biologistes et bioinformaticiens doivent prendre en considération telles types de contraintes.

3. Sélection d'un modèle d'évolution. Le but de cette étape est d'estimer les relations phylogénétiques (en termes de probabilités) dans les *taxa*. Les modèles d'évolution représentent le processus de substitution des nucléotides et des acides aminés dans l'histoire évolutive. Les méthodes existantes sont généralement interreliées. Elles diffèrent par le nombre de paramètres qu'elles comportent, les types de substitutions et les différences entre les probabilités de substitution. Nous trouvons des méthodes dédiées aux séquences ADN, comme GTR, K80 ou F81 (Cavalli-Sforza et Edwards, 1967) et des modèles dédiés aux protéines comme JTT et WAG (Abascal *et al.*, 2005). Il faut noter que, dans la plus part des temps, les

données s'apparient avec plusieurs modèles à la fois puisque chacun de ces modèles représente des propriétés différentes de l'évolution (*e.g.*, structure de codons, composition dynamique des nucléotides). Alors, il faut bien tester la relation entre les paramètres des modèles et les processus biologiques à étudier.

4. Inférence phylogénétique. Il s'agit de reconstruire un arbre phylogénétique décrivant l'histoire évolutive des *taxa* à étudier. Nous trouvons principalement deux types de méthodes, soit les méthodes basées sur la distance et les méthodes basées sur les caractères. Dans la première approche, il s'agit de calculer une matrice de distances génétiques, *e.g.*, distance de *Wagner*, *Neighbor Joining* (Lemey Philippe, 2009). Cette matrice représente une estimation du nombre de substitutions faites au cours de l'évolution. Dans la deuxième approche, nous trouvons des méthodes issues du maximum de vraisemblance et de parcimonie. Ici, il s'agit de calculer la probabilité de l'évolution à partir de différents paramètres du modèle d'évolution utilisé, les différentes structures de l'arbre phylogénétique et d'autres caractéristiques (Durbin *et al.*, 1998). Encore une fois, il faut bien étudier les arbres phylogénétiques inférés et le support (*e.g.*, probabilités postérieurs, distance de vraisemblance) qui vient avec chaque méthode. Prendre les mauvais paramètres peut engendrer des biais systémiques tels que l'Attraction des Longues Branches[10].

---

[10]L'attraction des longues branches est un artefact de reconstruction en phylogénie moléculaire qui provoque le regroupement des taxons qui évoluent d'une façon rapide sans refléter leur véritable lien de parenté. Ce phénomène est lié à l'accumulation de substitutions convergentes qui sont interprétées comme des synapomorphies (n'ayant pas évoluées dans une ou plusieurs 'branches' du taxon). L'attraction des longues branches affecte particulièrement les méthodes de parcimonie, notamment parce qu'elles n'autorisent pas de différence dans les taux de substitutions.

5. Validation. La robustesse des arbres estimés peut être évaluée via différentes approches de validation de topologie (groupes, clades) et/ou de calcul des erreurs aléatoires (échantillonnage de caractères, gènes non cohérents) ou systémiques (indépendances de caractères, distances corrigées, *etc.*). L'hypothèse représentée dans l'arbre estimé est généralement validée à l'aide des techniques de ré-échantillonnage telles que le *Jacknife* ou le *Bootstrap* (Lemey Philippe, 2009) . Dans le bootstrap paramétrique *seqBoot* (Plotree et Plotgram, 1989) par exemple : étant donné les séquences, une phylogénie et un modèle d'évolution, il s'agit de simuler l'évolution issues aléatoirement d'une distribution indépendante (Felsenstein, 1985). Ici aussi, il faut faire attention aux paramètres de la simulation (*e.g.*, type de distribution, échantillon de départ).

6. Analyse des arbres phylogénétiques. Il faut bien noter qu'une étude, sans une analyse de la significance des résultats dans un contexte biologique large, a peu de valeur scientifique. Dans cette étape, les arbres phylogénétiques générés sont analysés dans des études plus avancées comme dans les analyses de divergences des espèces dans le temps ou encore dans les analyses biogéographiques des organismes (dos Reis *et al.*, 2016). Nous trouvons ici des approches basées sur l'analyse de parcimonie comme *TreeFitter* (Ronquist, 2003) ou des approches statistiques comme *GeoDis* (Posada *et al.*, 2000) analysant les distributions des clades géographiques des haplotypes[11]. Effectivement, divers facteurs additionnels sont à considérer pour bien comprendre ces systèmes biologiques complexes. Par exemple, comprendre les niveaux d'expression d'un gène extrait à partir de plusieurs spécimens archéologiques, revient à étudier aussi la conservation de son ADN dans différentes conditions expérimentales selon plusieurs paramètres.

[11]L'ensemble de gènes hérités du même parent dans le même chromosome.

7. Visualisation de données. La visualisation des résultats d'analyses phylogénétiques pose un problème en soi, d'autant plus qu'il s'agit d'analyser les résultats de grands espaces de distributions. Divers logiciels de visualisation d'arbres sont disponibles comme *Phylodendron*, *FigTree* et *TreeView* (Saldanha, 2004). La majorité de ces programmes fournissent des outils statistiques et d'exportation de données sous différents formats portables. Par ailleurs, une mauvaise visualisation de données peut déformer l'interprétation des résultats.

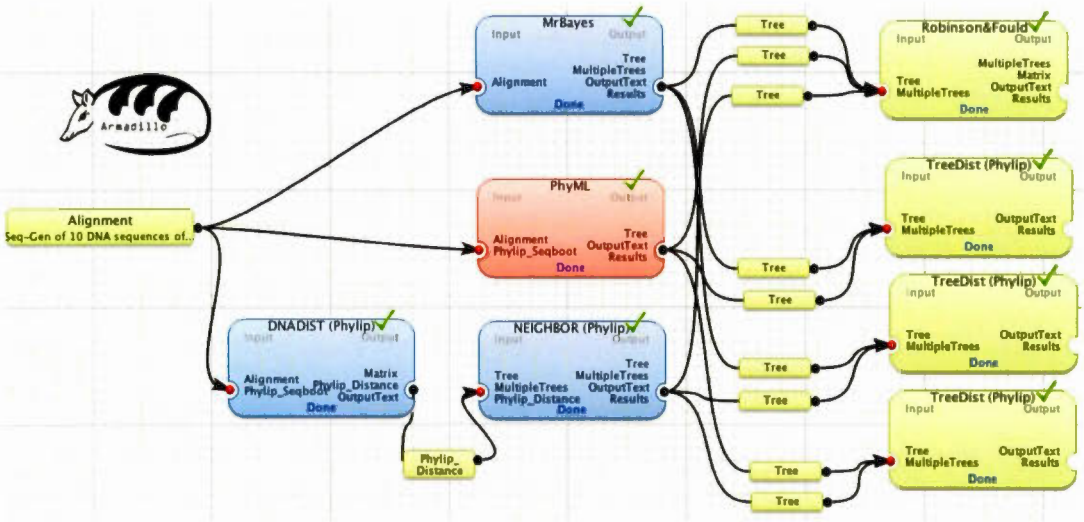


FIGURE 1.2: Exemple d'analyse phylogénétique des génomes mitochondriaux dans la plateforme Armadillo (Lord *et al.*, 2012)

Nous décrivons un scénario d'analyse phylogénétique dans la figure 1.2. Ce flux de travail décrit des processus phylogénétiques dont l'alignement est l'entrée de l'analyse. La distance phylogénétique a été calculée par *DNADist*[12] ; la reconstruction des arbres phylogénétiques été assurée par *MrBayes*[13], *PhyML*[14] et *NEIGH-*

[12] `http://evolution.genetics.washington.edu/phylip/doc/dnadist.html`

[13] `http://mrbayes.sourceforge.net`

[14] `http://www.atgc-montpellier.fr/phyml`

BOR[15] ; la validation des topologies générées été faite par l'algorithme de *Robinson & Fould*[16] et la validation des longueurs de branches a été assurée par *TreeDist*[17].

Les analyses phylogénétiques présentent différentes approches, algorithmes et programmes (logiciels) dans chacun de ses processus ainsi que différents types et formats de données posant ainsi des contraintes additionnelles complexes sur le problème à résoudre. Cette mosaïque d'objets que nous envisageons de modéliser l'acquisition présente différents défis liés au domaine de l'application et à la manipulation "informatique" de ses outils.

1.2 Flux de travaux et acquisition de connaissances de résolution de problèmes

Le concept *flux de travaux* a été initialement définit dans le domaine d'affaires en 1996 par la WfMC[18] (Workflow Management Coalition) comme : "*the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*". Par ailleurs, les flux de travaux scientifiques, reconnus entant "*paradigme utile pour décrire, gérer et partager des analyses scientifiques complexes*" (Van der Aalst, 2011) , sont utilisés dans la résolution automatique de problèmes en créant des processus comportant différentes étapes. Ainsi, nous proposons la définition suivante des flux de travaux :

> *un flux de travail représente des connaissances de résolution de problème dans un ensemble de tâches (activités) interconnectées par des*

---

[15] `http://evolution.genetics.washington.edu/phylip/doc/neighbor.html`

[16] `http://adn.biol.umontreal.ca/~numericalecology/old/robinson_foulds.html`

[17] `http://evolution.genetics.washington.edu/phylip/doc/treedist.html`

[18] `http://www.wfmc.org`

> *flux de contrôles (e.g., séquences, procédures parallèles, itérations, etc.) et des flux de données (d'entrées et de sorties). Ces objets (tâches et données) et interactions (relations entre ces objets) sont souvent annotés par des descripteurs sémantiques soit des métadonnées décrivant la provenance de données et les contraintes expérimentales.*

Ainsi, nous considérons qu'un flux de travail est un graphe étiqueté orienté (*Directed Acyclic Graphs - DAG* en anglais) : $W(V_W, E_W, L_W, \varrho_W)$ où V_W et E_W représenten l'ensemble d'objets et l'ensemble de relations, respectivement, et ϱ_W : $(V_W, E_W) \rightarrow L_W$ la fonction qui assigne les étiquettes $l \in L_W$ aux objets et aux relations. Afin de représenter le raisonnement derrière l'acquisition des connaissances processuelles L_W, nous avons besoin de comprendre les types de connaissances impliquées et les étapes de leur acquisition. Nous présentons dans ce qui suit une revue de la littérature servant à éclairer les concepts clés dans l'acquisition de connaissances processuelles.

1.2.1 Connaissances impliquées dans la résolution de problèmes

Toute connaissance scientifique, comme dans la phylogénie et tout autre domaine scientifique, est fondée sur trois types d'information – observation, lois et théories. Les observations sont généralisées dans des lois et ainsi expliquées par des théories. Cependant, dans un laboratoire, les praticiens, et en particulier les biologistes phylogénéticiens, fournissent "intensément" d'effort dans la construction de *protocoles de processus* et dans la collecte et le *pré-traitement* de données et d'outils. Ceci vient avant même de raisonner sur les observations ou les théories. Par conséquent, l'élément de base dans le processus de résolution de problèmes est la connaissance processuelle et procédurale représentant le *savoir faire* humain. Il est important de noter la différence entre les connaissances déclaratives, aussi dites propositionnelles ou explicites (savoir-quoi) et les connaissances processuelles

(savoir-comment). Tandis que le premier type de connaissances repose plutôt sur les faits, les connaissances processuelles sont basées sur l'expérience et la compétence des individus (Sahdra et Thagard, 2003). Myaeng *et al.* (2013) appellent les connaissances liées aux processus, des connaissances *expérientielles* :

> *"Experiential knowledge [...] as the knowledge gained through experience as opposed to a priori (before experience) knowledge" or "knowledge that can only be acquired through experience. [...] This definition has a few elements that affect the process of experience mining and the associated techniques when it is done automatically. First, something must "have happened" in the past if it is to be considered as experience. This is particularly important especially when a textual description is analyzed and discerned for existence of an experience. Second, an experience must have a context because it must have happened. A real event or activity cannot happen without a context regardless of whether or not it is described explicitly or assumed implicitly. Third, in order for an activity or event to become an experience, it must have influenced the way the experiencer think or behave."*

Dans leur définition, trois éléments interviennent dans l'acquisition de l'expérience : (1) les connaissances apriori, (2) le contexte de l'expérience et (3) le raisonnement sur l'activité/événement pour produire une expérience. Le savoir faire est ainsi le produit de ces éléments. Nous revenons plus tard sur ces trois concepts dans les sections suivantes.

Dans la philosophie du processus, en particulier dans la métaphysique d'Alfred North Whitehead (Bogost, 2010), tout est impliqué dans des processus dynamiques :

> *"The idea is that being is an event, beings are events, and that existence fundamentally involves flow and change and inheritance. [...] Everything is constantly renewing itself through a series of perishing occasions [...] Each actual entity is a process proceeding from phase to phase , each phase being the real basis which its successor proceeds toward the completion of the thing in question."*

Ici, Whitehead abandonne le matériel en faveur des processus; tout devient des événements et des continuités. Face à ce rejet du discret, nous trouvons que dans l'informatique est utilisé plutôt le terme "connaissances procédurales" (Corbett et Anderson, 1994). Dans ce contexte, un processus représente deux aspects : (1) la façon dont les choses fonctionnent (méthodes, techniques, *etc.*) comme dans les systèmes de gestion de flux de travaux bioinformatiques et (2) la façon dont une computation fonctionne. Afin de clarifier toute confusion entre processus et procédure, nous distinguons bien l'aspect *processuel* des processus de son aspect *computationnel*. Pour cela nous utiliserons le terme "*processus*" pour désigner un flux des travaux et les termes "*technique*" ou "*algorithme*" au lieu de "procédure" pour représenter les logiques (symboles) et leurs computations. Les connaissances de résolution de problèmes impliquent ainsi des connaissances processuelles, procédurales ainsi que déclaratives.

Toutes ces connaissances présentent plusieurs défis et ceci peu importe le niveau d'expertise des individus. Sternberg et Ben-Zeev (2001) ont identifié quatre caractéristiques distinguant un novice d'un expert en résolution de problèmes (voir le tableau 5.5 dans l'annexe *E*). Les recherches de Daley (Daley, 1999) ont abouti à découvrir que les novices tendent à trouver dans leur processus d'apprentissage un processus d'acquisition de nouvelles connaissances en liant le problème à résoudre à des cas ressemblant à ce qu'ils ont déjà vu. Dans cette approche (*best-fit approach*), il s'agit de trouver le ou les meilleurs modèles de résolution similaires au

problème à résoudre. Par ailleurs, Forneris et Alpine (Forneris et Peden-McAlpine, 2007) prouvent que le raisonnement critique se trouve plutôt dans la maitrise de la connaissance du domaine (déclarative) en impliquant des perspectives plus large de la connaissance. Dans (Sahdra et Thagard, 2003), les auteurs définissent les connaissances procédurales chez les experts à travers l'*Intuition* derrière la reconnaissance des motifs (patrons ou *patterns* en anglais) les plus pertinents FP de son environnement afin de prendre une décision et réagir. Selon Dreyfus and Dreyfus (1986) et Klein (1998), l'intuition dépend de l'utilisation de l'expérience pour reconnaitre les motifs clés qui indiquent la dynamicité d'une situation. Face à différentes tâches et grâce à son intuition, un expert reconnait facilement le motif clé FP et certaines attentes. La reconnaissance de FP permet ainsi de définir ses buts et ses actions. Si l'expert détecte une anomalie, il peut chercher les attributs qui correspondent mieux à la situation. Il peut ainsi modifier son expérience face aux changements et évaluer (grâce à une simulation mentale) ses décisions/actions en imaginant l'impact de ses actions sur les buts/motivations.

1.2.2 Raisonnement

Le but du raisonnement est de chercher à produire des stratégies basées sur l'apprentissage à partir des expériences antérieures. La résolution d'un problème prédéfini implique "une recherche systématique d'une solution dans un espace d'actions possibles" (Rousse, 2013).

Un espace de problèmes peut être défini comme suit :

> "*A problem space comprises symbolic structures and operators–actions that begin with an input and produce an output–as well as the problem itself. The problem is defined through a set of initial states, a set of goal states, and a set of path constraints*". (Sternberg et Ben-Zeev,

2001)

Par exemple, un chercheur biologiste doit trouver un/plusieurs chemin(s) clair(s) d'une solution et doit être capable d'identifier les étapes requises afin d'atteindre son but. En effet, l'identification et l'exécution de ces étapes présentent en soi le défi à affronter. Sternberg et Ben-Zeev (2001) présentent les stratégies de résolution dans un espace de problèmes en définissant non seulement les opérateurs d'actions mais aussi le coût de l'efficacité de ces opérateurs. Un expert est celui qui peut, en même temps, expliquer les étapes choisies et évaluer les actions qui supportent ses explications. En d'autres termes, un biologiste doit trouver les meilleurs outils dans chaque sous-problème afin de résoudre le problème global. Par exemple, une fois que les données génomiques sont collectées, il doit résoudre le problème d'alignement de ses séquences en tenant compte du type de séquences (ADN, ARN ou protéines), de l'espèce étudiée (plante, archée, ...), des fonctions moléculaires et cellulaires impliquées (enzymes, ligand, ...), *etc.* Ce problème est ainsi défini dans un espace d'objectifs/sous-objectifs à atteindre qui sont interreliés par des contraintes (dépendances logicielles – configurations ou du domaine –hypothèses).

Les nouvelles techniques de résolution de problèmes, notamment basées sur les flux de travaux et les services Web (Blankenberg *et al.*, 2001; Lord *et al.*, 2015; Chen et Chang, 2017), sont de plus en plus utilisés grace à leur "facilité" de mise en place, mais aussi par la qualité des solutions qu'elles sont capables de trouver en un temps "assez court". Cependant, elles se heurtent à un problème bien connu en informatique : la "malédiction de la dimension". Bien que les solutions actuelles de fouille de données se basent sur un nombre important de variables à optimiser (parfois plusieurs milliers), des heuristiques sont initialement mises au point sur des problèmes d'une dizaine de variables (Witten *et al.*, 2016). On admet dans la littérature que l'on peut qualifier de "problème à haute dimension" un problème

dont le nombre de variables est supérieur à 100. Ces problèmes nécessitent une approche différente. De fait, la plupart des méthodes cités ne sont pas robustes à l'augmentation de la dimension, et ne convergent plus en un temps raisonnable (d'exécution), lorsque ce nombre devient assez important. Le nombre important des combinaisons de paramètres (configuration de algorithmes), leur dépendances (non triviales à représenter) ainsi que la grande diversité des solutions actuelles bioinformatique et des hypothèses adoptées dans chaque étapes rendent la tâche d'optimisation difficile à réaliser avec les méthodes actuelles de fouille de données et d'apprentissage automatique.

Nous trouvons dans les littératures de la *cognition complexe* et de l'*apprentissage automatique* différentes stratégies de résolution de problèmes (Kahney, 1986; Mitchell *et al.*, 2004; Witten *et al.*, 2016). Dans l'approche *ascendante* il s'agit de commencer par la résolution du but final et ensuite trouver un chemin dans l'espace de problèmes qui valide l'hypothèse et l'état initial. Contrairement à l'approche *descendante* dans laquelle l'algorithme de résolution de problèmes cherche à trouver le meilleur chemin (dans l'espace de problèmes) en commençant par l'état initial.

En outre, le choix du/des meilleur(s) P dans cet espace de recherche revient à trouver la meilleure expérience antérieure et la *formaliser* à travers une inférence (consciemment ou pas) et ensuite la transformer en une action. Les théories computationnelles actuelles ne permettent pas de répondre "clairement" à la façon dont l'expert infère des nouvelles connaissances processuelles et procédurales. L'approche symbolique, avec Newell (1990) et Anderson (1993) de la cognition propose des modèles basés sur des règles : $IF < pattern > THEN < reaction >$. Dans ce contexte, l'expertise est représentée dans la prémisse de la règle, cependant, elle doit être verbalisée dans des clauses. Holyoak (1991) soutient plutôt l'approche connexionniste de la représentation de l'expertise dans laquelle la prise

de décision est basée sur la satisfaction parallèle des contraintes dans un modèle inspiré du réseau de neurones chez l'humain. Les neurones sont interconnectés par des contraintes positives et négatives et la décision requiert un processus de calcul de la meilleure fonction de satisfaction. Cependant, ce modèle est loin de la représentation "réelle" de la conscience humaine puisque l'accès parallèle à ces processus de satisfactions se fait de manière "inconsciente".

Par ailleurs, nous trouvons d'autres approches socio-cognitives simulant l'acquisition des connaissances de résolution de problèmes. Dans la théorie des activités (Kuutti, 1996), les chercheurs essayent de comprendre la relation entre les activités et l'esprit humain (conscience). Nous trouvons ainsi le modèle de Vygotsky qui modélise une activité par le triplet (*sujet*, *outil*, *objet*), le modèle de Engeström qui introduit l'aspect collaboratif (rôles des acteurs) dans les processus et d'autres modèles analysant les aspects, hiérarchique et sémantique, entre les trois éléments des activités (Myaeng *et al.*, 2013). Dans la théorie des "*Scripts*" de Schank et Abelson (voir leur nouvelle édition (Schank et Abelson, 2013)), un script permet de faire des économies de raisonnement en guidant l'interprétation d'événements rencontrés dans des récits/activités courantes sauvegardées dans une mémoire épisodique. Les scripts se reposent sur des schémas (Bartlette (1995) et d'actions qui garantissent la cohérence et "s'occupent" des détails de résolution de problèmes. Le Raisonnement à Partir des Cas (RàPC) est une implémentation de cette notion de scripts et de mémoire épisodique où les cas décrivent des solutions à des problèmes déjà rencontrés. Il s'agit après d'apparier les éléments du nouveau problème avec les éléments du schéma des scripts stockés dans la mémoire. Dans la théorie des "*Frames*" de Gordon (1998), une différente implémentation de la théorie de scripts est proposée. Une activité est représentée par un frame, qui consiste à remplir un ensemble de "slots" (e.g. événement, lieu, individu, *etc.*) lorsqu'une situation est rencontrée. Les valeurs de slots sont organisées dans un

thésaurus de termes (réseau sémantique) liés pour construire des clusters d'activités (activités communes). En changeant les frames, le contexte des activités communes changent aussi. Les frames montrent ainsi l'importance du *contexte* dans la description d'une activité.

1.2.3 Problèmes d'acquisition de connaissances processuelles

Désormais, avec le Web 2.0 et les nouvelles technologies du Web Sémantique (Faatz et Steinmetz, 2002), le Web devient de plus en plus structuré (mais encore loin de la structuration complète) et donc de plus en plus exploitable en tenant compte de la grande masse de textes représentant différents types de savoir faire : blogs, forums, how-to, *etc.* Les pages Web présentent alors une source intéressante pour la fouille d'expériences. Dans ce contexte, la tâche d'acquisition de connaissances de résolution de problèmes utilise des techniques de Traitement Automatique de Langage Naturelle (TALN) (Cunningham *et al.*, 2002a) afin de faciliter l'élicitation de données à partir des textes. Il ne s'agit pas de remplacer le rôle d'un expert dans l'acquisition des connaissances mais plutôt de l'aider à encoder et formaliser ses connaissances. Toutefois, les outils actuels du TALN engendrent des erreurs d'encodage liées au contexte et à d'autres ambiguïtés sémantiques (Zempleni *et al.*, 2007). Impliqué dans un processus d'annotation, le rôle d'un expert reste toujours primordial dans l'acquisition de processus. Néanmoins, plusieurs difficultés entravent son rôle (Sternberg et Ben-Zeev, 2001). Nous citons dans la suite quelques unes.

Habilités Cognitives. Dans la linguistique cognitive, la rigidité cognitive (ou *entrechment* en anglais) réfère à la non flexibilité de l'expérience des experts dans la stabilité/instabilité de leurs schémas du domaine. Dane (2010) a étudié ce compromis entre l'expertise et la rigidité cognitive :

> "*I adopt the term cognitive entrenchment (or, simply, entrenchment) to refer to a high level of stability in one's domain schemas. Although cognitive entrenchment may not arise solely among experts, the acquisition of domain expertise can lead to such entrenchment. The schema stability characterizing cognitive entrenchment may emerge, at least in part, from the frequency with which experts tend to draw on their domain schemas. In particular, because expertise is accrued through continual training, practice, and performance, the content and relations comprising an expert's domain schemas are likely to be activated and applied innumerable times. This repeated activation and use may stabilize the structure of the expert's schemas such that schema revision becomes relatively unlikely.*"

Indépendamment du niveau d'expertise de l'encodeur X, ce dernier peut être immergé dans des "motifs habituels" de raisonnement qui peuvent biaiser l'acquisition de connaissances. Par exemple, un biologiste X peut se tromper dans l'identification de la classe d'un outil o dans un texte en reconnaissant tout simplement le terme utilisé. Le programme *PhyML* [19] est généralement utilisé dans l'étape d'inférence d'arbres phylogénétiques. Toutefois, ce programme peut être aussi utilisé comme un paramètre dans un autre programme afin de résoudre une étape différente, comme par exemple dans le calcul de mesure de vraisemblance pendant l'étape de la validation des arbres phylogénétiques. X peut se tromper ainsi dans l'annotation de l'outil o parce qu'il a l'habitude de voir o dans la même étape.

---

[19] `http://www.atgc-montpellier.fr/phyml`

Charge cognitive excessive. La charge cognitive de X peut accroître lors de l'attribution des ressources à des activités prédéfinies. X doit décider quelles ressources sont impliquées dans la résolution d'une étape. Ceci peut engendrer chez X une sorte de stress et d'anxiété excessive lorsqu'on accorde une période de temps bien déterminée pour exécuter sa tâche d'annotation. L'acquisition de connaissances processuelles et procédurales peut être aussi assurée via des canaux de communication classiques comme la communication "de personne à personne". C'est là où nous trouvons d'autres types d'obstacles d'acquisition. Un expert est souvent occupé ou débordé et accorde souvent peu de temps à des discussions dont le but est d'extérioriser des connaissances processuelles touchant son savoir faire. Il peut, peu importe ses excuses, annuler un rendez vous ou tout simplement avoir des difficultés à répondre à une question. Aussi, lors d'une discussion, un expert peut développer un sentiment de crainte, voire de menace, et il peut donner par conséquent de fausses informations. Il n'existe pas de solutions spécifiques pour ce genre de problèmes. Il faut tout de même garder une bonne relation personnelle avec l'expert (Hart, 1988).

La question majeure que nous posons dans la suite est : *Comment rendre la grande masse de connaissances liées aux activités décrites dans des textes sous un format "compréhensible" par la machine (machine readable) avec un degré de granularité suffisant afin de les intégrer dans des applications "intelligentes".* Le majeur défi reste dans l'identification des processus d'activités, l'extraction, l'interprétation, l'abstraction et l'agrégation des éléments de l'expérience en différents niveaux de granularité.

1.2.4 Une approche basée sur les ontologies

Nous essayons de résoudre le problème d'acquisition des connaissances processuelles abstraites en nous appuyant sur une approche d'extraction basée sur une

ontologie du domaine afin de représenter les différents niveaux d'abstraction du savoir faire. Décrite dans un language formelle, comme avec les logiques de description (Baader, 2007), l'ontologie *définit un ensemble structuré catégorisant les termes du domaine dans des Concepts et des Relations représentant les différentes sémantiques dans différents niveaux d'abstraction*[20]. Ainsi, tout comme un flux de travail W, une ontologie Ω est considérée comme un DAG où les noeuds représentent les concepts et les arêtes représentent les relations entre ces concepts. Nous présentons dans la figure 1.3 un exemple d'ontologie dans le domaine bioinformatique.

Ces ontologies peuvent servir de support aux utilisateurs experts ou novices afin de résoudre leurs problèmes complexes selon leurs niveaux d'expertise. Dans un niveau concret de l'ontologie nous trouvons des concepts et des relations touchant à des problèmes spécifiques tels que le calcul de distances entre des séquences ou le calcul d'un modèle de substitution de nucléotides. Dans un niveau plus haut d'abstraction, nous trouvons plutôt des concepts génériques qui représentent le but global d'un problème : *e.g.*, l'inférence phylogénétique. La *généralisation* (appelée aussi *subsomption*) des concepts concrets en concepts abstraits doit prendre en considération les niveaux d'abstraction suffisants pour une modélisation de résolution de problèmes (Staab *et al.*, 2010). La généralisation doit ainsi se limiter à un niveau d'abstraction particulier. Par ailleurs, la finalité des représentations n'est pas dans la modélisation fidèle de la réalité d'un domaine. Ce n'est pas dans la théorie que nous devons aller chercher des formes explicites mais c'est plutôt dans certaines attendues pratiques. Le rôle de l'ontologie n'est pas alors théorique mais plutôt opératoire : *Elle vise à concevoir certaines aptitudes d'analyse*

[20]Il faut noter qu'une abstraction (généralisation) permet de regrouper un certain nombre de concepts/relations selon des caractéristiques communes. Un niveau d'abstraction décrit alors la profondeur du concept/relation dans l'ontologie.

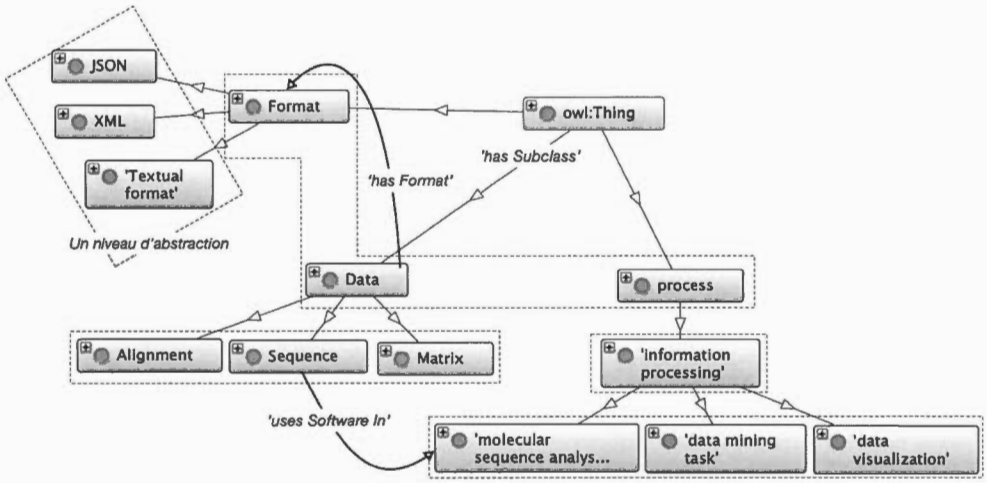


FIGURE 1.3: Portion de l'ontologie "The Software Ontology" (SWO) (Malone *et al.*, 2014). Les concepts sont représentés par des rectangles gris et les relations par des arêtes étiquetés. La hiérarchie des concepts est décrite par les relations '*has SubClass*' définissant ainsi les différents niveaux d'abstraction (rectangles pointillés). Par exemple l'ensemble des concepts '*Format*', '*Data*' et '*Process*' représente le premier niveau dans l'ontologie SWO.

phylogénétique offrant à l'humain une assistance efficace. Notre hypothèse d'acquisition de processus repose ainsi sur (1) l'agrégation des instances individuels des expériences dans des abstractions (catégories), (2) l'ancrage de la sémantique des concepts expérientiels (de processus) dans des différents niveaux d'abstraction, (3) l'organisation et la liaison de ces expériences pour obtenir des résultats interprétables directement avec un niveau raisonnable de confiance pour la prise de décision.

L'extraction du savoir faire en flux de travaux à partir des textes s'inscrit dans le contexte de découverte de données et en particulier dans les contextes de fouille de textes, fouille de motifs processuels et de recommandation basée sur une ontologie

du domaine. Nous détaillons dans ce qui suit le contexte générale de chacun[21].

1.3 Extraction de flux de travaux à partir des textes

Comme le souligne la plupart des travaux sur le sujet, notamment de Dufour-Lussier *et al.* (2014); Schumacher *et al.* (2013); Müller et Bergmann (2014), l'extraction automatique de processus à partir des textes est basée sur le RàPC et est appuyée par des techniques d'extraction de l'information (Gonzalez *et al.*, 2016) issues du domaine de TALN. Ces approches sont basées sur des bases de connaissances terminologiques afin d'extraire les différents éléments de flux de travaux W. Ces terminologies peuvent être représentées par des ontologies représentant des concepts et des relations du domaine d'application. Le processus global d'extraction procède par couches successives dans la représentation ontologique : termes, synonymes, concepts, hiérarchies, relations, *etc.* La difficulté principale est liée au fait que les concepts se manifestent à travers des unités lexicales qui admettent souvent plusieurs sens : la *polysémie* (Stevenson et Wilks, 2003). L'extraction de termes candidats vise les constructions susceptibles de désigner des concepts du domaine. Nous détaillons, dans ce qui suit, différentes approches utilisées dans le domaine d'extraction de l'information, en particulier l'extraction automatique des termes et celles des relations représentant des connaissances processuelles (voir aussi les tableaux récapitulatifs 5.6 et 5.7 dans l'Annexe F).

1.3.1 Extraction des entités nommées.

Une *entité nommée* est une expression linguistique qui fait souvent référence à un *terme* (un ou plusieurs *entités lexicales* –*tokens* en anglais), représentant une/plusieurs catégories de noms telles que gènes, espèces, approches d'inférence

[21]Nous revenons sur chacun de ses contextes d'une façon exhaustive dans les chapitres concernés.

phylogénétique ou de validation d'arbres. Nous reprenons ici des définitions connues dans le domaine de Reconnaissance des Entité Nommées (REN) comme celle de Grishman (2012) : "*Information Extraction is the automatic identification of selected types of entities, relations, or events in free text.*" et de Cardie (1997) "*Very generally, an information extraction system takes as input an unrestricted text and "summarizes" the text with respect to a prespecified topic or domain of interest : it finds useful information about the domain and encodes that information in a structured form, suitable for populating databases.*". Ainsi, la REN peut être définie comme : *la tâche d'identification et de classification des termes dans une catégorie conceptuelle.* Dans ce contexte de classification, les entités nommées sont décrites via l'ensemble d'attributs qui les définissent : caractéristiques associées aux termes afin de distinguer ses significations (classes). Dans ce cadre, nous considérons l'Apprentissage Automatique (Witten *et al.*, 2016) en tant qu'approche de classification des entités nommées. Nous trouvons plusieurs approches proposées dans la littérature des REN (Leaman *et al.*, 2015; Provoost et Moens, 2015; Choi *et al.*, 2016). Chacune aborde et redéfinit le problème de classification de termes d'une façon bien particulière. Nous avons classifié ces approches sous 4 catégories : (1) approche classique, (2) approche par apprentissage supervisé, (3) approche par apprentissage semi-supervisé et (4) approche par apprentissage non supervisé.

Approche classique. L'approche classique de reconnaissance est basée sur des motifs et des règles lexico-syntaxiques écrites à la main. Nous trouvons des travaux comme (Sekine et Nobata, 2004) utilisant une terminologie/dictionnaire de mots (*Gazetteer* en anglais) externe – une liste de mots déjà étiquetés – pour reconnaître un terme dans le texte. Cette technique simple et rapide ne résout pas les ambiguïtés sémantiques du langage naturel (Stevenson et Wilks, 2003). De plus, cette liste doit être mise à jour et maintenue d'une façon régulière. D'autres

techniques heuristiques sont aussi utilisées dans la classification des termes (Li *et al.*, 2012; Bontcheva *et al.*, 2013). Il s'agit de définir une grammaire spécifique pour identifier chaque catégorie sous forme de règles formelles. Il nous semble important de mentionner que cette approche est privilégiée lorsqu'un ensemble d'apprentissage n'est pas disponible.

Apprentissage supervisé. Les techniques d'apprentissage basées sur un modèle de classification supervisé sont les techniques les plus répandues dans le domaine de REN. Nous trouvons des systèmes d'apprentissage basés sur les Chaînes de Markov Cachées (HMM) (Zhou et Su, 2002), le Maximum d'Entropie (ME) (Chieu et Ng, 2002b), les Supports à Vaste Marges (SVM) (Kazama *et al.*, 2002) et les Champs Markoviens Conditionnels (CRF) (Settles, 2004) (voir aussi Nadeau et Sekine, 2007; Witten *et al.*, 2016; Aparna et Ku, 2016). Par exemple, de Bruijn *et al.* (2011) proposent d'utiliser un semi-HMM (Witten *et al.*, 2016) représentant une série de token-attributs (bag-of-token-features) représentant chaque concept à classer. Leur modèle est basé sur des attributs de type "niveau-de-discours" : *token* (*e.g.*, ponctuation, préfixe, stem[22]), *contexte* (*e.g.*, bi-tri-quad-grammes, tokens voisins), *phrase* (*e.g.*, longueur, premier mot) et *section* (*e.g.*, titres et sous-titres). Toutefois, il a été montré que la qualité d'annotation peut être améliorée en utilisant des lexiques externes. Xu *et al.* (2012), par exemple, proposent, en plus des attributs de *niveau-de-discours*, d'autres : de type *Partie-Du-Discours* [23] (ou *Part Of Speech* - POS en anglais) et de type sémantique en appariant

[22]En linguistique, la *racinisation* (ou *stemming* en anglais) est un procédé de transformation des flexions (ensemble des modifications subies par le *signifiant*) en leur radical ou racine, *i.e.* la partie restante d'un mot une fois que l'on a supprimé son (ses) préfixe(s) et suffixe(s).

[23]Les classes grammaticales (nature des mots), *e.g.*, Verbe, Nom propre, Pronom

les sac-de-tokens avec les mots dans les dictionnaires *MeSH*[24] (*Medical Subject Headings*) et UMLS[25] (Unified Medical Language System). Leaman *et al.* (2015) proposent aussi dans leur système *tmChem* des attributs spécifiques au domaine décrivant des motifs de formules chimiques, séquences nucléiques et protéiques, *etc.* Par ailleurs, la segmentation de textes en unités-lexicales est elle-même une tâche difficile surtout dans un domaine très particulier comme la phylogénie où différents vocabulaires sont utilisés comme l'informatique, les mathématiques et la biologie. Pour cela, nous trouvons, par exemple, des modèles comme *CRF* qui intègrent à la fois les deux tâches de segmentation et d'annotation dans un seul modèle couplé à des ressources lexicales riches (Aparna et Ku, 2016). Dans ce cadre, le texte est pris comme des séquences de mots étant dépendants l'un de l'autre par des liens causaux.

Apprentissage semi-supervisé. La technique de base utilisée dans cette catégorie est celle du *Bootstrapping* (Dietterich, 2000) dans laquelle un "petit" échantillon d'apprentissage est utilisé. Par exemple, un oracle (un expert) disposant d'un nombre limité et fini de noms de gènes, peut créer un vocabulaire de contexte à partir des noms de gènes déjà étiquetés dans le texte. Le processus d'apprentissage ré-applique ensuite les contextes créés dans d'autres échantillons d'exemples pour en découvrir des nouveaux contextes. En répétant le processus plusieurs fois, un grand nombre de noms de gènes et vocabulaires sont créés. Nous trouvons un

---

[24]MeSH est un système de métadonnées médicales en langue anglaise en se basant sur l'indexation d'articles en sciences de la vie. Cette initiative a été lancée et est toujours maintenue par la Bibliothèque nationale de médecine des États-Unis d'Amérique : `https://www.nlm.nih.gov/mesh/`

[25]UMLS est un méta-thesaurus médical catégorisant les mots-clés du domaine : `https://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html`

récent travail de Chen *et al.* (2015b) qui est basée sur une approche d'apprentissage actif[26] (Witten *et al.*, 2016) visant à réduire le coût d'annotation. Les auteurs proposent trois algorithmes basés sur : (1) l'incertitude, (2) la diversité et (3) un algorithme de base. Dans le premier algorithme, les phrases incertaines (en se basant sur l'Entropie des mots et des séquences (Ebeling et Nicolis, 1991)) sont considérées comme les plus discriminants. Dans l'hypothèse de diversité, des scores de similarité deux-à-deux entre les phrases sont calculés, à savoir : (i) la similarité syntaxique basée sur les fréquences des classes grammaticales dans les phrases et (ii) la *similarité cosinus* des concepts UMLS en se basant sur le plus petit ancêtre commun. L'algorithme de base calcule tout simplement le nombre de mots et de concepts dans chaque phrase et favorise les phrases courtes ayant le plus nombre de concepts du domaine. Basée sur une extension de l'algorithme SVM, Provoost et Moens (2015) proposent, afin de d'équilibrer les données positives et négatives, de re-positionner l'hyperplan séparant les deux classes en ajoutant deux paramètres (représentant les distributions des deux classes) dans la fonction de maximisation de la marge. De plus, un algorithme de pré-sélection de données non-annotées est proposé pour chercher les exemples négatifs selon un seuil de fonction de masse (de probabilités) sur les vecteurs d'attributs. Cependant, ce type d'apprentissage n'améliore la qualité d'annotation et de classification des entités nommées que lorsqu'une grande collection de documents est disponible.

Apprentissage non supervisé. Dans cette approche, il s'agit de retrouver des groupes (*clusters*) d'entités nommées en se basant sur leur similarité de contexte. Liu *et al.* (2015) proposent d'utiliser une variante d'apprentissage profond (*deep learning*) (LeCun *et al.*, 2015) avec un CRF en utilisant le modèle skip-gram (Levy et Goldberg, 2014) comme une méthode de représentation dynamique (*word em-*

[26] via des "feedbacks" interactifs avec l'expert

bedding en anglais) de cooccurrences de mots en des vecteurs de nombres réels afin de reconnaitre les noms de médicaments dans les résumés des articles MEDLINE[27]. Cette technique prédit le contexte d'un mot en se basant sur la représentation de ce mot en maximisant sa probabilité *softmax* avec son contexte. Plus récemment, Choi *et al.* (2016) proposent un nouveau modèle *Med2Vec* pour apprendre les représentations des cooccurrences de codes médicales dans une large base de données de ~3 millions de visites. Ce modèle, en deux couches cachées de Perceptron (Witten *et al.*, 2016), représente à la fois les cooccurrences de codes et les séquences de visites.

1.3.2 Extraction des relations

L'extraction automatique des relations sémantiques à partir d'un corpus de textes représente l'un des grands défis dans notre approche d'extraction de processus. Dans cette tâche, il s'agit de détecter et classifier des relations explicites et implicites dans le texte [28]. Par ailleurs, nous trouvons une multitude de méthodes et de techniques utilisées dans l'extraction automatique des relations. Nous avons catégorisé les différentes approches selon le type de l'apprentissage utilisé.

Approche classique. Les règles d'extraction écrites à la main (Zhou *et al.*, 2005; Poesio *et al.*, 2008; Cunningham *et al.*, 2013), comme proposées dans les applica-

---

[27]Bibliothèque nationale de médecine des États-Unis d'Amérique : `https://structuredabstracts.nlm.nih.gov`

[28]Selon l'ACE (`https://www.ldc.upenn.edu/collaborations/past-projects/ace`), les relations explicites figurent dans le texte avec une preuve explicite et donc syntaxique. Les relations implicites peuvent ne pas être définies dans le texte d'une façon syntaxique. Par exemple la relation "*est utilisé dans*" peut représenter différentes relations sémantiques comme "est un alignement de" ou "est une source de", ceci selon le contexte évoqué.

tions *Hearst* ou *NYU Proteus* (Bach et Badaskar, 2007), s'intéressent à modéliser des motifs lexico-syntaxiques comme "*such Y as X*" ou "*Y, especially X*" afin de détecter des Hyponymes (*is-a*), Hyperonymes (*not-is-a*), Holonymes (*has-part*) et des Méronymes (*is-part-of*). La limitation majeure de cette approche est dans le traitement manuel des motifs. Il s'agit de concevoir *toutes* les relations possibles ainsi que l'espace de motifs explorant ces relations.

Apprentissage supervisé. Construire des modèles statistiques, mathématiques, ou probabilistes, d'une façon supervisée revient à définir une base d'apprentissage et des attributs représentant les types de relations sémantiques à apprendre. Nous trouvons dans la littérature que des modèles comme les Séparateurs à Vastes Marges (SVM) (Culotta et Sorensen, 2004), les réseaux Bayésiens (Chieu et Ng, 2002a), ainsi que les modèles basés sur le Maximum d'Entropie (Kambhatla, 2004) (Zelenko *et al.*, 2003; Witten *et al.*, 2016, voir aussi) utilisent souvent les propriétés lexicales et syntaxiques du contexte des entités nommées qui sont concernées dans la relation, *i.e. domaine* et *co-domaine*[29]. Nous trouvons des attributs de "*léger poids*" comme les sacs-de-mots, les bigrams avant ou après les entités nommées concernées (domaine et co-domaine) aussi que des attributs exigeant un pré-traitement concentré comme la segmentation des phrases (*chunking*) afin de représenter le chemin de dépendances lexico-syntaxiques du contexte de la relation. Ce chemin représente la phrase englobant le *domaine* et le *co-domaine* de la relation. Aussi, nous trouvons dans la littérature d'autres approches utilisant une analyse syntaxique complète (*full parsing*) du texte afin de modéliser des constructions particulières dans la structure du chemin de la relation. Afin de reconnaitre les interactions Protein-Protein, Li *et al.* (2015) proposent un SVM

[29]Le co-domaine d'une relation est l'ensemble des valeurs qui sont images de la relation. Dans $f(x) = y$, x est le domaine et y est le co-domaine de la relation f.

basée sur un noyau d'arbres représentant les instances d'attributs dans des arbres de convolutions (Collins et Duffy, 2002) ayant des structures syntaxiques similaires[30]. Les attributs utilisés ici sont basés sur : les sacs-de-mots (occurrences de proteins, mots entre les proteins, *etc.*), les représentations dynamiques de mots basées sur le modèle *skip-gram* (Mikolov *et al.*, 2013) et les représentations basées sur les regroupements de vecteurs de mots (K-means, n-gram, et d'autres).

Apprentissage semi-supervisé. Si les textes sont peu annotés, on peut se servir des échantillons d'instances de relations disponibles pour construire les motifs qui les caractérisent. Ensuite, il s'agit d'agrandir l'échantillon de travail pour construire des nouvelles règles d'extraction (soit un *Bootstrapping*). Les systèmes classiques comme Snowball et DIPRE (Cunningham *et al.*, 2013) sont des exemples connus dans la littérature de la classification semi-supervisée des relations sémantiques. Plus récemment, Sun *et al.* (2011) proposent d'utiliser le regroupement statistique de Brown *et al.* (1992) qui assigne à chaque mot un *cluster* puis les combine en maximisant l'entropie moyen d'information (Quinlan, 1986). En traversant les étapes de combinaison de clusters, chaque mot est représenté par une séquence binaire représentant la hiérarchie du mot. Les attributs utilisés sont basés sur le tuple <*Before*, *Domain*, *Between*, *Range*, *After*> qui représente la relation entre le domaine (*Domain*) et le co-domaine (*Range*) dans les contextes des mots précédents (*Before*) et suivants (After) les entités de la relation. Ces attributs sont ensuite triés et filtrés par leur "importance linguistique" mesuré par le gain d'information de Quinlan (1986). Poon *et al.* (2015) proposent un modèle de reconnaissance de voies biologiques (Pathways) du Cancer dans la base de données PubMed. L'idée clé ici est d'introduire une variable latente pour tester si une relation R entre une entité nommée E_1 et une E_2 est "bonne" pour

[30]En calculant le nombre de sous-arbres communs.

chaque cooccurrence des E dans une phrase. Ensuite, l'algorithme génère d'une façon aléatoire, les exemples négatifs et calcule des attributs classiques comme : la direction, la distance et le chemin de dépendances lexico-syntaxiques entre E_1 et E_2. Cependant, les modèles construits ici sont trop sensibles à l'échantillon de départ. De plus, le changement du contexte (échantillon) dans chaque itération peut causer des ambiguités sémantiques. Ainsi, le taux de précision peut être faible. Cependant, l'un des avantages de cette approche est qu'on peut tolérer quelques attributs non discriminants et non sensibles à l'ensemble d'apprentissage.

Apprentissage non-supervisé. Quand une base d'apprentissage de textes n'est pas disponible, il s'agit de retrouver des similarités partagées dans les contextes cibles, soit, par exemple, les chemins de dépendances lexico-syntaxiques entre les paires d'entités. Si deux chemins représentent des contextes similaires qui correspondent à des phrases similaires, alors les significations tendent d'être similaires aussi, chose qui n'est pas toujours évidente dans un langage naturel. DIRT (Lin et Pantel, 2001) est un modèle d'extraction par règles d'inférence utilisant WordNet (Fellbaum, 1998) pour l'extraction des instances de relations par similarité de chemins de dépendances lexico-syntaxiques. Nous trouvons une amélioration du système DIRT dans (Yao *et al.*, 2012b) où des catégories de chemins de dépendances sont induites afin de représenter la synonymie sémantique d'une relation. Un clustering (regroupement) Hiérarchique Agglomératif (Day et Edelsbrunner, 1984) est appliqué pour calculer la similarité des vecteurs d'attributs construits. Afin de prédire les tokens déclencheurs d'événements, Nguyen et Grishman (2015) proposent d'utiliser trois représentations dynamiques (*embeddings*) dans un réseau de neurones convolutif (Tymoshenko *et al.*, 2016). En fixant le contexte d'un token dans un phrase, il s'agit de calculer les tables dynamiques de : (i) mots en utilisant un modèle déjà entrainé afin de capturer les motifs syntaxiques, (ii) positions i

(distances i entre les tokens du contexte) et (iii) types d'entités nommées de l'ACE en utilisant l'annotation classique BIO (*Begin*, *Inside*, *Outside*). Plus récemment, Alicante *et al.* (2016) proposent d'utiliser le méta-thesaurus médical UMLS et un ensemble de motifs grammaticaux afin de reconnaitre les entités nommées dans les textes. Une méthode de clustering de pairs d'attributs est ensuite utilisée afin de reconnaitre les relations. Les auteurs proposent ici d'utiliser les attributs de bi et trigrammes de mots ainsi que des attributs de "barrières" basés sur les paires de classes grammaticales (t, e) où t est le terme déclencheur (généralement un adjectif) et e, le premier terme situé à gauche de t. Même que ces approches ne demandent pas de base de connaissances préalables et des textes pré-annotés, les taux de précisions et de rappels restent faibles (entre 40% et 65%).

1.4 Fouille de motifs processuels basée sur une ontologie

La fouille de flux de travaux basée sur des motifs processuels cherche à découvrir des cooccurrences fréquentes entre ses items (événements). Les travaux de recherche existants ont porté principalement sur l'étude des motifs composés d'objets, et dans un certain nombre de cas, des catégories d'objets (*e.g.*, ontologies, taxonomies). Bien que les motifs d'objets sont trop spécifiques, et de ce fait peuvent être peu fréquents, les motifs de concepts ont divers niveaux d'abstraction et sont plus fréquents mais risquent d'être moins précis. Cependant, ces types de connaissances ont pour but d'augmenter à la fois la pertinence et l'intelligibilité des modèles découverts. Par exemple, l'analyse des associations de gènes-gènes (ou gènes proteins) et leurs annotations requiert des informations souvent non ou sous-représentées dans les algorithmes de fouille de motifs séquentiels classiques. Les structures de données complexes, comme les ontologies (*e.g.*, Gene Ontology), prennent en considération des concepts plus génériques dans la fouille. La problématique de la fouille de motifs devient de plus en plus complexe en combinant l'aspect structurel de données et l'aspect générique de l'ontologie. Ainsi, l'espace

de recherche de motifs s'élargit en augmentant la possibilité de découvrir plus de nouveaux motifs fréquents.

1.4.1 Problème de fouille de motifs

Dans l'approche Apriori (Agrawal et Srikant, 1994) de fouille de motifs, il s'agit de construire l'espace de recherche (appelé aussi *treillis de motifs*) en commençant par des ensembles qui se composent d'un seul item que l'on étend en y ajoutant d'autres items et/ou relations sur plusieurs balayages/niveaux (k). Le premier niveau sert à identifier les candidats CP_k[31] et à compter le nombre de fois qu'apparait chaque motif dans la base de données : *support.* Dans le cas de fouille de graphes (Kuramochi et Karypis, 2001), il s'agit plutôt de trouver les *structures isomorphes* : sous-graphes partageant des sommets et des arêtes de même types (Singh, 2016). Tous les candidats dont le support est plus grand qu'une valeur prédéterminée appelée min_supp (ou σ) constituent les $k-$motifs fréquents FP_k. Cet ensemble sert d'amorce pour générer l'ensemble de candidats CP_{k+1} en *étendant* les FP_k (y ajoutant des noeuds/arêtes). La fonction de calcul de support est une fonction *anti-monotone*, ainsi, si un motif p_1 est jugé fréquent et son support $supp(p_1) \geq \sigma$ alors tous ses sous-motifs candidats (extensions) ont des supports $supp(p_{1_i}) \leq supp(p_1)$.

La fouille de motifs basée sur des ontologies a été initialement utilisée dans le contexte de la fouille d'association et l'Analyse du Panier de la Ménagère (Han et Fu, 1995, 1999; Srikant et Agrawal, 1996; Plantevit *et al.*, 2006; Šebek *et al.*, 2012). Bien que ces algorithmes sont efficaces et scalables, ils ne sont pas applicables d'une façon directe au contexte de flux de travaux étant des graphes orientés acycliques (DAGs) dont les noeuds/arêtes peuvent avoir différents *labels* (étiquettes).

---

[31] Un motif candidat CP_k, dont on ne connait pas encore le support, est un motif généré dans un niveau k à partir des motifs $k-1$.

1.4.2 Fouille de graphes généralisés

Inokuchi (2004) propose une extension de l'algorithme AcGM (Apriori-based connected Graph Mining) (Inokuchi *et al.*, 2002) pour fouiller des sous-structures généralisées de graphes étiquetés. Les graphes sont représentés par des matrices d'adjacence associées avec des structures canoniques afin d'identifier les graphes isomorphes. Les motifs *sur-généralisés* sont élagués s'il existe d'autres motifs représentant la même "classe" de graphes[32]. En outre, la stratégie de parcours du treillis en largeur d'abord est elle même couteuse en terme de mémoire ce qui ne permet pas de gérer des taxonomies et des graphes de grandes tailles. De plus, les auteurs ne donnent pas de preuves concernant la complétude et la génération unique des motifs.

Cakmak et Ozsoyoglu (2008) proposent l'algorithme Taxogram qui n'utilise que la structure hiérarchique de l'ontologie. Taxogram décrit 3 étapes. (1) Transformer les données d'entrée en une base de graphes avec les concepts les plus généralisés (*root concepts*) en gardant les étiquettes (*labels*) originales des instances. (2) Utiliser l'algorithme gSpan (Yan et Han, 2002) pour fouiller les motifs généralisés en créant des représentations de motifs basées sur des indexes (*noeud, occurrences*) afin faciliter la tâche de calcul d'isomorphismes de graphes. Et (3) Spécialiser les motifs de niveau k à partir de ceux de niveau $k-1$ en remplaçant les noeuds/arêtes par leurs descendants hiérarchiques. On obtient alors des motifs de même niveaux. Cependant, puisqu'il n'y a pas de fonction d'ordre complet entre les items (noeuds et arêtes), l'algorithme peut générer des dupliquants. Ainsi, une étape de post-traitement basée sur la liste des indexes est proposée.

Dans le cadre de la fouille de motifs de relations sémantiques (*relationset*), Jiang

---

[32]L'ensemble de motifs contenant tout motif qui, en remplaçant ses noeuds/arêtes par leurs ancêtres ou descendants, on obtient le même motif.

et al. (2007) proposent l'algorithme GP-Close qui fouille les motifs fermés (Zaki et Hsiao, 2002) en (1) générant les motifs fermés locaux (dans chaque niveau) et (2) élaguant les candidats "enfants" (descendants de chaque *relationset*) redondants et leurs sous-arbres conséquents. Se focalisant seulement sur la génération de motifs fermés, les auteurs ne présentent pas les processus de généralisation de prédicats (relations) et ceux de calcul de supports.

Werth *et al.* (2009) proposent une approche de fouille de DAGs enrichie par une ontologie du domaine d'Abstractions Procédurales (Back, 1987) qui cherche les codes redondants dans un programme en extrayant des nouvelles procédures substituant les motifs de codes fréquents. Leur algorithme DAGMA propose une nouvelle représentation canonique basée sur l'ordre topologique des noeuds des graphes permettant d'accélérer la tâche de recherche de graphes isomorphes. La forme canonique consiste de créer des tuples (*index du label, index du prédécesseur*) attachés aux motifs dans chaque niveau. Afin d'étendre un graphe, un ensemble d'opérations canoniques est appliqué : (1) Insérer un noeud ancêtre (root concept), (2) Insérer un noeud/arête d'un autre niveau topologique ou (3) rester dans le même niveau et remplacer le noeud/arête par un autre. Et (4) Insérer une arête vers le dernier noeud (prédécesseur). Toutefois, puisqu'il n'ya pas de relation d'ordre entre les arêtes des graphes, l'approche ne garantit pas la génération unique des motifs. En plus, le temps de calcul des graphes isomorphes avec des noeuds non-connectés est supérieur à celui avec des noeuds connectés (plus de structures isomorphes sont retrouvées).

Adda *et al.* (2010) proposent une approche de fouille de motifs séquentiels enrichis par des relations ontologiques à partir des *logs* : historique de séquences de *clicks* sur des pages Web annotées par des métadonnées. Leur algorithme `xPMiner` décrit dans un premier lieu l'étape de transformation de séquences en des graphes en y ajoutant toutes les relations possibles (hiérarchiques et non hiérarchiques) à

partir de l'ontologie du domaine. Quatre opérateurs canoniques sont proposés afin d'ajouter/spécialiser les items (noeuds et relations) d'un motif généralisé. Néanmoins, l'algorithme ne peut être applicable directement à la fouille de flux de travaux. Ces derniers représentent des multi-ensembles d'items : un objet/concept peut être utilisé plusieurs fois dans la même séquence sur plusieurs transactions. De plus, malgré les contraintes proposées sur les opérateurs canoniques, les auteurs ne présentent pas de preuves sur la complétude et l'unicité des motifs générés.

Nguyen *et al.* (2014) proposent une approche de fouille de flux de travaux enrichie par une ontologie. Les DAGs de flux de travaux sont initialement transformés en des arbres (*parse trees*) en appliquant un ordre topologique puis lexicographique. Les arbres sont ensuite généralisés en se basant sur la distance (la longueur du chemin le plus court entre deux concepts dans la hiérarchie) pour réorganiser les subsomptions dans chaque étape de généralisation. Une fois que les arbres sont généralisés, l'approche reprend l'algorithme TreeMiner (Zaki, 2005) pour fouiller les sous-arbres fréquents. Toutefois, les auteurs ne démontrent pas la complétude de l'approche suite à cette transformation. En outre, leur méthode de subsomption de concept/relations dépend fortement de la structure de l'ontologie de leur domaine d'application[33] (Hilario *et al.*, 2011) qui représente les éléments de flux de travaux en : Tâches, Algorithmes, Opérateurs, Données et Modèles.

Afin de résumer les travaux décrits ci-avant, nous avons choisi de les restructurer selon leur domaine d'application, stratégie de parcours de l'ontologie et les différentes contributions et limitations (voir le tableau 5.8 dans l'Annexe *G*).

---

[33]Méta-apprentissage des algorithmes de fouille de données dans RapidMiner : `http://rapidminer.com`.

1.5 Systèmes de recommandation

Depuis l'émergence du domaine des systèmes de recommandation, plusieurs approches et méthodes ont été proposées (Bogers et Van Den Bosch, 2011; Aggarwal, 2016; Tarus *et al.*, 2017). Ces dernières sont généralement basées sur les types de données : (i) les interactions $Utilisateur \times Item$, comme les comportements d'achat ou les notes (ou *rating* en anglais) données sur un produit, et, (ii) les descriptions (attributs) sur les $Items$ et/ou $Utilisateurs$ comme les métadonnées ou les mots-clés pertinents. Les méthodes utilisant le premier type de données sont des méthodes, dites "*collaboratives*" (Najafabadi et Mahrin, 2016), basées sur les corrélations entre les motifs de notes à travers les utilisateurs afin de recommander des items. Les approches utilisant le deuxième type, appelées "*basées sur le contenu*" (ou *content-based* en anglais) (Aggarwal, 2016), sont plutôt centralisées sur les descriptions (attributs) d'items d'un seul utilisateur. Ces approches sont reconnues par leur pouvoir expressif des recommandations riches en "sémantique" par rapport aux approches collaboratives qui n'exploitent pas les différentes dimensions et niveaux d'abstraction des items. L'approche basée sur le contenu analyse l'historique de documents (textes) et/ou descriptions d'items d'un utilisateur afin de créer un modèle ou un profil d'intérêts et de préférences (généralement représentées sous un ensemble de motifs) adaptées aux attentes de l'utilisateur. Le processus de recommandation cherche à apparier les attributs du modèle avec les attributs de l'objet à recommander. L'ensemble des attributs et des propriétés des items sont généralement définies dans une ontologie représentant les sémantiques de concepts du domaine dans des relations et des métadonnées descriptives. Les défis majeurs dans cette approche sont liés à (i) la définition de l'ontologie, (ii) la représentation et l'annotation des items dans les textes et (iii) la définition d'une stratégie d'appariement item-profil. Nous décrivons, dans ce qui suit, différentes approches de recommandations basées des ontologies du domaine en

discutant dans chacune les solutions proposées pour chaque défis (voir aussi le tableau récapitulatif 5.9 dans l'Annexe *H*).

Magnini et Strapparava (2001) proposent *SiteIF* le premier système adoptant une approche basée sur le contenu des textes pour créer un modèle d'intérêts pour des sites Web d'information et d'actualité (*News*) multilingues. Les auteurs proposent d'utiliser MultiWordNet[34], une base de données lexicales multilingue, afin d'aligner les significations des langues, anglaise et italienne. Chaque terme analysé dans le texte est associé automatiquement à une liste de *Synsets*[35] utilisant une technique de désambiguïsation de mots polysémiques (Stevenson et Wilks, 2003). Chaque profil d'utilisateur créée est un réseau sémantique dont les noeuds représentent les synsets trouvés dans les textes. Le système reçoit en entrée la représentation du synset et le modèle d'utilisateur puis estime la pertinence du document par rapport aux co-occurrences des concepts dans différents contextes (autres documents).

Semeraro *et al.* (2009) proposent *ITR* (*Item Recommender*) une approche d'analyse sémantique représentant le "réseau de significations" de chaque document ainsi chaque item est représenté par un vecteur de synsets (Bag-of-synstes). Un classifieur Bayésien binaire est ensuite entrainé et appliqué afin d'identifier les items les plus "intéressants". L'appariement revient à calculer la probabilité d'appartenance d'un item à la classe "intéressant" selon les probabilité des synsets dans le modèle.

Vesin *et al.* (2012) proposent *Protus 2.0* utilisant deux ontologies de type OWL (*Web Ontology Language*)[36], une du domaine et une de tâches, décrivant le do-

---

[34] `http://multiwordnet.fbk.eu/english/home.php`

[35] Regroupement de mots dans des ensembles de synonymes.

[36] `https://www.w3.org/OWL`

maine d'application soit l'apprentissage des concepts du language de programmation JAVA[37]. Ici, les auteurs utilisent le moteur d'inférence SWRL (Semantic Web Rule Language)[38] et les règles Jess (Java Expert System Shell)[39] de la logique du premier ordre (Baader, 2007) afin de recommander des nouvelles ressources aux utilisateurs.

Cobos *et al.* (2013) proposent d'utiliser une approche hybride basée sur le contenu et le filtrage collaboratif. Leur système *RSPP* (*Recommendation System of Pedagogical Patterns*) aide les tuteurs à choisir des stratégies pédagogiques selon les données du contexte représentées dans une ontologie de type RDFS modélisant les : caractéristiques sur le cours à enseigner, objectifs à atteindre, profils d'étudiants, professeurs, *etc.* Ces données sont transformées, tout d'abord, en une matrice de valeurs normalisées (entre 0 et 1), ensuite, en une représentation compacte via une Décomposition en Valeurs Singulières (Golub et Reinsch, 1970). Enfin, il s'agit de calculer les voisins les plus proches à la requête de l'enseignant en utilisant la *similarité cosinus* (Aggarwal, 2016).

Pukkhem (2013) propose d'utiliser 3 ontologies OWL représentant : l'Apprenant, l'Objet d'apprentissage et les Règles de Mapping entre le style et l'objet d'apprentissage. Ces dernières sont décrites par un expert et utilisées pour extraire les caractéristiques des objets à apprendre dans les manuels de cours. Ces attributs sont ensuite validées par des experts. Une fois que le profil d'apprenant et le modèle d'objets associé sont construits, le système applique un algorithme de recommandation multi-agents (Arias *et al.*, 2012) qui apprend à calculer un score

---

[37] `http://www.oracle.com/technetwork/java`

[38] Un langage de règles combinant la logique de description de Horn et OWL : `https://www.w3.org/Submission/SWRL`

[39] Langage de règles basé sur JAVA : `http://herzberg.ca.sandia.gov`

de préférence basé sur la fréquence des caractéristiques de chaque objet/apprenant en interagissant avec l'environnement (autres objets, apprenants, *etc.*).

1.6 Objectifs de la recherche

1.6.1 Hypothèse fondatrice

Nous estimons, et ceci est l'hypothèse fondatrice de ce projet, que l'acquisition des connaissances pratiques phylogénétiques doit être assistée par la machine. Cette machine est "supposée" guider le phylogénéticien dans la résolution de ses problèmes tout en considérant le niveau de son expertise. En effet, l'extraction des ces éléments génériques (connaissances de différents niveaux d'abstraction) peut être facilitée par les spécificités des ressources que l'on dispose, soient les textes de la littérature scientifique. L'exploitation optimale de ces ressources nécessite toutefois la mise en oeuvre des techniques adéquates c'est-à-dire des techniques d'extraction automatique de l'information et de fouille de données ainsi que des techniques de recommandation appropriées.

1.6.2 Analyse des besoins

Nous présentons nos besoins de formalisation de connaissances en deux parties : les connaissances *factologiques* (déclaratives) et les connaissances *processuelles*. Le premier type de connaissances servira à bâtir une *ontologie du domaine* qui représentera les concepts théoriques de l'analyse phylogénétique. L'*ontologie processuelle* représentera donc la dimension pratique du domaine. Cette dernière aura pour but de représenter d'une façon explicite des flux de travaux abstraits dont les éléments font références aux éléments des flux de travaux concrets, soient des exécutions réelles des tâches trouvées dans la littérature. Les représentations factologiques quant à elles s'intéressent plutôt aux concepts structurels des flux de travaux : *e.g.*, les concepts qui touchent les types de données étudiés (*e.g.*, noms

d'espèces, nom des gènes) et les techniques utilisés dans l'analyse phylogénétique (*e.g.*, alignement multiple, inférence statistique).

Afin d'identifier les besoins systémiques, nous posons les questions suivantes :

1. Étant donné une ontologie et un ensemble de textes, comment peut-on supporter le processus d'extraction des instances factologiques et processuelles à partir des textes ? Concrètement, comment peut-on créer un modèle spécifique au domaine phylogénétique afin d'annoter automatiquement les objets de les relations d'un processus de résolution de problèmes dans un texte ? Et comment peut-on représenter un flux de travail concret et le reconstruire à partir des éléments annotés dans le texte ?
2. Comment représenter un motif abstrait de flux de travaux ? Comment calculer la pertinence de ces motifs dans un corpus de flux de travaux concrets ? Cette question mène à poser une autre : comment peut-on régler le problème d'appariement des flux de travaux abstraits étant des DAGs afin de calculer leurs fréquences dans une base de connaissances ?
3. Étant donnée une base de motifs abstraits de flux de travaux et une ontologie du domaine, comment peut-on recommander les objets concrets les plus *intéressants* pour un utilisateur final face à un problème donné ?

Le premier ensemble de questions vise à identifier les besoins d'extraction de flux de travaux à partir des textes. En fait, ces besoins se présentent dans (1) la reconnaissance et la classification des éléments de flux de travaux en : objets (données et tâches) et relations entre ces objets à partir de l'ontologie factuelle et (2) la reconstruction des flux de travaux à partir de ces éléments. Une fois qu'une base de connaissances $\mathcal{D}_W$ est extraite, il s'agit d'élaborer des techniques de flux de travaux abstraits. À partir d'un language de motifs abstraits Γ_W et un language de processus concret Δ_W, nous élaborons la structure d'un espace de motifs fréquents

$\mathcal{F}$ exploitant les connaissances enregistrées dans l'ontologie processuelle. Nous appliquons notre approche dans le cadre de recommandation de flux de travaux phylogénétiques. Nous élaborons ainsi une technique de recommandation basée sur le contenu essayant d'appairer le flux de travaux à recommander w_r avec la base de motifs abstraits $\mathcal{D}_W$ afin de suggérer les objets de la prochaine étape en tenant compte les contraintes dans w_r (en séquence d'objets et relations entre ces objets).

L'objectif global de ce projet est d'apporter une réponse claire à notre question principale, à savoir, dans quelle mesure l'ontologie du domaine favorise l'automatisation du processus d'acquisition de connaissances liées à la résolution de problèmes. La poursuite de cet objectif va conduire à poser un ensemble de questions plus concrètes. Des questions concernent l'élaboration des techniques (i) d'extraction de flux de travaux concrets à partir des textes (ii) de fouille de flux de travaux abstraits à partir des flux extraits et (iii) de recommandation et leurs application dans un domaine technique soit l'analyse phylogénétique.

Afin de matérialiser la réponse à ces questions, nous développons une méthodologie claire de notre approche que nous présentons les détails techniques dans les chapitres suivants.

CHAPITRE II

EXTRACTION DE FLUX DE TRAVAUX À PARTIR DES TEXTES

Le travail de ce chapitre sera prochainement soumis à la prestigieuse revue *TCBB* IEEE/ACM Transactions on Computational Biology and Bioinformatic sous le titre "Bioinformatic workflow extraction from scientific texts using word sense disambiguation". Nous présentons la version imprimée de l'article dans l'annexe *A*. Cet article présente la solution proposée afin d'extraire des flux de travaux concrets à partir des textes en se basant sur une ontologie du domaine. Les résultats de ce module constitueront l'entrée pour le prochain module, celui de la fouille de motifs de flux de travaux généralisés.

La rédaction de l'article, l'implémentation, les expérimentations et le montage des figures ont été tous réalisés par Ahmed Halioui sous la supervision des Prof. Abdoulaye Baniré Diallo et Petko Valtchev.

2.1 Résumé

Nous présentons dans ce papier une méthode d'extraction automatique de flux de travaux à partir des textes basée sur le Raisonnement à Partir de Cas (RàPC). Les systèmes actuelles de gestion de flux de travaux implémentent différentes solutions basées sur des représentations graphiques complexes des tâches et des services généralement distribués soit par exemple avec le *cloud computing* ou le *grid computation*. L'acquisition automatique de l'expérience, *i.e.* connaissances processuelles, à partir des textes décrit des représentations *sémantiquement riches* de cas structurées (annotations, raisonnement derrière le choix des outils, *etc.*). C'est dans ce contexte que nous présentons notre solution d'extraction de flux de travaux basée sur une ontologie du domaine. Notre approche est une extension des techniques classiques de TALN vers une approche d'extraction de processus enrichie par des connaissances ontologiques. Notre représentation des processus (flux de travaux) est basée sur des DAGs d'activités et les liens entre ces dernières représentent les dépendances du domaine et d'application. Une méthode de désambiguïsation de mots est aussi proposée afin de résoudre les classes sémantiques des tâches dans les textes. Nous avons appliqué notre solution à un problème bioinformatique, à savoir l'analyse phylogénétique. L'évaluation basée sur la similarité sémantique des flux de travaux dans un *Ground-Truth* nous a permis d'obtenir des résultats intéressants et encourageants dans le domaine de l'extraction de flux de travaux sémantiques. L'implémentation et les sources de données sont disponibles sur : `https://github.com/bioinfoUQAM/tgowler`.

2.2 Abstract

This paper introduces a method for automatic workflow extraction from texts using Process-Oriented Case-Based Reasoning (POCBR). While the current workflow management systems implement mostly different complicated graphical tasks based on advanced distributed solutions (*e.g.* cloud computing and grid computation), workflow knowledge acquisition from texts using case-based reasoning represents more expressive and semantic case representations. We propose in this context, an ontology-based workflow extraction framework to acquire processual knowledge from texts. Our methodology extends classic NLP techniques to extract and disambiguate tasks in texts. Using a graph-based representation of workflows and a domain ontology, our extraction process provide a context-aware approach to recognize workflow components : data and control flows. We applied our framework in a technical domain in bioinformatics : *i.e.* phylogenetic analyses. An evaluation based on workflow semantic similarities on a Ground-truth proves that our approach provides promising results in the process extraction domain. Both data and implementation of our framework are available in : `https://github.com/bioinfoUQAM/tgowler`.

2.3 Introduction

Current workflow management systems provide a step by step guidance for users through sophisticated graphical workflow modelling (Van der Aalst, 2016; Lord *et al.*, 2012; Blankenberg *et al.*, 2001). Tracking these executions can be used to adapt workflows to new situations. This can be supported by reasoning methods like Process Oriented Case-Based Reasoning (POCBR) (Minor et Montani, 2012) which is a newly merged branch from the Case Based Reasoning (CBR) field. Because case engineering is a complicated and a costly process, automatic case-based acquisition from texts has been a major focus research in the last decade (Dufour-Lussier *et al.*, 2014; Schumacher *et al.*, 2013; Müller et Bergmann, 2014). Textual CBR approaches vary widely in their scope according to their use of Natural Language Processing (NLP) methods, case representations, and are highly dependent on the intended applications. Hence, it is very difficult to reuse the proposed approaches to some problem-solving situations where a domain-specific vocabulary is used such as in Phylogenetics (Lemey Philippe, 2009) where, as a interdisciplinary domain, it combines different vocabularies from Computer Science, Statistics, Mathematics, and Engineering to analyze and interprete biological data.

Even though Lau *et al.* (2009) compared three different approaches : key-based, grammar-based and machine learning, they only consider imperative sentences which cover more than 50% of the whole data and their interpretation modules are limited to the command tuple representation (A : type of the action, textual parameter to an action, the target on which to act). Liu et Agah (2009) developed a workflow extractor trained with a few manually created positive samples. A set of extraction rules are generated and then applied to automatically crawled documents (from *e-how* Websites, *e.g.* www.wikihow.com). However, their model has only successfully extracted ~42% of these documents. Schumacher *et al.*

(2013) proposed to use a full fledged NLP solution to extract and enrich workflows from texts. Their frame-based information extraction approach to resolve *anaphoras*[1] (Mitkov, 2014) in texts uses *frame patterns*[2] to trigger phrases. A sequential pattern mining is also applied to mine anaphora-rules and then, search for the most frequent term cooccurrences to resolve an anaphoric reference. A similar approach is proposed by (Dufour-Lussier *et al.*, 2014) to resolve "universal references" using a domain-specific ontology. Given that ontology, they map the anaphoric reference to a concept. Yet, their resolution is based on heuristics (rules) highly dependent on the domain application. Addis et Borrajo (2010) proposed a Plan Acquisition Architecture to build an augmented plan in the form of predicates from extracted raw content by reducing the article into a list of two kinds of information : input-tool (predicate that describes the tool properties) and action (action description with action properties). They also use data mining and statistical algorithms to identify particular subsequences of plans to specific contexts. Instead of manually composed plans from a set of atomic actions, Tenorth *et al.* (2010) proposed to generate plans by transforming natural language task instructions from how-to sites into formal symbolic robot plans. Their evaluation results show that their planner is able to transform about 80% of the instructions. This number can be increased with a better syntax parser and more mappings to find connections between WordNet synsets (Fellbaum, 1998) and ontological concepts. However, this extension has never been proved. A recent approach to extracting situation ontology from how-to instructions is proposed by Ryu *et al.* (2013) where an automatic method based on syntactic patterns and

---

[1]The problem of resolving what a pronoun or a noun phrase refers to.

[2]A frame-based structure defines the meaning of a term in a sentence using *slots*. Each frame slot is filled using linguistic patterns (heuristic rules) mapped to syntactic roles (subject, direct object, and indirect object) to trigger phrases in texts.

a CRF (Conditional Random Fields) (Settles, 2004) probabilistic model are used in order to classify actions and objects such as time and place. Although their pattern-based approach achieves a high level of accuracy (>73%), its coverage is limited to rules constructed semi-automatically from small fraction of sentences (only imperative sentences) in *how-to* texts. Song *et al.* (2011) proposed a novel attempt to extract procedural knowledge from the PubMed abstracts[3] using the TAM model — a triple combination of (Target, Method, Action) and designed a text mining method with deep language processing analysis. This analysis contains a (1) preprocessing step (syntactic and ontological tagging), (2) purpose/solution sentence classification and (3) relationship between two processes is assigned to a control flow (sequential, parallel, causal, *etc.*). Poon *et al.* (2015) proposed a distant supervision approach[4] for *Cancer Pathway Extraction* from texts. The main idea is to test the significance (using a latent variable), for each sentence evoking two entities E_1 (theme) and E_2 (cause), of the relation R (between the two entities). Distant supervision is provided by enforcing during training that for each relational triple (E_1, R, E_2), at least one latent assignment is true if the database contains the relation, and none otherwise. Each instance is represented by a linear model with features over the sentence and entities. The feature weights are learned using online learning with Perceptron (Rosenblatt, 1958). Yet, their semi-supervised approach is sensitive to the starting sample. Changing the context at each iteration, while creating automatically positive examples, may generate different conflicting relations since the latent assignment is a simple comparison of R in the training set. Comparing contexts using the triple (E_1, R, $E2$) may

---

[3]`https://www.ncbi.nlm.nih.gov/pubmed`

[4]In distant supervision, we make use of an already existing database, such as KEGG (`http://www.genome.jp/kegg/pathway`), to collect examples for the relation that we want to extract. These examples are used to automatically generate the training data.

work only for binary relations ; lifting this limitation to handle n-ary and nested relations is an important future direction.

While those techniques will continue to be essential, new challenges also emerged as we are understanding more about the nature of human experience and its role in making decisions (Spjuth *et al.*, 2015). The expertise grounded in scientific texts is to be mined in order to acquire new experience acquisition. Yet, some complex semantic ambiguities are need to be deal with. To the best of our knowledge only anaphora resolutions have been studied in procedural texts. However, semantic ambiguities arise more difficulties : *e.g.*, word sense ambiguities. Word Sense Disambiguation (WSD) (Stevenson et Wilks, 2003) is the process to identify the sense or senses of a polysemic term. Although important, WSD as an 'intermediate' task, is unlikely that anyone other than linguistics would be interested in its results. However, we found that polysemic words are also common even in very technical domains, such as phylogenetic analyses, where specific software and programs could be also used for different purposes. Generally, programs may contain multiple packages and processes and offer multiple supports. Thus, even specific programs may be used in different contexts. Without a context-aware approach it is very difficult to reconstruct a phylogenetic workflow from texts with a simple term-matching process. In addition, identifying activities (tasks) and organizing them in a control-flow seem to be a complicated task, especially when it can be applied in different positions in a workflow. We tried to not limit ourselves to highly structured procedural texts but to a domain specific problem to extract domain-relevant meanings from it. In this paper, we propose an ontology-based approach to extract semantically rich bioinformatics workflows from texts using a WSD technique.

Section 2.4 presents the choices made in designing our ontology-based extraction process. Two formalisms are presented : workflow representation and ontologies.

Additionally, a running example is introduced to explain our representational choices for phylogenetic analyses. Section 2.5 is the main core of the paper describing in details the process of the workflow extraction from texts. A text is analyzed with a pipes-and-filters NLP chain and a dedicated WSD for workflow tagging. Section 2.6 presents experiments and results of the proposed processes by comparing automatically obtained workflows with an expert-curated dataset. Finally, section 2.7 discusses results and future work.

2.4 Formalisms for enriched workflow representations

2.4.1 Workflow representation

Traditionally, workflows are "*the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*" (Van der Aalst, 2016). In general, workflows consist of a set of *tasks* (activities) combined with a *control-flow* such as sequences, parallel (conjunction) or alternative branches (disjunction), as well as repeated executions (loops). We are interested, here, in workflows as sequences and concurrent runs of program tasks. These latter consume *data items i.e.* input sources and output products. The *data-flow* represents interactions (relationships) between data items and tasks. In addition, workflow items are annotated using semantic descriptors (metadata) describing constraints (*e.g.*,input/output) and data provenance (*e.g.*, database, species).

Different types of workflow representations are proposed in the literature of Process Oriented CBR (Minor et Montani, 2012). Petri Networks (Peterson, 1981) are the most popular design representing the dynamic aspect of processes. However, the representational bias (as in other languages) of such models should be considered while extracting workflows, *e.g.* multiple instantiations of a task or sub-task. Enriched workflows with additional data describing task parameters and

metadata are also to be considered while representing workflows in order to make the workflow more expressive and hence, easily to be reused in similar cases.

In this study, we illustrate our approach in the domain of phylogenetics Lemey Philippe (2009). Inferred from nucleic acid (DNA or RNA) or protein sequences, phylogenetic relationships represent the evolutionary history of a group of species. During phylogenetic steps, several bioinformatic programs (software) are used. Here, the tasks represent phylogenetic *programs* and their combination of *parameters* and data items referring to input/output *data* are enriched with *metadata*.

An example of a phylogenetic analysis of *Hemagglutinin* (HA) protein sequences, extracted from the article text of Liang *et al.* (2010), is illustrated in Fig. 2.1. This workflow represents the phylogenetic solution to study the evolutionary history of HA proteins according to the following steps : (1) multiple sequence alignment, (2) model representation, (3) tree inference and (4) result visualization. Each step is composed of partially ordered program tasks dependently linked with each other. These dependencies represent control and data flows enhanced with semantic annotations. For instance, BionJ (Gascuel, 1997), a neighbour joining program, gets *Hemagglutinin* sequences collected from the Genbank database[5] as input and produces phylogenetic trees. Duplicates in workflows represent different instantiations of one task. For instance, the first mention of *MEGA* (Kumar *et al.*, 2016) (at the $3r^{rd}$ step) is used with different parameters than the second instance at the same step. *MEGA* is used with both, a *Maximum Likelihood* (*ML*) method and a *Maximum Parsimony* (*MP*) (Tamura *et al.*, 2011).

Definition 2.4.1 (Workflow data language). *Given a universe of task items O^T (tasks and parameters) and data items O^D (data types and metadata), i.e. $\omega^O = O^T \cup O^D$, a workflow language Δ_W defines workflow records of partially*

[5] `https://www.ncbi.nlm.nih.gov/genbank/`

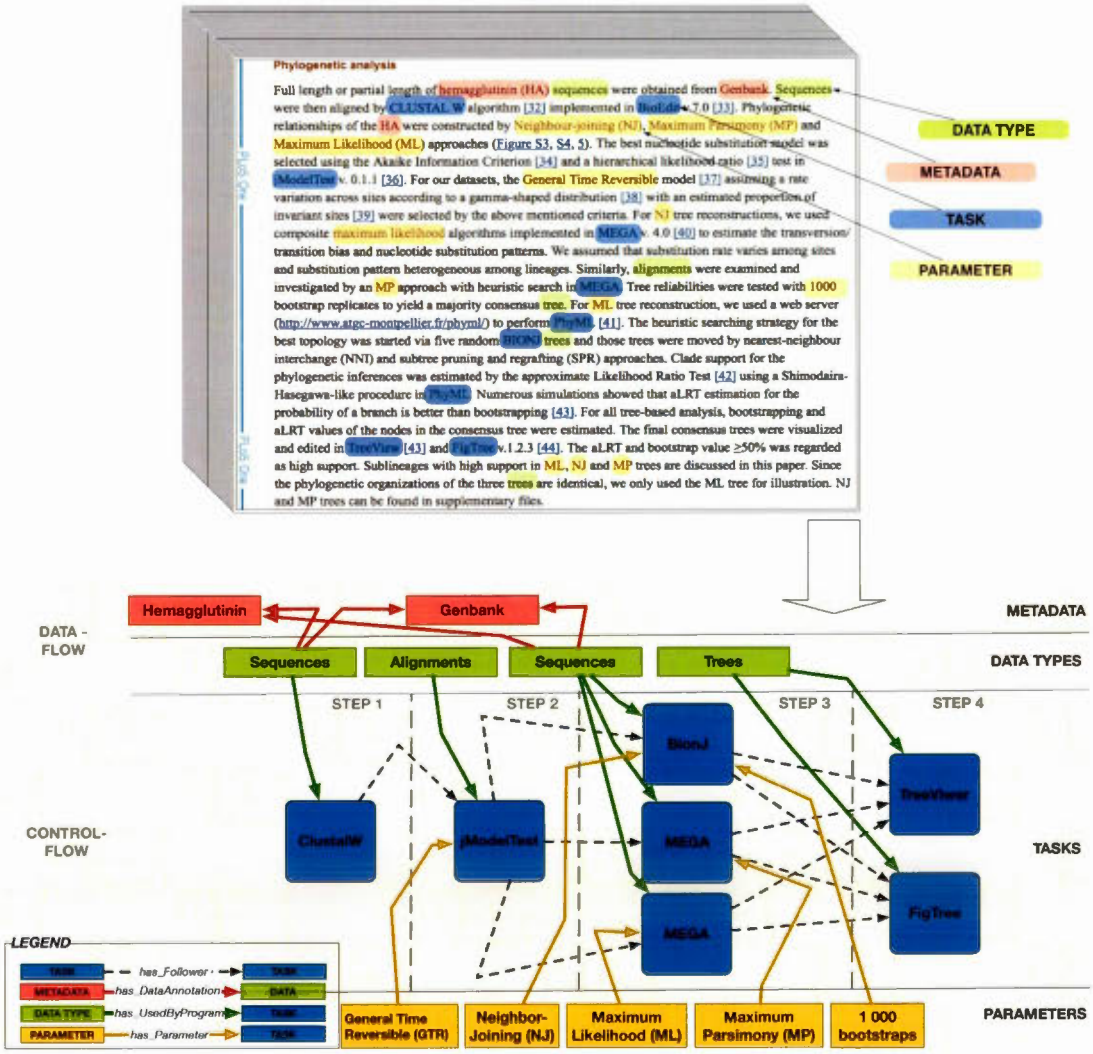


FIGURE 2.1: Example of a workflow excerpted from the article of Liang *et al.* (2010). We represent the workflow in four different layers. The first layers describe data and metadata annotations. The tasks layer represents the control flow and the last layer describes parameters. Relations between the different objects at each layer describe the control flow (links between tasks), the data flows (links between data and tasks) and other semantic flows (between data and parameters and/or metadata).

ordered sequences composed of inter-related object labels $o \in \omega^O$. *A workflow* $w \in \Delta_W$ *is a direct acyclic graph* $w = (\zeta, \Theta)$ *where* $\zeta = \langle \{o_1, o_2, ..., o_{i_1}\}, \{o_{i_1+1}, \ldots, o_{i_2}\}, \ldots, \{o_{i_{(m-1)+1}}, \ldots, o_{i_m}\}\rangle$ *is a sequence of object transactions (steps). In turn,* $\Theta = \{\theta_1, \theta_2, \theta_k, \ldots, \theta_p\}$ *is the set of triples where* $\theta_k = (i_1, l, i_2)$ *represents the link of type* $l \in \omega^{\mathcal{L}}$ *between the object at position* i_1 *in* $w.\zeta$, *i.e.* $w.\zeta[i_1]$ *and the object at the position* i_2, *i.e.* $w.\zeta[i_2]$.

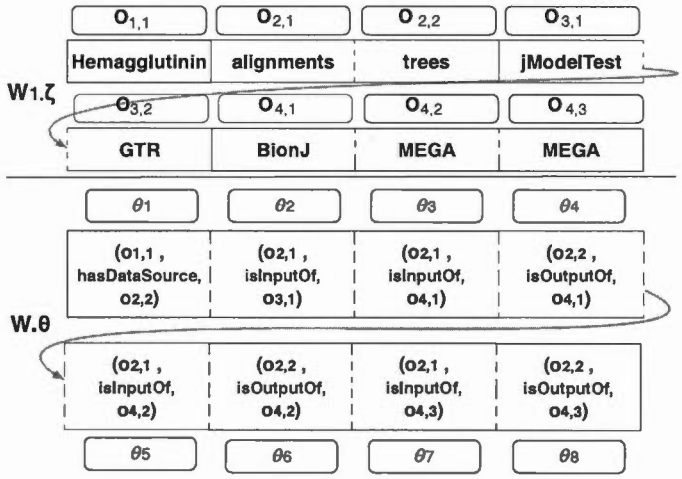


FIGURE 2.2: Example of a workflow data representation. A workflow w is a set of partially ordered objects $w.\zeta$ and a set of link triples $w.\Theta$. Here, items belonging to the same set are separated with dotted lines.

Fig. 2.2 shows a sub-workflow w_1 representation from the previous example in the figure 2.1. Phylogenetic programs and parameters and their input/output data and metadata are represented in a partially ordered sequence of object transactions (steps) and a set of links between them. For example, *Hemagglutinin alignments* ($o_{1,2}$) are the input (see θ_3) in the neighbour joining program *BionJ* ($o_{1,4}$) generating *phylogenetic trees* (see θ_4). The order between phylogenetic programs represents the partial execution of phylogenetic steps. The "parallel" execution of programs is represented by the transactions in $w.\zeta$, *e.g.* $\{BionJ, MEGA, MEGA\}$

are programs in the same step.

2.4.2 Ontology representation

The universe of objects $\omega^O = O$, links $\omega^{\mathcal{L}} = \mathcal{L}$ and their corresponding labels (references) is defined in the domain ontology Ω (see a portion from the PHAGE ontology[6] in *Fig.* 2.3).

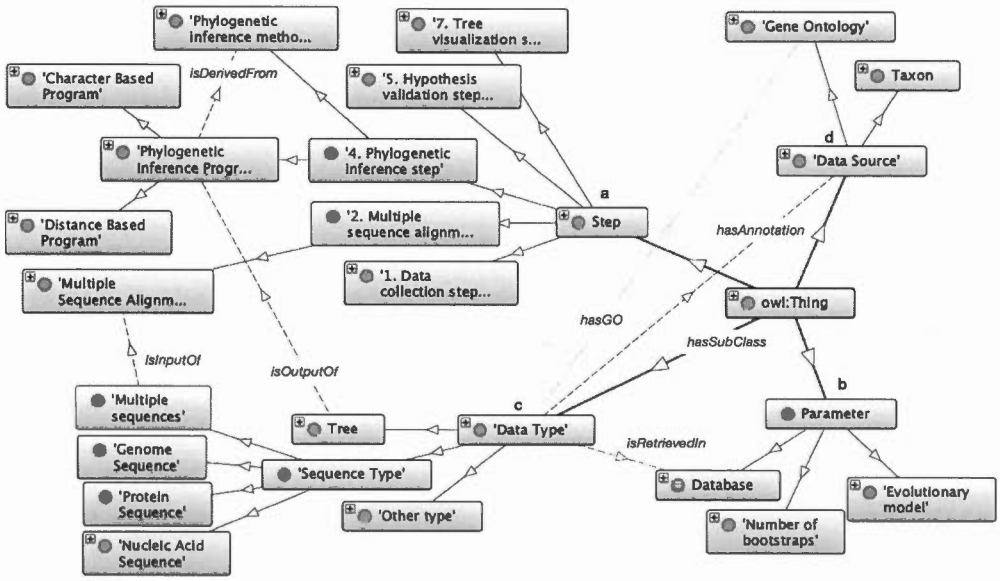


FIGURE 2.3: A sample from the phylogenetic domain ontology PHAGE. Grey rectangles illustrate concepts, where the top ones (a) and (b) correspond to task items (steps and program parameters, respectively), (c) and (d) concepts correspond to data items (data types and metadata, respectively). Coloured dotted arcs represent either annotation or input/output properties.

Definition 2.4.2 (Universe of objects). *Let* $\Omega = \langle O, C, \mathcal{L}, \mathcal{R} \leq_\Omega, \rho \rangle$ *be an ontology where* O *is the set of all objects, i.e. individual tasks and data types,*

[6] http://bioportal.bioontology.org/ontologies/PHAGE

C the set of all concepts, $\mathcal{L}$ the set of all links between objects, and $\mathcal{R}$ the set of all properties (relations) between concepts. $H_C = \langle C, \leq_\Omega \rangle$ and $H_\mathcal{R} = \langle \mathcal{R}, \leq_\Omega \rangle$ denote the concept and relation hierarchies of the ontology where $\leq_\Omega$ denotes the rdfs :subClassOf hierarchical relationship. Objects $o \in O$ and links $l \in \mathcal{L}$ are instances of concepts $c \in C$ and relations $r \in \mathcal{R}$, respectively. The ternary relation $\rho \subseteq C \times \mathcal{R} \times C$ is made of triples $c_1 \times r \times c_2$ expressing the incidence between a relation r and its domain c_1 and range c_2 concepts.

2.5 Extraction process

Our framework is based on a pipes-and-filters architecture (Zhu, 2005) using the NLP software GATE (General Architecture for Text Engineering) developed by Cunningham *et al.* (2013). A filter contains an element performing a data transformation on the data stream. The pipes connect the data stream from the output filter to the input one. Filters are independent so one could add or delete them without affecting the execution of others.

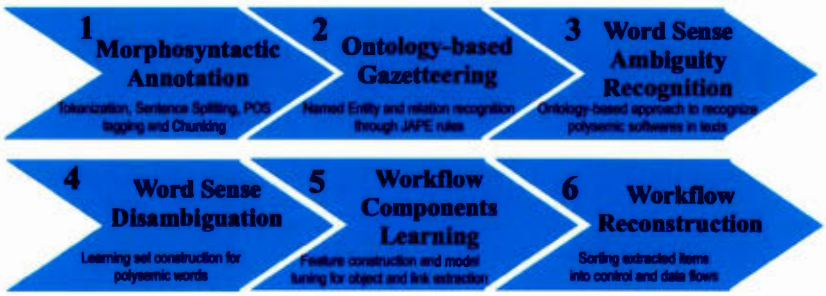


FIGURE 2.4: Pipeline of the workflow extraction from texts.

The following sections describe the different sub-tasks of our workflow extraction solution. Following the standard NLP pipeline, the very first steps are going from the morphological analysis of the text to the identification of syntactic classes of terms and sentences. Section 2.5.1 quickly goes through those steps. We present some commonly used techniques and tools and we discuss how we adapt them to

our issues. Taking the output of those steps, section 2.5.2 describes our solution to recognize domain terms based on the concepts and relations from a domain ontology Ω. Section 2.5.3 proposes a way to recognize word sense ambiguities using the hierarchy of concepts in Ω. Our word sense disambiguation (WSD) technique is supported by a supervised machine learning approach. Section 2.5.4 describes the way we create a training set to build a WSD model and a link extraction one. Sections 2.5.5 presents the features and the classifier that we use to learn WSD and relationships in texts. Section 2.5.6 details the process to reconstruct data and control flows from the extracted and disambiguated terms and finally section 2.5.7 presents the similarity measure to assess automatically extracted workflows based on expert-curated ones.

2.5.1 Morphosyntactic annotation

The initial step segments the text into lexical items called *tokens* equivalent to "words". In general, tokenization operates tokens with white spaces between them as far as we treat English texts. However, due to the specificity of the bioinformatics vocabulary, we use a specialized biomedical tokenizer which relies on a dedicated token lattice design pattern and an adapted Viterbi algorithm (Viterbi, 1967) for bioinformatics tokens (Barrett et Weber-Jahnke, 2011). For instance, "*don't*" is treated as two tokens, while the gene "*4'OMT2*" is really one. In addition, the text is segmented into sentences using punctuation-based patterns (Cunningham *et al.*, 2002b). The extracted tokens and sentences are further tagged with their appropriate grammatical and syntactic categories. At first, a Part-Of-Speech (POS) tagging is executed in order to identify grammatical classes of terms. We use the MedPost POS tagger (Smith *et al.*, 2004) which is based on a HMM (Hidden Markov Model) model trained over 1 000 XHTML PubMed texts. Their tagger achieved an accuracy of 97.43%. Next, we use a lightweight syntactic parser, the OpenNLP chunker (Buyko *et al.*, 2006), to identify active and passive verb phrases

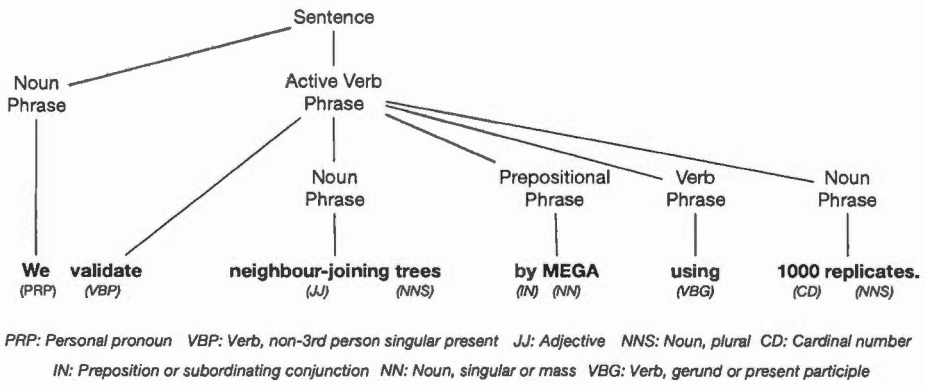


FIGURE 2.5: An example of a chunking tree.

in sentences. The depth of the "chunk tree" is strictly limited to 3 (root, phrases and POS tags). Chunker complexities are equivalent to a finite-state automaton and provide a very simple grammar which is sufficient to continue with the extraction process. We present in Fig. 2.5 an example of the generated chunking tree.

2.5.2 Semantic annotation

This step is about recognizing workflow items in texts : *i.e.* tasks, parameters, data, metadata and links between them. For this purpose, an ontology-based gazetteer (terminology) is used to map instances with text tokens. This step is important to any workflow extraction process. Yet, simple word matchings between ontology instances and tokens would not be sufficient to recognize workflow elements. Different syntactic and semantic ambiguities are present in any natural language (Nadeau et Sekine, 2007). For this purpose, we use (1) specific JAPE[7]

[7]Java Annotation Patterns Engine (JAPE). JAPE language is a finite state transducer on the GATE platform (Cunningham *et al.*, 2013) that operates over annotations based on regular expressions. Thus, it is useful for pattern-matching, semantic extraction, and many other operations over syntactic trees such as those produced by natural language parsers.

rules to extract and filter workflow concept and relation instances from texts and (2) a WSD model to clarify other semantic ambiguities : *i.e.* multi-class classification (see next sub-sections).

In order to filter concept bad matchings from the ontology, we use specific-domain stop-words, original XHTML markup annotations (from the texts) and regular expression (regex) patterns written in JAPE language. For example, when gene or protein names are nouns, verbs or adjectives, their probability to be exact matchings in a text are never 1 (Kim *et al.*, 2003). For instance, the gene term '*CAT*' is very similar to DNA codon sequences, *e.g.* in 'CAT ACT GTA ...' or even to other terms such 'caterpillar' or 'cats' . In addition, some bioinformatics software names have similar issues. For example, the word '*align*' in the context of phylogenetics presents a lexical ambiguity too. Commonly, '*align*' is a frequent verb in phylogenetics texts but it is also used as a program name, that is why the probability of this term is never 1. However, the probability of 'align' to be the name of a software written in lowercase is really near to zero (authors tend to use different orthography such as uppercase and italic to distinguish program names in texts). Thus, we build dedicated JAPE rules.

Rule 1 is an example of a pattern-based rule to eliminate program annotations where terms, as verbs, nouns or adjectives, are written in lower case. *ClassC.inst* is the instance ID in the ontology[8], *Token.category* is the POS tag of a token and *Token.orth* is its orthography. Hence, for all (most) specific program categories [9]

---

[8]Since instances might have "synonyms" (*e.g.* 'MCMC' and 'Markov Chain Monte Carlo') or might refer to software versions (*e.g.* 'MEGA 5.1'), we choose to identify each instance with unique identifiers having the prefix as the class name in the domain ontology (*e.g.* ProgramID_9482394).

[9]A most specific concept/relation is the leaf category (which has no child category) in the hierarchies H_C and $H_{\mathcal{R}}$.

in Ω, we build one rule for the most sensitive POS categories grounding lexical ambiguities : *i.e.* verbs (VB), nouns (NN) and adjectives (JJ) (Kim *et al.*, 2003).

Rule R_1 :
$(\{ClassC.inst =\sim$ "^$ProgramID_$",
$Token.category ==$ ("VB" || "NN" || "JJ"),
$Token.orth ==$ "*lowercase*"$\}) : class$
$-->$
$inputAS.remove(class)$;

Since phylogenetics sections describe processual events of how to complete specific tasks, sentences represent (a) a causal chain, (b) active, and (c) associated with task steps (Diehl et Mills, 2002). However, this rule is not always applied since, these sections use both, narrative and procedural texts. Thereof, for the link recognition step, we adopt the following hypothesis : "*a relation between two concepts (domain and range) may exist in the same sentence evoking the instances of those concepts. The link is defined in the verb of this sentence*". Since, both active and passive voices are used to describe phylogenetics events, we deal with both types of relations; actions between a domain and a range concept in both directions. For this purpose, we propose a frame-based (Gordon, 1998) approach using the ontology triple structure $\rho = (c_1, r, c_2)$ to map and match ontological relations $r \in R$ with links $l \in L$ in texts : $\theta = (o_1, l, o_2)$. Relation instances (links) are recognized based on all relation triples that exist in Ω compatible with its elements (domain, range and link type). The interpreter loads all the triples from the ontology and maps tagged concepts as domains and ranges in texts for each link. For instance, for the triple ("*BootstrapProgram*", "*hasParameter*", "*NbBootstraps*") $\in \Omega$, we search for all its domain and range terms (instances) from the same sentence in texts. For example, a sentence candidate, in this case, would be :

"We validate neighbour-joining trees by MEGA ***(domain)*** *using* ***(link)*** *1000 replicates* ***(range)***".

In addition, if a match is found, a set of syntactic patterns is applied for each link to filter bad matchings. For example, if the phrase voice is active, then, the verb evoking the relation domain and range might be between these terms taking into account the order of domain and range terms in the sentence (*i.e.* domain occurring before range). The rule R_2 annotates the link context in texts where *dStart*, *dEnd* and *rStart* correspond to 'S'tart and 'E'nd positions of 'd'omain/'r'ange terms in the triggered sentence.

Rule R_2 :

$(\{Active\ Verb\ Phrase\})\ contains\ (\{Domain\}\ \{Token\}+\ \{Range\})) : p$

$-->$

$\{for(Annotation\ d : domain)$

$for(Annotation\ r : range)$

$if(dStart < rStart\ \&\&\ dEnd < rStart\ \&\&\ p\ \text{is}\ active)$

$outputAS.add(dStart, rEnd, "ClassR", p.verb)$

$\}$

2.5.3 Word sense ambiguity recognition

The previous process of semantic annotation faces known hurdles rooted in the word sense ambiguities : *i.e. polysemy* (Stevenson et Wilks, 2003). The problem is that words often have more than one meaning (*e.g.* Metonymy[10]) and sometimes

[10]Metonymy is a "figure of speech" in which a thing or a concept is referred to the name of something closely associated with that thing or concept. For instance, "human" may refer to the *Homo Sapiens* species or to a "an expert" referring to a manual processing.

completely different meanings (*e.g.* Homonyms[11]). The meaning of a word in a particular usage can only be determined by examining its context. This is, in general, a trivial task in NLP. For example, in the following two sentences, the software MEGA (Kumar *et al.*, 2016), a multi-purposes package of programs, is used in different contexts.

- *We assess the neighbor-joining method by MEGA using 1000 replicates.*
- *Tree data was visualized using MEGA.*

In the first sentence, *MEGA* is used to validate the phylogenetic hypothesis which corresponds to the '*HypothesisValidationProgram*' concept. However, in the second sentence, the program *MEGA* is used in the visualization step which corresponds to the '*TreeVisualizationProgram*' concept. A software containing different packages (algorithms) is usually used for different purposes and contexts. Multi-purpose packages programs such as *MEGA* and *PHYLIP* (Plotree et Plotgram, 1989) refer to polysemic concepts in the domain ontology. This software-related polysemy is usually represented with the *multiple inheritance*[12] feature in the domain ontology. For this purpose A new concept '*GeneralPurposePackages*' is added to the ontology Ω. This is done automatically while interpreting the hierarchy of concepts H_C to search for individuals (instances) belonging to multiple task concepts.

For each parent concept $\hat{c}$ of a polysemic one c (where $c \prec_{can} \hat{c}$ is a *subClassOf* RDF relation) belonging to a multiple inheritance in the hierarchy H_C, we add the equivalence relationship between c and the ambiguous concept '*Multiple Purpose*

[11]Homonyms are words which have the same spelling and pronounciation, but have different meanings (*e.g.* date, crane)

[12]Multiple inheritance is an object-oriented feature in which a class (concept) or an object can inherit characteristics and features from more than one parent class.

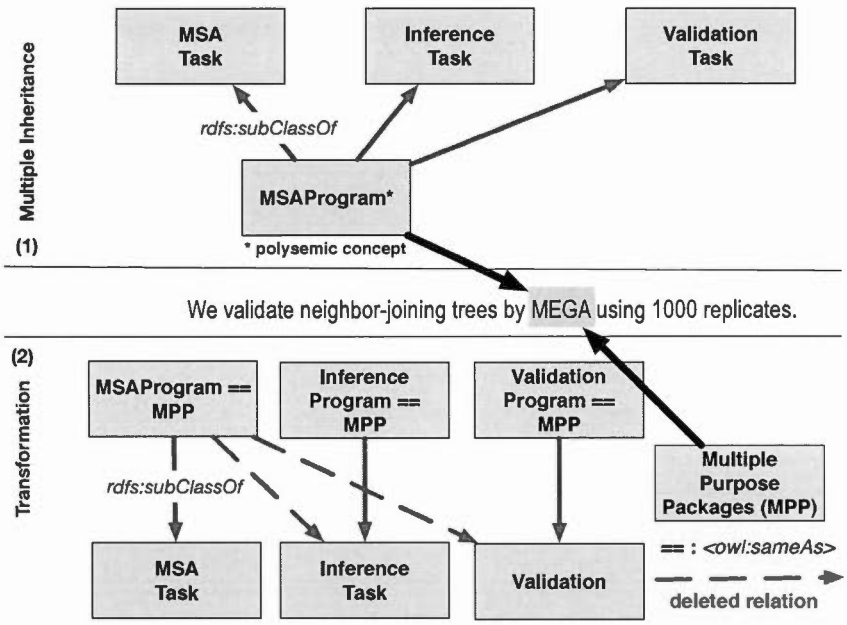


FIGURE 2.6: Ontology transformation of polysemic concepts. An example of multiple inheritance grounding polysemic concepts in Ω : (1) the ontology before (having multiple inheritance), (2) the ontology after keeping only one relation (the most left one) and adding a new concept equivalent to ambiguous tasks. Here, MSA refer to 'Multiple Sequence Alignment' and MPP to 'Multiple Purpose Packages'.

Packages', *i.e.* : $<: c > < owl : sameAs > <: MultiplePurposePackages >$ (see example in *Fig.* 2.6). For instance, *MEGA* could be used in three different steps. Hence, in order to identify polysemic words in the ontology, we create a new latent concept equivalent to all ambiguous concepts while eliminating multiple inheritance in the ontology.

2.5.4 Word Sense Disambiguation (WSD)

Next, after discovering polysemic concepts in the ontology, we tag their ambiguous instances in texts. We propose, here, a supervised machine learning approach to

WSD. The problem of WSD is defined as follows : "*given a set of pre-tagged polysemic words, WSD is to classify each word with its appropriate task concept (sense) with high accuracy and efficiency*". Pre-tagged words are usually created manually, but, here, we propose to construct the WSD ground truth in a semi automatic way using the domain ontology and syntactic tags from sentences.

$Rule\ R_3$:
$Priority$: 90
$((\{Active\ Verb\ Phrase\})\ contains\ (\{GeneralPurposePackages\}) : c) : context$
$-->$
$\{if(context\ is\ active\ \&\&$
$(context.verb.root\ in\ VerbSet : \{"valid","test",\ldots\}\ ||$
$context.object.root\ in\ ParamSet : \{"bootstrap","replicate",\ldots\}\ ))$
$c.putFeature("resolution","ValidationProgram");$
$\}$

Rule R_3 shows an example of JAPE rules to generate the WSD learning set. Here, both of the semantic tags (see section 2.5.2) and the syntactic tree (see section 2.5.1) are used to clarify the polysemy of the concept c. The *Verbset* are stem verbs extracted automatically from the WordNet synonyms (Miller, 1995) of the verb in the parent concept $\hat{c}$ (*e.g. valid*) in H_C (*i.e.* from the level immediately above). The *ParamSet* contains also WordNet synonyms of the corresponding program *Parameter* terms (*e.g.* number of bootstraps in validation programs). These latter are used (if they exist) as hints to classify the polysemic program to its right class. Hence, the pattern R_3 verifies if the verb, in the active voice, is about testing or validating something, thereof, the '*GeneralPurposePackages*' concept is disambiguated to the '*ValidationProgram*' one. Moreover, WSD rules have priority probabilities to fire them by order. Here, for each polysemic concept created in

the previous step, we generate one WSD rule. An additional rule is added by the expert to represent the default concept if no heuristic is valid (default rule).

2.5.5 Learning module

In order to create a model for automatic WSD, a set of features is generated representing the dataset of already tagged words in texts. For instance, in order to classify the term 'MEGA' (*ClassC*) from the example of Fig. 2.7, we generate the set of features (*featuresC*) reflecting its context (see alos table 2.1).

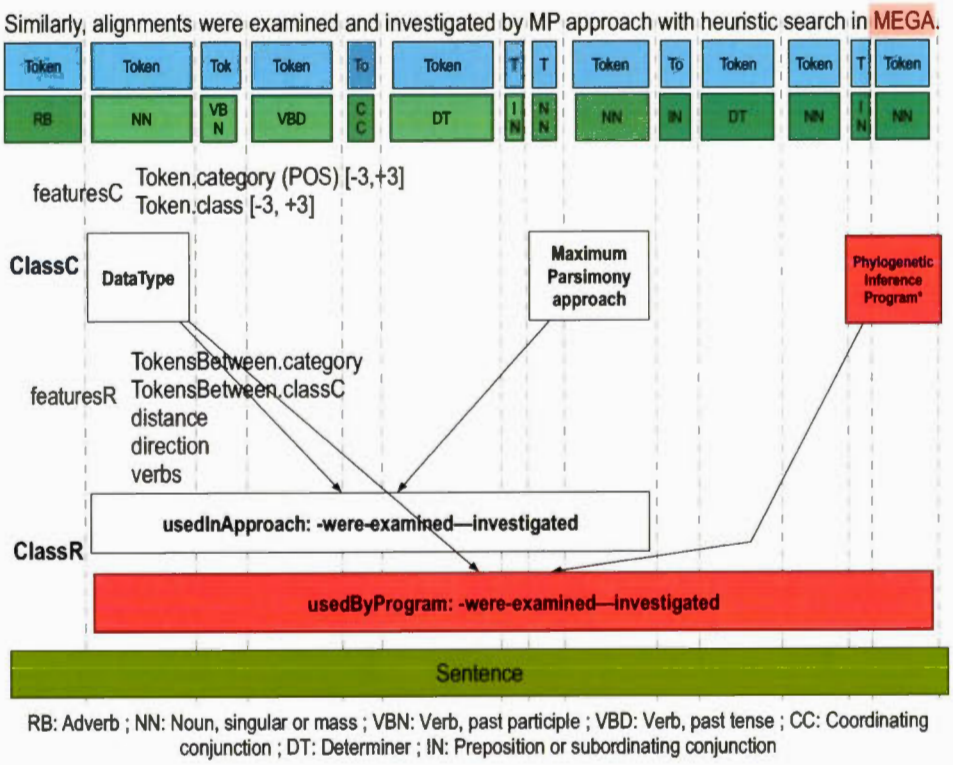


FIGURE 2.7: An example of learning features for WSD concepts and relation extraction from texts.

Here, the context covers a window made of : the Win_i concepts (concepts surrounding the term to classify) (*Token.class*) and POS tags (*Token.category*) preceding

Table 2.1: Feature descriptions

	Feature	Description
1	Token.category $[win_i+, win_i-]$ (Context POS tags)	The win_i, (number of) grammatical tags (POS) surrounding the ambiguous word in the sentence.
2	Token. class $[win_i+, win_i-]$ (Context concepts)	The win_i (number of) domain concepts surrounding the ambiguous word in the sentence.
3	TokensBetween.category (Surrounding categories)	The ordered list of POS tags between the domain and range terms.
4	TokensBetween.category (Surrounding concepts)	The ordered list of concepts between the domain and range terms.
5	Distance	The number of tokens between the domain and range terms.
6	Direction	0 if the two annotations overlap, -1 if range follows domain in the sentence, and -1 if the range precedes domain.
7	Verbs	The ordered list of verbs in the sentence evoking the link triple (domain, relation, range).

and the Win_i ones following the concerned word in the sentence. The window of concepts represents surrounded concepts in the sentence (*e.g.*, in a window of 3, '*MEGA*' is surrounded with a '*Maximum Parsimony approach*' and a '*DataType*' in Fig. 2.7). The window of concepts and POS tags (Win_i) is fixed in the learning step. In this example, $win_i = 3$, hence, *Token.class* = *{DataType, MaximumParsimonyApproach}* and *Token.category* = *{Determiner, Noun, Preposition}*.

In addition, in order to classify a link between the ambiguous word 'MEGA' (domain) and the data type 'alignments' (range), we define its context as the set of tokens between the domain and the range. Here, features (*featuresR*) represent the context token POS tags (*TokensBetween.category*), distance (in number of tokens between domain and range), direction (domain-to-range) and verbs between the concerned terms (see table 2.1).

Since proposed features are context-based, the model representing ambiguous

concept and relation terms in texts should take into account the type of these (features) representations. Hence, we chose to use the PAUM model (Perceptron Algorithm with Uneven Margins) (Li *et al.*, 2005) to learn WSD and relation classifications from on the extracted feature instances. PAUM is a hybrid SVM (Support Vector Machine) (Wang *et al.*, 2005) and Perceptron (Tsoumakas *et al.*, 2009) classifier that was designed especially for imbalanced data and has successfully been applied to various semantic annotation problems (Li et Bontcheva, 2007; Dědek, 2010). This algorithm differs from other classifiers, such as CRF (Settles, 2004), since it treats positive and negative examples differently by introducing two margin parameters τ_+ and τ_- into the updating rules for the positive and negative examples, respectively. Here, τ_+ and τ_- simulate the weights on context examples, hence, we could increase/decrease the distance between positive/negative contexts (see features 3-7) of an ambiguous concept and its surrounded ones. This reflects the intuition that "*the nearer a neighbouring concept is close to an ambiguous concept, the more important it is for classifying the given token*". It is about the same intuition for the relation classification. We represent the distance between positive/negative domain and range concept examples of one relation. PAUM learns how to update its learning rate and margin parameters to maximize this distance (Li *et al.*, 2005). Since we are not dealing with a huge number of features, we use the traditional Sequential Forward Selection (Doak, 1992) while adding f features at a time to select. The selected best set of features are then, used in the construction of a model for an automatic annotation of ambiguous words in texts and relations between them.

After tuning and training our models, we applied them to the test set (test corpus) which constitutes 1/3 of the available corpus (we note the other 2/3 are used for training). Extracted terms and links are then evaluated using different evaluation measures, *i.e.* Precision, Recall and F-Measure (Yang, 1999). Precision is the

number of *correctly* identified items (objects/links) as the percentage of identified items. In other words, it measures how many items that our system identified were correctly tagged, regardless of whether it also failed to retrieve correct items. The higher the precision, the better the system is at ensuring that what was identified is correct. Recall measures how many items should have been identified from the actual (real) set of correct annotations. We note that we consider annotations to be *partially* correct if the entity/relation type is correct and the spans (positions of characters in the text) are overlapping but not identical. Thus, partially correct responses are allocated a half weight.

$$Precision = \frac{Correct + 0.5 \times Partial}{Correct + Spurious + Partial} \tag{2.1}$$

$$Recall = \frac{Correct + 0.5 \times Partial}{Correct + Missing + Partial} \tag{2.2}$$

where :

- *Correct* : Number of correct annotations ;
- *Partial* : Number of partially correct annotations (if they overlap in terms) ;
- *Spurious* : Number of spurious annotations (in the test set) ;
- *Missing* : Number of missing annotations (in the training set).

The F-measure (FM) is often used in conjunction with Precision and Recall, as the harmonic average of the two.

$$FM = \frac{2 \times (Precision + Recall)}{Precision + Recall} \tag{2.3}$$

2.5.6 Workflow reconstruction

From recognized links between concept instances in texts, we build our concrete workflow knowledge database where items are the named entities ordered by the top-level pattern in the domain ontology. Concept and relation instances serve to identify respectively the data-flow and the control-flow. For example, from the tuples : ρ_1 = (Alignments, "isOutputOf", ClustalW), ρ_2 = (Alignments, "isInputOf", MEGA), we construct the control-flow : $w.\zeta = \langle$ {Alignments}, {ClustalW}, {MEGA} $\rangle$ and the data-flow : $w.\theta = (\rho_1, \rho_2, \rho_3)$ where ρ_3 = (Alignments, "genInputTo", ClustalW). The partial order in the sequence $w.\zeta$ is carried out with the 'generate input to' (*genInputTo*) relation which is the result of the transitive relation between input and output data from the ontology Ω.

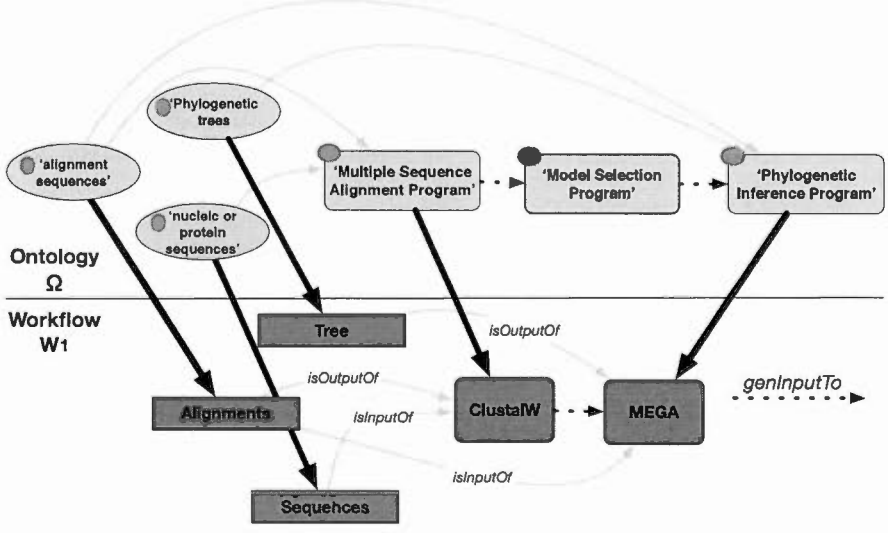


FIGURE 2.8: Example of a workflow reconstruction from a domain-ontology. Here, the hard black arcs represent the possible concept matchings between the ontological workflow pattern (grey rectangles) and the workflow sequence $w_1.\zeta$ (blue rectangles).

Fig. 2.8 shows an example of workflow reconstruction from a domain ontology. The

first set in $w_1.\zeta$: {Sequences, Alignments, Tree} represents the data types. While relations ρ between datatypes and programs '*ClustalW*' and '*MEGA*' represent the data-flow, the sequence itself in $w_1.\zeta$ represents the control-flow ('*genInputTo*').

2.5.7 Workflow similarity measure

Following the work of?, workflows are compared by the titles of their tasks and data descriptions (labels). Here, our algorithm consists of taking two workflows – an expert-curated workflow a and a predicted one p – and analyzing which terms appear in both of them respecting their step occurrences. Unlike the work of Starlinger *et al.* (2014) and Bergmann et Gil (2014) proposing computationally expensive workflow mapping algorithms (based on graph-edit distance or A* search), we propose a simple bag-of-words similarity measure which takes into account both object (data, metadata and tasks) and link (data-flow and control-flow) terms in texts and their ontological tags. The frequency of terms of scientific workflows with a high similarity value tends to be quite similar. Thus, we compare each workflow item term against all the others from the same step to discover how similar they are. We remind that each workflow a and p is represented by the couple (ζ, Θ) in Δ_W where ζ is the set of object labels organized into transactions (steps) and Θ is the the set of relation triples connecting a domain object to a range one from ζ using a link type $l \in \mathcal{L}$. The similarity between two workflows a and p is then given by the average similarity in both workflow components (ζ, Θ) :

$$sim_w(a,p) = \frac{sim_w^{\zeta}(a,p) + sim_w^{\Theta}(a,p)}{2} \tag{2.4}$$

Let $m = |a.\zeta|$ be the total number of steps, n the number of items per step and $k = |a.\Theta|$ the number of link triples in the actual workflow a. Each object $o_{i,j}$ is weighted by its *label correctness* measure $w^{\zeta} \in \{0..1\}$ and each *property correctness* (triples) θ_k is weighted by $w^{\Theta} = w^{\Theta_t} \times w^{\Theta_s}$ where the weight $w^{\Theta_t} \in \{0..1\}$ is

the degree of an *object term correctness* (domain and range) and $w^{\Theta_*} \in \{0..1\}$ is the degree of a *link term correctness*. We define term "correctness" based on the character matchings between a term and its referring label in the ontology Ω. A partial correctness of a term $v \in p.\zeta$ is defined as where v is a substring in $u = a.o_{i,j}$. We apply here the longest common substring algorithm Salton et Buckley (1988) to search for the actual term string in the $a.\zeta$ sequence of object labels.

$$sim_w^{\zeta}(a,p) = \frac{\sum_j^m \frac{\sum_i^n (w^{\zeta}(a.o_{i,j},\ p.\zeta))}{n}}{m} \tag{2.5}$$

$$sim_w^{\Theta}(a,p) = \frac{\sum_i^k (w^{\Theta}(a.\theta_j,\ p.\Theta))}{k} \tag{2.6}$$

2.6 Experiments and results

2.6.1 Dataset

For this study, we downloaded scientific articles from the digital repository PubMed Central PMC database (published from January until April 2015) in XHTML format. 320 texts have been extracted. At first we downloaded more than 1 000 recent articles, journals and conference papers having at least one section : 'phylogenetic analysis' or 'phylogenetic analyses'. However, these sections may have not sufficient information to reconstruct workflows from texts. Thus, we filtered the downloaded articles based on the smallest ontological pattern that any phylogenetic analysis should represent. A phylogenetic study might use at least one datatype and one phylogenetic inference program (see Fig. 2.3) to execute a workflow.

To conceive and implement the domain ontology, we build at first a relational database gazetteer (in MySQL) from different well-known sources of data in the

phylogenetic literature such as Gene Ontology (Ashburner *et al.*, 2000), NCBI[13], UniprotKB[14] and from Felsenstein Website [15] (Letovsky, 2006). The terminology is then reorganized in a domain ontology in order to recognize terms and relations in texts. We used the Quest-Ontop plateform (Rodriguez-Muro et Calvanese, 2012) to map MySQL classes to SPARQL concepts in a domain ontology. The generated ontology presents 17 different object properties (relations), 44 581 concepts (with 7 root concepts and 111 unique concepts) and more than 1 300 000 unique instances without counting the synonyms. Root concepts represent the seven top tasks (steps) of a phylogenetic analysis. The ontology is free to access under the BioPortal repository : `http://bioportal.bioontology.org/ontologies/PHAGE` and the Gate implementation of our workflow extractor is available in : `https://github.com/bioinfoUQAM/tgowler/tree/master/WfExtractor_1.0`.

In addition, an expert has annotated manually random 100 texts from the PubMed database to build a workflow *ground-truth*. Using the GATE platform and our ontology schema, he annotated only terms of concepts and relations that serve to reconstruct the phylogenetic workflows from texts.

2.6.2 Results

In the first experiment, we evaluate the WSD model (PAUM WSD). We use precision, recall and F-Measure metrics. A 10-fold cross validation was run over the PAUM classifier to choose the best parameters : $\tau_- = 5$, $\tau_+ = 50$, where τ_- and τ_+ are respectively the negative and positive margins. We have picked the best values for τ_- from $\{-10, -5, -1, 0, 1, 5, 10\}$ and for τ_+ from $\{-100, -50,$

[13] `http://www.ncbi.nlm.nih.gov`

[14] `http://www.uniprot.org`

[15] `http://evolution.genetics.washington.edu`

-10, 0, 10, 50, 100} based on the validation results. In addition, we chose the whole set of the proposed features since all POS and concept context tags show the best F-Measure with a context window $win_i = 5$.

In the second experiment, we evaluate the relation extraction model. Same as in the previous experiment, we ran a 10-fold cross validation over the PAUM Relation Extraction (PAUM RE) classifier. We chose the best features and best parameters as follows : $\tau_- = 1$, $\tau_+ = 10$. Selected features are : token POS tags, concepts, distance, direction and the list of verbs between domain d and range r (see table 2.1).

Table 2.2: PAUM Word Sense Disambiguation and relation extraction evaluation results.

	Precision (%)	Recall (%)	F-Measure (%)
PAUM WSD	99.21	66.66	79.74
PAUM RE	72.63	84.22	77.24

Table 2.2 shows results of WSD and relation extraction learning. For PAUM WSD, the recall is the ratio of total amount of ambiguous words correctly disambiguated over the total amount of ambiguous words in the expert-curated dataset. The precision is the ratio of the total amount of correctly disambiguated words over the total number of generated ambiguous words. Our PAUM WSD model is highly precise (99.21%) but predicted words represent 66.66% of the total number of ambiguous words in the exert dataset. This is understandable since the context of an ambiguous word could be out of its sentence context. Hence, this is one of the limitation of our context-aware hypothesis. PAUM RE model presents different results in terms of precision and recall from the PAUM WSD but it shows similar F-Measures. PAUM RE covers 84.22% from the actual relations in the expert-curated dataset, however, 72.63% of the generated relations are correctly

predicted.

In order to understand more about our results, we present in Fig. 2.9 the distribution of similarities sim_w between the predicted and the Ground-Truth concrete workflows. The overall similarity is 0.83 over the 100 extracted concrete workflows. This figure shows also details about similarity distributions of both concepts and relations in workflows. The overall mean of relation similarity is 0.89 and concept similarity is 0.79. The best concept results are phylogenetic inference programs where the similarities are above 0.8 except for 5 workflows. Moreover, in one side, gene ontology and taxa annotations in workflows are 0.6 similar within the actual workflows which is understandable since only full scientific terms are considered while tagging the name of species, genes and proteins in texts. Though, synonyms weren't considered for the mentioned concepts at the semantic annotation step. These are dismissed in order to minimize the error cost of the abstractions in common terms (*e.g.* cell, RNA, polymerase, *etc.*). In the other side, all program terms similarities are above 0.8 except for model selection and hypothesis validation programs (0.6). This is due to the context ambiguities within these categories. Bioinformaticians and biologists tend to use general purpose packages instead of specific tools for model selection and tree validation steps. The context of such tools are 40% of the cases out if its sentence area.

In the relation side, links between data types and tree visualization programs show the best similarities (the mean similarity is 0.95). Mean similarities of relations between : (1) data types and MSA (Multiple Sequence Alignment) programs, (2) data types and phylogenetic inference programs and (3) data types and species are all between 0.65 and 0.77. Both predicted relations between model selection programs and their model parameters, and predicted relations between bootstrapping programs and their number of replicates are 0.9 similar with the expected workflow relations. Even though the context-based pipeline supporting our sys-

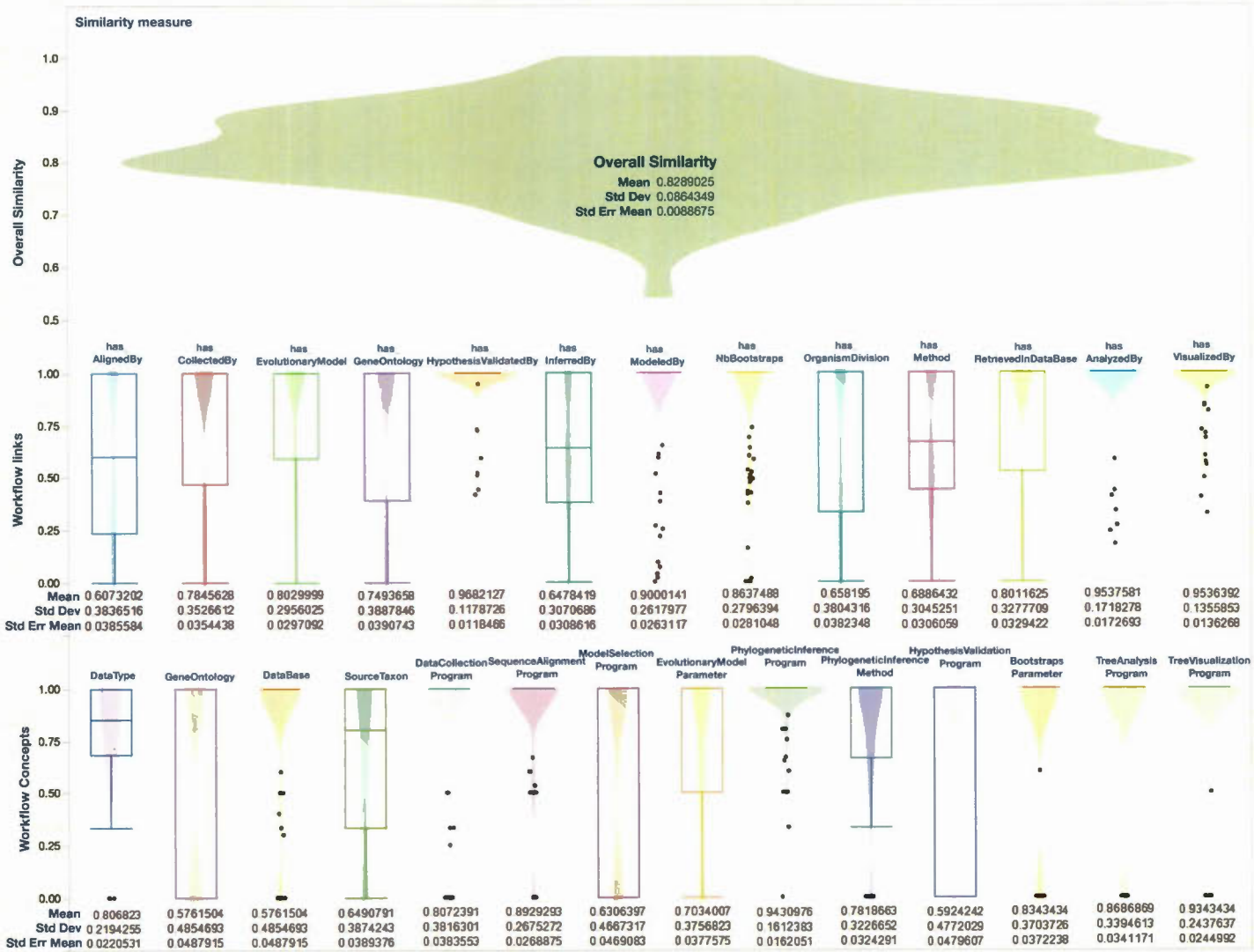


FIGURE 2.9: Workflow similarity distributions.

tem needs to be enhanced in its ability to capture more generic mentions (*e.g.* tree validation programs) in wider contexts and a more elaborate model to capture metadata in texts (specifically for metadata, *e.g.* GO annotations and name of organisms), experiments show satisfying similarity results (between 57% and 96%) with the expert-curated workflows and a good word sense disambiguation model with an F-Measure >77%.

2.7 Conclusion

This paper addresses the problem of the extraction of enriched workflows using a domain-ontology to resolve word sense ambiguities in texts. The framework is supported by an application in the bioinformatics domain. Our graph based workflow representation extends the classic POCBR workflow structures with a semantically richer relations that serve for both : extraction and resolution of polysemic tasks in texts. It is promising that our method reaches high similarities with an ontology-based word sense disambiguation approach for tasks, however, other term classifications need to be improved, mainly data type metadata such as the name of genes, proteins and species. Future work will extend the existing control-flow representation with more sophisticated relations representing disjunction and loop controls in workflows. Outside the context of the case acquisitions, we intend to include text adaptation where, in the case of extraction deficiencies (missing tasks), fragments from the workflows are automatically added based on more general contexts in texts instead of sentences : *e.g.* paragraphs.

Acknowledgment

This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D.

CHAPITRE III

FOUILLE GÉNÉRALISÉE DE FLUX DE TRAVAUX

Le travail de ce chapitre sera prochainement soumis à la prestigieuse conférence ECML PKDD 2018 : European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, sous le titre : "Ontology-powered pattern mining : discovery of frequent abstract workflows in phylogeny". Nous notons que la version courte du papier a été publiée dans la conférence ACM SIGAPP Symposium On Applied Computing (SAC), 2017, sous le titre : "Ontology-based Workflow Pattern Mining : Application to Bioinformatics Expertise Acquisition" (Halioui *et al.*, 2017a) . Nous présentons la version imprimée de l'article dans l'annexe *B*.

Cet article présente la solution proposée afin d'extraire des motifs abstraits de flux de travaux en se basant sur une ontologie processuelle et une base de flux de travaux concrets. Une nouvelle structure d'appariement basée la relation d'incidence entre les motifs générée est proposée afin d'accélérer la tâche de calcul de fréquence. La rédaction de l'article, la conception et l'implémentation de l'approche de fouille, les expérimentations ainsi que le montage des figures ont tous été réalisés par Ahmed Halioui sous la supervision des Prof. Abdoulaye Baniré Diallo et Petko Valtchev. Seulement l'implémentation de l'algorithme d'appariement de motifs a été réalisée par l'étudiant Tomas Martin, maitrise en informatique à

l'UQAM, sous la supervision du Prof. Petko Valtchev.

3.1 Résumé

Les systèmes de gestion de flux de travaux actuels permettent de construire des solutions à des problèmes du domaine (*e.g.*, analyse phylogénétique, fouille de données) dans des processus complexes à multiples étapes qui se composent de tâches élémentaires. Dans des domaines riches, comme dans la bioinformatique, différentes connaissances liées au domaine sont à maitriser (méthodes, outils, formats de données, paramètres, *etc.*). Ainsi, la fouille de processus est appliquée afin de découvrir des motifs de flux de travaux pertinents à partir des traces d'exécutions (*logs*) permettant de découvrir des modèles de "meilleurs pratiques". Cependant, à cause de la non pertinence (en terme de fréquence) des cooccurrences d'items concrets (d'où la "*sous-réutilisabité*" de motifs), les *motifs généralisés* semblent être plus "*intéressants*" à fouiller. Ce type de motifs permet de classifier (catégoriser) les éléments de flux de travaux en des concepts (classes) et des relations. Ainsi, nous proposons une nouvelle approche de fouille de motifs processuels basée sur une ontologie du domaine définissant les éléments d'un flux de travail (flux de données et flux de contrôle). Nous ramenons le problème de la fouille des processus au cadre de la fouille de motifs séquentiels généralisés avec des liens entre ses items. Notre méthode traverse l'espace de recherche de motifs en utilisant cinq opérateurs de raffinement exploitant les liens hiérarchiques dans l'ontologie. De plus, l'étape d'appariement entre les motifs abstraits et les flux de travaux concrets est supportée par une structure dédiée afin de réduire le nombre de comparaisons. Une étude de scalabilité de notre système ainsi que la comparaison avec `xPMiner` (une approche proche à notre solution) montre des résultats satisfaisantes en termes de temps d'exécution (génération de ~13 500 motifs en 0.5 secondes.). L'implémentation de la plateforme de fouille ainsi que les données d'expérimentations sont disponibles dans : `https://github.com/bioinfoUQAM/tgowler/tree/master/WfMiner_1.1`.

3.2 Abstract

In complex problem solving domains such bioinformatics, workflows driven by generic management models, are used to design and execute a wide range of complex analyses. However, a minimum of expertise is to be capitalized in order to guide non experts facing a technical problem, *e.g.*, phylogenetic analysis. As a solution, process mining is applied to extract workflow patterns of abstract components from system logs in order to cover local variations. However, in rich and complex structured domains, an additional issue is related to the right choice of abstraction. We propose, in this paper, a pattern-based workflow mining on the top of a domain ontology which categorizes workflow elements and their interactions. Our method relies on workflow representations as double-labeled DAGs. Hence the underlying problem is transformed into : mining generalized (abstract) sequential patterns with links between their items. Besides, we propose a refinement operator, made of a set of primitives following the ontology structure to traverse the pattern space. In addition, a parsimonious matching discipline exploiting a storage for partial pattern-to-data matchings is used. A study of scalability and a comparison with the `xPMiner` system[1] proves that our method behaves in a satisfactory manner in terms of mining time (output of 14 416 patterns in 0.75 seconds.).

Both data and implementation of our approach are available in : `https://github.com/bioinfoUQAM/tgowler/tree/master/WfMiner_1.1`.

[1] `xPMiner` provide a close approach to ours since it mines sequences (a simpler data structure) instead of workflows as DAGs.

3.3 Introduction

Workflows, as tools for conceiving and managing complex multi-step processes, are widely used in the resolution of complex knowledge-rich domain problems *e.g.*, Bioinformatics. Current workflow management systems offer a wide spectrum of analytical services (Wolstencroft *et al.*, 2013; Blankenberg *et al.*, 2001; Lord *et al.*, 2012) requiring to master different types of expertise in data collection, cloud computing, Web services, *etc* (Blankenberg *et al.*, 2001; Anisimova *et al.*, 2013). Therefore, in order to build a successful workflow, the user must typically master a large variety of methods, tools and data formats. In many cases, the ultimate success of the analysis depends on the right choice of algorithm parameters and even of software versions. In fact, such a demanding environment decreases the value of workflow management for non expert users. This, in turn, motivates the application of automated tools to the acquisition of problem solving expertise –coherent abstract knowledge about problem resolution–, *e.g.*, by distilling reusable parts from recorded past experience. Process mining (Van der Aalst, 2016), a branch of Knowledge Discovery from Data (KDD) (Fayyad *et al.*, 1996), studies the discovery of reusable knowledge from databases of process records, in particular, from workflow representations, *e.g.*, execution logs. As workflows have an inherent graph structure, effective process mining methods face related complexity issues.

Independently, as an approach taming the large variety of services and related process elements, domain ontologies have been increasingly adopted (Wolstencroft *et al.*, 2013). Ontologies offer, *inter alia*, a taxonomy of domain entities that typically comprises categories at various abstraction levels, plus, if applicable, a taxonomy of domain relations. While they might positively impact the mastering of the underlying problem domain by novice users, the real value lays in their ma-

chine readability : an ontology can be leveraged in the process of KDD in order to ensure a variety of benefits. To name a few, the intelligibility of the discovered patterns would increase, if stated in the ontological vocabulary, while easing their integration into the body of a formalized knowledge. In addition, ontologies, representing domain relations and constraints, might be used to limit the exploration of the pattern space and the novelty of the discovered knowledge *w.r.t.* what is already known could be assessed in a direct way. Yet, the benefits come with a cost. Ontology-compliant data is typically graph-shaped whereby vertices are *resources* typed by the ontological categories and edges are *links* labeled by relations. Thus, mining workflow patterns on the top of an ontology amounts to discover *(double) generalized graph patterns* where both concepts and relations are mined. Among the ensuing technical difficulties, it is noteworthy that the interplay between categories and relations induces a huge and highly irregular pattern space.

In this paper, we are investigating the benefits of mining ontology-based workflow patterns for the acquisition of bioinformatics expertise. In particular, we are focusing on *Phylogenetic Analyses* (Lemey Philippe, 2009) whereby datasets are workflows from various origins : traces from Web application logs (Blankenberg *et al.*, 2001; Lord *et al.*, 2012), published workflows as such (Wolstencroft *et al.*, 2013) or reconstructed from the text of scientific papers (Duck *et al.*, 2014; Halioui *et al.*, 2015). Our ultimate goal is to automate assistance to non expert analysts (see (Spjuth *et al.*, 2015) for a discussion). The underlying technical goal is the discovery of abstract workflows from the dataset of concrete workflow records. We approach it as a problem of mining generalized DAG patterns over an ontology. The single truly comparable work from the literature tackles click-stream analysis for an ontology-powered Web portal (Adda *et al.*, 2010) by applying a mining methodology from the work of De Raedt (2008). Similarly, we define two des-

criptive languages, one for Data (concrete workflows) and the other for Patterns (generalized workflows), and a pattern space on top of them with a dedicated refinement operator to traverse it. Patterns are exploited by a Top-Down Depth-First traversal method (Aggarwal, 2016) using support as a quality criterion for patterns.

The remainder of the paper is organized as follows. Section 3.4 summarizes current works on pattern mining with ontologies. Section 3.5 motivates the data format and describes the essential components of our mining problem. Section 3.6 describes the workflow data and pattern languages. While section 3.7 defines a total order between workflow and pattern elements, section 3.8 presents the pattern space and proposes a mining method based on canonical operations and a dedicated matching DAG-based pattern-to-data structure to calculate the *support* (frequency) of patterns. Section 3.9 shows a running example and section 4.5 reports experimental results and discusses practical validation of the entire approach. Finally, section 5.7.4 concludes the paper.

3.4 Related work

Ontology (Stumme *et al.*, 2006) is a versatile construct that may take various shapes : from a simple *is-a* taxonomy, or light-weight ontology, to a fully-blown formal ontology in OWL [2]. Light-weight ontologies have been used in data mining from its onset as a source of domain knowledge to improve the mining process (Aggarwal et Han, 2014). In sequence pattern mining (Han et Fu, 1995; Plantevit *et al.*, 2006; Šebek *et al.*, 2012), the generalized patterns comprise abstract items *i.e.* item categories from a domain taxonomy, *a.k.a.* concepts. For

[2]The W3C Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things – see `https://www.w3.org/OWL`

instance, GSP (Srikant et Agrawal, 1995) mines frequent patterns from sequences of user transactions. It traverses the pattern space in an Apriori-like level-wise descent (Agrawal et Srikant, 1994) by exploring the support monotony over the taxonomy. The workflows we deal with are similar to sequences where transactions (tasks) are only partially ordered. Besides, at some steps the ordering is flexible. Yet, a key difference here is that items can be connected by links labeled by ontological properties (*a.k.a.* relations). Thus, a data record forms a double layer of labeled DAG.

Moreover, our fully-fledged ontology also provides a taxonomy on relations as well as adjacency constraints on those (domain and range concepts). Respectively, target patterns are sequences of concept transactions with inter-concept relations which, due to the specific nature of workflows, amounts to a (double) generalized DAG where each node/link may lay at an *arbitrary* level within its respective taxonomy. As a matter of fact, determining the proper abstraction level is a key issue. Most miners detect generalized patterns made of same-level items (Han et Fu, 1995), with the noteworthy exception of (Fortin et Liu, 1996) where an Object-Oriented approach is used to combine levels.

Generalized graph patterns were formulated by Inokuchi (2004), yet, never fully explored since; arguably due to the inherent complexity of the resulting pattern space. For instance, Cakmak et Ozsoyoglu (2008) proposed a specific type of generalized graph patterns, yet, the approach includes a preliminary flattening of the data graphs : transforming all the nodes/edges to root concepts/relations, respectively from the ontology. Similarly, Jiang *et al.* (2007) proposed to mine generalized patterns from RDF[3] graphs with an RDF schema as a taxonomy. Yet, the proposed method decomposes at first the graphs into (vertex, edge, vertex) triples

[3] `https://www.w3.org/TR/rdf-primer`

and afterwards deals with them as transactions of triple-shaped items. Here, the generalization is operated separately on the individual items following the schema. In this paper, key shortages are the patterns –(multi-)sets of generalized triples– (1) might not constitute a connected graph or, alternatively, (2) compiling them back into a graph might not ensure the result is matched by the covered data graphs. Werth *et al.* (2009) proposed a tuple-based canonical form (labelIndex, predecessorIndex) attached to patterns with a 4-steps refinement operator : (1) adding a root node, (2) adding a node/edge from a different abstract level, (3) replacing a node/edge from the same abstract level and (4) insert an edge pointing to the predecessor. Yet, pattern extensions do not follow a specific order which yields to generate some duplicates.

Using an ontology to assist mining of usage logs[4] (Srivastava *et al.*, 2000) has proven popular in the last decade. A number of studies have used ontologies to enrich patterns mined from sequences of clicks, or similar navigation traces (Dai et Mobasher, 2002; Adda *et al.*, 2010; Senkul et Salin, 2012). The work of Adda *et al.* (2010) is an exception since it presents a complete framework to abstract patterns from click-streams (logs of a semantic Web porta)l powered by a domain ontology. At the preliminary step, logs are turned into sequences of content object IDs further enhanced with semantic links from the ontology. Then, an Apriori-like method, `xPMiner`, extracts patterns as are sequences of concepts labels connected with properties from the ontology. To traverse the pattern space (from the top to the bottom), it uses a refinement operator made of four primitives acting on concepts or properties (by adding/specializing the last concept/relation). Unfortunately, the paper does not present a true validation, *e.g.*, a study of the utility of the discovered patterns. Moreover, the mining task is inherently costly. It uses

[4]Frequent pattern mining techniques for discovering different types of patterns in a Weblog database.

an expensive data-to-pattern matching and it is not shown to avoid multiple candidate generation. Here, we deal with a comparable problem with a different input format. Instead of pure sequences, we deal with DAGs. Due to the complex nature of the underlying domain problem, DAGs are split into ordered layers, thus, our problem is restated as Mining Generalized Sequential Patterns (as opposed to sequence patterns in (Adda *et al.*, 2010)). Following the schema proposed in (De Raedt, 2008), we define dedicated data and pattern languages and show how these are combined into a pattern space *w.r.t.* a generality relation. We also identify levels in the space and provide a refinement operator to move a level down. However, unlike `xPMiner`, our method is a *depth-first* miner exploiting (1) an explicit storage for partial data-to-pattern matchings to be inherited along a downwards path and (2) linear extensions of the respective partial orders on concepts and properties to generate each frequent pattern exactly once. Below, we describe the key elements of our mining problem. To that end, we use, besides an isomorphism-base matching and an ontology-based refinement primitives, concept and relation rankings over the hierarchies in the ontology to ensure a total order among pattern components. This ensures two valuable properties : (1) a canonical representation for each pattern as a sequence of linked concept bags (reducing the matching cost), and (2) a unique order among unit refinements of a pattern that allows to trace it back to the root pattern (via a unique generation chain).

3.5 Motivating example

Phylogenetics (Lemey Philippe, 2009) is a core bioinformatics discipline. It reconstructs the evolutionary history of species based on molecular information, *i.e.* DNA, RNA and proteins. *Fig.* 3.1 shows a phylogenetic workflow designed with the Armadillo (Lord *et al.*, 2012) platform. With tools like Armadillo or Galaxy (Blankenberg *et al.*, 2001), users can build solutions to complex analytical problems from basic blocks, *e.g.*, algorithms, data types and sources. A workflow graph depicts

an analytical resolution in terms of blocks and their interactions –control/data flows– drawn as nodes and links, respectively. Fig. 3.1 represents the comparison of phylogenetic trees built with and without alignment processes, respectively. It comprises three steps : (1) aligning sequences using ClustalW (Thompson *et al.*, 2002) program; (2) inferring phylogenetic trees with PhyML (Guindon *et al.*, 2010) and (3) calculating tree distances using the TreeDist (Zhang et Shasha, 1989) and Robinson&Fould (Robinson et Foulds, 1981) algorithms.

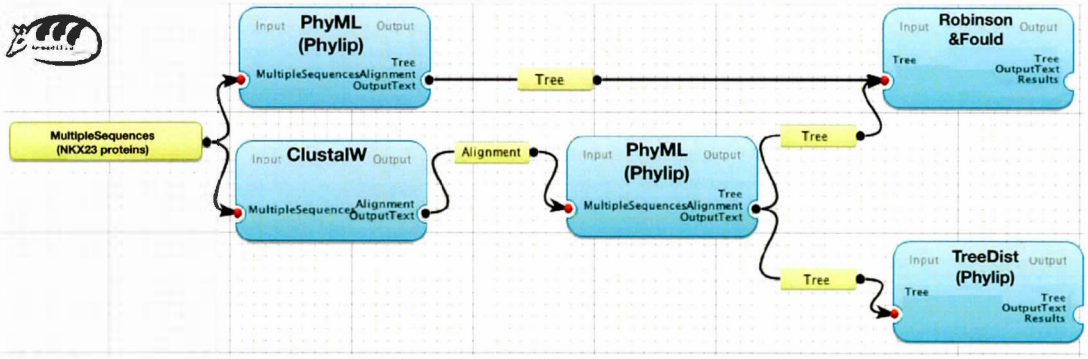


FIGURE 3.1: Bioinformatics workflow : a reconstruction of phylogenies from the NKX23 proteins (Lord *et al.*, 2012). While large rectangles depict tasks (*e.g.*, PhyML, ClustalW) and small ones depict data types (MultipleSequences, Tree, *etc.*), oriented links stand for data flows.

Composing the appropriate workflow is a challenging task since it requires the mastering of a large number of data formats, methods, tools, parameters, *etc.* Yet, recorded workflows might be used as guidelines for novices and even for "experts". Yet, given the huge variety of possible inputs, it is unlikely for a workflow to fully fit a new situation. A solution would be to mine higher-level of abstractions summarizing commonalities in a set of workflows. In our case, one such abstraction could be read as follows : "*Protein sequences involved in muscle contraction are inferred using a character-based method, on one hand, and a progressive alignment plus a character-based inference method, on the other hand, and then, results are*

fed into a distance-based tree validation method". In a hypothetical pattern language, this would become the following sequence of steps (bags of categories) : ⟨{*Muscle Contraction*, *Protein Sequence*}, {*Character Based*, *Progressive MSA* }, {*Character Based*}, {*Tree Distance*}⟩ with interactions as the triples[5] { (1, *isInput*, 2), (1, *isInput*, 3), ...}. Therefore, workflow tasks and data types are organized into a hierarchy of concepts while constraints and dependencies between them become instances of properties that are also organized in a hierarchy of relations. The respective *is-a* relationships are called *SubClassOf* and *SubPropertyOf* in RDF Schema[6]. Fig. 3.2 shows a part of the RDF phylogenetic ontology where both hierarchies are illustrated separately. Instances are also connected to their concepts/properties using the *is-a* property, *e.g.*, the ClustalW software and its concept Progressive MSA.

3.6 Data and pattern languages

The universe of workflow items, *i.e.* concepts C/objects O and relations $\mathcal{R}$/links $\mathcal{L}$, is defined by the ontology Ω.

Definition 3.6.1. *Ontology*
Let $\Omega = \langle O, C, \mathcal{L}, \mathcal{R}, \in_\Omega, \rho \rangle$ *be an ontology where* O *is the set of all objects, i.e. individual tasks and data types,* C *the set of all concepts,* $\mathcal{L}$ *the set of all links between objects, and* $\mathcal{R}$ *the set of all properties (relations) between concepts.* $H_C = \langle C, \in_\Omega \rangle$ *and* $H_\mathcal{R} = \langle \mathcal{R}, \in_\Omega \rangle$ *denote the concept and relation hierarchies of the ontology where* $\in_\Omega$ *denotes the is-a relationship. Objects* $o \in O$ *and links* $l \in \mathcal{L}$ *are instances of concepts* $c \in C$ *and relations* $r \in \mathcal{R}$*, respectively. The ternary relation* $\rho \subseteq C \times \mathcal{R} \times C$ *is made of triples* $c_1 \times r \times c_2$ *expressing the incidence*

[5]Triplets here represent links between the sequence positions instead of labels

[6]`https://www.w3.org/TR/rdf-schema/`

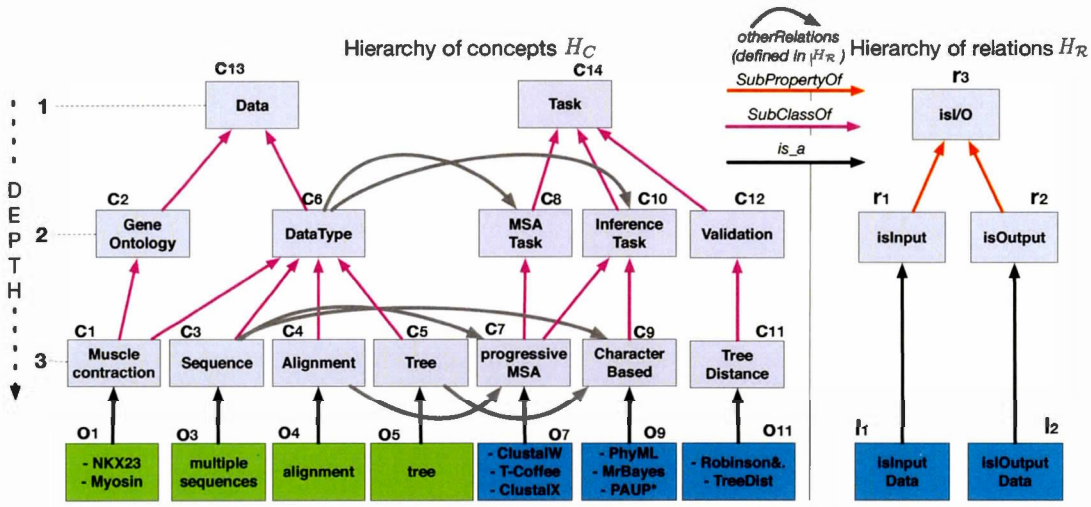


FIGURE 3.2: Hierarchies in the Phylogenetic PHAGE Ontology (a partial). In both hierarchies H_C and $H_{\mathcal{R}}$, concepts/relations are light grey rectangles and instances are dark-coloured ones with data types to the left, tasks to the right (of H_C) and links (of $H_{\mathcal{R}}$).

between a relation r and its domain c_1 and range c_2 concepts.

Remark (Labels in the ontology). In order to simplify the mining step, labels[7] are uniques for each concept $c \in C$ and relation $r \in \mathcal{R}$ in the ontology. For instance, the concept *Alignment* method has only one label *'Alignment'* representing the set of its individuals (instances) (*e.g.*, local, global and structure-based alignments).

Moreover, given a universe of tasks and data O and links $\mathcal{L}$, a workflow record represents a double-labeled DAG with vertex labels from O and edge labels from $\mathcal{L}$. Next, we will use the terms "object" and "link" instead of their corresponding labels (usages), respectively, in the ontology. In addition, vertices are split into

[7]A label is a self-identity and/or behaviour of individuals by the terms used to describe or classify them. In RDF Schema, for example, *rdfs :label* is an instance of *rdf :Property* that may be used to provide a human-readable version of a resource name.

subsequent layers following the steps of the phylogenetic analysis with possible further splits, *i.e.*, through a topological sort, so there is no intra-layer edges remain. In this way, the data record becomes a sequence of object transactions with a set of cross-transactional links.

Definition 3.6.2. *Data language*
Δ_W is made of pairs $w = (\zeta, \Theta)$ where $\zeta = \langle \{o_1, o_2, ..., o_{i_1}\}, \{o_{i_1+1}, \ldots, o_{i_2}\}, \ldots, \{o_{i_{(m-1)+1}}, \ldots, o_{i_m}\}\rangle$ is a sequence of object transactions. In turn, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_p\}$ is a set of triples where $\theta_k = (i_1, l, i_2)$ represents a link of type l between the object at position i_1 in $w.\zeta$, i.e. $w.\zeta[i_1]$ and the object $w.\zeta[i_2]$.

In the above definition, we assume a total order among the items of w, which, in turn, requires transactions to be totally ordered. We discuss a way to obtain such an order from the hierarchies in the following paragraphs. At this point, we define a mapping of the object ranks in that order into the transaction ranks within w. In a valid triple (i_1, l, i_2), i_1 and i_2 belonging to different transactions whereby the transaction of i_1 is before the one of i_2 :

Definition 3.6.3. *Transaction mapping*
Let $\tau_w : \mathbb{N} \to \mathbb{N}$ be the function that maps item positions to transaction IDs :

$$\forall \theta_k = (i_1, l, i_2) \in w.\Theta,\ \tau_p(i_1) < \tau_p(i_2)$$

Example 1. w_1 in *Fig.* 3.3 (A) is a schematic representation of the workflow from *Fig.* 3.1. It comprises tasks b, c, d, e and data type a as well as a set of links. Tasks are further split into steps, thus, inducing the transaction sequence $\langle \{a\}, \{b, c\}, \{b\}, \{d, e\}\rangle$ plus the triples $\{(1, l_2, 2), (1, l_2, 2), \ldots, (4, l_1, 6)\}$. Here, l_1 embodies the execution order in the workflow and l_2 is the input relation between data types and tasks.

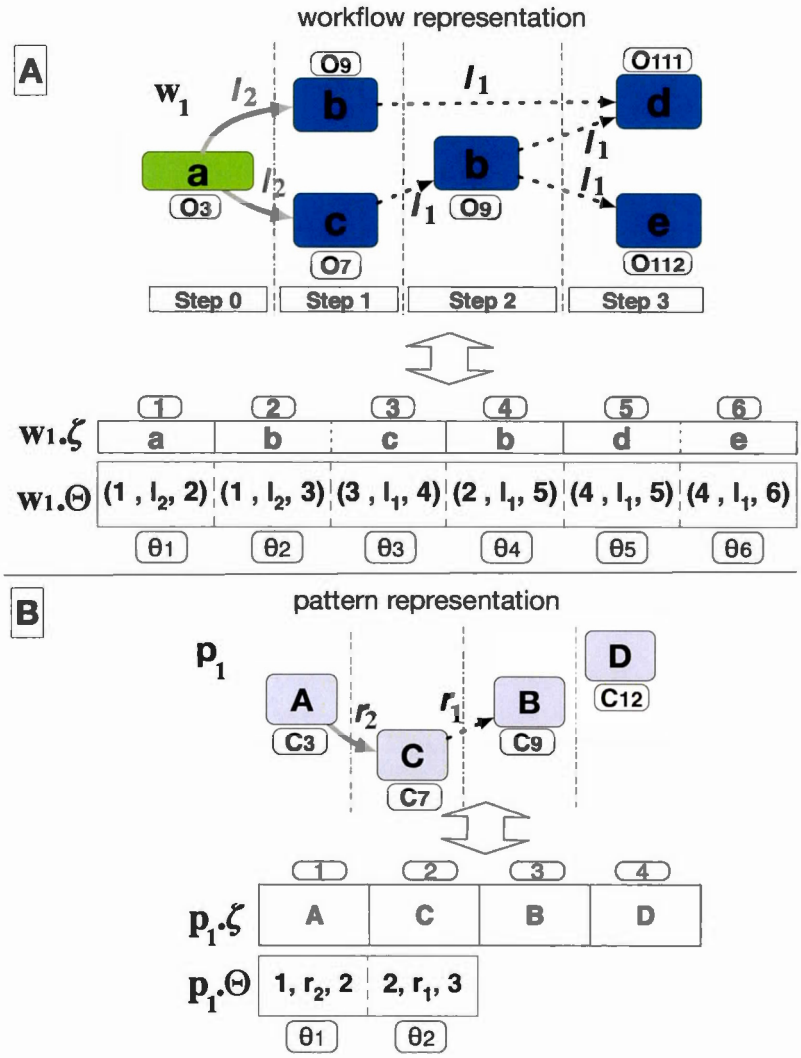


FIGURE 3.3: (A) Workflow representation of $w_1 \in \Delta_W$. (B) Pattern representation of $p_1 \in \Gamma_W$.

Notation-wise, workflow tasks and data types will, henceforth, be denoted by lowercase letters and pattern concepts and relations be denoted by uppercase ones. In addition, same for the pattern side, we define a second language representing different types of semantics : categories of objects and links.

Definition 3.6.4. *Pattern language*
Γ_W *is made of pairs* $p = (\zeta, \Theta)$ *where* $p.\zeta$ *is the sequence of concept (labels) bags and* $p.\Theta$ *is the set of relation triples.*

Remark (Concept and relation labels in patterns). We need to mention that concept usages (labels) in the pattern sequence ζ are not uniques, thus, pattern transactions are multisets[8].

Example 2. p_1 in *Fig.* 3.3 (B) represents the sequence $\langle \{A\}, \{C\}, \{B\}, \{D\} \rangle$ plus the triples $\theta_1 = (1, r_2, 2)$ and $\theta_2 = (2, r_1, 3)$ linking pattern components.

3.7 Total order

To streamline pattern mining, we define a canonical representation for each pattern that eases, among others, the detections of identical patterns. Such forms are typically based on a total ordering of pattern components whereby the *last position* is the focus of refinement primitives to guarantee a *non redundant* and a *unique* space traversal. Hence, given a pattern p, a chain of refinement operations can be built from p up to the root pattern by reversing its generation in terms of successive refinements. The chain is unique whenever at each step, the *last* refinement is unambiguously defined, both in terms of primitive and argument pattern.

---

[8]A multiset is a generalization of the concept "set", that, unlike a set, allows multiple instances of elements. For example, $\{a, a, b\}$ and $\{a, b\}$ are different multisets although they are the same set.

To that end, we define, at first, a total order between concepts/relations in Ω to help sorting each transaction in workflows and patterns. Since we follow a general-to-specific (top-to-bottom following the hierarchical relations) traversal of the pattern space, ranks need to be monotonous (the more general a concept/relation type, the higher its rank). Therefore, we define the functions $rank_C : C \rightarrow \mathbb{N}$ and $rank_{\mathcal{R}} : \mathcal{R} \rightarrow \mathbb{N}$ mapping concepts and relation types, respectively, into natural numbers. Ranking is obtained from the hierarchy H_C and $H_{\mathcal{R}}$ in the following way : (1) the topological sort of the hierarchy yields to a split of C and $\mathcal{R}$ into levels, (2) for each concept $c \in C$ and relation type $r \in \mathcal{R}$, the choice of a unique *canonical parent* (*e.g.*, c' from the level immediately above : $c \prec_{can} c'$) is done by ignoring multiple inheritance in the hierarchies[9], and (3) over the resulting spanning tree of the hierarchy, we assign ranks along a depth-first postfix traversal.

Definition 3.7.1. *Canonical parent*
By ignoring the multiple inheritance in H_C and $H_{\mathcal{R}}$ using a post-fix traversal, the canonical parent of a concept $c \in H_C$ is the unique concept c' whereas $c \prec_{can} c'$ denoting that c' is more general than c (c is more specific than c'). Same for properties, $r \prec_{can} r'$ denotes that r' is the canonical parent of r in $H_{\mathcal{R}}$.

In addition, in the taxonomies H_C and $H_{\mathcal{R}}$, a root concept $\hat{c}$ or a root relation $\hat{r}$, respectively, represent the set of the most general items in the hierarchy besides the empty set $\emptyset$, hence : $\hat{c} \prec_{can} \emptyset$ and $\hat{r} \prec_{can} \emptyset$.

Definition 3.7.2. *Concept/relation ranking functions*
The function $rank_C : C \rightarrow \mathbb{N}$ of a concept c is the position of c in the postfix topological order of H_C, i.e. $c \prec_{can} c' \implies rank_C(c) < rank_C(c')$. Similarly for

---

[9] In order to guarantee a *unique* generation of patterns from the ontology, this latter might present unique specialization paths in its hierarchies. Therefore, we ignore multiple inheritances in H_C and $H_{\mathcal{R}}$.

relations, the rank of a relation $rank_{\mathcal{R}} : \mathcal{R} \rightarrow \mathbb{N}$ *is the position of* r *in the postfix topological order of* $H_{\mathcal{R}}$, *i.e.* $r \prec_{can} r' \implies rank_{\mathcal{R}}(r) < rank_{\mathcal{R}}(r')$.

Example 3. The *Fig.* 3.2 shows the result of our ranking method on the phylogenetic ontology (see numbers above rectangles) : *e.g.*, $rank_C(Tasks) = 14$ while 'CharacterBased' $\prec_{can}$ 'Inference Task'.

Based on the $rank_C()$ and $rank_{\mathcal{R}}()$ functions, a pattern can be generated in its canonical representation. However, this is of little help for matching, even if the ranking is extended to instances in the hierarchies so that workflow transaction could be sorted *w.r.t.* a compatible order. Indeed, even in those cases, it is not possible to ensure a pattern and a workflow can be matched like pure sequences due to the partial order in transactions, *e.g.*, parallel execution of tasks in workflows. In other terms, while there might be no sequence matching over the canonical forms, local reordering in some of the workflow transactions might enable such a matching.

Example 4. Matching $p_2.\zeta$ with $w_2.\zeta$ (see *Fig.* 3.4) amounts to find a matching for every concept in $p_2.\zeta$ while there still be a non matched object in $w_2.\zeta$ or a complete match is found. In this example, the first iteration (ordering) in $w_2.\zeta$ finds the first matching between the first concept c_3 and the second object o_3. Next iteration places o_3 at the first position in $w_2.\zeta$ and shifts non matched objects to the right. A matching is found in this set between c_6 and the last object o_4. Reordering continues until all concept matchings are found (in the 3^{rd} iteration).

In addition, given a pattern p, the triples in $p.\Theta$ are *ordered w.r.t.* the involved positions. At first, we sort upon the destination positions (third element in the triple) and then by the source ones (first element in the triple). Since there could be concept repetitions in a pattern, concept positions in $p.\Theta$ are ordered according

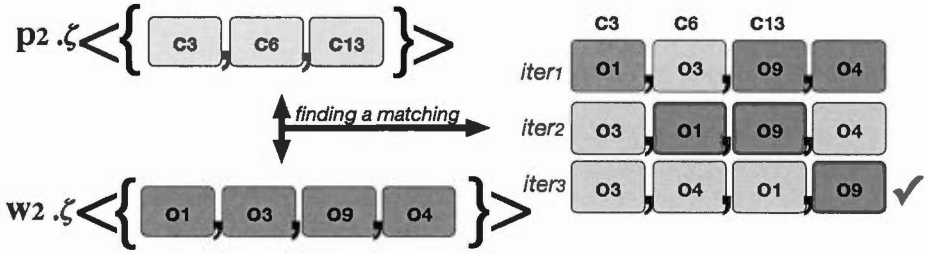


FIGURE 3.4: Matching example between concepts in $p_2.\zeta$ and objects in $w_2.\zeta$. Red rectangles show that the object is matched to the corresponding concepts in the pattern, *e.g.*, at the first iteration, $w_2.o_3$ is matched with $p_2.c_6$.

to source positions i_k and relation types r_k.

Definition 3.7.3. *Relation/link triples order*
Let (r_k, i_k) *and* (r_j, i_j) *be the prefixes of two different triples :* $\exists l$, $1 \leq m \leq l$, $i_{j_m} = i_{k_m}$ *and* $rank_{\mathcal{R}}(r_{j_m}) \leq rank_{\mathcal{R}}(r_{k_m}) \Rightarrow i_{j_{l+1}} < i_{k_{l+1}}$.

Since no triples share both their source and destination in p (with the same relation types), the above ordering is *total*. This enables the definition of the *last triple* in a pattern.

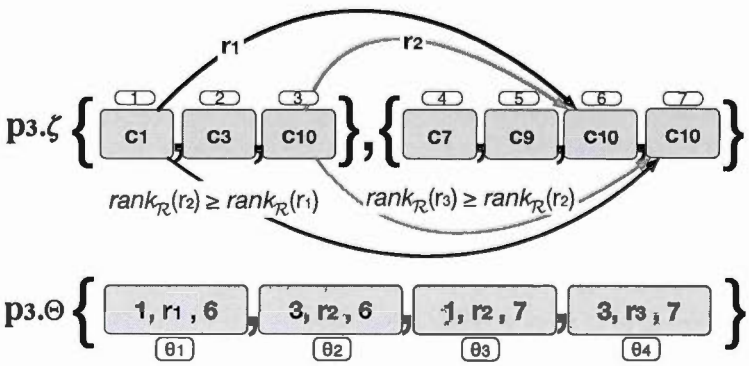


FIGURE 3.5: Example of triple set ranking.

Example 5. $p_3.\Theta$ in *Fig.* 3.5 shows a duplication of the concept c_{10} in the last

transaction. The question that we raise here is : how do we reorder duplicate concepts in the pattern sequence while calculating matching workflows ? According to the above definition, $p_3.\Theta$ is sorted according to relation types while considering (the order) the starting positions. Hence, relations on the last concept c_{10} (7^{th} position) have relation rank types greater or equal than their previous duplicates on the same starting positions. For instance, the rank of r_3 in θ_4 is greater than the rank of r_2 in θ_2.

3.8 Mining workflow patterns

3.8.1 Pattern space

In order to define the pattern space, we need to define the *generality* order ($\sqsubseteq$) between patterns which follows the *instantiation* relationship ($\lhd$) between workflows and patterns. For a workflow $w \in \Delta_W$ and a pattern $p \in \Gamma_W$, we assert $w \lhd p$ (said w *instantiates* p) whenever there exists a sub-structure in w isomorphic to p. The mapping function $\Phi : \{1, ..|p.\zeta|\} \rightarrow \{1, ..|w.\zeta|\}$ defines the instantiation relationship between w and p as follows :

Definition 3.8.1. *Instantiation relationship*

- Φ *is an injective function ;*
- $\forall i \in \{1, .., |p.\zeta|\}$, $w.\zeta[\Phi(i)] \in_\Omega p.\zeta[i]$;
- $\forall \theta_k = (i_1, r, i_2) \in p.\Theta$, $\exists l \in \mathcal{L}$ *s.t.* $l \in_\Omega r$ *and* $(\Phi(i_1), l, \Phi(i_2)) \in w.\Theta$.
- $\forall i, i' \in [1, |p.\zeta|]$, $\tau_p(i') \leq \tau_p(i)$ *iff* $\tau_w(\Phi(i')) \leq \tau_w(\Phi(i))$

The mapping function Φ maps every concept (second property) and triple (third property) in p with its corresponding item in w. The last property of Φ ensures that item instantiations between a pattern p and a workflow w should follow (1) the same order of items from the same transaction in p and w and (2) the order of mapped transactions between p and w.

Example 6. w_1 instantiates p_1 ($w_1 \lhd p_1$) in $\langle$ $\{a\}$, $\{c\}$, $\{b\}$, $\{d\}$ $\rangle$ and in links $\{l_1, l_2\}$ while $l_1 \in_\Omega r_1$, $l_2 \in_\Omega r_2$, $a \in_\Omega A$, $b \in_\Omega B$, *etc.*, $(\Phi(1), l_2, \Phi(3)) \in_\Omega p_1.\theta_1$ and $(\Phi(3), l_1, \Phi(4)) \in_\Omega p_2.\theta_2$ (see *Fig.* 3.3).

In fact, the generality order between patterns follows instantiations as a pattern p_1 is a super-pattern (*canonical parent*) of another one p'_1, whenever every data component that instantiates p'_1, instantiates also p_1, however the inverse is not true.

Definition 3.8.2. *Generality relationship*

Let p_1 and p'_1 be two patterns from Γ_W, p_1 is more general than p'_1 : $p'_1 \sqsubseteq p_1$ iff $\{w_i \in \Delta_W | w_i \lhd p'_1\} \subseteq \{w_j \in \Delta_W | w_j \lhd p_1\}$.

Remark (Limitation of the generality definition). We need to mention that, the more generic (real) generality relation between two patterns $p_2 \preceq p_1$ implies that, for all possible workflows (in our case a finite set[10]), the following property holds : $\{w_i \in \Delta_W | w_i \lhd p_2\} \subseteq \{w_j \in \Delta_W | w_j \lhd p_1\}$ (coverage).

Definition 3.8.3. *Levels in pattern space*

The level of a pattern, denoted as its generality degree g, is the sum of its component (concepts and relations) depths h from the ontology where a depth h is the maximal path (maxDepth) from a component to the root according to its hierarchy in Ω. Hence, $\forall c_i \in C$ and $\theta_j = (c^j_{domain}, r^j, c^j_{range}) \in \rho$, $g(p) = \sum_{c_i}^{|p.\zeta|} h(c_i) + \sum_{\theta_j}^{|p.\Theta|} h(\theta_j)$ where $h(\theta_j) = maxDepth(c_i, H_C)$ and $h(\theta_j) = maxDepth(r^j, H_\mathcal{R})$.

Example 7. The level of the pattern p_3 is $g(p_3) = h(c_1) + h(c_3) + h(c_7) + h(c_9) + 3 \times h(c_{10}) + h(r_1) + h(r_2) + h(r_3) = 23$ (see *Fig.* 3.2). Thus, a level in our pattern space is defined as the set of patterns having the same generality degree.

---

[10]Since the number of steps is fixed same as the number of each input/output task parameters (domain constraints), the number of all phylogenetic workflows is finite.

Furthermore, the generation of $k+1$-level patterns from k-level ones amounts to the calculation of the following five *canonical* operations : (1) root concept insertion in a new transaction, (2) root concept insertion at the end of one transaction, (3) concept specialization, (4) root relation insertion and (5) relation specialization. Since pattern elements are indexed using the total order induced by the ontology ranking, refining a pattern is always done from its last element (the highest rank).

Let c be the last concept in $p.\zeta$, r be the relation in the last triple of $p.\Theta$, $\hat{c}$ a root concept and $\hat{r}$ a root relation in the ontology. Refining p amounts to add a concept/relation in the pattern or, alternatively, to replace c by c' for whom c is the canonical parent concept : $c' \prec_{can} c$. Similarly, r can only be specialized to a relation r' that satisfies $r' \prec_{can} r$. Table 3.1 lists the precise set of conditions for a canonical operation to be applicable for all the five of them.

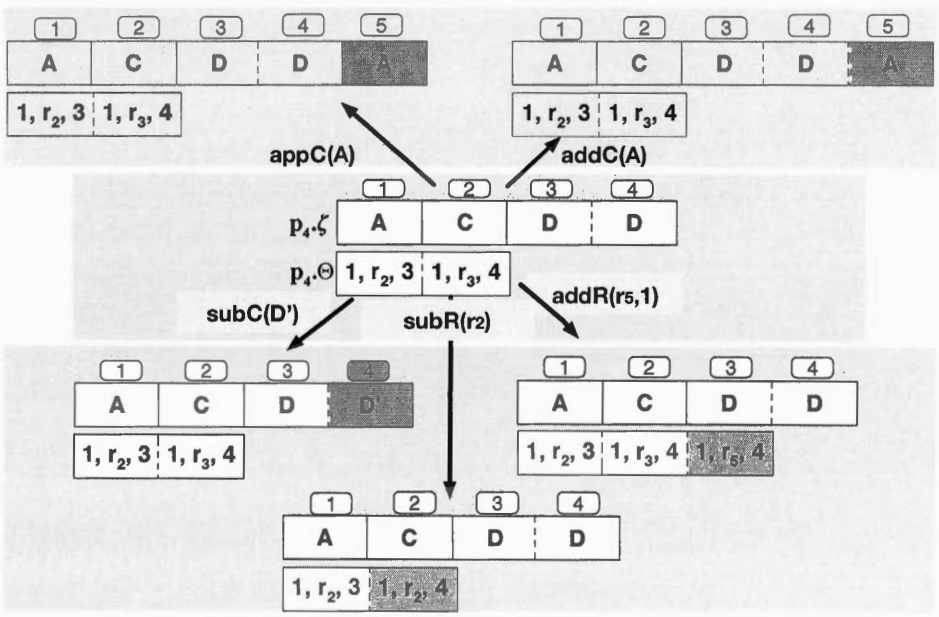


FIGURE 3.6: An example of canonical operations.

Table 3.1: Canonical operations.

Definition	Constraints
appC$(\hat{c})$: appends a new transaction with a root concept in it.	1. $\hat{c}$ is a root concept.
addC$(\hat{c})$: adds a root concept to the last transaction.	1. $\hat{c}$ is a root concept; 2. $rank_C(\hat{c}) \geq rank_C(p.\zeta[\|p.\zeta\|])$.
subC(c') : replaces the last concept with c'.	1. $c' \prec_{can} p.\zeta[\|p.\zeta\|]$; 2. $rank_C(p.\zeta[\|p.\zeta\|-1]) \leq rank_C(c')$ (where $rank_C(p.\zeta[\|p.\zeta\|-1])$ is the 2nd last concept).
addR$(\hat{r}, j)$: adds a relation $\hat{r}$ between position j and the last position.	1. $\hat{r}$ is a root relation; 2. $p.\zeta[\|p.\zeta\|] \in_\Omega ran(\hat{r})$; 3. $p.\zeta[j] \in_\Omega dom(\hat{r})$; 4. $\forall k\ \exists r_k$ with $(k, r_k, \|p.\zeta\|) \in p.\Theta \Rightarrow k < j$ (right most domain). 5. if $p.\zeta[\|p.\zeta\|-1] = p.\zeta[p.\zeta\|]$ then $\forall l, \exists r_l$, $(i_l, r_l, \|p.\zeta\|-1) \in p.\Theta \Rightarrow$ $rank_R(\hat{r}) \geq rank_R(r_l)$, $i_l \geq j$.
subR(r') : replaces the last relation of the pattern with $\hat{r}$.	let $j_m = max(j\|(j, r_i, \|p.\zeta\|) \in p.\Theta)$ and $r"$ s.t. $(j_m, r", \|p.\zeta\|) \in p.\Theta$ in : 1. $r' \prec_{can} r"$; 2. $p.\zeta[\|p.\zeta\|] \in_\Omega ran(r')$; 3. $p.\zeta[j_m] \in_\Omega dom(r')$. 4. if $p.\zeta[\|p.\zeta\|-1] = p.\zeta[p.\zeta\|]$ then $\forall l, \exists r_l$, $(i_l, r_l, \|p.\zeta\|-1) \in p.\Theta \Rightarrow$ $rank_R(r') \geq rank_R(r_l)$, $i_l \geq j_m$.

Here $ran(r')$ and $dom(r')$ functions return the corresponding range and domain concepts, respectively, of one relation r' from H_R.

Example 8. *Fig.* 3.6 illustrates all the canonical operations that we can apply on a pattern p_4. To append the concept A to a new transaction in $p_4.\zeta$, we only verify if A is a root concept. However, in order to add a concept A at the last transaction of $p_4.\zeta$, we check if A is a root and its rank is higher than the last rank in $p_4.\zeta$. Specializing the last concept $subC(D')$ in p_4 amounts to replace the last concept D with its descendent D' while its rank is higher than the rank of the previous concept (B) in last the transaction. The $addR(r_5, 1)$ operation on p_4 adds the relation r_5 between the first concept A and the last one D. To that end, we need at first to verify if there is at least one relation pointing to the last position. Next, we check the compatibilities between the concept A and the domain of r_5 and between D and the range of r_5 from the ontology. After that, we add the concerning triple after the right most one in $p_4.\Theta$ having 4 (last concept position) as the ending point (destination) and r_5 as a relation. We remind that $p_4.\Theta$ set is sorted by its ending and starting positions and then, by its relation type rank. The final operation $subR(r_2)$ specializes the last relation on r_3 having the concept A as a starting position (source). In order to preserve the order in $p_4.\Theta$, r_2, the canonical descendant of r_3 from the ontology, should start from the highest starting point (source) in the subset of triples of r_3 in $p_4.\Theta$ which is $\theta_2 = (1, r_3, 4)$. Hence, $subR(r_2)$ replaces θ_2 with the triple : $(1, r_2, 4)$.

3.8.2 Mining method

Mining abstract workflows from a database $\mathcal{D}_\mathcal{W}$ on top of an ontology Ω and a minimum support threshold σ amounts to find all the frequent patterns $\mathcal{F}_\sigma = \{p \in \Gamma_W | supp(p) \geq \sigma\}$. Here $supp(p)$ is the number of workflows from $\mathcal{D}_\mathcal{W}$ matching p $|w \lhd p|$. We propose to use a depth-first overall traversal in which we apply our refinement operator to find all candidate patterns from a given frequent pattern p. The algorithm starts with the generation of the frequent 1-patterns which are made of unique root concepts from the ontology. For each one of them, the gene-

ration is applied recursively to subsequent levels $k+1$. The level candidates are generated from frequent patterns at level k by applying the canonical operations.

Algorithm 1 Generate frequent patterns

```
1: procedure GENERATEANDTESTDF(f : frequent k-level pattern; MS : matching
   workflows; σ : minimum support )
2:     CP ← ADDC(f) ∪ APPC(f) ∪ SUBC(f) ∪ ADDR(f) ∪ SUBR(f);   ▷ Apply all
   the canonic operations (if the constraints are satisfied)
3:     for all p ∈ CP do                                        ▷ For each candidate
4:         M' ← FINDMATCHINGWORFKLOWS(p, MS);
5:         if size(M') >= σ * |D_W| then    ▷ If the support of p is above the throshold
6:             FP^{k+1}.add(p);                              ▷ Save k+1 Frequent Patterns
7:             GENERATEANDTESTDF(p, M', σ);                  ▷ Generate k+2 candidates
8:         end if
9:     end for
10: end procedure
```

Algorithm 1 provides the pseudo-code to expand a k-level frequent pattern to a set of $k+1$ candidate patterns and then, filter these *w.r.t.* the set of sequences matching the key pattern. Candidate generation from $f \in \mathcal{F}$ amounts to tentatively applying all of the five canonical operations. For each generated candidate, we search for its workflow matching set (M') to calculate its frequency starting from the last matched operation (on concept/relation) position. If the pattern is frequent, we save this latter and its matching set to calculate recursively pattern candidates of the next levels $k+2, k+3, ...$ until there is no frequent patterns to generate.

A legitimate question with our new mining method is whether it is *complete*, *i.e.* it generates all the frequent patterns. Another one is about the non redundancy in generation.

Proposition 1. *Non redundancy*

There is only one way to generate a pattern p^{k+1} from a parent pattern p^k.

Proof. If there is a root relation $\hat{r}$ on the last concept $\hat{c}$ in p^{k+1}, thereof, the last operation applied on p^k would be $addR(\hat{r}, j_m)$, where j_m is the most right domain (see the fourth constraint on $addR$ operation). However, if $\hat{r}$ is not a root relation, then, the last operation is $subR(\hat{r})$ while there is only one way to specialize a relation $r' : \hat{r} \prec_{can} r'$ from the spanning tree. Moreover, if the position of the last concept $\hat{c}$ is not j and $\hat{c}$ is a unique root concept in the last transaction (singleton), then, the last operation is $appC(\hat{c})$ (see the constraint on $appC$ operation). However, if $\hat{c}$ is a not a root concept in the last singleton transaction, then, the last operation is $subC(\hat{c})$ (see the definition of $subC$). If $\hat{c}$ is a root concept in the last transaction (non-singleton), then, the last operation is $addC(\hat{c})$ (see the definition of $addC$). In addition, if $\hat{c}$ is not a root concept in the last transaction (non-singleton), then, the last operation on p^k is $subC(\hat{c})$ while $\hat{c}$ is obtained from a unique $c' : \hat{c} \prec_{can} \hat{c}$ (see the second constraint on $subC$ operation). Finally, as we shown above, if there is only one way to obtain patterns p^{k+1} from p^k, then, recursively, there is only one way to obtain patterns p^1 from the root pattern $p_0 = \langle\emptyset\rangle$. p^1 patterns represent only root concepts c in singleton transactions, hence, the only operations that could be done on p_0 are $appC(c)$ operations. □

Proposition 2. *Completeness*

If frequent patterns at the level $k+1$ exist, then, any of them could be generated by at least one pattern from $f \in \mathcal{F}$ at level k by the means of the canonical operations.

Proof. According to the generality order defined in definition 3.8.2, $\forall p^k \in FPC^k$ is frequent ($supp(p^k) \geq \sigma$), its parent pattern p^{k-1} is frequent too (due to the

monotony of the generality relationship). Since patterns are generated recursively by applying canonical operations on the last element and there is alway one way to do it (see previous proof), then the root pattern p_0 is frequent. This is always true since the empty set is always frequent. □

Due to the combinatorial number of generated sub-patterns at each level, we propose to generate less patterns while there exists no super pattern (in some higher level) $p_i \sqsubseteq p_j$ with the same support or a p_i with $\{w_i \in \Delta_W | w_i \sqsubseteq p_i\} \subset \{w_j \in \Delta_W | w_j \sqsubseteq p_j\}$ (coverage property).

Remark (Not closed patterns). Since our pattern space is not built on the generic generality order[11], it is trickier to spot all the closed patterns or maximals (Zaki et Hsiao, 2002) due the proposed canonical operation. However, we only prune here a set of patterns "covering" the same set of workflows in order to prune more patterns for the end-user interpretability.

The algorithm 2 resumes the first algorithm of mining while pruning all patterns p_i satisfying the above "covered" pattern assumption. At the first place, we eliminate all candidate patterns $p_i \in CP$ having the same support as their parent f (lines 6–9), then, verify if p_i has no super pattern $p_j \supset p_i$ verifying the "coverage" property (see lines 11–17).

Matching workflows

In order to improve the matching operation, we propose to calculate frequent patterns $p \in \mathcal{F}_\sigma$ starting from the last visited concept or relation. Hence, we do not need to recalculate all the matchings for each concept and relation positions in

[11]Each level in the pattern space is defined by the five canonical operators in order to maintain a total order between pattern items.

Algorithm 2 Prune "covered" patterns

1: **procedure** PRUNEPATTERNS(f : frequent k-level pattern ; MS : matching workflows ; σ : minimum support)
2: $CP \leftarrow$ ADDC(f) $\cup$ APPC(f) $\cup$ SUBC(f) $\cup$ ADDR(f) $\cup$ SUBR(f) ;
3: **for all** $p_i \in CP$ **do**
4: *covered* $\leftarrow$ **False** ;
5: $M' \leftarrow$ FINDMATCHINGWORFKLOWS(p_i, MS) ;
6: **if** size(M') $>= \sigma * |\mathcal{D}_{\mathcal{W}}|$ **then**
7: **if** size(M') = size(MS) **then** ▷ *if the number of a candidate matchings is equals to the number of its parent matchings*
8: removeFromFP(p_i) ; ▷ *the candidate is covered by its parent*
9: *covered* $\leftarrow$ **True** ;
10: **else** ▷ *Search for covered patterns within the set of all other candidates*
11: **for all** $p_j \in CP$ **do**
12: **if** $\{w_i \in \Delta_W | w_i \sqsubseteq p_i\} \subset \{w_j \in \Delta_W | w_j \sqsubseteq p_j\}$ **then** ▷ *if all candidate matchings is covered by (intersects with) an another candidate*
13: removeFromFP(p_i) ;
14: *covered* $\leftarrow$ **True** ;
15: **break** ;
16: **end if**
17: **end for**
18: **end if**
19: **if** *covered* = **False then**
20: $FPC^{k+1}.add(p_i)$; ▷ *if the frequent pattern is not covered by any other pattern, then, we generate its frequent sub-patterns*
21: PRUNEPATTERNS(p_i, M', σ) ;
22: **end if**
23: **end if**
24: **end for**
25: **end procedure**

the pattern. Besides, calculating the frequency of a candidate pattern generated from a canonical parent amounts to find all the workflows that match with the candidate based on the workflows matching its parent. For example, if $p'_1 \sqsubseteq p_1$ and $w_i \lhd p_1$, then, all we need is to verify if the matchings between p'_1 still hold starting from the last operation on p'_1. Therefore, the matching set of pattern candidates is reduced correspondingly to the matching set of parent patterns.

For this purpose, we propose a linked list of canonical operations on pattern items that we call *blocks*. Each block in the matching structure corresponds to a canonical operation divided into concept blocks and relation blocks. Concept blocks represent the operations of appending, adding and specializing concepts and relation blocks represent constraints on concept blocks *i.e.* operations on *Source* and *Destination* concepts of adding and specializing relations. Hence, we divided relation blocks into two types of blocks : *source* blocks representing the starting points of a relation and *destination* blocks representing the ending ones. Concept and relation blocks are ordered so each matching structure represents the sequence of operations to generate a pattern (see *Fig.* 3.7).

Adding a new relation $addR(r, i_1)$ amounts to create a *source* block and a *destination* one in the matching structure MS. At first, we search for the matching on the *destination* position i_2 based on previous matchings of the parent $MS(p)$ and create a *destination* block in the right place (according to relation constraints if they exist). Moreover, the *source* position i_1 validations start from the *destination* solutions of i_2.

Looking for the triple in $w.\Theta$ which matches in *destination* with the position i_2 is presented in the Algorithm 4. The search for workflow instantiating $MS(p)$ is done on the set of workflows instantiating its parent (line 3). The array m stores matching positions between $w.\Theta$ and $p.\Theta$ (line 5). The search begins from the last

Algorithm 3 Adding relation blocks in $MS(p)$

1: **procedure** ADDRELATIONBLOCKS($MS(p)$: matching structure of a pattern p ;r : relation to add, i_1 : position of starting concept)

▷ *destination validation*

2: $sib \leftarrow \emptyset$;

3: $i_2 \leftarrow |p.\zeta|$;

4: $ranB \leftarrow MS(p).getCptBlock(i_2)$;

5: $mRan \leftarrow$ MATCHINGDESTINATION ($MS(p)$, $ranB$, i_2, r) ;

6: **if** $ranB.lastRelationOn(i_2)$ **then** ▷ *create a link with the previous position*

7: $sib \leftarrow MS(p).nextRelBlock(i_2)$;

8: **else**

9: $sib \leftarrow MS(p).getCptBlock(i_2)$;

10: **end if**

11: CREATEDESTINATIONBLOCK ($ranB$, $mRan$, i_2, sib) ;

▷ *source validation*

12: $domB \leftarrow MS(p).getCptBlock(i_1)$;

13: $mDom \leftarrow$ MATCHINGSOURCE($MS(p)$, $domB$, i_1, i_2, r) ; ▷ *if there's no other relation constraint on* c_1

14: **if** $domB.lastRelationOn(i_1)$ **then** ▷ *create a link with the previous position*

15: $sib \leftarrow MS(p).nextRelBlock(i_1)$;

16: **else**

17: $sib \leftarrow MS(p).getCptBlock(i_1)$;

18: **end if**

19: CREATESOURCEBLOCK($domB$, $mDom$, i_1, sib) ;

▷ *update MS(P)*

20: $MS(p).lastAddedsource \leftarrow domB$;

21: $MS(p).lastAddeddestination \leftarrow ranB$;

22: $MS(p).lastAddedRelation \leftarrow r$;

23: **return** $MS(p)$;

24: **end procedure**

Algorithm 4 Verifying destination matchings

```
1: procedure MATCHINGDESTINATION(MS(p) ; ranB ; i2 ; r)
2:     result ← ∅ ;
3:     while w ∈ MS(p).listW() do
4:         notFound ← true ;                ▷ for each w which previously matched MS(p)
5:         m ← w.Θ.getMatchedPos(MS(p)) ;   ▷ get all matched positions in w
6:         if ranB.hasRan() then            ▷ if the source block points to a source already
7:             j ← m.getLastPos(ranB.getRelPos()) ; ▷ search begins from that source
8:         else
9:             j ← m.getLastPos(|p.Θ|) ;
10:        end if
11:        i ← |w.Θ| ;
12:        while i >= j and notFound do     ▷ for each relation in w
13:            match_R ← r ∈Ω w.Θ[i].lnk() ;
14:            match_r ← p.ζ[i2] ∈Ω w.Θ[i].ran() ;
15:            if match_R and match_r then
16:                m[w.Θ[i]] = |p.Θ| + 1 ;
17:                notFound ← false ;
18:            end if
19:            i − − ;
20:        end while
21:        result.add(w_ID, MS(p)_ID, m) ;
22:    end while
23:    return result ;
24: end procedure
```

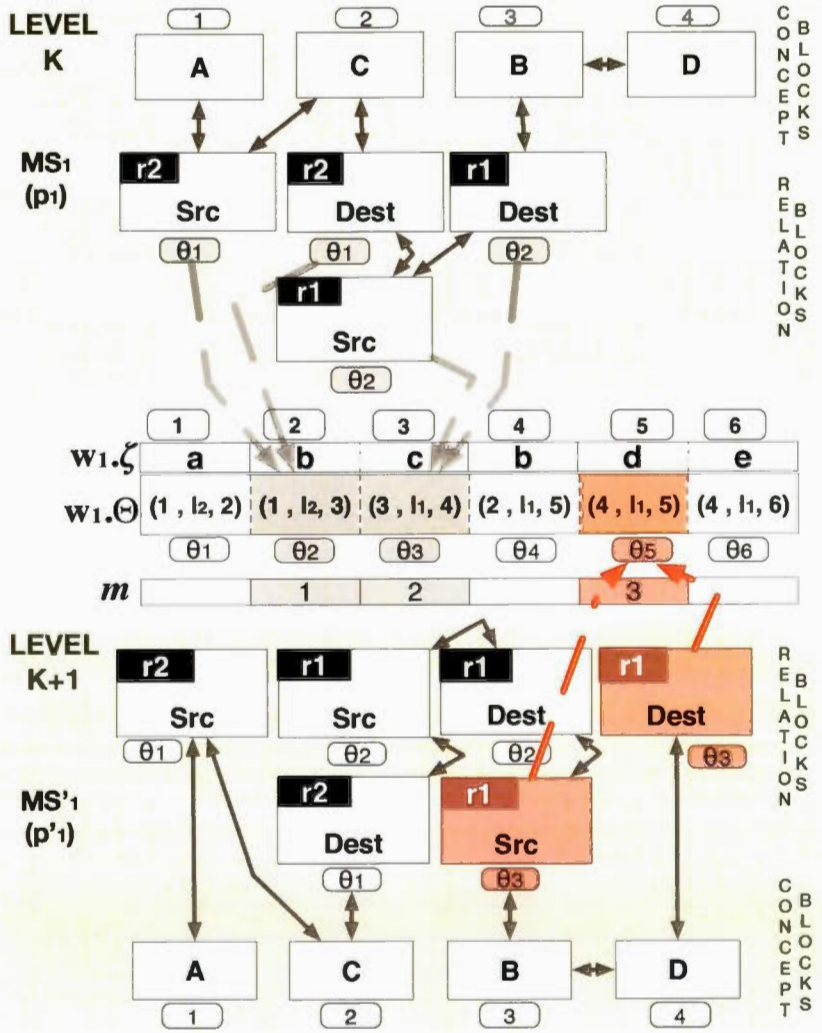


FIGURE 3.7: Example of adding a relation $addR(r_1, 3)$ in the matching structure of $MS'(p'_1)$ based on $MS(p_1)$. Rectangles in MS_1 and MS'_1 correspond to blocks showing the output of canonical operations. m stores matched triple positions between $w_1.\Theta$ and $p_1.\Theta$. For instance, the value 2 in the 3^{rd} cell of m represents the matching between $p_1.\Theta[2]$ and $w_1.\Theta[3]$. Grey rectangles in w_1 and m represent previously matched positions. Red rectangles represent the newly matched positions after adding the relation $addR(r_1, 3)$.

Algorithm 5 Verifying source matchings

```
1: procedure MATCHINGSOURCE(MS(p) ; domB ; i1 ; i2 ; r)
2:     result ← ∅ ;
3:     while w ∈ MS(p).listW() do
4:         notFound ← true ;
5:         m ← w.θ.getMatchedPos(MS(p)) ;
6:         i ← m.getLastPos(|p.Θ| + 1) ;
7:         if not (p.ζ(i1) ∈Ω w.Θ[i].dom()) then
8:             i − − ;
9:             while i > 0 and notFound do
10:                is_R ← r ∈Ω w.Θ[i].lnk() ;
11:                is_r ← p.ζ[i2] ∈Ω w.Θ[i].ran() ;
12:                is_d ← p.ζ[i1] ∈Ω w.Θ[i].dom() ;
13:                if is_R and is_r and is_d then
14:                    notFound ← false ;
15:                end if
16:                i − − ;
17:            end while
18:            update(wID, MS(p)ID, m) ;
19:        end if
20:    end while
21:    return result ;
22: end procedure
```

element in $w_1.\Theta$ and ends at the last found matching in m (lines 6–9). Next, we search in $w.\Theta$ the triple $\Theta[i]$ that matches in the *destination* i_2 and in the relation r. If $\Theta[i]$ is found, then, we add the new triple occurrence at the i^{th} position in m (line 15). Finally, we store the matching informations in the hash table *result* (line 18). For example, in *Fig.* 3.7, adding a relation r_1 between the 3^{rd} concept B

and the last one D amounts to find, at first, a matching to its *destination* element D in the workflow triple set $w.\Theta$. The search begins from the last triple θ_6 and it ends at the last matched triple in m which is θ_3. The first matching is found in θ_5 since the last element in p_1 matches with the 5^{th} element in $w_1.\zeta$ $(d \in_\Omega D)$ and $l_1 \in_\Omega r_1$. Hence, a new element 3 is added at the 5^{th} position of m representing a matching between $w_1.\Theta[5]$ and $p_1.\Theta[3]$.

The algorithm 5 describes how *source* solutions are found based on previous matchings from the last operation on the element of i_2. All potential workflows instantiating the pattern p are already stored in the hash table *result* from the previous matching validation on i_2. The search for the *source* matching of i_1 in a workflow starts from the last triple matching found in m (line 5). If this triple still matches in the *source* position i_1, then, a final solution is found and m will not be updated (line 7). However, if the triple does not match, then, we restart the search in $w.\Theta$ while reverifying all operation constraints until a new matching is found, then, we tentatively update m (line 16). For instance, in order to find the *source* matching the 3^{rd} concept (B) in the workflow w_1 (see Fig.3.7), we need at first to point to the last *destination* solution in m which is the the 5^{th} triple in $(w_1.\Theta)$. The search begins from this triple position until a possible solution is found. The next object instantiating B is the 4^{th} object (b) found also in $w_1.\Theta[5]$. The search stops here since a full matching is found. No matching update is required in this case.

3.9 Running example

Consider the table 3.2 introducing a set of three workflows, whereas the membership classes of each individual object and link are given in the phylogenetics ontology in *Fig.* 3.2. Assume, also, that the relative minimum of support is $\sigma = 2$.

Each $1-$frequent pattern is tentatively extended by the means of all the five canonical operations to produce the set of $2-$frequent patterns (see the second

Table 3.2: Example of Workflow sequences.

Object transactions.

ID	Objects
w_1	⟨ {KCC2D, sequence, alignment, tree}, {ClustalW}, {MrBayes} ⟩
w_2	⟨ {multipleSequences, alignments, trees}, {PhyML}, {PhyML} ⟩
w_3	⟨ {Myosin-7, multipleSequences, alignment}, {T-Coffee} ⟩

Link triple sets.

ID	Links
w_1	(2, isInput, 5), (3, isOutput, 5), (3, isInput, 6), (4, isOutput, 6)
w_2	(1, isInput, 5), (2, isOutput, 5), (2, isInput, 6), (3, isOutput, 6)
w_3	(2, isInputA, 4), (3, isOutputA, 5)

Table 3.3: Frequent patterns, their associated supports (Supp.) and corresponding workflow instantiation IDs (Inst.).

Frequent 1-patterns.

ID	Pattern	Supp.	Inst.
p_1	⟨ ⟨ {Data} ⟩, {} ⟩	3	1, 2, 3
p_2	⟨ ⟨ {Task} ⟩, {} ⟩	3	1, 2, 3

Frequent 2-patterns.

ID	Pattern	Supp.	Inst.
p_3	⟨ ⟨ {Data, Data} ⟩, {} ⟩	3	1, 2, 3
p_4	⟨ ⟨ {Data}, {Task} ⟩, {} ⟩	3	1, 2, 3
p_5	⟨ ⟨ {Task}, {Task} ⟩, {} ⟩	3	1, 2, 3
p_6	⟨ ⟨ {GeneOntology} ⟩, {} ⟩	2	1, 3
p_7	⟨ ⟨ {DataType} ⟩, {} ⟩	3	1, 2, 3
p_8	⟨ ⟨ {MSATask} ⟩, {} ⟩	3	1, 2, 3
p_9	⟨ ⟨ {InferenceTask} ⟩, {} ⟩	2	1, 2

sub-table in table 3.2). For instance, concept specialization on the last concept of p_2 : *i.e.* $subC('Task')$, produces the patterns p_8 and p_9. This latter is successfully instantiated by the first and the second workflow but it fails the matching with the third one, hence its support is 2. At the first step, generated patterns p_1 and p_2 have only one concept generated from the root concepts in Ω using the 'appC' operation on $\{\emptyset\}$ (see the first sub-table in table 3.3). The matching

Frequent 3-patterns.

ID	Pattern	Supp.	Inst.
p_{10}	⟨ ⟨ {Data, Data, Data} ⟩, {} ⟩	3	1, 2, 3
p_{11}	⟨ ⟨ {Data, Data}, {Task} ⟩, {} ⟩	3	1, 2, 3
p_{12}	⟨ ⟨ {Data}, {Task}, {Task} ⟩, {} ⟩	2	1, 2
p_{13}	⟨ ⟨ {MuscleContraction} ⟩, {} ⟩	2	1, 3
p_{14}	⟨ ⟨ {Alignment} ⟩, {} ⟩	3	1, 2, 3
p_{15}	⟨ ⟨ {Sequence} ⟩, {} ⟩	3	1, 2, 3
p_{16}	⟨ ⟨ {Tree} ⟩, {} ⟩	2	1, 2
p_{17}	⟨ ⟨ {progressiveMSA} ⟩, {} ⟩	2	1, 3
p_{18}	⟨ ⟨ {CharacterBased} ⟩, {} ⟩	2	1, 2
p_{19}	⟨ ⟨ {Data}, {Task} ⟩, {(1, isI/O, 2)} ⟩	3	1, 2, 3
p_{20}	⟨ ⟨ {Data, Sequence} ⟩, {} ⟩	3	1, 2, 3
p_{21}	⟨ ⟨ {Data, Alignment} ⟩, {} ⟩	3	1, 2, 3
p_{22}	⟨ ⟨ {Data, Tree} ⟩, {} ⟩	2	1, 2
p_{23}	⟨ ⟨ {Data}, {MSATask} ⟩, {} ⟩	3	1, 2, 3
p_{24}	⟨ ⟨ {Data}, {InferenceTask} ⟩, {} ⟩	2	1, 2
p_{25}	⟨ ⟨ {Task}, {InferenceTask} ⟩, {} ⟩	2	1, 2

procedure calculates, for each candidate pattern, the number of its workflow instantiations in the database. Here, both patterns matches with each workflow in the database, hence, its support is 3. For compactness reasons, we only show, here, frequent patterns for the first three steps and eliminate infrequent candidates. Generated sub-patterns from 2-frequent patterns are presented in the third sub-table of table 3.2. As an illustration, consider the extension of the pattern p_4. Extending this latter yields to generate the frequent patterns (1) p_{12} by using the $appC('Task')$ operation, (2) p_{19} by adding a new relation $addR('isI/O', 1)$

between 'Data' and 'Task' and finally (3) p_{23} and p_{24} by using the $subC('Task')$ operation.

3.10 Experimental study

At the preparatory step, we used a dedicated text miner (Halioui *et al.*, 2015) to extract phylogenetic workflows out of a corpus of 2 500+ articles from the PubMed PMC repository. We also used the PHAGE phylogenetic ontology published in Bioportal[12] with the following metrics : 114 concepts, 1 000+ objects (we ignore here vocabularies on species, genes and proteins) and 1 105 links of 14 different properties. Out of the discovered workflows, we composed two datasets : (1) *WFG*, contains 500 expert-validated workflows (12 objects and 6 links, on average), constitutes our ground truth, (2) *WFPub_1*, with 2 783 workflows (11 objects and 7 links), is our mining target.

In addition, and in order to study the scalability of our system, we formed another two datasets, *WFPub_2* and *WFPub_3* of different sizes (Halioui *et al.*, 2015). These are described in Table 3.4 which provides metrics such as the number of workflows as well as the average workflow size in terms of objects and links. In addition, we created two other categories of datasets with different object and link sizes, based on a simulation on the top of the domain ontology. In the first group, *WFSimO_X*, we kept the average number of objects in the workflow under control whereas in the second one, *WFSimL_X*, we did the same for the links. In generating these datasets, we put a maximal effort on plausibility : on the general knowledge from the processual ontology, we used 15 top-level workflow patterns provided by the aforementioned expert (*e.g.*, $p_E = \langle\{Data\ Collection\}, \{Sequence\ Alignment\}, \{Phylogenetic\ Inference\}\rangle$). Hence, these patterns are used

[12] http://bioportal.bioontology.org/ontologies/PHAGE

Table 3.4: Workflow datasets.

	#wf	avg. #obj.	avg. #links
WFPub_1	2 783	11	7
WFPub_2	631	12	6
WFPub_3	100	11	5
WFG	500	12	6
WFSimO_05	100	5	11
WFSimO_15	100	15	39
WFSimO_25	100	25	88
WFSimL_05	100	5	5
WFSimL_15	100	10	15
WFSimL_25	100	12	25
WFSimL_200	100	34	200

The **#wf** column presents the total number of concrete workflows in each database. The numbers in **avg. #obj.** and **avg. #links** columns are the average number of objects and the average number of links, per workflow in the database, respectively.

to generate randomly workflows as multisets of objects and triple sets .

3.10.1 Sample mining output

Fig. 3.8 presents a sample of patterns extracted from the intersection of the *WFPub_1* dataset (automatically extracted workflows) and the expert-curated one *WFG* starting from the third level of the ontology. The first pattern P_1 is ~60%

frequent in the database. It represents *Bayesian phylogenetic tree reconstruction* from *aligned* set of *proteins*. The second pattern P_2 (support of ~30%) also represents a *protein*-related workflow. At first, *sequences* are *aligned* with a *multiple sequence alignment program*, then, the result is used in a *character based method* from a *general purpose packages program* to infer *phylogenetic trees*. The latter is further used in order to *validate* trees using *more than 1 000 bootstrap replicates*.

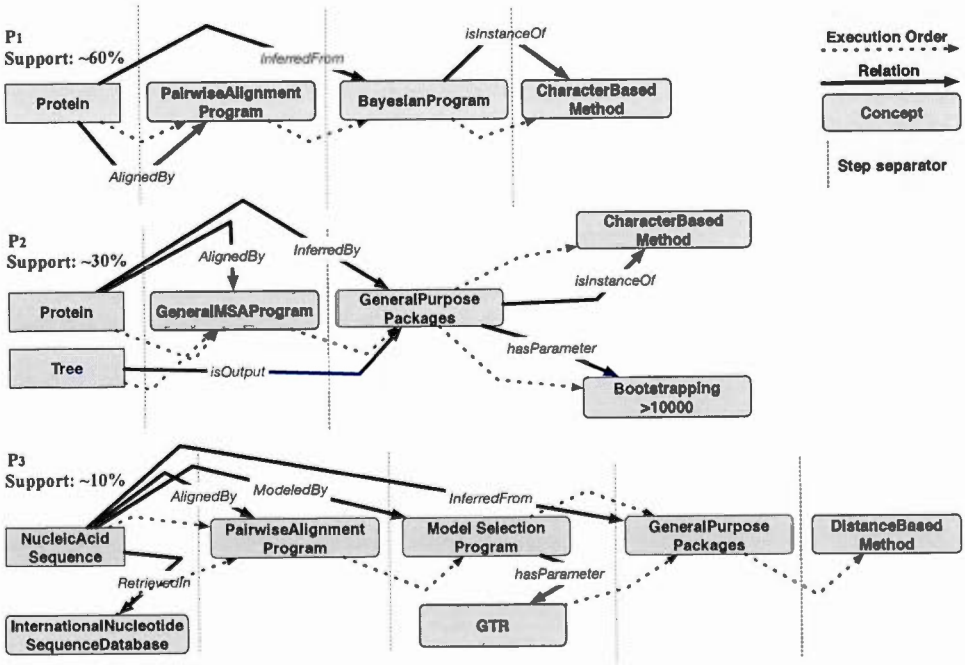


FIGURE 3.8: Example of generated patterns.

The third pattern P_3 is even less frequent, yet, it comprises more specific concepts and relations. It is about *DNA* and *RNA* sequences extracted from an *international nucleotide sequence database* using a *pairwise alignment program*. The output *alignments* were used in a *model selection program* with a *GTR* (Yang, 1994) as a model. Then a *phylogenetic tree* is inferred using a *multiple-purpose packages program* based on the *distance-based* approach.

3.10.2 Scale-up study

Fig. 3.9 (A) shows the number of generated patterns while increasing values of σ over 3 workflow databases diverging on their size. The curve shapes indicate that dataset size is not a universally decisive factor for the size of the output. However, unsurprisingly, decreasing σ leads to larger pattern sets.

Fig. 3.9 (B), illustrates the evolution of total time spent on matching upon increase in the ontology level (with a constant $\sigma = 0.7$). Unexpectedly, we found a pick in the matching time at level one (except in WDPub_1 database). At this point, since patterns are pre-filtered lately on roots, an important number of matching combinations might occur from level 0 to level 1. Hence, the matching cost at level one is superior to the matching cost starting from the most general concepts.

Fig. 3.9 (C) and (D) show the impact of increasing workflow size on the total matching time. In both cases, as the number of objects or links increases, the number of matching operations increases too (more so in the case of links, up to an exponential function). This is expected since more matching operations are done.

Fig. 3.9 (E) and (F) show the mean running time over increasing prefixes of the *WFPub_1* and *WFPub_3* datasets (increments of ca. 10%). The overall shape of the curves indicates that there is a clear correlation between the number of workflows and the time spent to generate the underlying sets. In addition, pruning already *covered patterns* yields to decrease significantly the number of generated patterns (~15 orders of magnitude). Even if our method of pruning is not monotone, with such rapidity (processing of ~3K workflows in 45 seconds), the overall cost remains reasonable.

In the next experiment, we compared our method to `xPMiner` (Adda *et al.*, 2010)

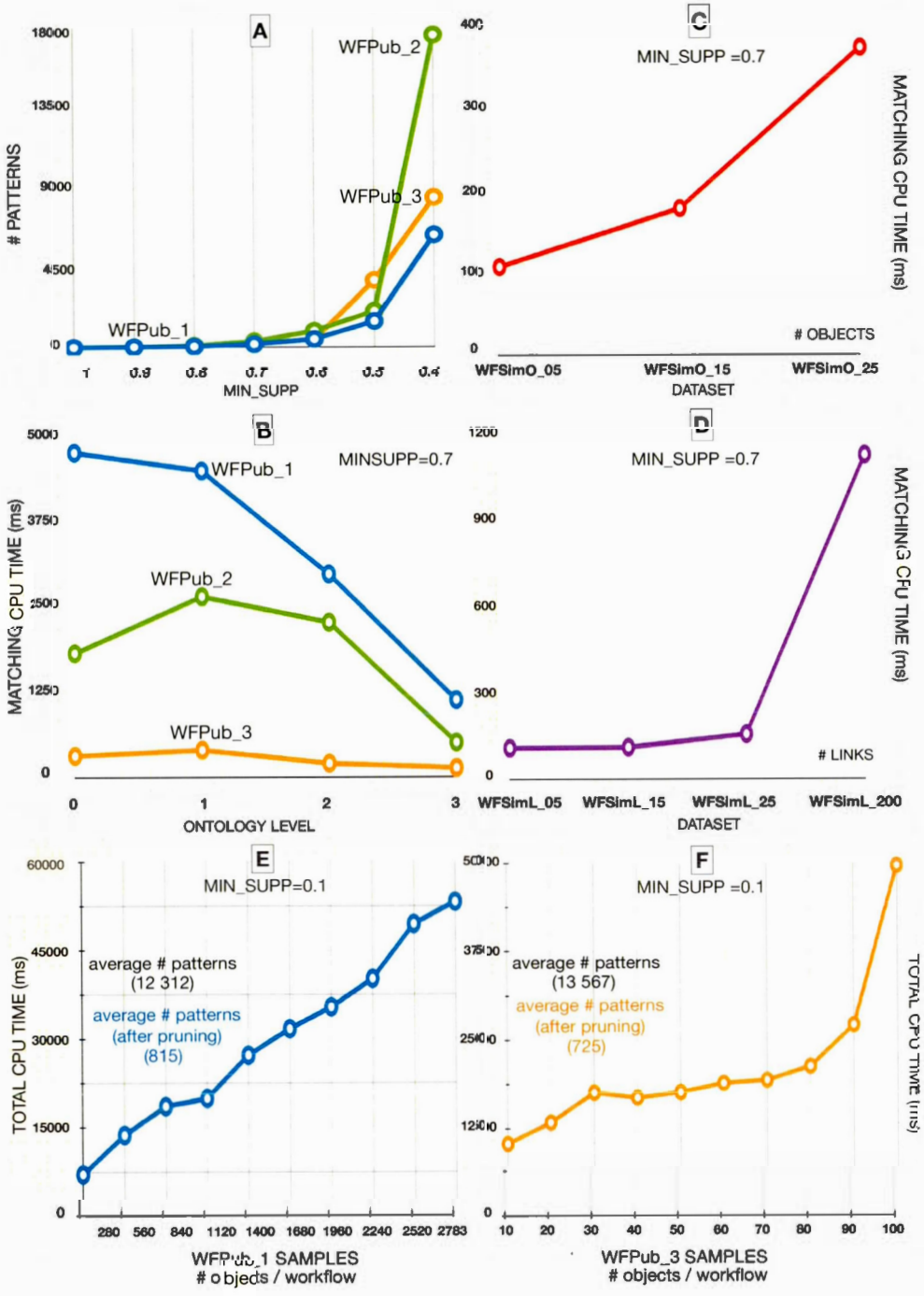


FIGURE 3.9: Scalability experiments.

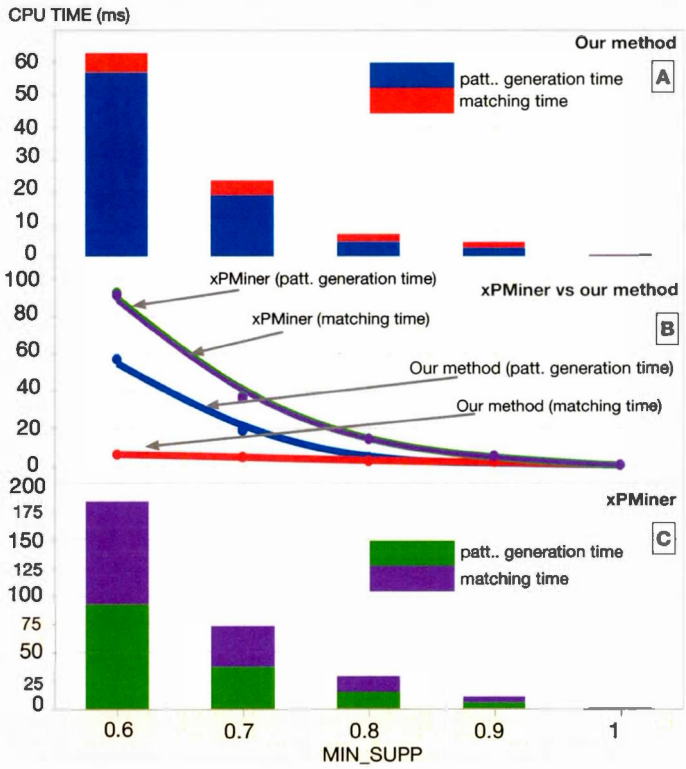


FIGURE 3.10: *Our method* vs **xPMiner** using the WFPub_3 dataset.

while running on the *WFG* dataset (expert-curated). Even if that **xPMiner** does not handle workflows, we managed to transform our input into sequences while "ignoring" the partial order of steps. We compared the running time executions for the entire mining task as well as the proportion spent on matching for a range of values of σ. The outcome is depicted in the *Fig.* 3.10. The top diagram shows that our algorithm spends only a tiny proportion on matching for lower values of σ (larger pattern sets). For example, with $\sigma = 0.6$, only $\sim$10% of the total time goes to the matching task. However, in **xPMiner** (bottom diagram), the matching takes $\sim$50% of the total time, a ratio of our method reaches with σ above 0.9. In addition, over the entire mining process our method steadily outperforms **xPMiner** by a factor of three even with much complex representations (workflows) instead

of sequences in `xPMiner`.

3.11 Conclusion

In this paper, we proposed a generalized pattern mining method as a way to automate the acquisition of problem solving experience (in terms of workflows) that effectively exploits a domain ontology. The major challenges we faced, here, were the design of an (1) expressive, yet tractable pattern languages and (2) efficient mining methods preserving a total order between workflow items in order to guarantee that all patterns are generated exactly once. To that end, we described the pattern space (data and pattern languages, instantiation and generality relations) and a refinement operator to move downwards in the pattern space. Finally, we designed a depth-first mining method to generate non redundant patterns (patterns covered by the same set of workflows) exploiting the generalization relationship to reduce the matching computations. Our method was validated on different simulated and expert-curated datasets of phylogenetic workflows retrieved from the domain literature. A challenging research avenue to look at would be the design of a domain-specific *interestingness* criteria –instead of support–, *e.g.*, exploiting the ontological knowledge to measure the "semantic" similarity between abstract patterns and concrete workflows.

Acknowledgment

This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D.

CHAPITRE IV

UN SYSTÈME D'ACQUISITION DE FLUX DE TRAVAUX ABSTRAITS

Le travail de ce chapitre a été accepté et est en cours de publication dans le journal Journal of Computational Biology (JCB), 2017, sous le titre "T-Gowler : Discovering Generalized Process Models within Texts" (Halioui *et al.*, 2017b). Nous présentons la version imprimée de l'article dans l'annexe *C*. Nous notons que la version courte du papier a été publiée dans la conférence IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), 2015, sous le titre "Acquisition of generic problem solving knowledge through information extraction and pattern mining" (Halioui *et al.*, 2015).

Ce chapitre consiste à introduire l'architecture de notre système `T-Gowler` dans ses deux modules : (1) extraction de flux de travaux concrets à partir des textes et (2) fouille de motifs généralisés. Dans cet article, nous reprenons les deux contributions dans leur cadre systémique afin de présenter l'architecture complète de notre plateforme d'acquisition de flux de travaux abstraits.

La rédaction de l'article, les expérimentations et le montage des figures ont été tous réalisés par Ahmed Halioui sous la supervision des Prof. Abdoulaye Baniré Diallo et Petko Valtchev.

4.1 Résumé

Le problème d'acquisition de connaissances génériques de résolution de problèmes présente des limites bien que d'énormes quantités de données processuelles concrètes existent dans la littérature du domaine. L'apprentissage automatique des cas de résolution de problèmes est confronté à plusieurs défis majeurs, à savoir, la complexité du domaine d'application (en concepts, relations, événements, processus, *etc.*) et la "lisibilité" par des machines automatiques (*machine-readability*). Nous étudions dans ce papier le problème d'acquisition d'expertise dans le domaine de l'analyse phylogénétique qui présente, dans sa littérature de textes, des sources d'information riches en connaissances processuelles et factuelles. En effet, nous proposons le système `T-Gowler` qui offre une solution d'extraction de connaissances processuelles en utilisant des outils d'analyse de textes et de fouille de motifs basée sur une ontologie. `T-Gowler` génère des abstractions de motifs généralisés à partir des flux de travaux concrets extraits du texte. Ainsi, notre système comporte deux modules (1) un extracteur de flux de travaux et (2) un moteur de fouille de motifs processuels. Ces deux modules sont basés sur deux types d'ontologies : une ontologie processuelle et une deuxième factuelle (du domaine).

4.2 Abstract

Contemporary workflow management systems are driven by explicit process models specifying the inter-dependencies between tasks. Creating these models is a challenging and time-consuming task. Existing approaches to mining concrete workflows into models tackle design aspects related to the diverging abstraction levels of the tasks. Concrete workflow logs represent tasks and cases of concrete events –partially or totally ordered– grounding hidden multi-level (abstract) semantics and contexts. Relevant generalized events could be rediscovered within these processes. We propose in this paper an ontology-based workflow mining system to generate patterns from sequences of events which are themselves extracted from texts. Our system `T-Gowler` (Generalized Ontology-based WorkfLow minER within Texts) is based on two ontology-based modules : a workflow extractor and a pattern miner. To this end, it uses two different ontologies : a domain one (to support workflow extraction from texts) and a processual one (to mine generalized patterns from extracted workflows). `T-Gowler` implementation and data sources are available in : `https://github.com/bioinfoUQAM/tgowler`.

4.3 Introduction

Workflows (Taylor *et al.*, 2014) are increasingly designed as an efficient way of managing complex multi-step processes for the resolution of complex problems from knowledge-rich domains such as medicine, biology, chemistry and bioinformatics. This conducts to a growing popularity of workflow platforms offering a wide spectrum of analytical services (Wolstencroft *et al.*, 2013; Blankenberg *et al.*, 2001; Lord *et al.*, 2012). However, in order to build successful workflows, the user must typically master a large variety of methods, tools, data formats, *etc.* In many cases, the ultimate success of the workflows depends on the right choice of the included methods, software parameters and versions, *etc.* In fact, such a demanding environment reduces the value of Workflow Management Systems (WMS) for non expert users. Facing these challenges, *Process Mining* (Ellis *et al.*, 2006; Van der Aalst, 2016), a branch of Knowledge Discovery from Data (KDD) (Fayyad *et al.*, 1996), studies the discovery of reusable knowledge from databases of process records, in particular, from workflow representations such as execution logs. As workflows have an inherent graph structure, effective process mining methods face the related complexity issues. A workflow graph depicts an analytical resolution in terms of blocks and their interactions (control/data flows) drawn as nodes and links, respectively. Yet, given the large spectrum of possible inputs, it is unlikely for a recorded workflow to fully fit a new situation. A solution could be to mine higher-level of abstractions summarizing commonalities of a set of workflows. Representing abstraction levels using ontologies to assist mining of usage logs has proven popular in the last decade (Dai et Mobasher, 2002; Adda *et al.*, 2007; Senkul et Salin, 2012). A complete framework for abstracting ontology-based patterns out of click-streams (from the logs of a semantic Web portal powered by a domain ontology) provides an alternative view (Adda *et al.*, 2010). Yet, here, we deal with a comparable problem on a different type of input format. Instead

of pure sequences, we consider workflows as DAGs. However, due to the complex nature[1] of the underlying domain problem, the DAGs are split into ordered layers. In the next section, we define first, the different elements of the ontology, followed by two descriptive languages : Data and Pattern languages. We also introduce the pattern space, the instantiation relationship between workflows and patterns, and the generality order between patterns. Finally, we describe the pattern space and the refinement operators to generate patterns at each level from that space. Generated pattern are then, exploited by a *Top-Down Depth-First* traversal method using the *support* metric as a quality criterion for the generated patterns.

Since we already described the technical details about the proposed methods to extract and mine generalized workflow patterns from texts, we only present. here, a brief summary of the proposed methods (see chapters 2 and 3 for more details) and goes quickly through the empirical results of our system in the Phylogenetics domain.

4.4 T-GOWler : ontology-based workflow pattern acquisition from texts

We describe, in this section, the pipeline of the methods to extract abstract patterns within texts. Our `T-Gowler` system contains two modules : (A) `WfExtractor` extracts workflows W from texts and builds a workflow database $\mathcal{D}_W$ and (B) `WfMiner` mines frequent generalized patterns $\mathcal{F}$ from $\mathcal{D}_W$ (see *Fig.*4.1). The first module (developed on GATE platform[2] by Cunningham *et al.* (2013)) requires

[1]The definition of the domain concepts and constraints is a complex task itself.

[2]GATE (`www.gate.ac.uk`) is a well-known environment for annotation tasks using different NLP (Natural Language Processing) techniques such as OpenNLP parser and Stanford POS tagger.

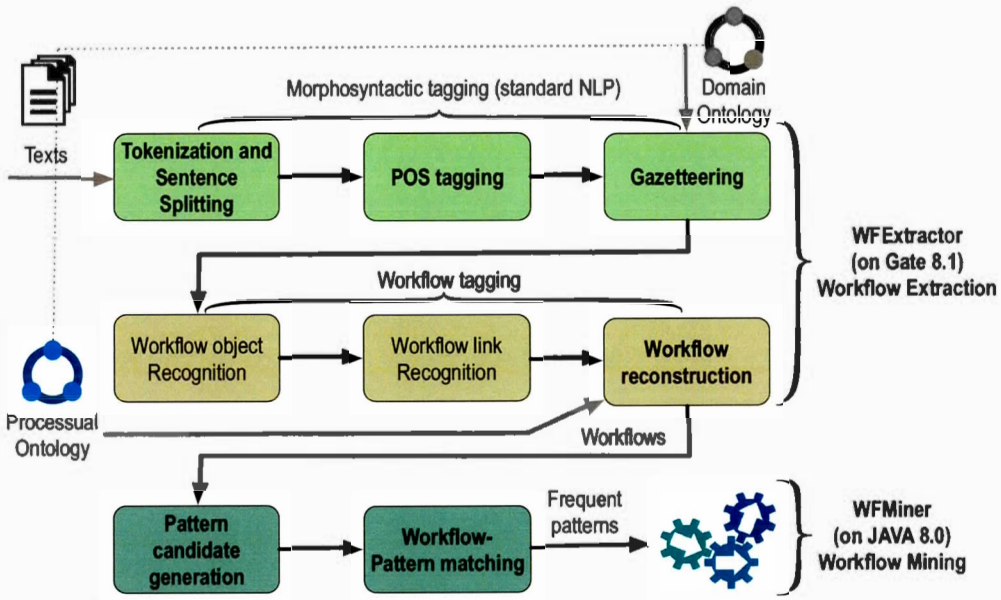


FIGURE 4.1: `T-Gowler` system architecture.

plain text articles and an RDF[3] domain ontology representing the domain terminology (*Gazetteer*) categorizing, into abstract concepts and relations, domain terms. The extraction pipeline is based on a simple NLP chain and a specific workflow tagger to classify workflow components, *i.e.* objects O and links $\mathcal{L}$, into their ontological categories, *i.e.* concepts C and relations $\mathcal{R}$, respectively. In order to preserve processes order in workflows, abstract patterns from the processual ontology are used. Once the ontology is loaded and concrete workflows are reconstructed, the `WfMiner` tool deals with the generation of frequent patterns using a level-wise pattern generation on the top of the ontology and a specific pattern-to-workflow matching algorithm to filter non frequent patterns.

[3] `https://www.w3.org/TR/rdf-primer`

4.4.1 WfExtractor : workflow extraction from texts

The extraction process takes place 3 different steps : (1) named entity ($o \in O$) extraction, (2) link ($l \in \mathcal{L}$) extraction and (3) workflow $w \in \mathcal{D}_W$ reconstruction from the extracted elements. Yet, before workflow component extraction, a classic NLP tool chain is executed in order to tag terms with their morphosyntactic tags, *i.e.* : grammatical classes within sentences. This chain is composed from : (1) the specialized biomedical tokenizer (Settles, 2004) and a simple sentence splitter (Bontcheva *et al.*, 2013), (2) the MedPost/SKR POS tagger (Smith *et al.*, 2004) and (3) the lightweight OpenNLP chunker (Buyko *et al.*, 2006) to identify active and passive verb phrases in texts.

For both named entity and link extraction, a domain ontology is applied as a Gazetteer in order to recognize workflow elements in texts. However, this process faces some hurdles due to the syntactic and semantic ambiguities, *i.e. polysemy* (Stevenson et Wilks, 2003) in some domain concepts. For instance, the program term *MEGA* (Kumar *et al.*, 2016) could be used to either validate the phylogenetics hypothesis or to visualize output data, *e.g.* phylogenetic trees. Thereof, we propose to use a supervised machine learning approach (Nadeau et Sekine, 2007) to disambiguate software contexts using specific JAPE rules[4]. For example, when the passive verb phrase (context) contains a 'program' term in its object and the root verb is a synonym to "valid" (from the *Hypothesis Validation* concept), then, its probability to be a validation program is very high. Hence, for each polysemic term in the texts, we create two rules : one for the active and the other for the

[4]Java Annotation Patterns Engine (JAPE). JAPE language is a finite state transducer on the GATE platform (Cunningham *et al.*, 2013) that operates over annotations based on regular expressions. Thus, it is useful for pattern-matching, semantic extraction, and many other operations over syntactic trees such as those produced by natural language parsers.

passive voice phrases. In such way, a learning set is constructed. Next, two sets of features are calculated to annotate ambiguous terms automatically, *i.e.* : (1) POS annotations and (2) surrounded concepts (domain concepts in the context sentence of the ambiguous word) in a window of the five next and previous concepts and tokens. In order to represent our context-based extracted features, the PAUM (Perceptron Algorithm with Uneven Margins) model (Li *et al.*, 2002) is applied to learn Word Sense Disambiguation (WSD) (Stevenson et Wilks, 2003) and relation classifications from the extracted feature instances. PAUM is specifically designed to simulate the distance between positive and negative examples which fits the nature of our features since positive and negative contexts are represented with distance-based features (in POS tags and concepts in the same sentence).

For the relation side, we adopt the following hypothesis : A link between two related concepts exists in the same sentence evoking the instances of its domain (source term) and range (destination term). Such link is represented in the verb located between the concerned terms (depending on the sentence voice). Hence, in order to recognize links in texts, we map the couple c_1, c_2, from the triple $\rho = (c_1, r, c_2)$ in the ontology, on text sentences. For example, for the relation instances of r : $isRetrievedIn$, defined with c_1 : $DataType$ as the relation domain, and c_2 : $Database$ as the relation range, we tag the context of the link with its relation type, *e.g. alined in* in the text is tagged with the $isRetrievedIn$ relationship. A PAUM model is then constructed using the following set of features : (1) token POS tags, (2) distance, (3) direction and (3) the list of verbs between domain and range instances in the context phrase (see table 2.1).

Finally, from the recognized links between named entities in texts, we build a concrete workflow knowledge database $\mathcal{D}_W$ where items are named entities ordered by the *genInputTo* properties (relations) from the processual ontology. For example, from the tuples : (ρ_1) (*16srRNA*, "*isInputOf*", *ClustalW*), (ρ_2) (*Clus-*

talW, "*genInputTo*", *PHYLIP*) and (ρ_3) (*PHYLIP*, "*genInputTo*", *TreeView*), we construct the sequence $\langle$ { {*16srRNA*}, {*ClustalW*}, {*PHYLIP*}, {*TreeView*} }, { (1, "*isInputOf*", 2) } $\rangle$ where 1 and 2 correspond the domain and range positions, respectively, in the workflow sequence.

4.4.2 WfMiner : mining generalized workflows

Here, we propose an *Depth-First* approach to generate generalized sub-graphs of workflows out of concrete ones. We propose two descriptive languages to represent the pattern space. The data language Δ_w is derived from sequences of objects and sets of link IDs from the processual ontology Ω. For instance, the workflow DAG w_1 in *Fig.* 4.2 is a poset $w.\zeta$ of object items $i \in \{1,..7\}$ and a set of link triples $w.\theta$ of with link types $l \in \{1,..6\}$. The second language Γ_w is used to represent a pattern p as an abstraction of $w \in \mathcal{D}_W$ (*e.g.*, $w_1 \lhd p_1 : w_1$ instantiates p_1). Hence, we represent a pattern p as a sequence of concept itemsets $p.\zeta$ and a set of relation triples $p.\theta$.

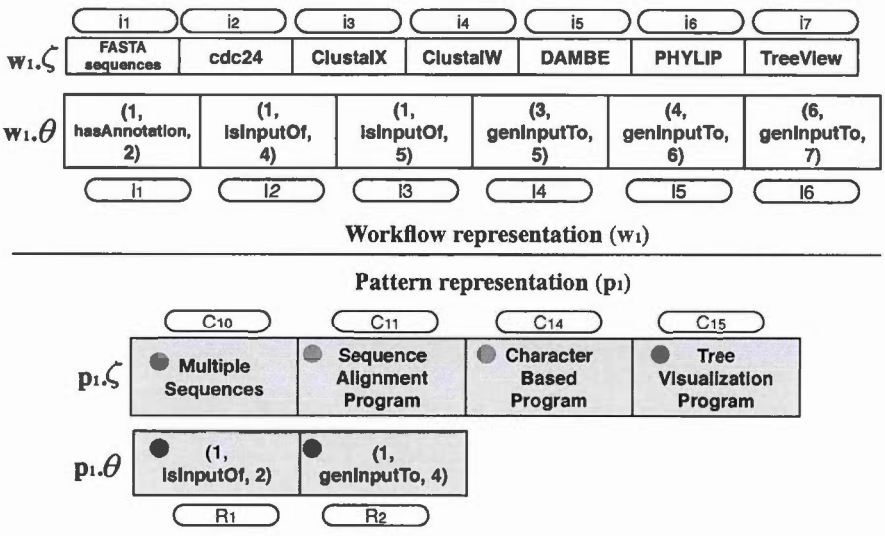


FIGURE 4.2: Workflow and pattern representations.

Level-wise descent in the pattern space relies on the generation of $k+1$-level patterns from k-level ones, which amounts to the computation of five canonical operations in a Depth-First search strategy from the ontology : (1) appC - appends a root concept to a new transaction, (2) addC - adds a root concept to the last transaction, (3) addR - adds a root relation between a concept and the last one in the last transaction, (4) subC - specializes the last concept and (5) subR - specializes the last relation in the pattern. Hence, for each pattern p, we apply all the five canonical operations. Although these operations constitute an effective generation mechanism. Their unrestricted use is redundancy-prone and hence, would harm the efficiency of the global mining process. In the previous work of Adda *et al.* (2007), calculating the support of a pattern p amounts to match p with all the workflow concrete sequences in $\mathcal{D}_W$. For this purpose, we introduce a new data structure to prune patterns while calculating their supports using the generality relationship between them. Calculating frequent patterns p^{k+1} begins from the last visited concept/relation in the parent pattern p^k and hence, we don't need to recalculate all the matching possibilities for each concept and relation previously matched at level k. The proposed matching structure MS is a "stack" of operation results (adding/specializing concepts/relations) on parent patterns. For example, the arrows in *Fig.* 4.3 show the way we obtain a pattern p_2 using the refinement operator. Calculating the support of p_2 is done starting from the last operation on p_1, *i.e.* addR(r_2, c_{11}). If the next operation in the $k+1^{th}$ iteration is to add another relation addR(r_3, c_{14}) to p_2, then, the matching structure of p_2 ($MS(p_2)$) puts relation blocks ('*source*' and '*destination*' relation blocks s) in their proper positions and calculates the possible matchings from the matching set of the parent pattern $MS(p_1)$.

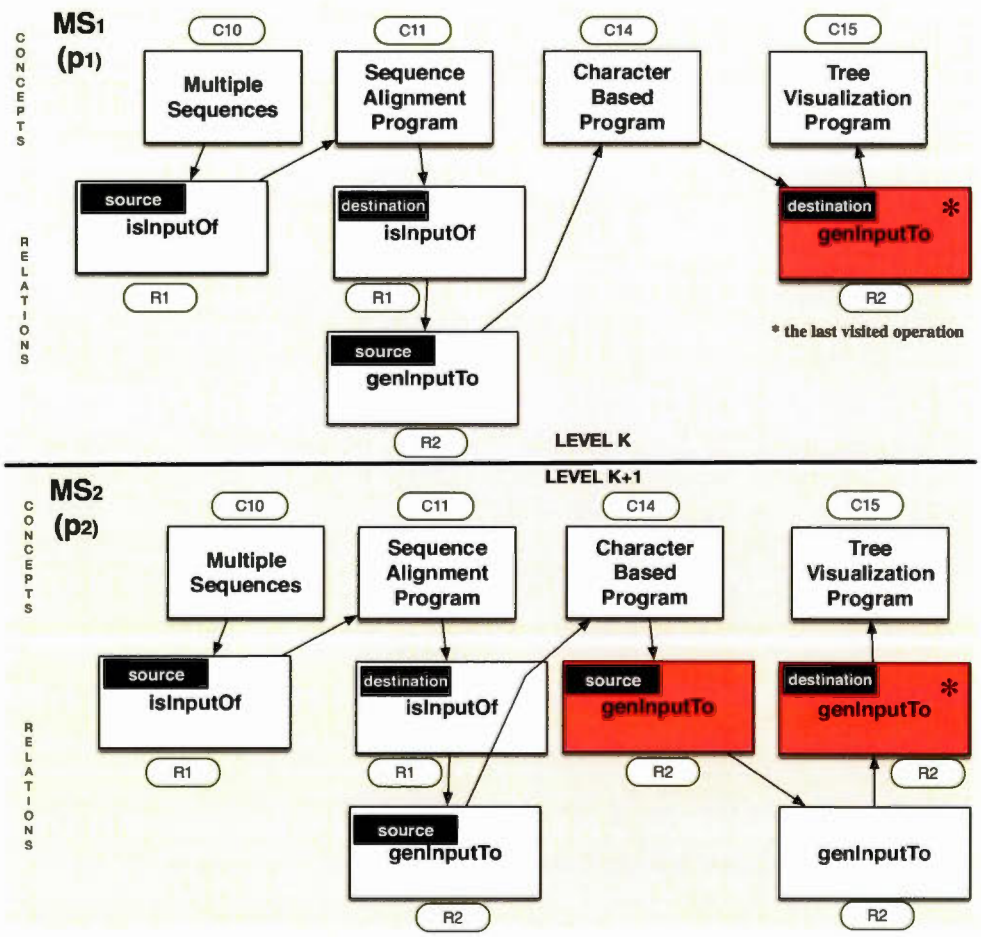


FIGURE 4.3: Pattern matching structures over levels k and $k+1$.

4.5 Experiments and results

4.5.1 Case study : phylogenetics analyses

Phylogeny (Lemey Philippe, 2009) represents the evolutionary history of a group of organisms having a common ancestor. Inferred from nucleic acid (DNA or RNA) or protein sequences, phylogenetic relations are discovered with tools detecting different evolutionary phenomena such as duplications, insertions and deletions (see Anisimova *et al.*, 2013).

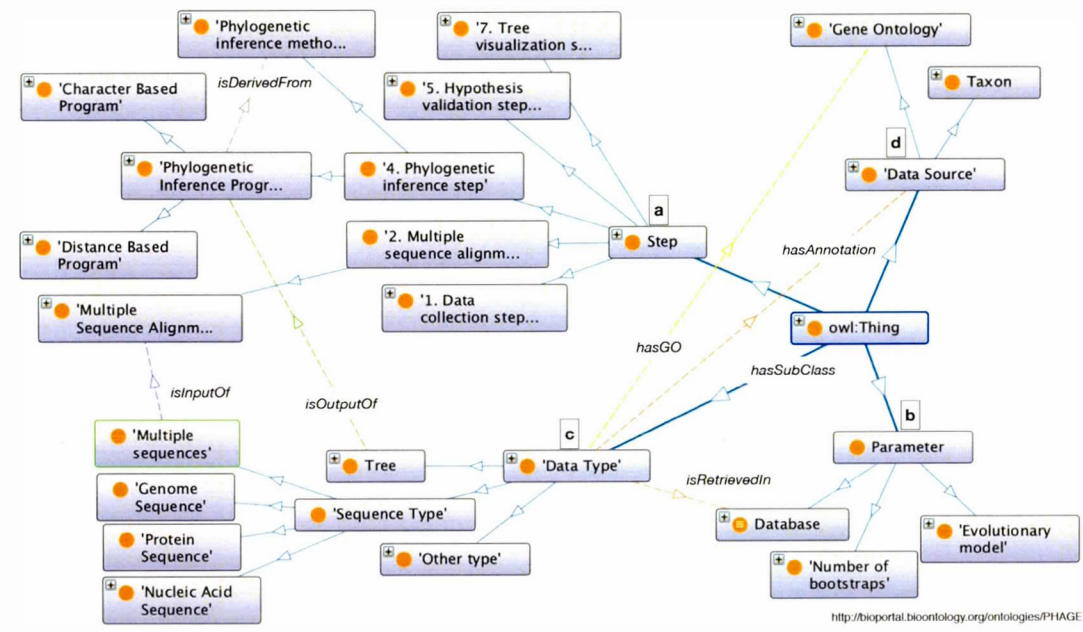


FIGURE 4.4: A portion from the phylogenetic PHAGE ontology (which merges domain and processual ontologies). Step concepts (a) represent processual tasks (activities). Parameter concepts (b) are program parameters from the step subhierarchy. Data type concepts (c) are input/output data and data sources (d) are annotations on data (provenance). PHAGE is composed of 37 different object properties (relations), 47 031 concepts (with 4 root concepts and 111 unique concepts) and more than 1 000 000 unique individuals (instances) without counting synonyms.

We downloaded more than 3 000 scientific articles (published from 2013 until April 2015) about phylogenetics studies from the PubMed Central (PMC) repository[5] in XHTML format. In addition, we constructed a terminology from different well-known databases in the phylogenetics literature such as the UniprotKB[6] (for gene

[5] http://www.ncbi.nlm.nih.gov/pmc

[6] http://www.uniprot.org

and protein names), Gene Ontology (Ashburner *et al.*, 2000), Taxonomy Browser[7] and the Felsenstein Website [8]. This gazetteer is then, reorganized in a domain ontology in order to recognize terms and relations in texts like described in the `WfExtractor` module. Next, we build manually a processual ontology with the help of experts. An example of an abstract processual knowledge is : (1) *Protein collected from the PDB database*[9] *are fed to* (2) *a homology search program and* (3) *a phylogenetic inference one.* Finally, both domain and processual ontologies are merged into one[10] (see Fig. 4.4) in order to mine frequent patterns from the workflow database using the `WfMiner` engine.

We present, next, the obtained results from the workflow extraction and the pattern mining modules.

4.5.2 Named entity learning results

An expert has annotated manually 300 random phylogenetic sections to build a *Ground Truth* workflow database that we call here *PhyloFlows.* Extracted terms are then, compared to the PAUM model results in order to validate their accuracies (with a 10-fold cross-validation) in terms of strict Precision, Recall and F-Measure (Salton et Buckley, 1988) (see table 4.1). More details about the validation strategy is described in the previous section 2.5.5.

Model context features have well learned and applied several top-level domain relation and concept terms, *i.e.* : data types, gene ontology annotations, phylo-

---

[7] `http://www.ncbi.nlm.nih.gov/Taxonomy`

[8] `http://evolution.genetics.washington.edu/programs.html`

[9] `http://www.rcsb.org/pdb`

[10] `http://bioportal.bioontology.org/ontologies/PHAGE`

Table 4.1: Evaluation results of the extracted workflow elements based on the expert curated dataset from the PubMed PMC articles (published from January 2013 to April 2015).

Concept/Relation	# Instances	Precision	Recall	F-Measure
DataType	597	0.68	0.75	0.71
GeneOntology	139	0.94	0.44	0.60
Taxon	202	0.72	0.58	0.64
Method	326	0.84	0.84	0.84
InferenceProgram	205	0.56	0.88	0.68
Overall (concepts)	1 469	0.74	0.69	0.71
Overall (relations)	325	0.72	0.84	0.78
Overall	**1 794**	**0.74**	**0.72**	**0.73**

The table shows the number of annotated instances of (some from the top level of abstraction) concepts and relations and their Precision, Recall and F-Measure evaluation values.

genetic methods and phylogenetic inference programs. In total, 1 469 workflow objects have been extracted from the phylogenetic analyses sections with an average of 71% of F-Measure. In the relation side, the system annotated 325 links with 72% of precision and 84% of recall (78% of F-Measure). Hence, workflow overall F-Measure is 73%. Indeed, with only context POS tags and concept neighbours, we did recognize workflow components in texts with good accuracies.

4.5.3 Generalized-pattern results

For the rest of our experiments, we consider only expert curated datasets (that we call here *PhyloFlows*) in order to assess the `WfMiner` module. This database is composed from 300 workflows with an average of 11 object items and 9 links

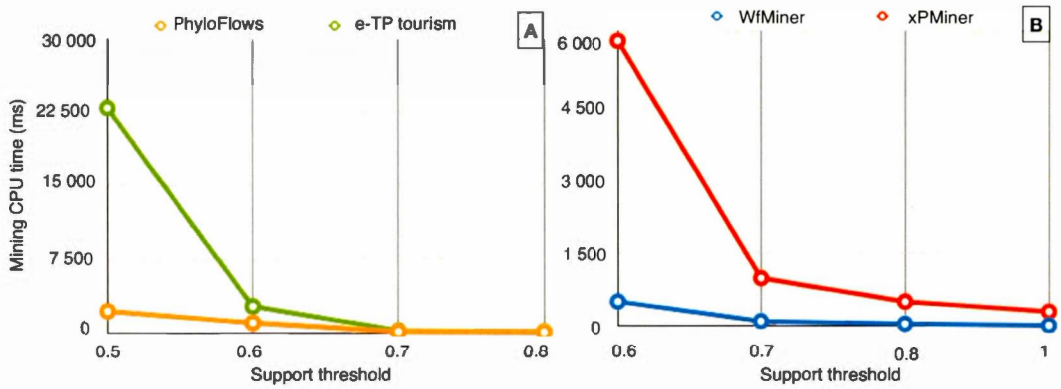


FIGURE 4.5: (A) Varying support for the *PhyloFlows* and *e-TP* tourism databases. (B) Comparing the mining running time of `WfMiner` and `xPMiner`.

per workflow. In addition, we use the *eTP-tourism* (Electronic Tourism Platform) workflow database which is a portion of a log file describing Web pages navigation of 56 different users in a Web travel site (Adda *et al.*, 2007). This database has an average of 19 different items and 6 links per sequence and enhanced by an ontology of 157 concepts and 35 object properties.

Fig. 4.5 (A) shows the difference in mining time between the *PhyloFlows* and *eTP-tourism* databases. Running time does eventually increases when more concepts and links are mined. Our system spends ~10 times the mining time to generate patterns with more 500 items and 100 links. While *eTP-tourism* database presents more dense graphs, matching sub-graphs and mining processes become heavier to handle. However, taking 23 seconds to do the whole process of matching and generating ~6 000 frequent graph patterns is still a reasonable time. In addition, our `WfMiner` outperforms `xPMiner` by Adda *et al.* (2007) (see *Fig.* 4.5 (B)) with 2 times order of magnitude in high threshold supports (>0.8). With 60% of minimum support, our approach executes ~10 times faster the mining algorithm than `xPMiner`.

Table 4.2: A sample from the phylogenetic extracted patterns within the texts in the PubMed PMC database (articles published from January 2013 to April 2015).

Workflow patterns		σ
p.ζ (concepts)	**p.Θ (relations)**	
$\langle\{DataType\}$, $\{DataCollection\}$, $\{MultipleSequenceAlignment\}$, $\{PhylogeneticInference\}\rangle$	(1, *isInputOf*, 2), (1, *isOutputOf*, 4), (2, *genInputTo*, 3), (3, *genInputTo*, 4)	40%
$\langle\{RNA$, *Tree*$\}$, $\{HomologySearch\}$, $\{CharacterBasedProgram\}$, $\{TreePlottingProgram\}$, $\{GTR\}\rangle$	(1, *isInputOf*, 2), (2 ,*isOutputOf*, 5), (3, *genInputTo*, 4), (3, *hasParameter*, 6), (4, *genInputTo*, 5)	10%
$\langle\{ProteinSequence$, *Alignments*, *Trees*$\}$, $\{Virus\}$, $\{GlobalMSA\}$, $\{ModelSelection\}$, $\{MaximumLikelihood\}$, $\{Bootstrapping\}$, $\{PDB, JTT\}\rangle$	(1, *hasTaxon*, 4), (1, *isInputOf*, 5), (2 , *isOutputOf*, 5), (2 , *isInputOf*, 6), (5, *genInputTo*, 6), (5, *hasParameter* , 9), (6, *genInputTo*, 7), (6, *hasParameter* , 10), (7, *genInputTo*, 8)	1%

Generated DAG patterns are sequences of concept multisets and sets of relation triples. For instance, in the 1^{st} pattern : (1, *isInputOf*, 2) represents the input relation between the 1^{st} concept (*DataType*) and the 2^{nd} one (*DataCollection*) in $p.\zeta$.

Finally, we present in table 4.2 a sample from the generated set of patterns obtained from the *PhyloFlows* database. These patterns could be used in a semantically rich recommendation framework considering the diversity of levels of abstraction in pattern representations. For instance, if we know that the $16s\ rRNA$ gene is used as input in the *ClustalW* alignment program (Thompson *et al.*, 2002), then a concrete recommendation could offer for example *MrBayes* (Ronquist et Huelsenbeck, 2003) as a phylogenetic inference tool (~1% frequent). However, offering "any" character based Inference program (see 2^{nd} pattern in table 4.2) is a more relevant recommendation due to its (pattern) higher frequency (~10%) in the database : since it contains more "general" concepts.

4.6 Conclusion

Our ontology-based approach to mine generalized patterns from texts proves that different kinds of knowledge could be extracted from the Phylogenetics literature. Processual and domain knowledge from articles show interesting patterns to mine. Different ontologies have been used to represent the different types of knowledge categorized into concepts and relations. Extracted workflows are enhanced with ontological annotations to produce generalized frequent patterns. These latter, covering a wider range of values than concrete elements, could be used to assist practitioners during their problem-solving task with a richer vocabulary. The proposed system could also be applied to different domains when expertise is confronted to domain knowledge. Video games, for example, is a good domain application to our approach when gamers are constantly faced to cognitive processual gestures (*e.g.* facing other character players or events). Workflows of events could be extracted from game sessions and mined in order to recommend and predict generalized patterns like : what type of event is going to happen during a game based on gamer logs or what kind of item a player is willing to buy to accomplish a mission.

Acknowledgment

This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D.

CHAPITRE V

VERS UN SYSTÈME DE RECOMMANDATION DE FLUX DE TRAVAUX

Le travail de ce chapitre est divisé en deux parties. La première partie présente le cadre *applicatif* de notre approche d'extraction et de fouille de motifs abstraits, soit en analyse des méthodes d'inférence phylogénétique. Nous présentons aussi des résultats avancés et nous discutons leurs pertinences dans le domaine de l'analyse phylogénétique.

Nous avons soumis la première partie de ce chapitre au journal *BMC Bioinformatics*, 2017, sous le titre "*Towards an ontology-based recommender system for relevant bioinformatics workflows*". Nous présentons la version imprimée de l'article dans l'annexe *D*. La rédaction de l'article, les expérimentations et le montage des figures ont été tous réalisés par Ahmed Halioui sous la supervision des Prof. Abdoulaye Baniré Diallo et Petko Valtchev. Dans la seconde partie de ce chapitre, nous proposons une esquisse de solution vers le développement d'un système de recommandation de processus de la prochaine étape à partir des flux abstraits générés. Cette esquisse décrit un modèle de recommandation basé sur les hiérarchies de concepts et de relations dans les motifs généralisés ainsi que sur des mesures de qualité des Top-k meilleurs objets recommandés.

5.1 Résumé

Les solutions bioinformatiques actuelles deviennent de plus en plus complexes et couteuses en terme computationnel, surtout avec l'explosion quantitative de grandes masses de données biologiques. Des nouvelles compétences doivent être acquises par les chercheurs scientifiques afin de suivre cette évolution. D'ailleurs, les solutions actuelles font face à plusieurs défis dont la *réutilisation des modèles (motifs) abstraits* pour des tâches *spécifiques*. La grande masse de données ainsi le cout qui vient avec les outils issus des nouvelles technologies peuvent faire "tromper" les utilisateurs dans leurs analyses. Afin de pallier à ces problèmes, nous proposons une solution basée sur des motifs abstraits (cooccurrences fréquentes de processus) de flux de travaux afin d'assister les biologistes et les bioinformaticiens dans leurs tâches de résolution de problèmes bioinformatiques. Conséquemment, des flux de travaux *concrets* sont extraits à partir de la littérature de textes scientifiques. Découvrir des motifs fréquents à partir des flux de travaux extraits aidera à monter un système de recommandation basé sur des abstractions de différents niveaux d'abstraction. Dans ce papier, nous étudions des sujets spécifiques en analyse phylogénétique. Nous avons annoté automatiquement plus de 2 500 articles publiés récents (du Janvier 2015 au Mars 2017) en utilisant les différents concepts et relations ontologiques. Les supports (fréquences) relatifs des flux de travaux abstraits fouillés montrent des résultats intéressants dans l'utilisation des outils et des types de données "pertinents" dans les différentes étapes de l'analyse phylogénétique. Ces motifs abstraits rendent les flux de travaux "plus expressifs", ainsi, faciles à être *réutilisés* dans un système de recommandation de "meilleures pratiques" phylogénétiques.

5.2 Abstract

Contemporary Workflow Management Systems are driven by explicit process models specifying the inter-dependencies between tasks. Creating these models is a challenging and time-consuming task. Existing approaches to mine concrete workflows into models (abstractions) tackle design aspects related to the diverging abstraction levels of the tasks. Concrete workflow logs represent tasks and cases of concrete events –partially or totally ordered– grounding "hidden" multi-level (abstract) semantics and contexts. Relevant generalized events could be rediscovered within these processes. We propose, in this paper, an ontology-based workflow mining system to generate patterns from sequences of events which are themselves extracted from texts. Our system `T-Gowler` (Generalized Ontology-based WorkfLow minER within Texts) is based on two ontology-based modules : a workflow extractor and a pattern miner. To this end, it uses two different ontologies : a domain one (to support workflow extraction from texts) and a processual one (to mine generalized patterns from the extracted workflows). `T-Gowler` implementation and data sources are available in : `https://github.com/bioinfoUQAM/tgowler`.

5.3 Introduction

The advances in biotechnology have revolutionized bioinformatics into a data-intensive discipline. The rapidly increasing amounts of available data, from new large scale analyses, have made data processing without automated pipelines (*a.k.a.* workflows) difficult to handle (Anisimova *et al.*, 2013). This situation is mainly due to the complexity of integrating diverse data formats and databanks, interfacing numerous programs with various parameters, as well as to the assessment of data inputs and outputs (Leigh *et al.*, 2011). Current bioinformatic platforms offer a variety of computational solutions using High-Performance (Becker *et al.*, 1998) and Cloud Computing (Armbrust *et al.*, 2010) with different levels of computational abstractions (Spjuth *et al.*, 2015). Two types of models are employed by Workflow Management Systems (WMS) : (1) concrete ones, in which the tasks are bound to specific resources, and (2) abstract ones, representing an abstract view of the resources used in tasks. Service-oriented systems such as Pegasys (Barnard, 2000), Taverna (Wolstencroft *et al.*, 2013), Galaxy (Blankenberg *et al.*, 2001) or Armadillo (Lord *et al.*, 2012) offer models based on Web Services providing an abstraction layer which separates the workflow platform from programming-language details. The state-of-the-art analysis revealed that biomedical services are currently used in an ad-hoc style with a low *reusability* (Leipzig, 2016). This suggests a need for techniques that help scientists to better locate interesting services and workflow fragments and reuse them to attain their research purposes. Current frameworks such Taverna (Wolstencroft *et al.*, 2013) and Galaxy (Blankenberg *et al.*, 2001) offer different approaches for Web Services Discovery (Stevens *et al.*, 2003) and a semi/full automated way to workflow composition. However, none of them provide recommendations to guide users and suggest the next steps to do while taking into account the specific nature of data (complete vs partial genomes, viral proteins, plant DNA, type of the evolutio-

nary model, *etc.*) and relevant analyses (phylogenetic analyses, function prediction, homology studies, Genome Wide Association studies, *etc.*). For example, in phylogenetics (Lemey Philippe, 2009), the evolutionary history of a group of organisms is inferred using different types of data including nucleic acid (DNA or RNA) or protein sequences. Depending on tools which one would use, different types of evolutionary events could be included in the study such as point mutations, duplications, insertions and deletions, genome rearrangements (Anisimova *et al.*, 2013). Such a study requires to define several complex steps, specific tool parameterizations and the acquisition of raw data. Current bioinformatics workflow repositories like myExperiment (Goble *et al.*, 2010) offer a large but a very generic scope of workflow-based analyses over different kinds of platforms. Solving very specific bioinformatic problems such phylogenetic analyses is still limited to few published and shared -non validated- workflows[1]. Moreover, the literature of scientific texts offers a more concise and complete information about bioinformatics analyses. Phylogenetic analysis constitutes a typical bioinformatic workflows in life sciences. This is witnessed by the large number of publications on the topic. This resource could be exploited to enrich workflow solutions. The difficulties in the process of workflow reconstruction from scientific text publications are illustrated in the following example.

Example 1. A biologist U_1 downloads a set of 9 Hemagglutinin sequences of different Influenza viruses from the NCBI database[2]. To carry out the analysis of

[1]Less than 100 workflows representing phylogenetic analyses in the myExperiment database.

[2]`https://www.ncbi.nlm.nih.gov/protein`

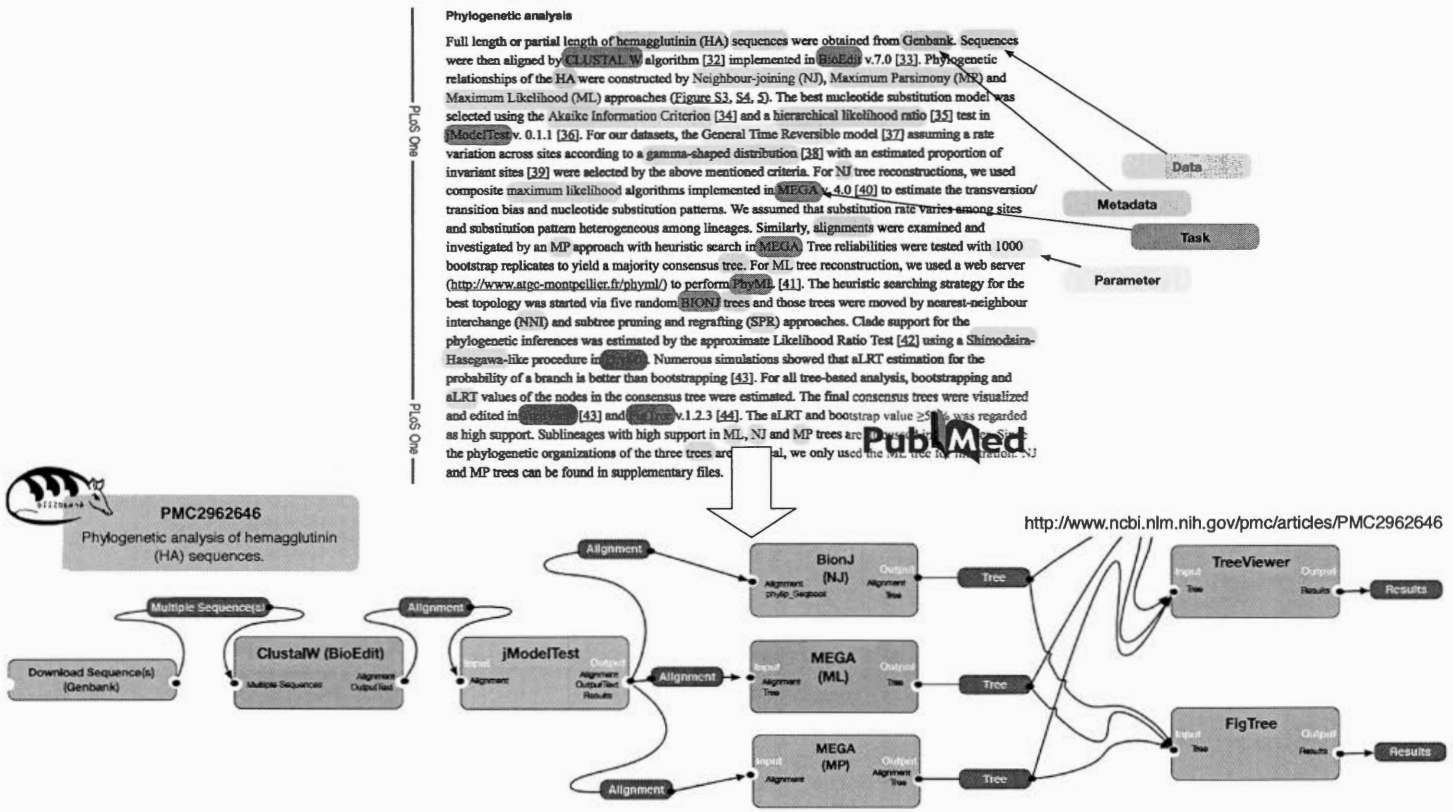


FIGURE 5.1: A Phylogenetic analysis example drawn from the publication of Liang *et al.* (2010) using the Armadillo platform (Lord *et al.*, 2012). From the highlighted software, data, descriptive metadata and parameters in texts, an expert drew a phylogenetic workflow. Task programs are represented with coloured rectangles and data flow with labeled links (input/output parameters). Duplicates of the program *MEGA* at the third step represent different execution instances of it. *Trees* from the *Alignments* generated from the *jModelTest* (Posada, 2008) are inferred using once the Maximum Likelihood instance of MEGA and again with the Maximum Parsimony estimator.

those sequences, U_1 chooses the Armadillo platform[3] (Lord *et al.*, 2012). Several algorithmic options are displayed : *i.e.* software parameters (*e.g.* the number of bootstraps and mutation rate) and data constraints (*e.g.* input/output to programs). U_1 prior *abstract* experience does not cover all the available methods and tools, therefore, the backup solution is to adapt a workflow from the literature (see Fig. 5.1). Yet, none of the workflows drawn from scientific publications could exactly match his needs. These workflows may not properly execute since a part of the analysis requires some *specific* software versions and dependencies.

The above example shows several problems which could occur while constructing bioinformatic workflows. Even simple routine projects can involve a number of different software requiring specific but also abstract problem solving knowledge. Even though software experimental conditions are related to several choices, the amount of data and the misuse of tools could also deteriorate the results of studies. This is where the user skills and experience do make the difference.

Example 2. Suppose that a bioinformatician U_2 chooses to study the same protein sequences of the Hemagglutinin. He runs *ClustalX* (Thompson *et al.*, 2002) to align sequences and then, chooses the *JTT* evolutionary model (Jones *et al.*, 1992) during the tree reconstruction. If we want to recommend a concrete phylogenetic program to U_2 such as the program *MEGA* (Kumar *et al.*, 2016), we propose to calculate the frequency of the sequence $s_0 : \langle \{ClustalX\}, \{JTT\}, \{MEGA\} \rangle$ in the extracted workflow database. However, such sequence is very rare due to the vast varieties of the available tools for each specific task. Thus, using frequent sequences of tool categories such as the "*Maximum Parsimony (MP)*" or "*Maximum Likelihood (ML)*" in a recommendation scenario is more relevant (due to

[3]Armadillo (www.bioinfo.uqam.ca/armadillo) is a drag and drop workflow platform dedicated to phylogenetics as well as general bioinformatics analysis.

the higher support of tool categories over instances). An abstract pattern such p_0 : $\langle$ $\{LocalAlignment\}$, $\{ProteinModelSelection\}$, $\{MaximumLikelihood\}$ $\rangle$ is more frequent than the concrete pattern s_0.

Here, we propose to use ontologies (Ashburner *et al.*, 2000) to represent the different categories (*concepts*) of tools and input/output data at different levels of abstractions. The constraints on combining tools and parameter values are expressed using properties (*relations*) between categories. In fact, the ontological representations are effectively exploited by two key processes : (1) the extraction of workflow terms and relations from texts (*concrete workflows*), and (2) the extraction of generalized workflow patterns (*abstract workflows*). Thereof, we distinguish *concrete* workflows made of individuals (instances) and links between them, from *abstract* workflows, comprising labels of concepts and properties at various levels of abstraction. In our approach, mining processual abstract patterns out of the concrete workflow cases is based on the hierarchies of both, concepts and relations, from the domain ontology. This could be the foundation of a semantically rich recommendation system to suggest the best next step methods.

The remainder of the paper is organized as follows : section 5.4 presents related work to workflow recommender systems in bioinformatics. Section 5.5 presents the experimental evaluation and results of the extracted workflows and patterns from texts. Finally, section 5.6 discusses the quality of our approach.

5.4 Related work

Current bioinformatics WMS have emerged as global *intelligent* infrastructures[4] by integrating new technologies to maintain a certain level of reusability through

[4]For example, these systems do not suggest local configurations for tools but a global way to manage tasks through Web Services.

grid and cloud computing (Yu et Buyya, 2005; Saltz *et al.*, 2006; Spjuth *et al.*, 2015). For example, they may offer an efficient way to abstract *services* for data transformation and searching resources in distributed secure databases. Yet, in practice, the sharing service-based capabilities may open a gateway to *reuse* workflows, however, the degree of reuse is still very low (Yao *et al.*, 2012b). For example, repositories like myExperiment, CrowdLabs and SHIWA (see (Leipzig, 2016)) offer a restricted number of workflows (*e.g* ~2 500 workflows in myExperiment) for very various disciplines, *e.g.*, Chemistry, Astronomy, Sequence Analyses, and in different data formats, *e.g.*, Taverna 1 XML[5], Taverna 2 SCUFL2[6], Galaxy .ga format[7], *etc.* which decreases the *reusability* of workflows. Hence, in the literature of bioinformatics best practises, more work reported on service oriented pipelines and/or provide expert-based suggestions (Anisimova *et al.*, 2013). In this vein, Spjuth *et al.* (2015) review the most advanced projects defining general guidelines for the methodological choices at various stages of a bioinformatics pipeline. The authors recommend dedicated practices in maintaining reproducibility and standardization in local and distributed environments. Anisimova *et al.* (2013) criticize the outdated methodology and provides practical guidance on issues and constraints faced by each of the phylogenetic analysis steps. Garijo *et al.* (2014) propose a set of high-level abstract motifs (patterns) as a view of activities undertaken within workflows. They performed their manual analysis over a set of real-world scientific workflows extracted from Taverna (Wolstencroft *et al.*, 2013) and Wings (Gil *et al.*, 2007) systems. However, expert recommendations are better used in solving problems and making decisions if they are supported by the help of Knowledge Management (KM) discipline. KM is the subject that provides stra-

---

[5]`https://taverna.incubator.apache.org`

[6]`https://taverna.incubator.apache.org/documentation/scufl2`

[7]`https://galaxyproject.org`

tegy, process and technology to share information and expertise among users. For instance, Semantic Web (SW) technologies (Staab *et al.*, 2010), as a special case of KM, allow automatic discovery and execution of Web Services that can handle the workflow tasks using ontologies to improve knowledge integration, querying, and sharing (Hashmi *et al.*, 2012).

Next, we present current solutions for bioinformatics recommendation and automatic workflow composition into three categories : (1) semantic-based recommendation, (2) similarity-based recommendation and (3) pattern-based recommendation.

Semantic-based recommendation. Proteus (Cannataro *et al.*, 2004) is one of the first projects to integrate an ontology-based design in bioinformatic workflows. It exploits a *DAML + OIL* ontology for the data mining domain describing resources and processes of knowledge discovery in databases. Lee *et al.* (2007) describe a SW solution assuming a grid computing environment. The proposed Bio-STEER system offers SW services in OWL-S[8]. A small-size ontology is employed in the representation of both abstract workflows (for Grid resources) and concrete ones (to bind tasks to specific resources). Ison *et al.* (2013) introduce the EDAM ontology for bioinformatic operations, types of data, identifiers, topics and formats. EDAM supports semantic annotations for Web Services, databases, interactive tools and libraries. Offered annotations have been integrated in a set of frameworks such

[8]OWL-S is an ontology built on top of WebOntology Language (OWL) for describing Semantic Web Services. It enables users and software agents to automatically discover, invoke, compose, and monitor Webresources offering services.

as EMBOSS [9] and eSysbio [10]. However, these systems do not exploit the entire richness of EDAM semantics. For example, eSysbio uses "EDAM Data and Format" to decide how to handle data by an adequate visualization and search. This navigation, as static as it is[11], does not take benefits from the relation semantics between concepts. In the same scope, we find the Software Ontology (SWO) (Malone *et al.*, 2014) offering an extended version of EDAM through new concepts and relations for information processing tasks (phylogenetic construction, protein structure analysis, *etc.*), algorithms (ANOVA, t-test, *etc.*), software and software versions. The ontology has been used so far in only service discovery frameworks like BioMedBridges[12] and the Gene Expression Atlas repository[13]. Yet, EDAM and SWO represent only very generic concepts/relations that still need to be specified for very specific technical domains such as phylogenetics (Lemey Philippe, 2009) or genome rearrangements (Bafna et Pevzner, 1996). Moreover, GBrowse (Wilkinson, 2006) and then, Seahawk (Gordon et Sensen, 2007) are two end-user oriented with ontology-based data centric approaches. Users can either specify input and/or output data and the application determines the corrected data type and proposes what to do with it instead of proposing a specific service. DiBernardo *et al.* (2008) use the BioMoby (Wilkinson et Links, 2002) services registry and construct the workflow at each step by ranking the services according to the si-

---

[9] www.ebi.ac.uk/Tools/emboss

[10] www.esysbio.org

[11] The current version of eSysbio uses a limited subset of EDAM for static navigation (allowing grouping and filtering of data by their type), without taking into account the relations other than the closure of "*is format of*".

[12] www.biomedbridges.eu

[13] www.ebi.ac.uk/gxa/home

milarity of their data type inputs and other non-functional measurements like the number of retrievals of a service in the definition metadata. Finally, we find the SADI project (Wilkinson *et al.*, 2009) aiming to maximize the interoperability between Web Services unlike BioMoby which may contain inconsistent data (Withers *et al.*, 2010). SADI indexes both data and properties and offers a search engine which is designed to mimic the linguistics of scientific queries through very specific ontological relationships. Navigating and searching for the "best" tool in the set of task categories may not be enough to guide users to achieve their analysis. In addition, formulating queries could be a very difficult task if users do not know what is the next step and the best method/tool and configurations for that step.

Similarity-based recommendation. Similarity-based approaches aim to tackle two important problems of workflow reuse : (1) the retrieval of similar workflows form potentially large repositories and (2) the adaptation of workflows in new case studies. Here, the Process-Oriented Case-Based reasoning (Schumacher *et al.*, 2013; Bergmann et Gil, 2014) comes to support and partially automates experience-based problem solving by creating new workflows and adapt them for new purposes and circumstances. Bergmann et Gil (2014) propose a graph-based approach with generalization of workflows over the concept taxonomy in the domain ontology. The similarity between a case workflow and a query one is calculated using a local similarity assessing the similarity between two nodes/edges of the same type which is aggregated into a global similarity within a graph mapping process. Their similarity measures are based on an Object-Oriented approach (Bergmann et Stahl, 1998) calculating intra and inter class similarities between abstract and concrete objects. Their approach is used later on with Müller et Bergmann (2014) where during adaptation, deficiencies in the retrieved

case are incrementally compensated by replacing fragments of the retrieved case by appropriate workflow streams. Yet, their work is limited to strongly structured texts like cook recipes and software manuals where sentences (imperative phrases) explicitly describe series of actions.

Pattern-based recommendation. Most of bioinformatic pattern-based recommenders use sequential pattern mining algorithms in order to discover recurring software and data usage patterns (Aggarwal, 2016). For example, Duck *et al.* (2014) propose to use Natural Language Processing (NLP) tools to extract software and database terms from scientific texts. The process uses the bioNerDS terminology [14] in order to categorize terms in texts. In the pattern mining step, a straightforward mining algorithm discovers frequently occurring resource pairs. Each resource is paired with the one that immediately follows it in a text while accounting for the direction within the ensuing pairs. However, the order of the terms in the text is not always reliable. Without a context-aware extraction approach, the occurrence of such a pair could mean nothing but simple mentions. Ríos *et al.* (2009) propose Magallenes, a Web service discovery and an automatic workflow composition tool which associates data types to connect available and compatible Web Services into workflows that can process data sequentially to reach desired output given a particular input. A Breadth-First search with pruning algorithm is used to find the shortest path from source to target services. In the same scope, Yao *et al.* (2012a) use the ServiceMap algorithm (Tan *et al.*, 2011) to model and evaluate workflow "reputations" based on the number of downloads, views, credits and friends taking into account reputation time fading and its popularity. They propose a service-based recommender platform based on the paths from input to output data using the classic Apriori approach (Agrawal et Srikant, 1994) to

[14] www.bionerds.sourceforge.net

find frequent associations between services and then, find a path between two services/operations in the reputation network. In a different vein, methods for mining patterns and associations on top of an ontology have been proposed in (Svatek *et al.*, 2006). As an example, the GSP (Generalized Sequential Patterns) algorithm (Srikant et Agrawal, 1996) mines frequent patterns from sequences of user transactions. It traverses the pattern space with an Apriori-like level-wise discipline and by exploring the anti-monotony property of itemset frequencies over the taxonomy. Besides, using an ontology to enrich patterns of Webusages has been researched on for at least a decade (Dai et Mobasher, 2002; Senkul et Salin, 2012). Yet, these latter only process the class hierarchy. Adda *et al.* (2010) present a complete framework for Webusage mining from the logs of a Webpage clicks content where its (page resources) objects are described by an ontology. The input data are sequences of clicks further translated into IDs of content objects with their semantic links. Patterns are also sequences, yet, made of concept IDs linked with ontological properties. Their proposed `xPMiner` method defines a total order between sequence items and applies a level-wise mining through the pattern space with a refinement operator (De Raedt, 2008) made of four primitives following the hierarchies from the ontology : addition/specialization of concepts/relations.

Inspired by the model of Adda *et al.* (2010) and beside that input data is automatically extracted from texts, we offer a new pattern miner, where, instead of sequences, we deal with workflows as posets[15] (where transactions represent steps) of objets linked with each other, thus, forming (double)-layered DAGs (Direct Acyclic Graphs). Here, the DAG nodes represent task, data and metadata items and edges represent the control and data flows. In addition, on top of the workflow DAGs, an ontology provides two taxonomies, one for concepts and one

[15]Partially ordered sets which formalize and generalize the intuitive concept of an ordering, sequencing, or arrangement of the elements of a set.

for relations (*a.k.a.* generalized items). Hence, patterns are made of concepts with inter-concept relations which may belong to different generality levels in the ontology. Interested readers may refer to our publication (Halioui *et al.*, 2015) which represents the framework of our project and present preliminary results of the extracted concepts and relations from texts. Yet, we present in this work the full description of our approach of workflow extraction and pattern mining and offer a domain validation of the extracted patterns in the phylogenetics experience acquisition domain.

5.5 Advanced experiments

5.5.1 Dataset

The experiments are derived from a sample of +5 000 recent articles, journals and conference papers collected from the digital repository PubMed Central (PMC)[16] in the XHTML format using the following query :

> *"open access"[filter] AND "2017/01/01"[PubDate] : "2017/03/10"[PubDate] AND ("Phylogenetic analysis"[Section Title] OR "Phylogenetic analyses"[Section Title]).*

However, phylogenetic analyses sections may not have sufficient information to reconstruct the full workflows from texts. Thus, we filtered the downloaded articles based on the smallest pattern that any phylogenetic analysis should present : $\langle \{DataType\}, \{PhylogeneticInferenceProgram\} \rangle$. A phylogenetic study might use at least one data type and one phylogenetic inference program (see Fig. 5.3) to produce a phylogenetic tree. Finally, we retained a set of 2 969 articles.

---

[16] `https://www.ncbi.nlm.nih.gov/pmc/`

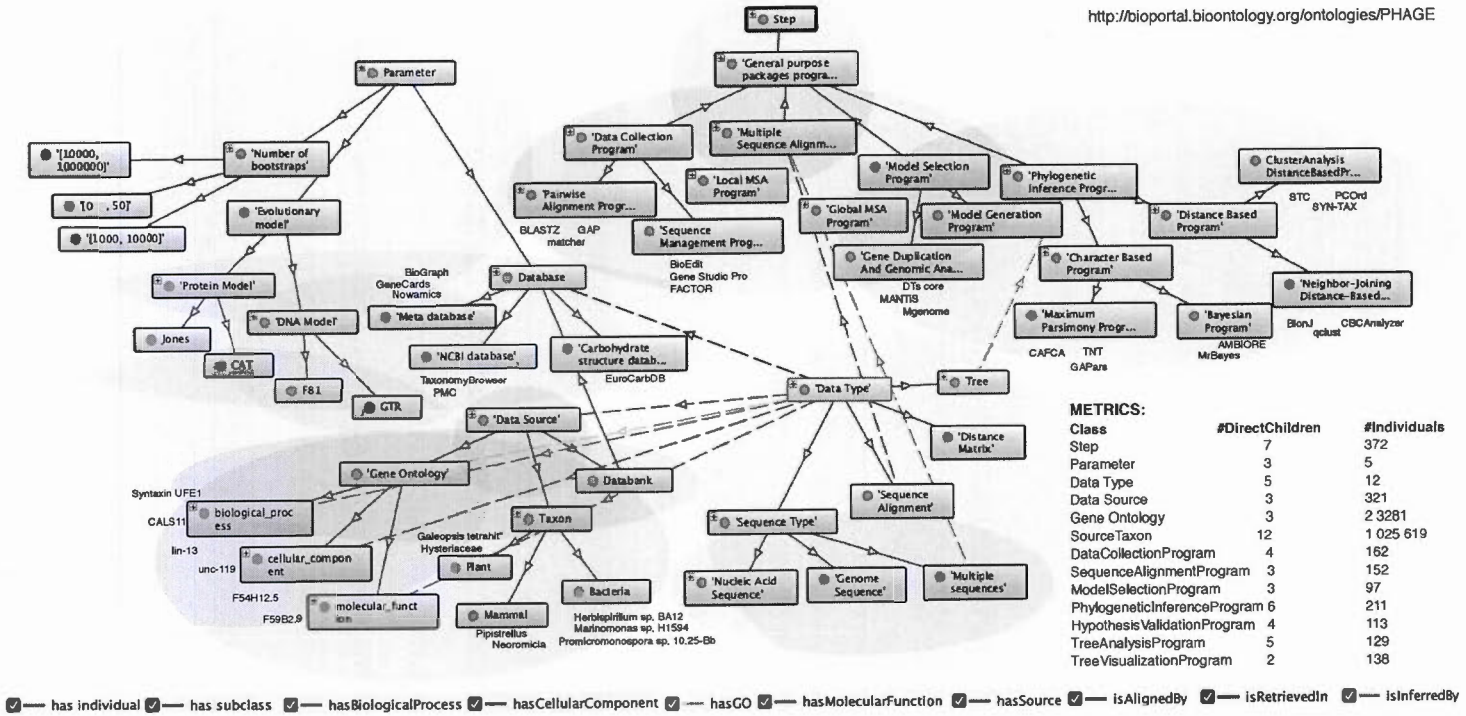


FIGURE 5.2: A sample from the proposed PHAGE ontology published in the BioPortal repository (see the link on the top right corner). Grey rectangles represent workflow concept references to : steps and programs (in the blue cloud), data types (in the green cloud), sources (in the purple cloud) and parameters (in the yellow cloud). The hard arcs represent hierarchical relations and the dotted arcs represent other semantic properties (domain constraints and input/output relations). Terms under concepts show some individuals (instances) from categories.

Our public PHAGE ontology on BioPortal [17] is composed of 37 different object properties (relations), 47 031 concepts (with 4 root concepts and 111 unique concepts) and more than 1 000 000 unique individuals (instances) without counting synonyms. Fig. 5.3 shows a sample from the proposed PHAGE ontology. '*Root concepts*' represent data types, their provenance (data sources), steps and tool parameters. '*Step concepts*' represent the seven steps of phylogenetic analyses and *relations* between concepts represent data flows and control flows. '*Program concepts*' represent the software concepts (*e.g. Local Alignment*, *Distance based approaches*). '*Parameter concepts*' represent the different types method/program parameters (*e.g.* number of bootstraps, DNA/protein evolutionary models) and '*Data type concepts*' represent the different data types (*DNA*, *RNA*, *distance matrices*). Finally, '*Data source concepts*' represent provenance metadata (*e.g.* data formats, databases, GO annotations, types of organisms). The proposed ontology have been collected in a semi-automated way from well-known databases in the phylogenetics domain. From the NCBI Taxonomy browser (Federhen, 2012), we extracted +1 000 000 (all) scientific terms and synonyms of organisms and organized them into 12 ranks (plants, bacteria, mammals, *etc.*). From the Gene Ontology (Ashburner *et al.*, 2000), we extracted recent (2016/04/12) GO concepts and relations. This ontology presents 112 unique concepts and 9 object relations. More than 116 000 (all) reviewed unique gene and protein terms were extracted from the UniProtKB database (Kiefer *et al.*, 2009). Each term was then inserted as an instance for the related GO concept. Finally, the Felsenstein's Website (Felsenstein et Felenstein, 2004) is crawled using simple data transformation scripts in XSLT leading to create more than 500 software terms categorized into 32 categories.

---

[17] `https://bioportal.bioontology.org/ontologies/PHAGE`

5.5.2 Extracted workflows from texts

Using the GATE platform[18], a human expert has corrected (created, deleted, updated) manually annotations from randomly chosen 500 texts to build a workflow ground-truth database. This latter serves as a test set for the evaluation step. The expert annotates the given texts using our domain ontology schema on the platform GATE. He annotated only terms of concepts and relations that serve to reconstruct the phylogenetic workflows from texts. Although this task is costly in terms of time/energy ($\sim$ 80 hours), the expert validation step is important to guarantee a credible perspective of our approach.

Next, we measure whether a term is annotated correctly, partially or simply missing. We show in Table 5.1 details about the total number of the discovered annotations. We calculate the classic Information Retrieval F-Measure (Salton et Buckley, 1988) based on correct, partially correct and spurious annotations (see also section 2.5.5). In total, our system has annotated 2 561 terms while the expert has annotated 2 641. The overall strict F-Measure (overlaps discarded) is 70% and the overall lenient F-Measure (overlaps included) is 76%. We notice that 118 annotations ($\sim$20%) represent ambiguous program terms (GuessedProgram). In this case, the average F-Measure (between strict and lenient) is 84% with a precision of 73% and a recall of 98%.

The manual annotation of ambiguous programs highlights all the textual mentions referring to a multiple purposes tool as indirect references. In some cases, correct results from our workflow tagger were not fully annotated, leading to false positives. The following sentences :

---

[18]GATE (www.gate.ac.uk) is a well-known environment for annotation tasks using widely known NLP techniques such as the OpenNLP parser (`https://wiki.apache.org/incubator/OpenNLPProposal`) and the Stanford POS tagger(Toutanova *et al.*, 2003).

> *A phylogenetic tree was constructed with* ***MEGA*** *5 using the aligned sequences [...] The phylogenetic tree was drawn using the neighbor-joining method with p-distance and 1,000 bootstrap replications.*

show an example of such case where our tagger correctly detects *MEGA* (Kumar *et al.*, 2016) as a phylogenetic tree inference in the first sentence but in the following sentence, *MEGA*, a neighbor-joining method, is also used to validate trees using 1 000 bootstrap replications. The class of MEGA (*HypothesisValidation*) was not detected by our context-based approach. The reference in the second sentence is implicit (hidden in the *Neighbor-Joining* concept), hence, it is difficult to detect that trees generated by *MEGA* are also validated using the bootstrap method. In addition, both of the verbs in the two sentences are very generic and misleading for an automatic tagging, *i.e.*, *aligned* and *drawn*. *MEGA*, in the previous sentences, is not used to align or visualize sequences or trees. Consequently, the precisions obtained by our pipeline for tree validation tasks are very low (~40%). In fact, 30% of the hypothesis validation programs are ambiguous programs (multiple purpose packages). Yet, our context-based pipeline supporting our system needs to be enhanced in its ability to capture more generic mentions in wider contexts (*e.g.* paragraphs).

Although strict and lenient F-Measure values are very close (~0.3% of difference), some categories, *i.e. DataCollectionProgram*, *TreeVisualizationPrgram* and *GuessedGeneralPackagesCategories*, show the same values for strict and lenient F-Measures (see also the 'O' column). Our tagger is able to recognize the exact mentions in texts : these categories does not present non-standard program names (*e.g.* common names) or generic mentions. In addition, we note that almost the half of the presented categories show reasonably high lenient F-Measure agreement (> 80%). Yet, besides *hypotheses validation* software names, *data collection* programs, *species* names, *protein* and *GO annotations* show the worst tagging

F-Measure results (with a lenient F-Measure between 60 and 67%). The recall values in data collection programs explain the weak F-Measure (for both strict and lenient agreements). Thereof, our database for this category need to be updated. In addition, since no specific JAPE rules are modelled for the metadata (GO annotations and species), some common false negatives (*i.e.* missed resource mentions) are discovered including common names of genes (*e.g.* the *g20* gene) or species (*e.g. birds*).

Fig. 5.3 shows the workflow similarity distribution results based on the evaluation of the 500 expert-curated workflows. Generated workflows are 82.89% similar with the actual workflows extracted by the expert (first figure on the top). While, the best mean workflow similarity is 100%, the worst similarity is 54.68% which means that, in the worst case scenarios, our workflow extraction framework will (probably) extracts half of the "actual" workflow components and the mean of object (tasks) similarities is 78.71% (second figure in Fig.5.3).

The best workflow task similarity is found in *phylogenetic tree inference* programs (100%). We found that *phylogenetic methods* are frequently co-occurred in the context of the *phylogenetic inference programs* which explains the high values of the mean similarities of this task. However, the worst task similarity is obtained by the *HypothesisValidationProgram*. This is due to the context of this task which is ambiguous to be resolved in one sentence as we explained previously. Concepts and POS categories surrounding these programs in one sentence are not very conclusive to disambiguate this category of software.

The relation side of workflows represents the data and control flows (third figure in Fig.5.3). The mean similarity in the control flow is 89.1% and 88.92% in the data flow. While the worst similarities in the data-flow is 74.43% which represents the relations between *Data Types* and *Multiple Sequence Alignment* programs,

Table 5.1: Concepts Evaluation Distribution

	S	G	P	O	SP	SR	SFM	LP	LR	LFM
BootstrapsParms	61	19	8	1	0.87	0.75	0.80	0.88	0.76	0.82
DataBase	43	19	13	3	0.72	0.66	0.69	0.77	0.70	0.74
DataCollectionProgram	36	29	5	0	0.87	0.55	0.67	0.87	0.55	0.67
DataType	597	103	186	89	0.68	0.75	0.71	0.78	0.86	0.82
GeneOntology	139	172	5	3	0.94	0.44	0.60	0.96	0.45	0.61
HypothesisValidationProgram	25	40	29	2	0.44	0.37	0.40	0.48	0.40	0.43
Method	326	48	46	14	0.84	0.84	0.84	0.88	0.87	0.87
ModelSelectionProgram	42	50	17	1	0.70	0.45	0.54	0.71	0.46	0.56
PhylogeneticInferenceProgram	205	16	146	10	0.56	0.88	0.69	0.59	0.93	0.72
SequenceAlignmentProgram	70	15	19	4	0.75	0.78	0.76	0.79	0.83	0.81
SourceTaxon	202	125	57	19	0.72	0.58	0.64	0.79	0.63	0.70
TreeVisualizationProgram	18	7	2	0	0.90	0.72	0.80	0.90	0.72	0.80
GuessedGeneralPackagesCategories	87	1	31	0	0.73	0.98	0.84	0.73	0.98	0.84
Sum	1851	644	564	146						
Macro summary					0.75	0.67	0.69	0.78	0.70	0.72
Micro summary					0.72	0.69	0.70	0.78	0.74	0.76

The 'S' column represents the number of Successfully recognized annotations. The 'G' (only in Gold set) column represents missed annotations while the 'P' (only in Prediction set) column shows the number of unknown annotations. The 'O' (Overlap) column shows the number of (partially recognized) annotations. While the 'SFM' and 'LFM' columns shows the Strict (overlaps discarded) and Lenient (overlaps included) F-Measures, respectively, the 'SP', 'ST' and 'LP', 'LR' columns represent Strict and Lenient values of Precision and Recall, respectively. Micro averaging essentially treats the corpus as one large document while Macro averaging calculates the F-Measure on a per document basis, and then, averages the results.

the best data-flow similarity is the relation similarity between *Data Types* and *Hypothesis Validation* programs (100%). Hence, this part of data flows is reconstructed perfectly. Verbs like "validate" and "test" are commonly and precisely used between *Data Types* and *Tree Validation* programs which explains the high

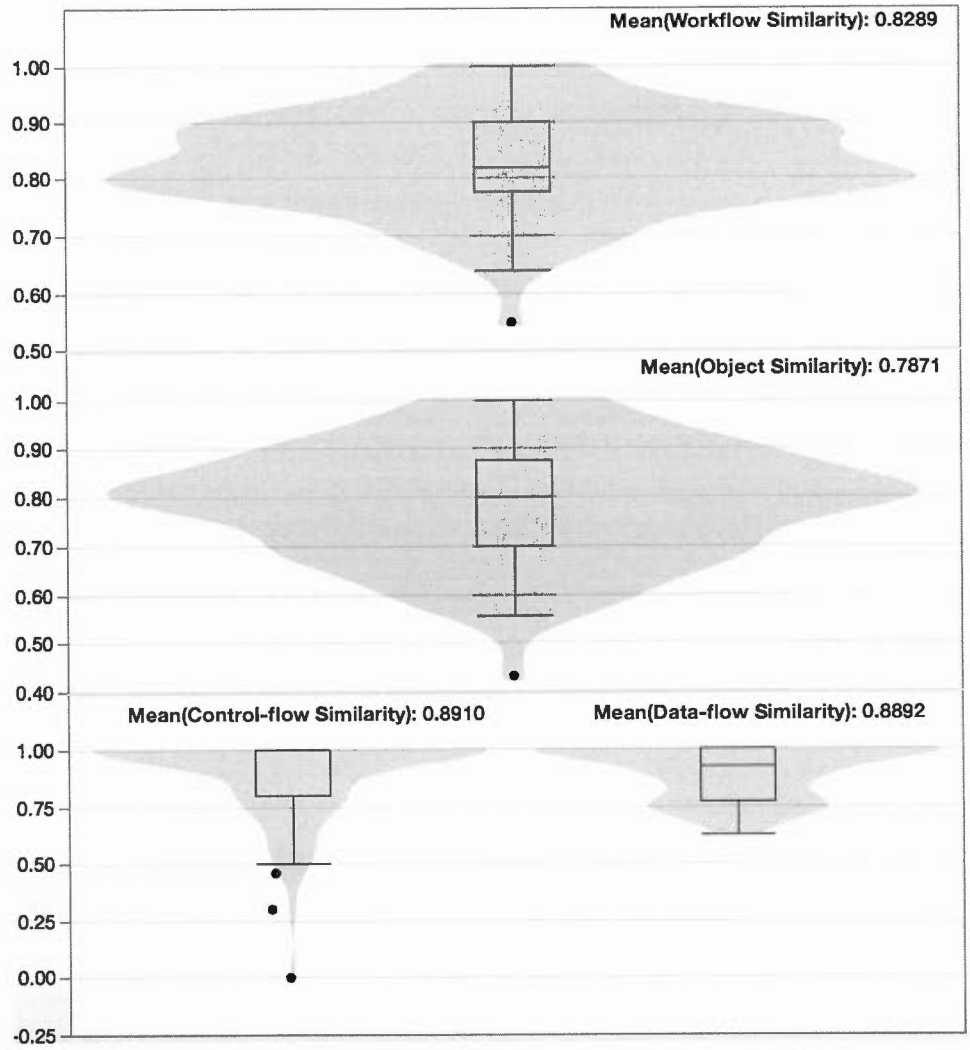


FIGURE 5.3: Workflow similarities results. The top figure represents the overall similarity between the actual a (expert curated) workflows and the predicted p ones. The second and third figures represent box-plots and contour distributions of concept and relation similarities between a and p. Bold points represent the interquartile range outliers.

similarity in this concept. The worst similarities in the control-flow are found in the links between *Phylogenetic Inference* programs and their types of *Methods*

(approaches) with a mean of 79.84%. However, links between *Program Parameters* like the *Number of Bootstraps* and the type of *Evolutionary Models* are, generally, explicitly described in the sentence which explains also its high similarity in the control-flow with (100%).

5.5.3 Generalized bioinformatics patterns

At the mining step, we used an ontology-based sequential mining algorithm that exploits the domain ontology to output the most frequent generalized workflow patterns. In an unrestrained run, the algorithm scans the pattern space from the top downwards, at each step, exploring the patterns at a specific generality level. After the first few levels, each step generates a considerable number of patterns. Hence, we prune patterns that cover the same set of workflows (see Algorithm 2).

Next, in order to characterize the "interestingness" of the generated patterns (Geng et Hamilton, 2006), we propose to use the depth of pattern elements in the ontology to represent their abstraction level. We remind that the abstraction level of a pattern defines how pattern elements are specific or abstract in the domain ontology. Yet, the ultimate interestingness of a pattern depends on the decision-maker (subjective criteria) and does not solely depend on the statistical strength of the pattern (objective criteria, *e.g.* support or concept depth). Albeit useful, objective measures are not enough to determine the interestingness of a pattern, particularly, in complex structured domains like phylogenetics. Our intended solution is to let users choose their levels of interest in the ontology *min_level*[19]. Then, ranking could be used to avoid checking all of the generated patterns (Collard et

[19]The expert proposes the minimum of abstraction level that interests him. This level is normalized by the maximum depth in the ontology (in both of concepts and relations). So a score of 1 means that the expert demands only the most specific patterns and the score of 0 means that he want all of the generated abstract patterns.

Vansnick, 2007).

In order to motivate our ranking method, we propose to use the patterns p_1 : ($\langle$ {*Protein*}, {*Bayesian*} $\rangle$, {(1, *isDataInAlignment*, 2) }) and p_2 : ($\langle$ {*Protein*, *MuscleContraction*}, {*MSA*}, {*ML*} $\rangle$, {(1, isDataInAlignment, 2), (1, isDataInInference, 3)}) and the ontology from the Fig.5.3.

Depth ranking score ($rank_{depth}$). In order to rank the pattern abstraction levels based on the distance from the minimum of depth suggested by the expert *min_level* we use here, the nonparametric two-sample Wilcoxon Rank-Sum Test (Wilcoxon *et al.*, 1963) where pattern depth lists are compared two-by-two. Suppose that we have a set of concept depth lists $depth(p_i) = \{maxDepth(c_j \in p_i.\zeta, H_C)\}$ (see definition 3.8.3) and we wish to test the hypothesis that the distribution of concept depths F_1 of a pattern p_1 is the same as that (F_2) in p_2 and the means μ of both of the distributions are greater or equals to the *min_level* given by the expert. Hence, our *null* hypothesis is $H_0 : F_1 = F_2 \Rightarrow \mu_1 = \mu_2$, $\mu_1 \geq min_level$. The two sided-alternative hypothesis would be, $H_1 : F_1 \neq F_2 \Rightarrow \mu_1 \neq \mu_2$, $(\mu_1, \mu_2) < min_level$. We remind that the Wilcoxon rank-sum test is based upon ranking the $|depth(p_1)| + |depth(p_2)|$ observations of the combined sample. Each depth is assigned to a rank : *e.g.*, the smallest has rank 1, the 2^{nd} smallest rank has 2, and so on. The rank-sum test statistic is, hence, the sum of the ranks for depth observations from one of the patterns. Next, the test finds all the possible permutations of the ranks and calculates their *p_values*. If there are depth ties, where concepts from different patterns have the same ranks, we assign, to each observation in a tie, its median rank[20].

[20] We choose to use the median, here, instead of averages since $\mu_1 - \mu_2$ is strongly influenced by some of the "*very specific*" or "*very abstract*" concepts in a pattern.

Next, we calculate the quantity of information in a pattern in terms of the number of elements covered from the domain ontology. Patterns representing the maximum amount of phylogenetic informations are preferred, hence, we use the coverage measure as a ranking feature to sort our patterns.

Coverage of steps (CC_S). Let $n_S(\Omega)$ be the number of all '*root concept steps*' in the ontology. The fraction of steps that a pattern covers is : $|p.\zeta|/n_S(\Omega)$. For instance, if the ontology defines 7 steps, as in the phylogenetic domain (see *Fig.* 5.3) and the pattern presents only 6 transactions, then, its step coverage is ~85.7%.

Coverage of data (CC_D). Let $n_D(p, \Omega)$ be the number of all data concepts[21] that could be plugged into pattern program concepts from the ontology. The data coverage of a pattern is, then, the number of data used in the pattern divided by $n_D(\Omega)$. For instance, given the pattern p_2, if the number of relations (incoming and outgoing) that could be connected to both of *MSA* and *ML* in the ontology is 4, then, $CC_D(p_2, \Omega) = 2/4 = 50\%$.

Coverage of metadata (CC_M). Let $n_M(p, \Omega)$ be the number of all metadata concepts that could be connected to pattern data concepts from the ontology. The metadata coverage of a pattern is, then, the number of its metadata divided by he number of all metadata in Ω that are connected to pattern data. For example, if the number of metadata connected to *Protein* in the ontology is 4, then, $CC_M(p_2, \Omega) = 1/4 = 25\%$.

[21] Data from the lowest abstraction level.

Coverage of programs (CC_P). Let $n_P(p, \Omega)$ be the number of all program concepts that could be used in the pattern steps from the ontology. The pattern program coverage is, then, the number of program concepts in $p.\zeta$ divided by n_P. For instance, if the number of program concepts in the second and the third step of the ontology is 10, then, $n_P(p_2, \Omega) = 2/10 = 20\%$.

Coverage of relations (CC_R). Let $|\lambda_p(t)|$ and $|\lambda_p(u)|$ be the numbers of elements at the t^{th} and u^{th} transaction where $t \neq u$, then, $n_R(p) = \sum_{t}^{|p.\zeta|}(|\lambda_p(t)| \times \sum_{u \neq t}^{|p.\zeta|-1} |\lambda_p(u)|)$ is the number of all possible combinations between pattern elements in $p.\zeta$. The pattern relation coverage is, hence, the number of triples in $p.\Theta$ divided by $n_R(p)$, *e.g.* $n_R(p_2) = 2/10 = 20\%$.

Finally, patterns are ranked using the weighted ranking function $rank_{pattern}$ (see 5.1) where α and β are the weights given, for the coverage score and the abstraction level score, respectively, and $avgC$ is the average function calculating the mean of coverage scores of a pattern p.

$$rank_{pattern} = \alpha.rank_{coverage} + \beta.rank_{depth} \tag{5.1}$$

where :

$$rank_{coverage} = avgC(CC_S, CC_D, CC_M, CC_P, CC_R). \tag{5.2}$$

We remind that, we are ranking patterns given their abstraction level towards the expert abstraction threshold and the information quantity of a pattern in terms of the number of covered elements from the domain ontology. Patterns with high values $avgC$ (containing more elements from the ontology) and $rank_{depth}$ are assigned with the highest ranks $rank_{pattern}$.

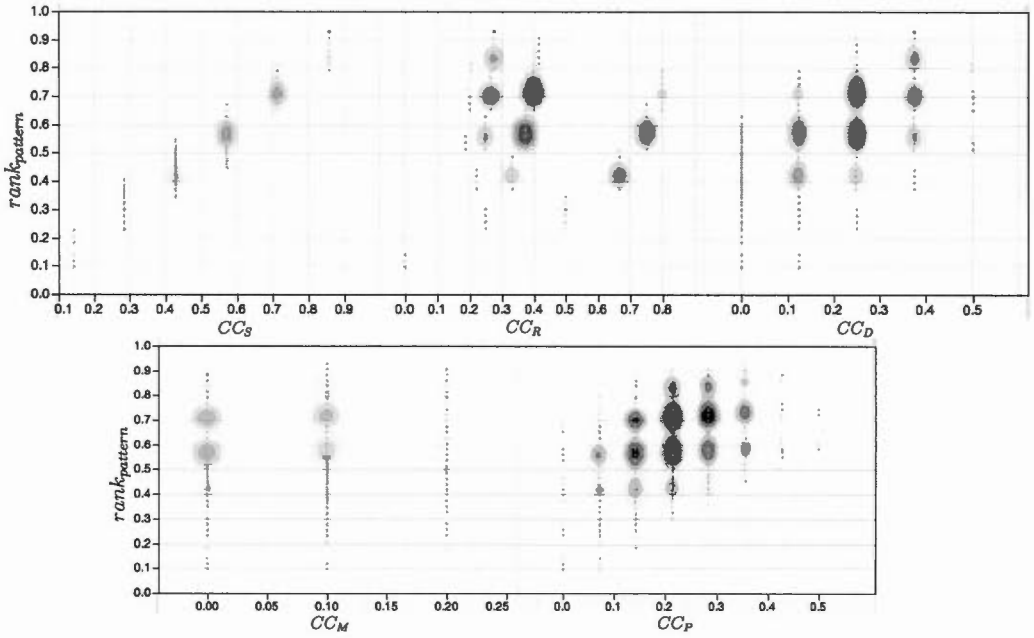


FIGURE 5.4: Pattern coverage metrics distributions Vs. pattern ranking function.

In our next experiment, we apply the pattern ranking function to the generated patterns with the following parameters : $\alpha = \beta = 0.5$ and $min_level = 0$ (to generate all possible patterns) where $\sigma = 0.1$. *Fig.* 5.4 shows the correlation results between pattern ranks and the proposed features. The distribution of the number of steps demonstrates a good correlation with the pattern ranking function. The figure shows that our patterns cover mostly 60-70% of the total number of the phylogenetic steps which is a good coverage. These patterns contain information about most of the phylogenetic steps and thus, they could be useful for novice users. In addition, generated patterns contain an average of ~25% of data (from the ontology) annotated with a maximum of 15% from all ontological metadata per pattern. The distribution of program frequencies shows that patterns use up to 20% of program categories per pattern and most of the discovered pattern concepts are 80% linked with each other. This means that generated pattern workflows are

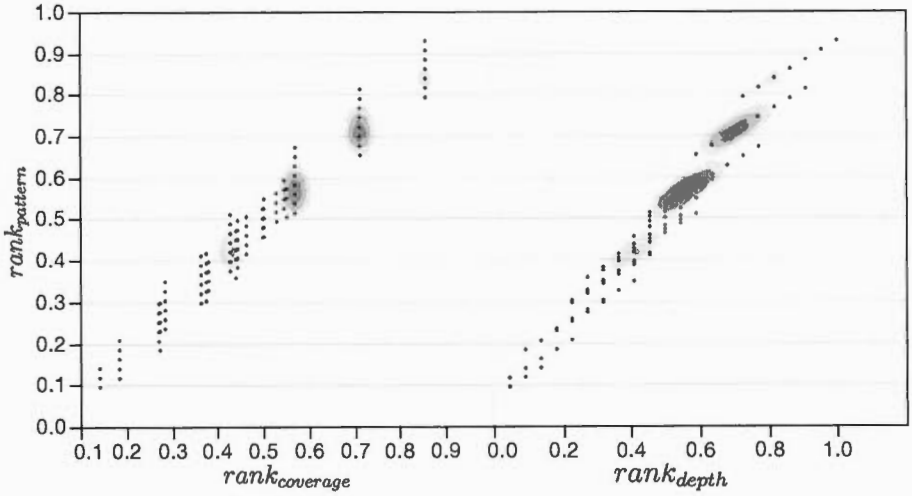


FIGURE 5.5: Pattern raking function Vs. coverage and depth rankings.

"highly" connected digraphs.

The *Fig.* 5.5 shows the correlations between the pattern ranking function, the coverage and depth ranking measures. The figure shows similar correlations between the depth ranking and the coverage ranking which means that patterns covering the most (85%) of elements are very specific too (with low abstraction levels). Thereof, discovered patterns with a low support ($\sigma = 0.1$) might contain the most interesting patterns (for expert users) since they are both large (covering most of the elements from the domain ontology) and contain very specific elements.

5.6 Discussion

A well-known regularity in patterns over the hierarchy of items can easily be observed here : patterns with high support (σ) tend to be too general, *i.e.* involve concepts and properties from higher levels of the ontology, and therefore, they have little practical value. Conversely, specific patterns –whose component concepts and properties lay close to the leafs of the ontological hierarchies– typically have lower

supports and thus, they might not make it to the mining result. Hence, without a very conclusive subjective criteria, given by "an expert" (or experts), it is hard to choose what are the most interesting patterns. Besides objective criteria such support and coverage, we ranked patterns based on their concept depths in the ontology and the minimum of depth level given by the expert. Yet, it is really hard to represent the minimum of abstraction level based on one minimum of depth, especially when concept semantics are very dissimilar at one level. For instance, if, at a level l in the ontology, we represent different types of concepts, *e.g.*, the *16sRNA* protein and the *bootstrap>1000* parameter, then, the $min_level = l$ means that the expert demands more specific concepts than both of the 16sRNA concept and the range of $[1000, +\infty)$ one at the same time. One simple depth level threshold might not be a very conclusive criteria to measure the *different types of interrestingess* for users (according to the different semantics in workflows, *e.g.*, tasks, metadata, parameters).

Moreover, recommendation-based validation experiments would present more interesting results to study the interrestingess of the generated patterns. The ranked abstract workflows generated, here, could be used to suggest the next tool/data to the user who might got lost after some number of steps. This could be the ultimate validation process. To that end, the system has to match the partial workflow corresponding to the user session against (the prefixes of) the abstract patterns from our pattern base. The remaining part (suffix) of a matched pattern represents a potential hint to the way the user session might continue from that step on. More specifically, if we take *Example 2.* as an illustrative basis, we can apply the pattern p_2 (see section 5.5.3) since it matches the sequence of the concrete workflow $s_2 : \langle \{Histatin\,3\}, \{ClustalX\} \rangle$. s_2 matches with p_2 on the data type *Protein Sequences*' and in the alignment method too ('*Multiple Sequence Alignment*'). Thereof, from p_2 we can recommend to use the *DAMBE* (Xia

et Xie, 2001) program for instance, as a '*Maximum Likelihood Inference*' approach. With a concept-based recommendation, the system could work on different granularity levels and, hence, could offer more flexibility to users. Indeed, instead of objects as concrete tools and resources from concrete workflows, we suggest a set of sorted objects and links to assist users with more semantically rich recommendations. Semantic suggestions could guide novices and even experts to compose more complex pipelines by presenting likely compositions. Users shouldn't be worried about very specific details (software versions, very specific parameters, technical supports, *etc.*) while building their workflows.

Acknowledgment

This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D.

5.7 Vers un système de recommandation

5.7.1 Une nouvelle approche de recommandation

Nous présentons dans ce qui suit une esquisse de solution pour la construction d'un système de recommandation basé sur le contenu des motifs abstraits générés. Un état de l'art des approches de recommandation existantes est présenté dans (Aggarwal, 2016) – voir aussi (Nguyen *et al.*, 2014; Chen *et al.*, 2015a; Tarus *et al.*, 2017). Les systèmes de recommandations actuels peuvent être classifiés dans deux grandes catégories : soit les systèmes de recommandation *orientés items* ("object-based recommender systems") et les systèmes *orientés concepts* ("concept-based recommender systems"). Dans la première catégorie, les objets fréquents utilisés par un utilisateur U sont recommandés lorsqu'au moins un de ces objets est utilisé par le nouvel utilisateur. Dans les systèmes orientés-concept, les supports (fréquences) des objets sont interprétés par les supports de leurs concepts (classes d'objets) à différents niveaux d'abstraction. Ensuite, les instances des concepts qui apparaissent ensemble sont recommandées. Cependant, les systèmes orientés concept se basent seulement sur la taxonomie de concepts afin de suggérer une classe d'objets. Nous proposons, ici, une stratégie de recommandation basée, à la fois, sur les concepts et les propriétés (relations) dans les flux de travaux. En particulier, nous nous intéressons à la recommandation de/des items (objets - selon les contraintes de leurs liens) de la prochaine transaction (étape) en se basant sur des motifs de flux de travaux générés à partir de la littérature.

Notre stratégie se présente comme suit :

1. À partir d'une base de motifs fréquents $\mathcal{F}_\sigma$, une base de flux de travaux $\mathcal{D}_W$ et une base de test $\mathcal{T}_W$, nous recommandons pour chaque scénario un ensemble d'items pour la prochaine étape, *i.e.* en "cachant", chaque fois, le restant des transactions à partir d'une transaction ≥ 2 pour chaque $w \in$

$\mathcal{T}_W$[22]. Par exemple, pour le flux de travail test $w_0 \in \mathcal{T}_W = \langle \{KCC2D\}, \{BLAST\}, \{ClustalX\}, \{PhyML, JTT\} \rangle$ de taille 4, nous générons les deux scénarios suivants : (1) $w_{01} = \langle \{KCC2D\}, \{BLAST\} \rangle$ et (2) $w_{02} = \langle \{KCC2D\}, \{BLAST\}, \{ClustalX\} \rangle$. Le problème de recommandation s'agit de prédire les items de l'étape suivante d'un scénario.

2. Trouver tous les appariements (*matchings*) entre le processus en cours d'exécution (le scénario $w \in \mathcal{T}_W$) et les préfixes A des motifs fréquents sous la forme d'associations (A, B) (où A est l'ensemble de concepts du préfixe et B est la décision sur la prochaine transaction (étape) en respectant les contraintes relationnelles des concepts de A et B dans w. Ainsi, pour chaque motif p_i, nous créons les règles r_i de prémisse A_i et de conclusion B_i.
3. Trouver tous les objets o_{r_i} dans la base de connaissances $\mathcal{D}_W$ qui, en les ajoutant à $A_i(w)$ (le préfixe maximal de p_i qui s'apparie avec w_j), restent encore appariés avec B_i.

Afin d'expliquer notre approche, nous prenons l'exemple de recommandation décrit dans le tableau 5.2. Soit la base de flux de travaux $\mathcal{D}_W$ décrivant les 3 premiers flux d'analyse phylogénétique du tableau. Soit le scénario de recommandation décrit dans $w_{ID} = 4$ (noté aussi w_4). Ce scénario simule le fait qu'un utilisateur U s'arrête devant l'exécution du programme d'alignement '*TCoffee*' (Notredame *et al.*, 2000).

Nous décrivons dans l'algorithme 6 les étapes de recherche des objets à recommander o_r en testant chaque fois l'appariement du couple (p, w) et en mettant à jour la liste des objets à recommander pour tous les motifs $p \in \mathcal{F}_\sigma$. En premier

---

[22]C'est généralement à partir de la deuxième étape (une fois que les données sont collectées) que l'utilisateur commence à poser des questions sur la/les façons optimales qui devraient achever son analyse.

Tableau 5.2: Exemple de base de flux de travaux.

w_{ID}	Sequence d'objets	Ensemble de liens
1	$\langle$ {*KCC2D, GO0060048*}, {*ClustalX*}, {*PhyML*} $\rangle$	{ (1, *isDataInAlignment*, 3), (1, *isDataInInference*, 4)}
2	$\langle$ {Myosin-7, GO0060048}, {*ClustalW*}, {*PhyML*}, {*PhyML, >1000Bootstraps*} $\rangle$	{(1, *isDataInAlignment*, 3), (1, *isDataInInference*, 4), (1, *isDataInValidation*, 5)}
3	$\langle$ {*CAV3*}, {*MrBayes*} $\rangle$	{(1, *isDataInAlignment*, 2), (1, *isDataInInference*, 3)}
4	$\langle$ {*NKX23, GO0006936*}, {*TCoffee*}, ...? $\rangle$	{(1, *isDataInAlignment*, 3), ...?}

lieu, l'algorithme recherche les images du préfixe A dans w s'appariant avec le flux de travail w (voir la ligne 7 et ensuite l'algorithme 7 qui est très similaire à l'algorithme FINDMATCHINGWORFKLOWS() de test d'instantiation basé sur la nouvelle structure d'appariement dans l'algorithme 1). Ainsi nous construisons la liste $A(w)$ contenant les appariements A(w). Ensuite, nous calculons les indices du début du suffixe B, par rapport à w à l'aide de la fonction TESTNEXTSTEP(), s'ils existent (voir ligne 8). Finalement, la procédure FINDNEXTMATCHINGS() est appelée afin de calculer la liste des objets o_r à partir de la base de connaissances $\mathcal{D}_w$ (voir ligne 9). En fait, cette liste correspond à l'ensemble des images qui s'apparient avec B selon la prémisse A dans $\mathcal{D}_W$.

La recherche du préfixe dans un motif revient à chercher tous les appariements en concepts et en relations avec les items du w. L'algorithme 7 cherche cet appariement maximal. À cette fin, nous sauvegardons dans les variables $posIMax$, et $imgsPot$ la plus grande position de tous les appariements déjà testés ainsi que les

Algorithm 6 Get recommended objects for w from closed patterns

1: **procedure** GETRECOMMENDATIONS($\mathcal{F}_\sigma$: the set of frequent patterns from $\mathcal{D}_W$; w : recommendation scenario from $\mathcal{T}_W$)
2: $o_r \leftarrow \emptyset$; ▷ *recommended objets*
3: **for each** $p \in \mathcal{F}_\sigma$ **do** ▷ *for each frequent pattern*
4: $A(w) \leftarrow \emptyset$; ▷ *premiss of a pattern which matches with the workflow scenario*
5: **if** $|p.\zeta| < |w.\zeta|$ **and** $|p.\Theta| < |w.\Theta|$ **then** ▷ *if the pattern is larger then the workflow*
6: **return** $A(w)$; ▷ *break*
7: $A(w) \leftarrow$ FINDPREFIXMATCHING(p, w) ; ▷ *find the max. prefix*
8: **if** $A(w) <> \emptyset$ **and** TESTNEXTSTEP($B \leftarrow p - A$) **then** ▷ *test if the next step (conclusion) is plausible*
9: $o_r.add$(FINDNEXTMATCHINGS(A, B, $\mathcal{D}_W$)) ; ▷ *add the conclusion matching (from the learning set) as a recommended object*
10: **end if**
11: **end if**
12: **end for**
13: **return**(o_r) ;
14: **end procedure**

images associées à ces appariements. Lorsqu'on remet en cause un appariement (si l'indice du dernier concept est supérieur à $posIMax$), nous mettons à jour les valeurs de $posIMax$ et $imgsPot$ (voir les lignes 9 à 12). L'algorithme se termine soit une fois qu'un appariement complet est retrouvé (voir la ligne 28), soit que tous les appariements possibles ont été testés (voir les lignes 6 à 7). Dans les deux cas, la position de la dernière classe de l'appariement maximal (s'il existe) est

Algorithm 7 Find the maximal matching between w and p

```
1: procedure FINDPREFIXMATCHING(p; w)
2:     posIMax, i ← 0; ▷ maximal position of a matching (image)
3:     imgsPot ← FINDALLIMAGES(p, w); ▷ a table containing all the potential matchings of each concept
4:     while i < |p.ζ| do                          ▷ iterate though the concepts of p
5:         if imgsPot[i].next ==false then          ▷ if there is no matching to verify
6:             if i == 0 then
7:                 return (posIMax, Images)                                  ▷ fail
8:             else
9:                 if i > posIMax then    ▷ if a max. image (object) is found ▷ success
10:                    posIMax ← i;
11:                    Images ← imgsPot.GETELEMENTS(posImax − 1);    ▷ get the objects matching with the last concept
12:                end if
13:                imgsPot[i].BEFOREFIRST();
14:                i − −;                              ▷ get the other images as well
15:            end if
16:        end if
17:        if i > 1 and imgsPot[i].NEXT() < imgsPot[i − 1].CURRENT() then ▷ if the next matching is equals to the current matching
18:            break;
19:        end if
20:        if TRIPLEMATCHING(p, w, i, imgsPot, imgsPot[i].NEXT()) then ▷ verify if the matching is hold where there is a link on the matched position
21:            i + +;
22:        end if
23:    end while
```

24: **if** $i == |p.\zeta|$ **then** ▷ *if we reach the end of the pattern concepts*
25: $Images \leftarrow imgsPot.\text{GET}()$;
26: $posIMax \leftarrow |p.\zeta|$;
27: **end if**
28: **return** $(posImax, Images)$; ▷ *return the max position (prefix) and its images*
29: **end procedure**

retournée conjointement avec toutes les images associées.

Étant donné la base de donnée D_W (décrite dans les trois premières lignes du tableau 5.2), la base de motifs fréquents $\mathcal{F}_\sigma$ (décrite dans le tableau 5.3 et les instances de chaque concept sont décrites dans le tableau 5.4), et un scénario de recommandation w_4 (dernière ligne du tableau 5.2), la tâche de recommandation consiste à trouver tous les motifs, dont les préfixes maximaux A qui couvrent w_4, et de lui suggérer des instances à partir des suffixes B s'appariant dans la base de connaissances $\mathcal{D}_W$. Soit, par exemple, le motif p_2 dans le tableau 5.2. L'algorithme 7 trouve, tout d'abord, le prefix maximal entre p_2 et w_4. Après avoir déterminé la liste d'images potentielles de chaque concept dans p_2 (*Protein*, *MuscleContraction*, *MSA* et *ML*), soit dans notre cas les positions $imgsPot = [\{\{1\},\{2\}\},\{\{3\}\},\{\{\}\}]$ dans w_4, l'algorithme vérifie bien si la position maximale potentielle correspond au dernier concept apparié. Dans notre cas, le troisième objet '*TCoffee*' s'apparie avec l'avant dernier concept '*MSA*', ainsi, $posIMax = 3$. L'algorithme vérifie ensuite si les contraintes relationnelles sont respectées (un appariement est retrouvé entre $p_2.\theta_1$ et $w_4.\theta_1$) et renvoie le couple ($posImax$ et $Images = imgsPot$). Ainsi, le préfixe maximal retrouvé dans p_2 est : $A = \langle\{Protein, MuscleContraction\}, \{\ MSA\ \}\rangle$. L'algorithme recommande ensuite tous les objets qui s'apparient avec le suffixe de p_2 : $B = \{ML\}$ dans $\mathcal{D}_W$. Étant

Tableau 5.3: Exemple de base de motifs.

p_{ID}	Sequence de concepts	Ensemble de relations
1	$\langle \{Protein\}, \{Bayesian\} \rangle$	$\{(1, isDataInAlignment, 2)\}$
2	$\langle \{Protein, MuscleContraction\}, \{MSA\}, \{ML\} \rangle$	$\{(1, isDataInAlignment, 3), (1, isDataInInference, 4)\}$

donnée que p_2 s'apparie avec w_1 et w_2 dans la base de connaissance, l'algorithme suggère, ainsi, la liste des objets $o_{r_2} = \{PhyML, > 1000 Bootstraps\}$ pour w_4.

Tableau 5.4: Mise en correspondance concept/instance.

Niveau	Concept (classe)	Objets (instance)
3	*MuscleContraction*	*GO0060048, GO0006936*
3	*Protein*	*KCC2D, Myosin-7, NKX23*
1	*MSA*	*ClustalX, ClustalW, TCoffee, ALIGN*
3	*ML*	*PhyML, RAxML*
3	*Bayesian*	*MrBayes, PhyloBayes*

5.7.2 Stratégie d'évaluation

Nous revenons ici sur notre question principale : comment peut-on valider les motifs abstraits générés dans le cadre de la recommandation basée sur le contenu ?

Étant donnée que très peu de données de validation sont disponibles sur le Web et que notre système n'est pas encore déployé dans une infrastructure en ligne, nous adoptons une stratégie de validation hors-ligne (Shani et Gunawardana, 2011) qui cherche à valider les motifs générés sur un ensemble de flux de travaux déjà construits qui servira à créer des scénarios de recommandation. Nous reprenons ainsi la base d'entrainement $\mathcal{D}_W$ (à partir de laquelle nous avons généré les asso-

ciations) et une base différente de test $\mathcal{T}_W$ extraite manuellement par un expert. Chaque règle d'association $r_i = (A_i, B_i)$ générée à partir d'un motif $p_i \in \mathcal{F}_\sigma$ servira à recommander un ensemble d'objets (items) o_{r_i} à partir de $\mathcal{D}_W$ pour chaque scénario $\in \mathcal{T}_W$. Finalement, la liste de tous les items suggérés, par toutes les associations possibles, est recommandée. Cependant, cet ensemble peut présenter un grand nombre d'items (selon le nombre d'appariements valides dans $\mathcal{D}_W$). Ainsi nous choisissons de ne présenter que les meilleurs top-k items selon la "pertinence" des associations impliquées dans une recommandation.

Top-n associations

Ici les associations sont triés selon leur degré de *confiance* et de *support* dans la base de connaissances $\mathcal{D}_W$. La confiance d'une recommandation peut être définie par la "certitude" du système dans ses recommandations générées (Bogers et Van Den Bosch, 2011). Ainsi le degré de confiance donnée à une règle d'association peut être mesurée par son indice de support (Agrawal et Srikant, 1994). La confiance détermine la fréquence de la cooccurrence des items de B dans la liste des transactions contenantes A. La définition formelle de la confiance est :

$$conf(r_i) = conf(A_i, B_i) = \frac{supp(A_i.B_i)}{supp(A_i)}. \tag{5.3}$$

Nous rappelons que le support d'une association (Agrawal et Srikant, 1994) définit la fréquence d'une association dans D_W :

$$supp(r_i) = supp(A_i, B_i) = \frac{supp(A_i.B_i)}{|D_W|}. \tag{5.4}$$

La mesure de confiance, ainsi que celle de support, sont utilisées dans une méthode heuristique d'agrégation de prédictions dans le cas où plusieurs règles r recom-

mandent le même objet o pour la prochaine étape dans w. Ainsi, nous proposons d'utiliser la fonction $MPR(r_i)$ (Mean Percentage Ranking) (Hu *et al.*, 2008) qui combine les alternatives de tri de motifs selon la confiance et le support de chacun.

$$MPR(r_i) = \frac{\sum_{r_i} u_{r_i} \overline{rank_{r_i}}}{\sum_{r_i} u_{r_i}} \tag{5.5}$$

Ici, u_{r_i} (utilité[23]) indique si une règle d'association r_i est utilisée dans un scénario de recommandation ou non et $\overline{rank_{r_i}}$ dénote le rang centile de chaque r_i dans la liste ordonnée des associations (selon la confiance puis le support). Des valeurs faibles du MPR ($< 50\%$) sont désirées. $\overline{rank_{r_i}} = 0\%$ indique que la règle r_i est située dans le top de la liste.

Top-k recommandations

Afin de mesurer la pertinence (ou non) des items recommandés, nous utilisons les métriques issues de la recherche d'information comme la Précision et le Rappel (Schütze, 2008). La Précision est la fraction des items recommandés qui sont *pertinents*. Étant donné que le nombre de ces items dépend fortement de la taille de l'ensemble de toutes les données "valides" possibles (ou au moins du *ground-truth*), nous ne prenons ici que les top-k meilleurs objets o (Deshpande et Karypis, 2004) en se basant sur les confiances des associations qui sont recommandés o. Par

[23]L'utilité d'une association est une mesure additive obtenue par la somme des recommandations individuelles. Nous utilisons ici la forme la plus basique de l'utilité : nombre de fois que le système de recommandation utilise une règle. Le lecteur peur se référer au travail de Aggarwal (2016) pour de plus amples informations.

conséquent, les items sont ordonnés de la façon suivante :

$$rank_{objet}(o) = \max(\frac{(100 - MPR(r_i)) \times |o|}{|o_{r_i}|}) \tag{5.6}$$

Le rang d'un item o correspond au maximum des rangs de règles d'associations aux top-n (voir la section précédente) qui recommandent le même item o. Nous favorisons ici les associations qui recommandent le moindre nombre d'items. La fraction $|o|/|o_{r_i}|$, où $|o|$ est le nombre de règles recommandant l'objet o et $|o_{r_i}|$ est le nombre d'objet recommandé par r_i, représente le poids, c'est à dire la fréquence d'items recommandés par la règle qui a la plus grande confiance.

Précision et Rappel

Admettant qu'un expert a fixé le paramètre k d'objets à recommander pour un scénario donné. Notons $\mathcal{S}(k)$ l'ensemble contenant les k items : $|\mathcal{S}(k)| = k$. Soit $\mathcal{G}$ l'ensemble d'items pertinents contenant les objets positifs dans $w \in \mathcal{T}_W$ pour la prochaine étape (la partie cachée d'un scénario de recommandation). Par exemple, pour le scénario $w_{02} = \langle \{KCC2D\}, \{BLAST\}, \{ClustalX\} \rangle$ extrait à partir de $w_0 \in \mathcal{T}_W = \langle \{KCC2D\}, \{BLAST\}, \{ClustalX\}, \{PhyML,\ JTT\} \rangle$, on obtient $\mathcal{G} = \{PhyML,\ JTT\}$. Ainsi, la *Précision* des k-items est définie par le pourcentage des objets recommandés qui sont effectivement positifs (consommés par le flux de travaux test $w \in \mathcal{T}_W$) :

$$Precision(k) = 100.\frac{|\mathcal{S}(k)|}{|\mathcal{G}|} \tag{5.7}$$

La valeur de la *Précision* évalue le système de recommandation dans le contexte d'un seul scénario de recommandation. Le *Rappel* défini le pourcentage des positifs du *ground-truth* qui ont été recommandés par le système dans la liste de taille k.

Cependant, le *Rappel* n'est pas utile dans la recherche d'information si nous ne connaissons pas l'ensemble de tous les items pertinents possibles pour un flux de travail. Ainsi, nous prenons en considération le biais qui vient avec la définition du Rappel dans un ensemble de test. De plus, étant donnée que l'ensemble d'items $\mathcal{G}$ peut contenir à la fois des objets et des métadonnées sur ces objets, par exemple le nom d'un logiciel (*e.g. PhyML*) et/ou ses paramètres (*e.g. JTT*). De ce fait, nous définissons ici deux façons pour calculer le nombre de *hits* (vrais positifs) dans l'ensemble du *gound-truth* afin de d'identifier si notre système recommande le bon logiciel avec les bons paramètres. Soit $\mathcal{G}^*$ l'ensemble des objets pertinents excluant les objets de types "métadonnée" :

$$Rappel_1(k) = 100.\frac{|\mathcal{S}(k) \cap \mathcal{G}^*|}{|\mathcal{G}|} \tag{5.8}$$

$$Rappel_2(k) = 100.\frac{|\mathcal{S}(k) \cap \mathcal{G}|}{|\mathcal{G}|} \tag{5.9}$$

Tandis qu'un compromis naturel existe entre la *Précision* et le *Rappel*, qui n'est nécessairement pas un compromis non monotone, la $F - Mesure$ représente une façon d'agrégation des deux mesures par leur moyenne harmonique :

$$F - mesure = 2 \times \frac{Precision \times Rappel}{Precision + Rappel}. \tag{5.10}$$

5.7.3 Résultats préliminaires

Afin de tester le pouvoir prédictif de notre système de recommandation, nous avons pris l'ensemble de flux de travaux extrait à partir des textes afin de générer des motifs abstraits fréquents et ainsi tester ces derniers sur un ensemble de flux de travaux validés annotés par un expert. Nous rappelons que ce lot se compose

de 500 flux de travaux (avec une moyenne de 12 objets et 6 liens) et le lot extrait à partir des textes se compose de 2 783 flux de travaux (avec 11 objets et 7 liens en moyenne). Nous avons testé les tops k meilleurs recommandations en reprenant notre modèle basé sur les motifs extraits. Afin d'évaluer la sensitivité de notre algorithme de recommandation, nous prenons différentes valeurs de k : 5, 10, 20, 30, 40 et 50 (voir figure 5.6). Comme déjà mentionné dans la section précédente, l'exactitude de notre approche est sensible à la définition même du rappel (voir l'équation 5.9) et ainsi aux données du lot de test.

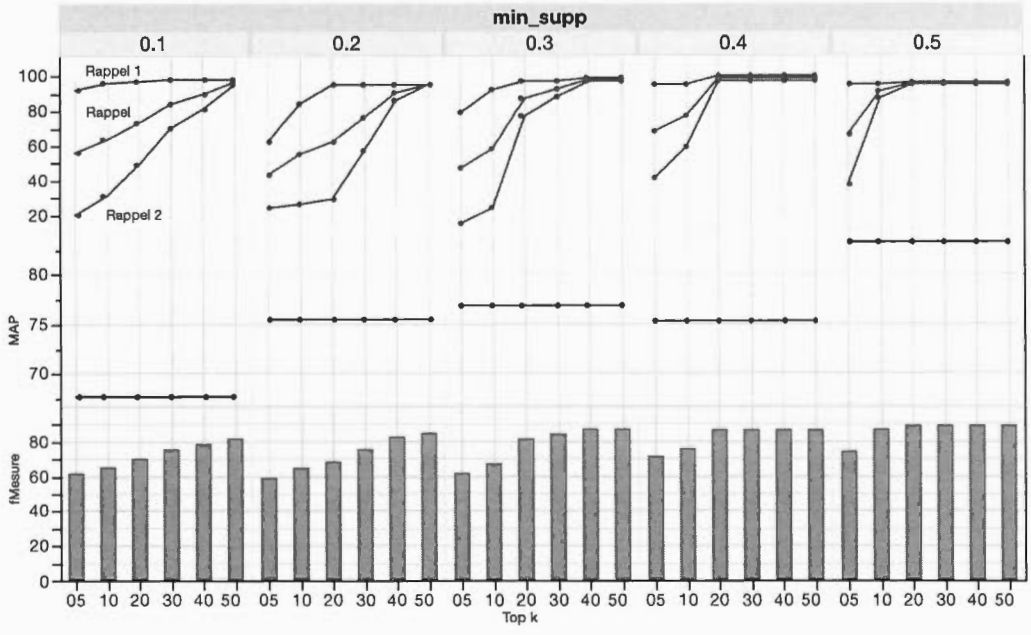


FIGURE 5.6: Qualité de Top k meilleures recommandations.

Nous remarquons qu'à partir des expérimentations, les taux de Rappel, de Précision et ainsi la F-Mesure baissent en diminuant le seuil de supports σ (min_supp) à l'exception de σ=0.3. Les valeurs élevées du seuil représentent des motifs génériques couvrant la majorité des classes d'où les taux élevés du Rappel. De plus, augmentant la valeur de k revient à améliorer les performances du système. Nous remarquons que d'autant plus que σ est élevée d'autant k deviendra plus stable

avec des valeurs faibles. Par exemple, pour $k > 10$, le taux de Rappel devient stable (92%) avec σ=0.5 et σ=0.4. La différence entre les deux types de Rappel est moindre pour des valeurs élevées de k mais elle est frappante pour $k = 5$. Cela revient à la définition stricte du Rappel (où plusieurs positions sont à recommander chaque fois). Pour le calcul de la F-Mesure, nous ne prenons que la moyenne des deux Rappels. Ici, la moyenne des fMesures s'améliore progressivement par 3.75%, 4.75%, 5.5%, 2.75% et 3.5% en variant k de 5 à 50 respectivement. Tout en gardant des taux de fMesure raisonnables (>60%), k reste un paramètre important dans l'amélioration des performances de notre système. Nous rappelons que la sélection des top k est basée sur les confiances des règles générés et donc elle est sensible aux données d'entrée.

Nous présentons dans la figure 5.7 les distribution de la fonction d'agrégation de la pertinence des motifs MPR. Nous rappelons que cette fonction trie les associations selon leurs degré de confiance et de support. La première figure (a) montre la corrélation entre le MPR, la Précision et la fréquence d'utilisation de motifs (utilité) pour un seuil de support $\sigma = 0.5$. Dans ce cas, peu de motifs sont générés (12 motifs) et les meilleurs motifs (top $n = 40\%$) mènent à recommander des objets avec une Précision élevée de 85%. Les figures (b), (c), (d) et (e) décrivent la même évolution de MPR, Précision et Rappel. Diminuer σ tend à baisser légèrement la qualité des top n motifs tout en gardant des valeurs satisfaisantes de Rappel et de Précision. De plus, la couverture (utilité) des motifs baisse en diminuant la valeur de σ. Les meilleures motifs pour $\sigma = 0.1$ (top 8% $\approx$ 70 motifs) présentent une moyenne de fMesure ~86.5%. Ainsi, nous constatons que les performances de notre système sont à la fois stables (entre 82% et 89%) et satisfaisantes (>82%) en termes de *fMesure*, tant pour des motifs génériques que pour des motifs spécifiques. Cependant, afin d'obtenir une idée plus claire de la qualité des recommandations, la façon la plus directe est de mesurer l'efficacité

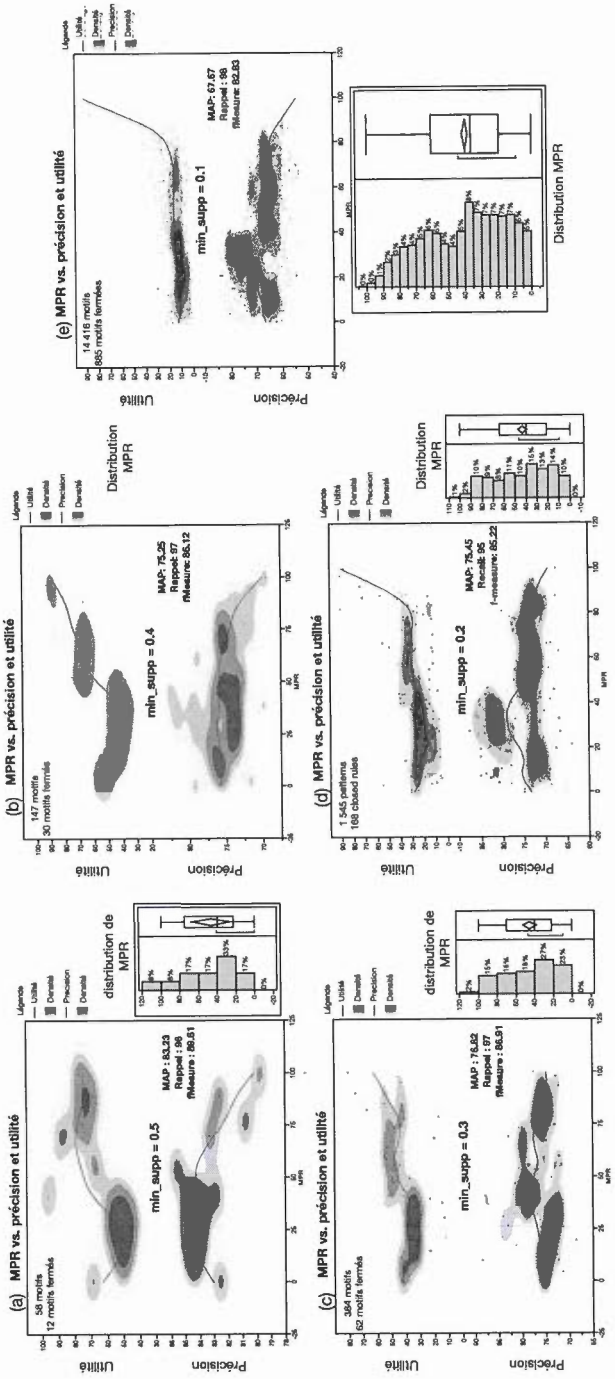


FIGURE 5.7: Qualité des Top n meilleurs motifs en se basant sur leur rang centiles (MPR).

de notre système dans son utilisation avec différents utilisateurs. Néanmoins, une telle étude est difficile à implémenter : une communauté assez large doit avoir accès à notre infrastructure en ligne. Notre stratégie de validation hors ligne vient avec l'avantage de l'utilisation d'une base de connaissances représentant l'historique des meilleurs pratiques phylognétiques dans la littérature du domaine. Toutefois, l'étude de l'efficacité vient souvent avec les biais liés aux mesures de scallabilité, couverture, prédiction, *etc.* Par exemple, le taux de Rappel est sensible à la représentativité des vrais positifs dans la base de connaissances. Nous trouvons dans la littérature de recommandation d'autres différentes mesures (Tarus *et al.*, 2017) (*e.g.*, calcul d'erreur quadratique, diversité, sérendipité) étudiant chacun un aspect bien particulier de la recommandation. Par exemple, la notion de diversité de la liste des top-k items mesure la "dissimilarité" entre les items recommandés et la liste $\mathcal{G}$. Il serait ainsi intéressant, dans notre cas, de proposer une mesure semblable qui étudie la diversité de classes d'items recommandés dans l'ontologie du domaine. Un algorithme de recherche du plus petit ancêtre commun entre les items recommandés peut être utilisé pour le calcul de la distance "sémantique" entre les items des deux listes.

5.7.4 Conclusion

Nous venons de montrer l'efficacité de notre approche de recommandation en termes de Précision de Rappel. En effet, l'inclusion des concepts et des relations dans le processus de recommandation nous a permis d'avoir une bonne Précision tout en raisonnant sur différents niveaux d'abstraction variables. Notre approche a l'originalité d'être basée sur une double procédure de filtrage. Au premier niveau, nous calculons un ensemble de motifs fréquents, représentant au mieux les flux de travaux extraits à partir de la littérature phylogénétique. Au deuxième niveau de filtrage, les top k prochaines sous-tâches sont recommandées. En outre, il serait intéressant, dans la continuité de ce projet, d'intégrer les motifs générés

dans une plateforme de construction de flux de travaux bioinformatiques comme Armadillo (Lord *et al.*, 2012) ou Galaxy (Blankenberg *et al.*, 2001) afin de tester l'efficacité de notre approche dans son cadre réel de recommandation en ligne.

CONCLUSION

Les flux de travaux en sciences deviennent des techniques de plus en plus exploitées et utilisées surtout par les chercheurs et les praticiens. Les plateformes actuelles de gestion de flux de travaux offrent des solutions généralistes qui ne fournissent pas de niveaux conceptuels spécifiques adaptés aux niveaux d'expertise de ses utilisateurs. Ces derniers sont généralement confrontés à des situations d'acquisition de nouvelles connaissances afin de résoudre leurs problèmes. Bien que les conditions expérimentales et le choix des logiciels d'analyses soient dépendants des ressources informatiques disponibles, souvent en relation avec le volume des données à analyser, d'autres conditions telles que la méconnaissance du problème et les biais reliés à chaque outil sont responsables de la divergence des résultats.

Dans cette thèse, nous avons proposé de résoudre un problème bien particulier d'acquisition de connaissances processuelles celui de l'analyse phylogénétique. Nous avons proposé d'utiliser une approche basée sur les ontologies dans chacune des couches, représentationnelle et fonctionnelle, de notre système `T-Gowler`. Ce dernier utilise deux types d'ontologie. Une ontologie du domaine dont le but est de représenter les données descriptives de l'analyse phylogénétique et une ontologie processuelle qui est utilisée afin de représenter les connaissances processuelles des flux de travaux. La première ontologie a été spécifiquement utilisée dans l'extraction des éléments de flux de travaux concrets à partir des textes. La deuxième quant à elle, a été utilisée dans la couche du raisonnement sur les connaissances extraites afin de générer des motifs abstraits de flux de travaux. Cependant, la fouille de motifs généralisés requiert le développement préalable d'un schéma ontologique détaillé du contenu du domaine d'application. Cette exigence peut s'avérer

contraignante dans d'autres domaines où les données théoriques ne sont pas toujours structurés ou même accessibles comme en mathématiques ou en didactique.

En outre, il est nécessaire de prendre conscience que la conceptualisation des ontologies n'est pas toujours fourni aux utilisateurs, même pour un expert du domaine. Nous pensons que l'accès aux ontologies par des utilisateurs, autres personnes que leurs créateurs, nécessite une maintenance périodique des termes, concepts et relations en langage naturel représentant les attentes pratiques de ces utilisateurs. Ainsi, soit qu'il faut séparer les termes et les concepts d'un domaine dans des terminologies ou des thesaurus sémantiques, soit s'écarter de la formalisation des ontologies en cherchant des proximités conceptuelles permettant d'en appréhender intuitivement la complexité : par exemple en utilisant des bases de graphes pour représenter des groupes de termes dans un réseau sémantique (Angles et Gutierrez, 2008; Bollacker *et al.*, 2008; Shao *et al.*, 2013).

Nous avons discuté dans le chapitre 2 de cette thèse notre première contribution principale, soit en extraction de flux de travaux concrets à partir des textes. Ce chapitre décrit la solution proposée afin d'extraire, enrichir et reconstruire les différents éléments de flux de travaux dans le texte. Nous avons proposé un langage spécifique aux flux de travaux basé sur la structure des DAGs. Ces graphes représentent les différentes activités et types de données dans des flux de : contrôle et de données. Nous avons aussi proposé un modèle d'apprentissage automatique de désambiguisation des éléments processuels dans le texte. Ce modèle a servi pour améliorer la qualité d'annotation automatique des termes et des relations dans le texte. En outre, il serait important, dans la continuité de ce projet, de développer une extension de notre modèle de désambiguïsation afin de prendre en considération un contexte plus large de termes polysémiques : soit l'analyse par section (ou paragraphe) au lieu des phrases. Il serait aussi possible d'étendre le système d'extraction avec une modélisation des paramètres dans les flux de travaux en

utilisant une approche floue de représentation de ses domaines de valeurs. Par exemple, les instances du concept *'Nombre de bootstraps'* peuvent être représentées par des "sous-domaines de valeurs" : nombre élevée, peu élevée, toutes les valeurs possibles (*all*) ou pas de valeur (*null*). Cette représentation reviendrait à enrichir la sémantique des paramètres en générant des motifs de flux de travaux plus intéressants (dans la configuration des tâches) et plus faciles à interpréter par les utilisateurs finaux.

Dans le chapitre 3 de la thèse nous avons présenté la deuxième contribution principale de notre approche soit la fouille de motifs abstraits à partir des flux de travaux extraits. Notre solution, basée sur une de fouille de données, applique une stratégie de parcours descendante en profondeur d'abord de l'espace des motifs construits à partir de l'ontologie. Nous nous sommes inspirés du langage de motifs d'Adda *et al.* (2010) et nous avons proposé une structure avancée adaptée aux flux de travaux étant des DAGs à plusieurs couches. Nous avons aussi proposée une structure d'appariement de graphes qui profite de la relation de *précédence* entre les motifs afin d'accélérer le processus de calcul de support. Par ailleurs, dans la génération des motifs candidats dans la fouille généralisée, il serait aussi important de proposer une méthode de pré-élagage de motifs afin de ne garder que les motifs maximaux (Zaki et Hsiao, 2002; Yan *et al.*, 2003).

Ensuite, dans le chapitre 4 de la thèse nous avons présenté l'architecture du système au complet dans ses deux modules d'extraction et de fouille de connaissances processuelles abstraites. Enfin, c'est dans le chapitre 5 que nous avons étudié les résultats obtenus et leurs pertinences dans le domaine de l'analyse phylogénétique. Nous avons aussi proposé une esquisse de solution pour le développement d'un système de recommandation de flux de travaux à partir des motifs abstraits générés. Nous avons pu montrer, à travers notre approche, la pertinence et le pouvoir prédictif de différents motifs fréquents dans l'acquisition automatique des connais-

sances de résolution de problèmes. La représentation de concepts et de relations ontologiques a permis de générer des motifs de meilleures pratiques de différents niveaux d'abstraction avec des taux de précision et de rappel satisfaisants. Cependant, nous n'avons pas pu valider les motifs générés dans leur *réutilisabilité* : la recommandation pour *différents* utilisateurs. Il serait intéressant de tester les modèles générés dans différents cas d'utilisation de flux de travaux, suivis par plusieurs phylogéniticiens de différents niveaux d'expertise. C'est à ce moment que nous pourrions tester et valider les différents niveaux d'abstraction utilisés dans les motifs processuels générés.

ANNEXE A

Bioinformatic workflow extraction from scientific texts using word sense disambiguation

Ahmed Halioui, Petko Valtchev and Abdoulaye Baniré Diallo

Abstract—This paper introduces a method for automatic workflow extraction from texts using Process-Oriented Case-Based Reasoning (POCBR). While the current workflow management systems implement mostly different complicated graphical tasks based on advanced distributed solutions (*e.g.* cloud computing and grid computation), workflow knowledge acquisition from texts using case-based reasoning represents more expressive and semantic case representations. We propose in this context, an ontology-based workflow extraction framework to acquire processual knowledge from texts. Our methodology extends classic NLP techniques to extract and disambiguate tasks in texts. Using a graph-based representation of workflows and a domain ontology, our extraction process provides a context-aware approach to recognize workflow components: data and control flows. In order to validate our approach, we applied our platform to a very known bioinformatic domain, *i.e.* phylogenetic analyses. An evaluation based on workflow semantic similarities on a Ground-truth proves that our approach provides promising results in the identification of semantically enriched workflows from scientific texts. Both data and implementation of our framework are available in: https://github.com/bioinfoUQAM/tgowler.

Index Terms—Workflow extraction, Word Sense Disambiguation, Ontologies, Workflow Similarity.

1 Introduction

Current workflow management systems provide a step by step guidance for users through sophisticated graphical workflow modelling [1]–[3]. Tracking these executions can be used to adapt workflows to new situations. This can be supported by reasoning methods like Process Oriented Case-Based Reasoning (POCBR) [4] which is a newly merged branch from the Case Based Reasoning (CBR) field. Because case engineering is a complicated and a costly process, automatic case-based acquisition from texts has been a major focus research in the last decade [5]–[7]. Textual CBR approaches vary widely in their scope according to their use of Natural Language Processing (NLP) methods, case representations, and are highly dependent on the intended applications. Hence, it is very difficult to reuse the proposed approaches to some problem-solving situations where a domain-specific vocabulary is used such as in Phylogenetics [8] where, as a interdisciplinary domain, it combines different vocabularies from Computer Science, Statistics, Mathematics, and Engineering to analyze and interpret biological data.

Even though Lau *et al* [9] compared three different approaches: key-based, grammar-based and machine learning, they only consider imperative sentences which cover more than 50% of the whole data and their interpretation modules are limited to the command tuple representation (A: type of the action, textual parameter to an action, the target on which to act). Liu *et* Agah [10] developed a workflow extractor trained with a few manually created positive samples. A set of extraction rules are generated and then applied to automatically crawled documents (from *e-how* Websites, *e.g.* www.wikihow.com). However, their model has only successfully extracted ~42% of these documents. Schumacher *et al.* [6] proposed to use a full fledged NLP solution to extract and enrich workflows from texts. Their frame-based information extraction approach to resolve *anaphoras*[1] [11] in texts uses *frame patterns*[2] to trigger phrases. A sequential pattern mining is also applied to mine anaphora-rules and then, search for the most frequent term cooccurrences to resolve an anaphoric reference. A similar approach is proposed in [5] to resolve "universal references" using a domain-specific ontology. Given that ontology, they map the anaphoric reference to a concept. Yet, their resolution is based on heuristics (rules) highly dependent on the domain application. Addis *et* Borrajo [12] proposed a Plan Acquisition Architecture to build an augmented plan in the form of predicates from extracted raw content by reducing the article into a list of two kinds of information: (i) input-tool (a predicate that describes the tool properties) and (ii) action (description with properties). They also use data mining and statistical algorithms to identify particular subsequences of plans to specific contexts. A recent approach to extracting situation ontology from how-to instructions is proposed in [13] where an automatic method based on syntactic patterns and a CRF (Conditional Random Fields) [14] probabilistic model are used in order to classify actions and objects such as time and place. Although their pattern-based approach achieves a high level of accuracy (>73%), its coverage is limited to rules constructed semi-automatically from small fraction of sentences (only imperative sentences) in *how-to* texts. Song *et al.* [15] proposed a novel attempt to extract procedural

• *All authors are with the Department Département d'informatique, Université du Québec à Montréal. H2X 3Y7, Québec, CANADA. E-mail: diallo.abdoulaye@uqam.ca*

1. The problem of resolving what a pronoun or a noun phrase refers to.

2. A frame-based structure defines the meaning of a term in a sentence using *slots*. Each frame slot is filled using linguistic patterns (heuristic rules) mapped to syntactic roles (subject, direct object, and indirect object) to trigger phrases in texts.

knowledge from the PubMed abstracts[3] using the TAM model - a triple combination of (Target, Method, Action) and designed a text mining method with deep language processing analysis. This analysis contains a (1) preprocessing step (syntactic and ontological tagging), (2) purpose/solution sentence classification and (3) relationship between two processes is assigned to a control flow (sequential, parallel, causal, *etc.*). [16] proposed a distant supervision approach[4] for *Cancer Pathway Extraction* from texts. The main idea is to test the significance (using a latent variable), for each sentence evoking two entities E_1 (theme) and E_2 (cause), of the relation R (between the two entities). Distant supervision is provided by enforcing during training that for each relational triple (E_1, R, E_2), at least one latent assignment is true if the database contains the relation, and none otherwise. Each instance is represented by a linear model with features over the sentence and entities. The feature weights are learned using online learning with Perceptron [17]. Yet, their semi-supervised approach is so sensitive to the starting sample. Changing the context at each iteration, while creating automatically positive examples, may generate different conflicting relations since the latent assignment is a simple comparison of R in the training set. Comparing contexts using the triple (E_1, R, $E2$) may work only for binary relations; lifting this limitation to handle n-ary and nested relations is an important future direction.

While those techniques will continue to be essential, new challenges also emerged as we are understanding more about the nature of human experience and its role in making decisions [18]. Processual knowledge, grounded in scientific texts are exploited to enhance the experience acquistion. Yet, some complex semantic ambiguities are need to be dealt with. To the best of our knowledge only anaphora resolutions have been studied in procedural texts. However, semantic ambiguities arise more difficulties: *e.g.*, word sense ambiguities. Word Sense Disambiguation (WSD) [19] is the process to identify the sense or senses of a polysemic term. Although important, WSD as an 'intermediate' task, is unlikely that anyone other than linguistics would be interested in its results. However, we found that polysemic words are also common even in very technical domains, such as phylogenetic analyses, where specific software and programs could be also used for different purposes. Generally, programs may contain multiple packages and processes and offer multiple supports. Thus, even very specific tools may be used in different contexts. Without a context-aware approach, it is very difficult to reconstruct a phylogenetic workflow from texts with a simple term-matching process. In addition, identifying activities (tasks) and organizing them in a control-flow seem to be a complicated task, especially when it can be applied in different positions in a workflow. Thereof, we tried not limit ourselves to highly structured procedural texts but to a domain specific problem to extract domain-relevant meanings from it. In this paper, we propose an ontology-based approach to extract semantically rich bioinformatics workflows from texts using a WSD technique. Here we use the domain ontology for two purposes: (1) to encode non ambiguous concepts (of different levels of abstraction) and the relations between them and thus, the domain terminology is modelled as a DAG of different level and (2) to classify terms from texts using this terminology. Its very difficult to represent the a DAG of hierarchical concepts and relations using relational databases. Section 2 presents the choices that we made to design our ontology-based extraction process. Two formalisms are presented: workflow representation and ontologies. Additionally, a running example is introduced to explain our representational choices for phylogenetic analyses. Section 3 is the main core of the paper describing in details the process of the workflow extraction from texts. A text is analyzed with a pipes-and-filters NLP chain and a dedicated WSD for workflow tagging. Section 4 presents experiments and results of the proposed processes by comparing automatically obtained workflows with an expert-curated dataset. Finally, section 5 discusses results and future work.

3. https://www.ncbi.nlm.nih.gov/pubmed

4. In distant supervision, we make use of an already existing database, such as KEGG (http://www.genome.jp/kegg/pathway), to collect examples for the relation that we want to extract. These examples are used to automatically generate the training data.

2 Formalisms for Enriched Workflow Representations

2.1 Workflow representation

Traditionally, workflows are *"the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules"* [1]. In general, workflows consist of a set of *tasks* (activities) combined with a *control-flow* such as sequences, parallel (conjunction) or alternative branches (disjunction), as well as repeated executions (loops). We are interested, here, in workflows as sequences and concurrent runs of program tasks. These latter consume *data items i.e.* input sources and output products. The *data-flow* represents interactions (relationships) between them (data items) and tasks. In addition, workflow items are annotated using semantic descriptors (metadata) for domain constraints and data provenance. Different types of workflow representations are proposed in the literature of POCBR [4]. Petri Networks [20] are the most popular design representing the dynamic aspect of processes. However, the representational bias (as in other languages) of such models should be considered while extracting workflows, *e.g.* multiple instantiations of a task or sub-task. Enriched workflows with semantics describing task parameters and metadata on data items are also to be considered while representing workflows in order to make the workflow more expressive and hence, easily to be reused in similar cases.

In this study, we illustrate our approach in the domain of phylogenetics [8]. Inferred from nucleic acid (DNA or RNA) or protein sequences, phylogenetic relationships represent the evolutionary history of a group of species. During phylogenetic steps, multiple bioinformatics programs (software) are used. Here, the tasks represent phylogenetic *programs* and their combination of *parameters* and data items referring to input/output *data* are enriched with *metadata*.

An example of a phylogenetic analysis of *Hemagglutinin (HA)* protein sequences, extracted from the article text of [21], is illustrated in Fig. 1. This workflow represents the phylogenetic solution to study the evolutionary history of

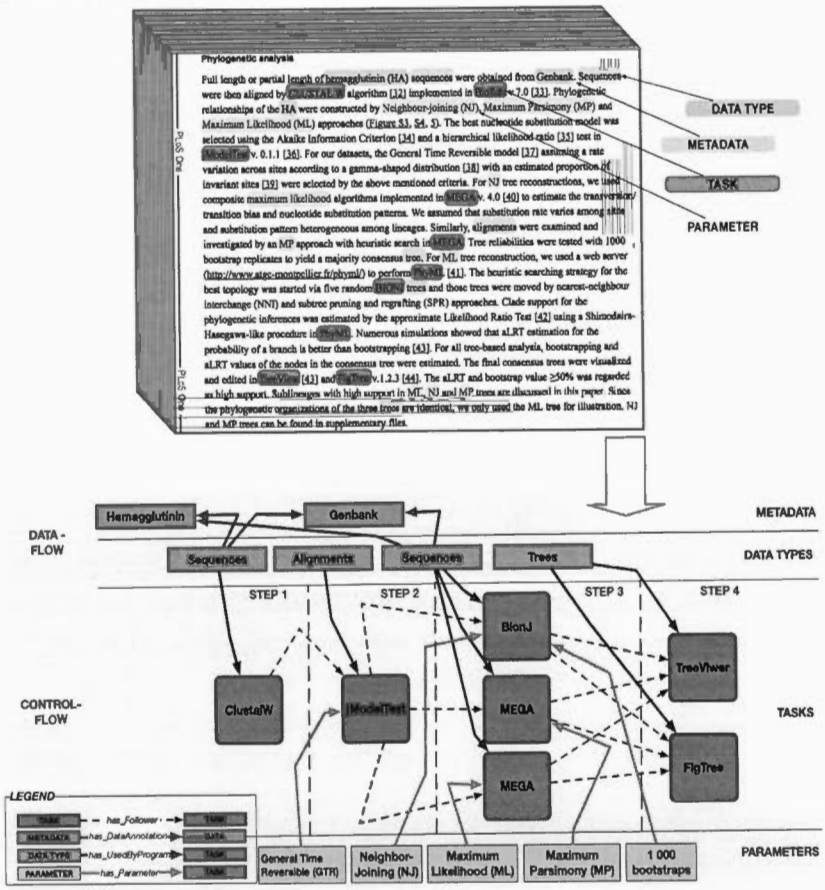


Fig. 1. Example excerpts from the article of Liang *et al.* [21]. We represent the workflow in four different layers. The first layers describe data and metadata annotations. The tasks layer represents the control flow and the last layer describes parameters. Relations between the different objects at each layer describe the control flow (links between tasks), the data flows (links between data and tasks) and other semantic flows (between data and parameters and/or metadata).

HA proteins according to the following steps: (1) multiple sequence alignment, (2) model representation, (3) tree inference and (4) result visualization. Each step is composed of partially ordered program tasks dependently linked with each other. These dependencies represent control and data flows enhanced with semantic annotations. For instance, BionJ [22], a neighbour joining program, gets *Hemagglutinin* sequences collected from the Genbank database[5] as input and produces phylogenetic trees. Duplicates in workflows represent different instantiations of one task. For instance, the first $MEGA$ [23] instance (at the $3r^{rd}$ step) is used with different parameters than the second instance at the same step. $MEGA$ is used with both, a *Maximum Likelihood* (*ML*) method and a *Maximum Parsimony* (*MP*) one [24].

Definition 2.1 (Workflow data language). Given a universe of task items (tasks and parameters) O^T and data items (data types and metadata) O^D, *i.e.* $\omega^O = O^T \cup O^D$, a workflow language Δ_W defines workflow records of partially ordered sequences composed of inter-related object labels $o \in \omega^O$. A workflow $w \in \Delta_W$ is a direct acyclic graph $w = (\zeta, \Theta)$ where $\zeta = \langle\{O_1, O_2 \ldots, O_n\}\rangle$ is the sequence of object transactions $O_i = \{o_{i,j}\}$, where j ranges from 1 to the size of O_i (the number of element at the ith step). In turn, $\Theta = \{\theta_1.\theta_2, \theta_k, \ldots, \theta_p\}$ is the set of triples where $\theta_k = (i_1, l, i_2)$ represents the link of type $l \in \omega^{\mathcal{L}}$ between the object at position i_1 in $w.\zeta$, *i.e.* $w.\zeta[i_1]$ and the object at

5. https://www.ncbi.nlm.nih.gov/genbank/

the position i_2, *i.e.* $w.\zeta[i_2]$.

Fig. 2 shows a sub-workflow w_1 representation from the previous example in the figure 1. Phylogenetic programs and parameters and their input/output data and metadata are represented in a partially ordered sequence of object transactions (steps) and a set of links between them. For example, *Hemagglutinin alignments* ($o_{1,2}$) are the input (see θ_3) in the neighbour joining program *BionJ* ($o_{1,4}$) generating *phylogenetic trees* (see θ_4). The order between phylogenetic programs represents the partial execution of phylogenetic steps. The "parallel" execution of programs is represented by the transactions in $w.\zeta$, *e.g.* $\{BionJ, MEGA, MEGA\}$ are programs in the same step.

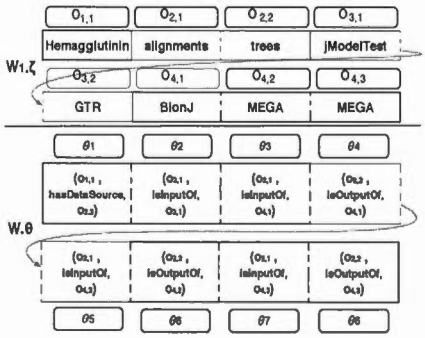


Fig. 2. Example of workflow data representation. A workflow w is a set of partially ordered objects $w.\zeta$ and a set of link triples $w.\Theta$. Here, items belonging to the same set are separated with dotted lines.

2.2 Ontology representation

The universe of objects $\omega^O = O$, links $\omega^{\mathcal{L}} = \mathcal{L}$ and their corresponding labels (references) is defined in the domain ontology Ω.

Definition 2.2 (Universe of objects and links). Let $\Omega = \langle O, C, \mathcal{L}, \mathcal{R}, \leq_\Omega, \rho \rangle$ be an ontology where O is the set of all objects, *i.e.* individual tasks and data types, C the set of all concepts, $\mathcal{L}$ the set of all links between objects, and $\mathcal{R}$ the set of all properties (relations) between concepts. $H_C = \langle C, \leq_\Omega \rangle$ and $H_\mathcal{R} = \langle \mathcal{R}, \leq_\Omega \rangle$ denote the concept and relation hierarchies of the ontology where $\leq_\Omega$ denotes the *rdfs:subClassOf* hierarchical relationship. Objects $o \in O$ and links $l \in \mathcal{L}$ are instances of concepts $c \in C$ and relations $r \in \mathcal{R}$, respectively. The ternary relation $\rho \subseteq C \times \mathcal{R} \times C$ is made of triples $c_1 \times r \times c_2$ expressing the incidence between a relation r and its domain c_1 and range c_2 concepts.

Fig. 3 shows a sample from the ontology of the phylogenetic analyses domain PHAGE [6].

3 Extraction Process

The problem of workflow extraction from texts consists of (1) recognizing workflow elements, *i.e.* O and $\mathcal{L}$ – object and link terms in the text – and (2) classify them to their ontological classes, *i.e.* C and $\mathcal{R}$ – concepts and relations – , respectively. The following sections describe the different sub-tasks of our workflow extraction solution (see *Fig.* 4). Following the standard NLP pipeline, the very first steps are going from the morphological analysis of the text to the identification of syntactic classes of terms and sentences. Section 3.1 quickly goes through those steps. We present some commonly used techniques and tools and we discuss how we adapt them to our issues. Taking the output of those steps, section 3.2 describes our solution to recognize domain terms based on the concepts and relations from a domain ontology Ω. Section 3.3 proposes a way to recognize word sense ambiguities using the hierarchy of concepts in Ω. Our word sense disambiguation (WSD) technique is supported by a super distant machine learning approach where the learning set is generated automatically using ontological rules. Section 3.4 describes the way we create a training set to build a WSD model and a link extraction one. Sections 3.5 presents the features and the classifier that we use to learn WSD and relationships in texts. Section 3.6 details the process to reconstruct data and control flows from the extracted and disambiguated terms and finally section 3.7 presents the similarity measure to assess automatically extracted workflows based on expert-curated ones.

3.1 Morphosyntactic annotation

The initial step segments the text into lexical items called *tokens* equivalent to "words". In general, tokenization operates tokens with white spaces between them as far as we treat English texts. However, due to the specificity of the bioinformatics vocabulary, we use a specialized biomedical tokenizer which relies on a dedicated token lattice design pattern for bioinformatics tokens [25]. For instance, "*don't*" is treated as two tokens, while the gene "*4'OMT2*" is really one. In addition, the text is segmented into sentences using punctuation-based patterns. The extracted tokens and sentences are further tagged with their appropriate grammatical and syntactic categories. At first, a Part-Of-Speech (POS) tagging is executed in order to identify grammatical classes of terms. We use the MedPost POS tagger [26] which is based on a HMM (Hidden Markov Model) model trained over 1 000 XHTML PubMed texts. Their tagger achieved an accuracy of 97.43%. Next, we use a lightweight syntactic parser, the OpenNLP chunker [27], to identify active and passive verb phrases in sentences. The depth of the "chunk tree" is strictly limited to 3 (root, phrases and POS tags). Chunker complexities are equivalent to a finite-state automaton and provide a very simple grammar which is sufficient to continue with the extraction process.

3.2 Semantic annotation

This step is about recognizing workflow items in texts: *i.e.* tasks, parameters, data, metadata and links between them. For this purpose, an ontology-based gazetteer (terminology) is used to map instances with text tokens. This step is important to any workflow extraction process. Yet, simple word matchings between ontology instances and tokens would not be sufficient to recognize workflow elements. Different syntactic and semantic ambiguities are present in

6. http://bioportal.bioontology.org/ontologies/PHAGE

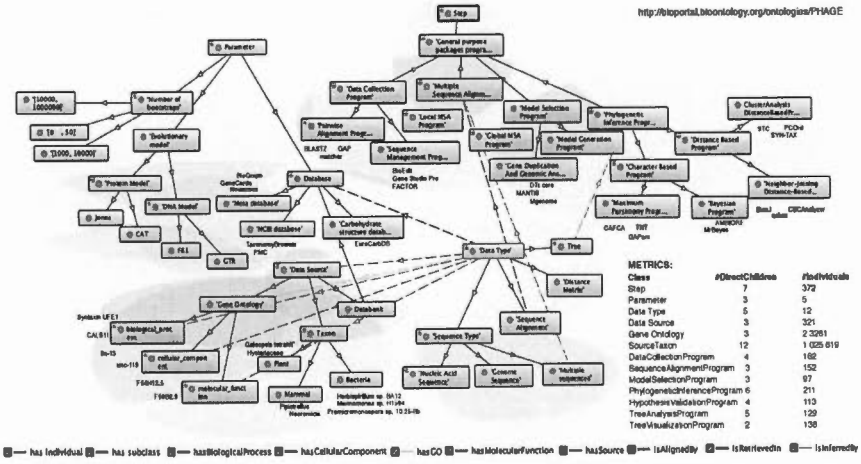


Fig. 3. A sample from the phylogenetic domain ontology PHAGE. Grey rectangles illustrate concepts, where the top ones (a) and (b) correspond to task items (steps and program parameters, respectively), (c) and (d) concepts correspond to data items (data types and metadata, respectively). Coloured dotted arcs represent either annotation or input/output properties.



Fig. 4. A sample from the proposed PHAGE ontology published in the BioPortal repository (see the link on the top right corner). Grey rectangles represent workflow concept references to: steps and programs (in the blue cloud), data types (in the green cloud), sources (in the purple cloud) and parameters (in the yellow cloud). The hard arcs represent hierarchical relations and the dotted arcs represent other semantic properties (domain constraints and input/output relations). Terms under concepts show some individuals (instances) from categories.

any natural language [28]. For this purpose, we use (1) specific JAPE[7] rules to extract and filter workflow concept and relation instances in texts and (2) a WSD model to clarify other semantic ambiguities: *i.e.* multi-class classification (see next sub-sections).

In order to filter concept bad matchings from the ontology, we use specific-domain stop-words, original XHTML markup annotations (from the texts) and regular expression (regex) patterns written in JAPE language. For example, when gene or protein names are nouns, verbs or adjectives, their probability to be exact matchings in a text are never 1 [30]. For instance, the gene term *'CAT'* is very similar to DNA codon sequences, *e.g.* in 'CAT ACT GTA ...' or to other terms such 'caterpillar' or 'cats' . In addition, some bioinformatics software names have similar issues. For example, the word *'align'* in the context of phylogenetics presents a lexical ambiguity too. Commonly, *'align'* is a frequent verb in phylogenetics texts but it is also used as a program name, that is why the probability of this term is never 1. However, the probability of 'align' to be the name of a software written in lowercase is really near to zero[8]. *Rule* 1 is an example of a pattern-based rule to eliminate program annotations where terms, as verbs, nouns or adjectives, are written in lower case. *ClassC.inst* is the instance ID in the ontology[9], *Token.category* is the POS tag of a token and *Token.orth* is its orthography. Hence, for all (most) specific program categories [10] in Ω, we build one rule for the most sensitive POS categories grounding lexical ambiguities: *i.e.* verbs (VB), nouns (NN) and adjectives (JJ) [30].

Rule R_1 :
$(\{ClassC.inst =\sim$ "$ProgramID_$",
$Token.category ==$ ("VB"||"NN"||"JJ"),
$Token.orth ==$ "$lowercase$"$\}) : class$
$-->$
$inputAS.remove(class)$;

Procedural texts describe processual events of how to complete specific tasks, hence, sentences represent (a) a

7. Java Annotation Patterns Engine (JAPE). JAPE language is a finite state transducer on the GATE platform [29] that operates over annotations based on regular expressions. Thus, it is useful for pattern-matching, semantic extraction, and many other operations over syntactic trees such as those produced by natural language parsers.

8. Authors tend to use different orthography such as uppercase and italic to distinguish program names in texts from other terms.

9. Since instances might have "synonyms" (*e.g.* 'MCMC' and 'Markov Chain Monte Carlo') or might refer to software versions (*e.g.* 'MEGA 5.1'), we choose to identify each instance with unique identifiers having the prefix as the class name in the domain ontology (*e.g.* ProgramID_9482394).

10. A most specific concept/relation is the leaf category (which has no child category) in the hierarchies H_C and H_R.

causal chain, in an (b) active voice (c) associated with task steps [31]. However, this rule is not always applied since, the phylogenetic analyses sections use both, narrative and procedural texts. Thereof, for the link recognition step, we adopt the following hypothesis: *"a relation between two concepts (domain and range) may exist in the same sentence evoking the instances of those concepts. The link is defined in the verb of this sentence"*. Since, both active and passive voices are used to describe phylogenetics events, we deal, here, with relations describing actions between *domain* and *range* concepts in both directions. For this purpose, we propose a frame-based [32] approach using the ontology triple structure $\rho = (c_1, r, c_2)$ to map and match ontological relations $r \in \mathcal{R}$ with links $l \in \mathcal{RL}$ in texts: $\theta = (o_1, l, o_2)$. Relation instances (links) are recognized based on all relation triples that exist in $\mathcal{R}$ compatible with its elements (domain, range and link type). The interpreter loads all the triples from the ontology and maps tagged concepts as domains and ranges in texts for each link. For instance, for the triple ("$BootstrapProgram$", "$hasParameter$", "$NbBootstraps$") $\in \mathcal{R}$, we search for all its domain and range terms (instances) from the same sentence in texts. For example, a sentence candidate, in this case, would be:

"We validate neighbour-joining trees by MEGA (domain) using (link) 1000 replicates (range)".

In addition, if a match is found, a set of syntactic patterns is applied for each link to filter bad matchings. For example, if the phrase voice is active, then, the verb evoking the relation domain and range might be between these terms taking into account the order of domain and range terms in the sentence (*i.e.* domain occurring before range). The rule R_2 annotates the link context in texts where *dStart*, *dEnd* and *rStart* correspond to 'S'tart and 'E'nd positions of 'd'omain/'r'ange terms in the triggered sentence.

```
Rule R2 :
(({Active Verb Phrase}) contains ({Domain} {Token}+
{Range})) : p
-->
{for(Annotation d : domain)
    for(Annotation r : range)
        if (dStart < rStart && dEnd < rStart &&
            p is active)
            outputAS.add(dStart, rEnd, "ClassR",
                         p.verb)
}
```

3.3 Word sense ambiguity recognition

The previous process of semantic annotation faces known hurdles rooted in the word sense ambiguities: *i.e. polysemy* [19]. The problem is that words often have more than one meaning (*e.g.* metonymy) and sometimes completely different meanings (*e.g.* homonyms). The meaning of a word in a particular usage can only be determined by examining its context. This is, in general, a trivial task in NLP. For example, in the following two sentences, the software MEGA [23], a multi-purposes package of programs, is used in different contexts.

- *We assess the neighbor-joining method by MEGA using 1000 replicates.*
- *Tree data was visualized using MEGA.*

In the first sentence, *MEGA* is used to validate the phylogenetic hypothesis which corresponds to the *'HypothesisValidationProgram'* concept. However, in the second sentence, the program *MEGA* is used in the visualization step which corresponds to the *'TreeVisualizationProgram'* concept. A software containing different packages (algorithms) is usually used for different purposes such as in *MEGA* [23] and *PHYLIP* [33], thus, these are referring to polysemic concepts in the domain ontology. This software-related polysemy is represented with the *multiple inheritance*[11] feature in the domain ontology. Hence, in order to identify polysemic words in the ontology, we create a latent concept *'GeneralPurposePackages'* equivalent to all ambiguous concepts while eliminating multiple inheritance in the ontology. This is done automatically while interpreting the hierarchy of concepts H_C to search for individuals (instances) belonging to multiple task concepts. For each parent concept $\hat{c}$ of a polysemic one c (where $c \prec_{sub} \hat{c}$ is a *subClassOf* RDF relation) belonging to a multiple inheritance in the hierarchy H_C, we add the equivalence relationship between c and the ambiguous concept *'Multiple Purpose Packages'*, *i.e.*: $< c > < owl : sameAs > < MultiplePurposePackages >$.

3.4 Word Sense Disambiguation (WSD)

Next, after discovering polysemic concepts in the ontology, we tag their ambiguous instances in texts. We propose, here, a supervised machine learning approach to WSD. The problem of WSD is defined as follows: *"given a set of pre-tagged polysemic words, WSD is to classify each word with its appropriate task concept (sense) with high accuracy and efficiency"*. Pre-tagged words are usually created manually, but, here, we propose to construct the WSD ground truth in a semi automatic way using the domain ontology and syntactic tags from sentences.

```
Rule R3 :
Priority : 90
(({Active Verb Phrase}) contains
({GeneralPurposePackages}) : c) : context
-->
{if(context is active &&
      (context.verb.root in VerbSet : {"valid","test",
...} || context.object.root in ParamSet : {"bootstrap",
"replicate",... } ))
    c.putFeature("resolution","ValidationProgram");
}
```

Rule R_3 shows an example of JAPE rules to generate the WSD learning set. Here, both of the syntactic tree (see section 3.1) and the semantic tags (see section 3.2) are used to clarify the polysemy of the concept *c*. The *Verbset* are stem verbs extracted automatically from the WordNet synonyms [34] of the verb in the parent concept $\hat{c}$ (*e.g. valid*) in H_C (*i.e.* from the level immediately above). The *ParamSet* contains also WordNet synonyms of the corresponding program *Parameter* terms (*e.g.* number of bootstraps in validation programs). These latter are used (if they exist) as

11. Multiple inheritance is an object-oriented feature in which a class (concept) or an object can inherit characteristics and features from more than one parent class.

hints to classify the polysemic program to its right class. Hence, the pattern R_3 verifies if the verb, in the active voice, is about testing or validating something, thereof, the *'GeneralPurposePackages'* concept is disambiguated to the *'ValidationProgram'* one. Moreover, WSD rules have priority probabilities to fire them by order (*e.g.*, the priority of R_3 is 90% among the other rules). Here, for each polysemic concept created in the previous step, we generate one WSD rule. An additional rule is added by the expert to represent the default concept if no heuristic is valid (default rule).

3.5 Learning module

In order to create a model for automatic WSD, a set of features is generated representing the dataset of already tagged words in texts. For instance, in order to classify the term 'MEGA' (*ClassC*) from the example of Fig. 5, we generate the set of features (*featuresC*) reflecting its context (see also table 1 - features 1 and 2).

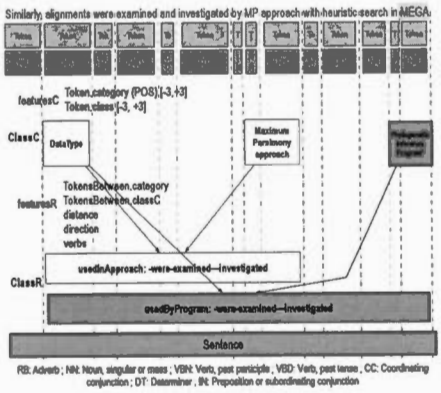


Fig. 5. An example of learning features for WSD concepts and relation extraction from texts.

In addition, in order to classify a link between the ambiguous word 'MEGA' (domain) and the data type 'alignments' (range), we define its context as the set of tokens between the domain and the range. Here, features (*featuresR*) represent the context token POS tags (*TokensBetween.category*), distance (in number of tokens between domain and range), direction (domain-to-range) and verbs between the concerned terms (see table 1 - features 3 and 7).

Since proposed features are context-based, the model representing ambiguous concept and relation terms in texts should take into account the type of these (features) representations. Hence, we chose to use the PAUM model (Perceptron Algorithm with Uneven Margins) [35] to learn WSD and relation classifications from on the extracted feature instances. PAUM was designed spefically for imbalanced data and has successfully been applied to various semantic annotation problems [36], [37]. This algorithm differs from other classifiers, such as CRF [14], since it treats positive and negative examples differently by introducing two margin

TABLE 1
Feature descriptions

	Feature	Description
1	Token.category [win_r] (Context POS tags)	The win_r (number of) grammatical tags (POS) surrounding the ambiguous word in the sentence.
2	Token. class [win_r] (Context concepts)	The win_r (number of) concepts surrounding the ambiguous word in the sentence.
3	TokensBetween.category (Surrounding categories)	The ordered list of POS tags between the domain and range terms.
4	TokensBetween.concepts (Surrounding concepts)	The ordered list of concepts between the domain and range terms.
5	Distance	The number of tokens between the domain and range terms.
6	Direction	0 if the two annotations overlap, +1 if the range follows the domain and -1 if the range precedes the domain.
7	Verbs	The ordered list of verbs in the sentence evoking the link triple.

parameters τ_+ and τ_- into the updating rules for the positive and negative examples, respectively. Here, τ_+ and τ_- simulate the weights on context examples, hence, we could increase/decrease the distance between positive/negative contexts (see features 3-7) of an ambiguous concept and its surrounded ones. This reflects the intuition that *"the nearer a neighbouring concept is close to an ambiguous concept, the more important it is for classifying the given token"*. It is about the same intuition for the relation classification. We represent the distance between positive/negative domain and range concept examples of one relation. PAUM learns how to update its learning rate and margin parameters to maximize this distance [35]. Since we are not dealing with a huge number of features, we use the traditional Sequential Forward Selection [38] while adding f features at a time to select. The selected best set of features are then, used in the construction of a model for an automatic annotation of ambiguous words in texts and relations between them.

After tuning and training our models, we applied them to the test set (test corpus) which constitutes 1/3 of the available corpus (we note the other 2/3 are used for training). Extracted terms and links are then evaluated using different evaluation measures, *i.e.* Precision, Recall and F-Measure [39]. Precision is the number of *correctly* identified items (objects/links) as the percentage of identified items. In other words, it measures how many items that our system identified were correctly tagged, regardless of whether it also failed to retrieve correct items. The higher the precision, the better the system is at ensuring that what was identified is correct. Recall measures how many items should have been identified from the actual (real) set of correct annotations. We note that we consider annotations to be *partially* correct if the entity/relation type is correct and the spans (positions of characters in the text) are overlapping but not identical. Thus, partially correct responses are allocated a half weight.

$$Precision = \frac{Correct + 0.5 \times Partial}{Correct + Spurious + Partial} \quad (1)$$

$$Recall = \frac{Correct + 0.5 \times Partial}{Correct + Missing + Partial} \quad (2)$$

where:

- *Correct*: Number of correct annotations;
- *Partial*: Number of partially correct annotations (if they overlap in terms);
- *Spurious*: Number of spurious annotations (in the test set);
- *Missing*: Number of missing annotations (in the training set).

The F-measure (FM) is often used in conjunction with Precision and Recall, as the harmonic average of the two.

$$FM = \frac{2 \times (Precision + Recall)}{Precision + Recall} \quad (3)$$

3.6 Workflow reconstruction

From recognized links between concept instances in texts, we build our concrete workflow knowledge database where items are the named entities ordered by the top-level pattern in the domain ontology. Concept and relation instances serve to identify, respectively, the data-flow and the control-flow. For example, from the tuples: ρ_1 = (Alignments, "isOutputOf", ClustalW), ρ_2 = (Alignments, "isInputOf", MEGA), we construct the control-flow: $w.\zeta$ = ⟨ {Alignments}, {ClustalW}, {MEGA} ⟩ and the data-flow: $w.\theta = (\rho_1, \rho_2, \rho_3)$ where ρ_3 = (Alignments, "genInputTo", ClustalW). The partial order in the sequence $w.\zeta$ is carried out with the 'generate input to' (*genInputTo*) relation which is the result of the transitive relation between input and output data from the ontology Ω.

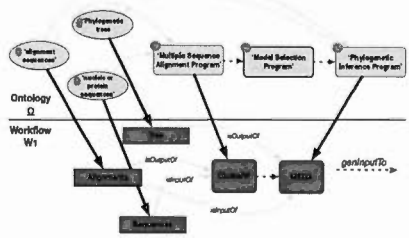


Fig. 6. Example of a workflow reconstruction from a domain-ontology. Here, the hard black arcs represent the possible concept matchings between the ontological workflow pattern (grey rectangles) and the workflow sequence $w_1.\zeta$ (blue rectangles).

Fig. 6 shows an example of workflow reconstruction from a domain ontology. The first set in $w_1.\zeta$: {Sequences, Alignments, Tree} represents the data types. While relations ρ between datatypes and programs *'ClustalW'* and *'MEGA'* represent the data-flow, the sequence itself in $w_1.\zeta$ represents the control-flow (*'genInputTo'*).

3.7 Workflow similarity measure

Here, we calculate the similarity measure $sim_w(a,p) \in \{0..1\}$ which calculates the similarity between an expert-curated workflow a and a predicted one p. Unlike the work of [40] and [41] proposing computationally expensive workflow mapping algorithms (based on graph-edit distance or A* search), we propose a simple bag-of-words similarity measure which takes into account both object (data, metadata and tasks) and link (data-flow and control-flow) terms in texts and their ontological labels. We remind that each workflow a and p is represented by the couple (ζ, Θ) in Δ_W where ζ is the set of object labels organized in transactions (steps) and Θ is the the set of relation triples connecting a domain object to a range one from ζ using a link type $l \in \mathcal{L}$. The similarity between two workflows a and p is given by the following formulas:

$$sim_w(a,p) = \frac{sim_w^{\zeta}(a,p) + sim_w^{\Theta}(a.p)}{2} \quad (4)$$

$$sim_w^{\zeta}(a,p) = \frac{\sum_j^m \frac{\sum_i^n (w^{\zeta}(a.o_{i,j}, p.\zeta))}{n}}{m} \quad (5)$$

$$sim_w^{\Theta}(a,p) = \frac{\sum_i^k (w^{\Theta}(a.\theta_k, p.\Theta))}{k} \quad (6)$$

Here, $m = |a.\zeta|$, n and $k = |a.\Theta|$ are the total number of steps, items per step and link triples, respectively, in the actual workflow a. Each object $o_{i,j}$ is weighted by its *label "correctness"* measure $w^{\zeta} = \{w_1^{\zeta_t}, w_2^{\zeta_t}, w_3^{\zeta_t}\}$ and each property (triple) θ_k is weighted by $w^{\Theta} = w^{\Theta_t} \times w^{\Theta_*}$ where the weight $w^{\Theta_t} \in \{0..1\}$ is the degree of an object term correctness (domain and range) and $w^{\Theta_*} \in \{0..1\}$ is the degree of a link term correctness. We define the term "correctness" based on the character matchings between a term and its referring label in the ontology Ω. A partial correctness of a term $v \in p.\zeta$ is defined as where v is a substring in $u = a.o_{i,j}$. We apply here the longest common substring algorithm to search for the actual term string in the $a.\zeta$ sequence of object labels.

$w_1^{\zeta_t} = 1$ is the weight given if the object $a.o_{i,j}$ is found in $p.\zeta$ and is correct. $w_2^{\zeta_t} = 0.6$ is the weight given for a term if the object $a.o_{i,j}$ is found in $p.\zeta$ and is partially correct. Finally, $w_3^{\zeta_t} = 0$ if the object $a.o_{i,j}$ is not found in $p.\zeta$. In the relation side, we tuned the weights as follows: $w_1^{\Theta_t} = 1$, $w_2^{\Theta_t} = 0.6$, $w_3^{\Theta_t} = 0.1$, $w_1^{\Theta_*} = 1$, $w_2^{\Theta_*} = 0.6$ and $w_3^{\Theta_*} = 0$. In addition, w^{Θ_t} are the weights given if the term of a link (*e.g. 'were-examined'*) is correct, partially correct or not. Same for w^{Θ_*}, the weights represent the correctness degrees of the relations structure: domain and range terms. Here, weights values are inspired from the work of [42] where lenient precisions and recalls are calculated given a partial overlap measure of terms.

4 Experiments and Results

4.1 Dataset

For this study, we downloaded scientific articles from the digital repository PubMed Central PMC database (published from January until April 2015) in XHTML format. 320 texts have been extracted. At first we downloaded more than 1 000 recent articles, journals and conference papers having at least one section: 'phylogenetic analyses'. However, these sections may have not sufficient information to reconstruct workflows from texts. Thus, we filtered the downloaded articles based on the smallest ontological pattern that any phylogenetic analysis should represent. A

phylogenetic study might use at least one datatype and one phylogenetic inference program in order to execute a phylogenetic analysis.

To conceive and implement the domain ontology, we build at first a relational database gazetteer (in MySQL) from different well-known sources of data in the phylogenetic literature such as Gene Ontology[12], NCBI[13], UniprotKB[14] and from Felsenstein Website [15]. The terminology is then reorganized in a domain ontology in order to recognize terms and relations in texts. We used the Quest-Ontop platform [43] to map MySQL classes to SPARQL concepts in a domain ontology. The generated ontology presents 37 different object properties (relations), 48 543 concepts (classified into 4 root concepts – besides the most abstract concept *owl:Thing*) and 111 unique concepts[16]) and more than 1 300 000 unique instances without counting the synonyms. Here the root concepts represent the four different layers of a workflow – see Fig.1. The ontology is free to access under the BioPortal repository[17], the Gate implementation of our workflow extractor and extracted workflows and result are available in: https://github.com/bioinfoUQAM/tgowler/tree/master/WfExtractor_1.0.

Using the GATE platform and the Stanford POS tagger [44], a human expert has extracted (created, deleted, updated) manually annotations from randomly chosen 100 texts to build a workflow ground-truth database. This latter serves as a test set for the evaluation step. The expert annotates the given texts using our domain ontology schema on the platform GATE. He annotated only terms of concepts and relations that serve to reconstruct the phylogenetic workflows from texts. Although this task is costly in terms of time/energy ($\sim$ 10 hours), the expert validation step is important to guarantee a credible perspective of our approach. We note, that we don't prefer an inter-annotator agreement validation since it was demonstrated that Kappa measure (not used to measure the accuracy) works only for discrete and mutually exclusive classes [45].

4.2 Results

4.2.1 Workflow elements

In the first experiment, we evaluate the WSD model (PAUM WSD). We use precision, recall and F-Measure metrics. A 10-fold cross validation was run over the PAUM classifier to choose the best parameters: $\tau_- = 5$, $\tau_+ = 50$, where τ_- and τ_+ are respectively the negative and positive margins. We have picked the best values for τ_- from $\{-10, -5, -1, 0, 1, 5, 10\}$ and for τ_+ from $\{-100, -50, -10, 0, 10, 50, 100\}$ based on the validation results. In addition, we chose the whole set of the proposed features since all POS and concept context tags show the best F-Measure with a context window $win_i = 5$.

In the second experiment, we evaluate the relation extraction model. Same as in the previous experiment, we ran a 10-fold cross validation over the PAUM Relation Extraction (PAUM RE) classifier. We chose the best features and best parameters as follows: $\tau_- = 1$, $\tau_+ = 10$. Selected features are: token POS tags, concepts, distance, direction and the list of verbs between domain d and range r (see table 1).

TABLE 2
PAUM Word Sense Disambiguation and relation extraction evaluation results.

	Precision (%)	Recall (%)	F-Measure (%)
PAUM WSD	99.21	66.66	79.74
PAUM RE	72.63	84.22	77.24

Table 2 shows results of WSD and relation extraction learning. For PAUM WSD, the recall is the ratio of total amount of ambiguous words correctly disambiguated over the total amount of ambiguous words in the expert-curated dataset. The precision is the ratio of the total amount of correctly disambiguated words over the total number of generated ambiguous words. Our PAUM WSD model is highly precise (99.21%) but predicted words represent 66.66% of the total number of ambiguous words in the exert dataset. This could be explained by the range of the chosen context of ambiguous words, *i.e.* the sentence. The PAUM RE model presents different results in terms of precision and recall from the PAUM WSD but it shows similar F-Measures. PAUM RE covers 84.22% from the actual relations in the expert-curated dataset, however, 72.63% of the generated relations are correctly predicted.

We show in Table 3 details about the total number of the discovered annotations. We calculate the classic Information Retrieval F-Measure [42] based on correct, partially correct and spurious annotations (see also section 3.5). In total, our system has annotated 2 561 terms while the expert has annotated 2 641. The overall strict F-Measure (overlaps discarded) is 70% and the overall lenient F-Measure (overlaps included) is 76%. We notice that 118 annotations (~20%) represent ambiguous program terms (GuessedProgram). In this case, the average F-Measure (between strict and lenient) is 84% with a precision of 73% and a recall of 98%. Although strict and lenient F-Measure values are very close (~0.3%), some categories, *i.e. DataCollectionProgram*, *TreeVisualizationPrgram* and *GuessedGeneralPackagesCategories*, show the same values for strict and lenient F-Measures (see also the 'O' column). Our tagger is able to recognize (or not) the exact mentions in texts: these categories does not present non-standard program names (*e.g.* common names) or generic mentions. In addition, we note that almost the half of the presented categories show reasonably high lenient F-Measure agreement (> 80%). Yet, besides *hypotheses validation* software names, *data collection* programs, *species* names, *protein* and *GO annotations* show the worst tagging F-Measure results (with a lenient F-Measure between 60 and 67%). The recall values in data collection programs explain the weak F-Measure (for both strict and lenient agreements). Thereof, our database for this category need to be updated. In addition, since no specific JAPE rules are modelled for the metadata (GO annotations and species), some common false negatives (*i.e.* missed resource mentions) are discovered including common names of genes (*e.g.* the *g20* gene) or species (*e.g. birds*).

12. http://www.geneontology.org
13. http://www.ncbi.nlm.nih.gov
14. http://www.uniprot.org
15. http://evolution.genetics.washington.edu
16. The rest of the concepts are synonyms to the 111 ones.
17. http://bioportal.bioontology.org/ontologies/PHAGE

TABLE 3
Concept Evaluation Distribution

	S	G	P	O	SP	SR	SFM	LP	LR	LFM
BootstrapsParms	61	19	8	1	0.87	0.75	0.80	0.88	0.76	0.82
DataBase	43	19	13	3	0.72	0.66	0.69	0.77	0.70	0.74
DataCollectionProgram	36	29	5	0	0.87	0.55	0.67	0.87	0.55	0.67
DataType	597	103	186	89	0.68	0.75	0.71	0.78	0.86	0.82
GeneOntology	139	172	5	3	0.94	0.44	0.60	0.96	0.45	0.61
HypothesisValidationProgram	25	40	29	2	0.44	0.37	0.40	0.48	0.40	0.43
Method	326	48	46	14	0.84	0.84	0.84	0.88	0.87	0.87
ModelSelectionProgram	42	50	17	1	0.70	0.45	0.54	0.71	0.46	0.56
PhylogeneticInferenceProgram	205	16	146	10	0.56	0.88	0.69	0.59	0.93	0.72
SequenceAlignmentProgram	70	15	19	4	0.75	0.78	0.76	0.79	0.83	0.81
SourceTaxon	202	125	57	19	0.72	0.58	0.64	0.79	0.63	0.70
TreeVisualizationProgram	18	7	2	0	0.90	0.72	0.80	0.90	0.72	0.80
GuessedGeneralPackagesCategories	87	1	31	0	0.73	0.98	0.84	0.73	0.98	0.84
Sum	1851	644	564	146						
Macro summary					0.75	0.67	0.69	0.78	0.70	0.72
Micro summary					0.72	0.69	0.70	0.78	0.74	0.76

The 'S' column represents the number of Successfully recognized annotations. The 'G' (only in Gold set) column represents missed annotations while the 'P' (only in Prediction set) column shows the number of unknown annotations. The 'O' (Overlap) column shows the number of (partially recognized) annotations. While the 'SFM' and 'LFM' columns shows the Strict (overlaps discarded) and Lenient (overlaps included) F-Measures, respectively, the 'SP', 'ST' and 'LP', 'LR' columns represent Strict and Lenient values of Precision and Recall, respectively. Micro averaging essentially treats the corpus as one large document while Macro averaging calculates the F-Measure on a per document basis, and then, averages the results.

The manual annotation of ambiguous programs highlights all the textual mentions referring to a multiple purposes tool as indirect references. In some cases, correct results from our workflow tagger were not fully annotated, leading to false positives. The following sentences:

A phylogenetic tree was constructed with MEGA 5 using the aligned sequences [...] The phylogenetic tree was drawn using the neighbor-joining method with p-distance and 1,000 bootstrap replications.

show an example of such case where our tagger correctly detects *MEGA* [23] as a phylogenetic tree inference in the first sentence but in the following sentence, *MEGA*, a neighbor-joining method, is also used to validate trees using 1 000 bootstrap replications. The class of MEGA (*HypothesisValidation*) was not detected by our context-based approach. The reference in the second sentence is implicit (hidden in the *Neighbor-Joining* concept), hence, it is difficult to detect that trees generated by *MEGA* are also validated using the bootstrap method. In addition, both of the verbs in the two sentences are very generic and misleading for an automatic tagging, *i.e.*, *aligned* and *drawn*. *MEGA*, in the previous sentences, is not used to align or visualize sequences or trees. Consequently, the precision obtained by our pipeline for tree validation tasks are very low (~40%). In fact, 30% of the hypothesis validation programs are ambiguous programs (multiple purpose packages). Yet, our context-based pipeline supporting our system needs to be enhanced in its ability to capture more generic mentions in wider contexts (*e.g.* paragraphs).

4.2.2 Workflow similarities with a the ground truth

We present in Fig. 7 the distribution of similarities sim_w between the predicted and the Ground-Truth concrete workflows. The overall similarity is 0.83 over the 100 expert curated workflows. This figure shows also details about similarity distributions of both concepts and relations in workflows. The overall mean of relation similarity is 0.89 and concept similarity is 0.79. The best concept results are phylogenetic inference programs where the similarities are above 0.8 except for 5 workflows. Moreover, in one side, gene ontology and taxa annotations in workflows are 0.6 similar within the actual workflows which is understandable since only full scientific terms are considered while tagging the name of species, genes and proteins in texts. Though, synonyms weren't considered for the mentioned concepts at the semantic annotation step. These are dismissed in order to minimize the error cost of the abstractions in common terms (*e.g.* cell, RNA, polymerase, *etc.*). In the other side, all program terms similarities are above 0.8 except for model selection and hypothesis validation programs (0.6). This is due to the context ambiguities within these categories. Bioinformaticians and biologists tend to use general purpose packages instead of specific tools for model selection and tree validation steps. The context of such tools are 40% of the cases out if its sentence area.

In the relation side, links between data types and tree visualization programs show the best similarities (the mean similarity is 0.95). Mean similarities of relations between : (1) data types and MSA (Multiple Sequence Alignment) programs, (2) data types and phylogenetic inference programs and (3) data types and species are all between 0.65 and 0.77. Both predicted relations between model selection programs and their model parameters, and predicted relations between bootstrapping programs and their number of replicates are 0.9 similar with the expected workflow relations. Even though the context-based pipeline supporting our system needs to be enhanced in its ability to capture

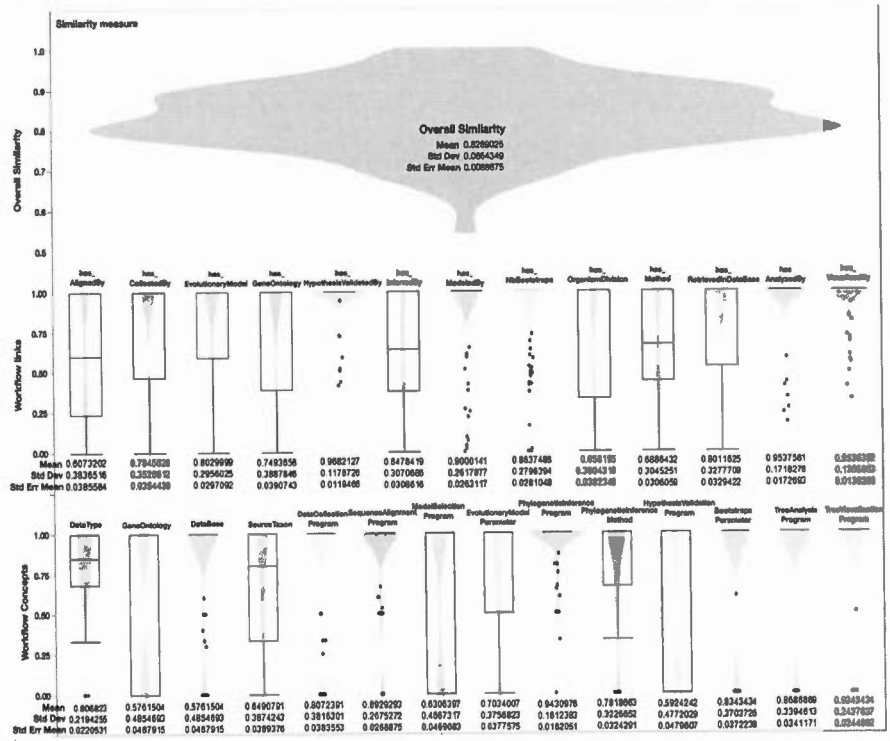


Fig. 7. Workflow similarity distributions with the ground truth.

more generic mentions (*e.g.* tree validation programs) in wider contexts and a more elaborate model to capture metadata in texts,*e.g.* Gene Ontology annotations and name of species, experiments show satisfying similarity results (between 57% and 96%) with the expert-curated workflows and a good word sense disambiguation model with an F-Measure >77%.

5 Conclusion

This paper addresses the problem of the extraction of enriched workflows using a domain-ontology to resolve word sense ambiguities in texts. The framework is supported by an application in the bioinformatics domain. Our graph based workflow representation extends the classic POCBR workflow structures with a semantically richer relations that serve for both: extraction and resolution of polysemic tasks in texts. It is promising that our method reaches high similarities with an ontology-based word sense disambiguation approach for tasks, however, other term classifications need to be improved, mainly data type metadata such as the name of genes, proteins and species. Besides, the implication of an expert is always mandatory for a good validation of extracted terms. We intend to extend our ontology-based mining framework in order to consider a variety of criteria, for example other method parameters (*e.g.* Gamma rates, transition ratios, *etc.*), more complex control structures (*e.g.* parallel executions, if-else statements) and resource quality in terms of time (the most recent tools/databases), author (focusing on domain's experts), journals, popularity, etc,). Updating our text corpora with the most recent articles will help to produce more relevant patterns. Moreover, the proposed generalized patterns could be used to improve even journals quality to select the most relevant articles using the abstract and concrete workflow information within texts. For instance, abstracting the proposed workflow in a submitted article is used to assist the reviewer assessment.

Acknowledgment

This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D.

References

[1] W. van der Aalst, "Process mining software," in *Process Mining*. Springer, 2016, pp. 325–352.

[2] E. Lord *et al.*, "Armadillo 1.1: An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and Simulations," *PloS One*, vol. 7, no. 1, Jan. 2012.
[3] D. Blankenberg *et al.*, "Galaxy: A Web-Based Genome Analysis Tool for Experimentalists," *Current protocols in molecular biology*, vol. Chapter 19, Jan. 2001.
[4] M. Minor and S. Montani, "Process-oriented case-based reasoning," *Luc Lamontagne and Juan A. Recio-Garcıa (Editors)*, p. 9, 2012.
[5] V. Dufour-Lussier *et al.*, "Automatic case acquisition from texts for process-oriented case-based reasoning," *Information Systems*, vol. 40, pp. 153–167, 2014.
[6] P. Schumacher *et al.*, "Extracting and enriching workflows from text," pp. 285–292, 2013.
[7] G. Müller *et al.*, "Workflow streams: A means for compositional adaptation in process-oriented cbr," in *Case-Based Reasoning Research and Development*. Springer, 2014, pp. 315–329.
[8] P. Lemey *et al.*, *The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press, 2009.
[9] T. A. Lau, C. Drews, and J. Nichols, "Interpreting written how-to instructions." pp. 1433–1438, 2009.
[10] Y. Liu *et al.*, "Crawling and extracting process data from the web," Springer, pp. 545–552, 2009.
[11] R. Mitkov, *Anaphora resolution*. Routledge, 2014.
[12] A. Addis *et al.*, "From unstructured web knowledge to plan descriptions," in *Information Retrieval and Mining in Distributed Environments*. Springer, 2010, pp. 41–59.
[13] J. Ryu, H. Ihm, and S.-H. Myaeng, "Automatic organization of human task goals for web-scale problem solving knowledge," ACM, pp. 81–88, 2013.
[14] B. Settles, "Biomedical named entity recognition using conditional random fields and rich feature sets," *Proc. of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications - JNLPBA '04*, p. 104, 2004.
[15] S.-k. Song, H.-s. Oh, S. H. Myaeng, S.-P. Choi, H.-W. Chun, Y.-S. Choi, and C.-H. Jeong, "Procedural knowledge extraction on medline abstracts," Springer, pp. 345–354, 2011.
[16] H. Poon *et al.*, "Distant supervision for cancer pathway extraction from text." *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, vol. 20, pp. 120–31, 2015.
[17] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.
[18] O. Spjuth *et al.*, "Experiences with workflows for automating data-intensive bioinformatics," *Biology Direct*, pp. 1–12, Aug. 2015.
[19] M. Stevenson and Y. Wilks, "Word sense disambiguation," *The Oxford Handbook of Comp. Linguistics*, pp. 249–265, 2003.
[20] J. L. Peterson, "Petri net theory and the modeling of systems," 1981.
[21] L. Liang *et al.*, "Combining spatial-temporal and phylogenetic analysis approaches for improved understanding on global h5n1 transmission," *PLoS One*, vol. 5, no. 10, 2010.
[22] O. Gascuel, "Bionj: an improved version of the nj algorithm based on a simple model of sequence data." *Molecular biology and evolution*, vol. 14, no. 7, pp. 685–695, 1997.
[23] S. Kumar *et al.*, "Mega7: Molecular evolutionary genetics analysis version 7.0 for bigger datasets," *Molecular biology and evolution*, p. msw054, 2016.
[24] K. Tamura *et al.*, "Mega5: molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods," *Molecular biology and evolution*, vol. 28, no. 10, pp. 2731–2739, 2011.
[25] N. Barrett *et al.*, "Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm," *BMC bioinformatics*, vol. 12, 2011.
[26] L. Smith, T. Rindflesch, and W. J. Wilbur, "MedPost: a part-of-speech tagger for bioMedical text," *Bioinformatics*, vol. 20, no. 14, pp. 2320–2321, Sep. 2004.
[27] E. Buyko *et al.*, "Automatically adapting an nlp core engine to the biology domain," pp. 65–68, 2006.
[28] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.
[29] H. Cunningham *et al.*, "Getting more out of biomedical documents with gate's full lifecycle open source text analytics," *PLoS*, vol. 9, no. 2, 2013.
[30] J.-D. Kim *et al.*, "Genia corpus a semantically annotated corpus for bio-textmining," *Bioinformatics*, vol. 19, no. suppl 1, pp. i180–i182, 2003.
[31] V. A. Diehl *et al.*, "Procedural text structure and reader perceptions and performance," *The Journal of general psychology*, vol. 129, no. 1, pp. 18–35, 2002.
[32] A. S. Gordon, "The design of knowledge-rich browsing interfaces for retrieval in digital libraries," Ph.D. dissertation, Northwestern University, 1998.
[33] D. Plotree and D. Plotgram, "Phylip-phylogeny inference package (version 3.2)," *cladistics*, vol. 5, no. 163, p. 6, 1989.
[34] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[35] Y. Li *et al.*, "Using uneven margins svm and perceptron for information extraction," Stroudsburg, PA, USA, pp. 72–79, 2005.
[36] Y. Liu *et al.*, "Hierarchical, perceptron-like learning for ontology-based information extraction," ACM, pp. 777–786, 2007.
[37] J. Dědek, "Towards semantic annotation supported by dependency linguistics and ilp," in *The Semantic Web–ISWC 2010*. Springer, 2010, pp. 297–304.
[38] J. Doak, *An evaluation of feature selection methods and their application to computer security*. University of California, Computer Science, 1992.
[39] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.
[40] J. Starlinger *et al.*, "Similarity search for scientific workflows," *Proc. of the VLDB Endowment*, vol. 7, no. 12, pp. 1143–1154, 2014.
[41] R. Bergmann *et al.*, "Similarity assessment and efficient retrieval of semantic workflows," *Information Systems*, vol. 40, pp. 115–127, 2014.
[42] G. Salton *et al.*, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.
[43] M. Rodriguez-Muro and D. Calvanese, "Quest, an owl 2 ql reasoner for ontology-based data access," *OWLED 2012*, 2012.
[44] K. Toutanova *et al.*, "Feature-rich part-of-speech tagging with a cyclic dependency network," Association for Computational Linguistics, pp. 173–180, 2003.
[45] S. Pyysalo, F. Ginter, J. Heimonen, J. Björne, J. Boberg, J. Järvinen, and T. Salakoski, "Bioinfer: a corpus for information extraction in the biomedical domain," *BMC Bioinformatics*, vol. 8, no. 1, p. 50, Feb 2007. [Online]. Available: https://doi.org/10.1186/1471-2105-8-50

Ahmed Hallioui received the PhD in Cognitive Computing at the Université du Québec à Montréal in 2017. Currently, he is working at the MiMs company to implement an artificial research assistant for life scientists. He is interested in procedural knowledge acquisition: *i.e.* information extraction, ontologies and data mining. During his PhD, he developed the T-GOWLeR system as an ontology based system to extract and mine abstract workflow models from texts.

Petko Valtchev completed his undergraduate studies in 1992 first at Sofia University St Klyment of Ohrid and then at Vienna University. Petko holds a M.Sc. from the National Polytechnic Institute of Grenoble (INPG), France, and a Doctoral degree from J. Fourier University, Grenoble (1999). His Ph.D. thesis on knowledge discovery for knowledge base construction was prepared at INRIA.

Abdoulaye Baniré Diallo is fascinated by the usability and efficiency of bioinformatics services to solve biological questions and transform biologist main needs into accessible services. Abdoulaye has a Ph.D. in computer science (option bioinformatics) from the McGill University. He is a former research associate at the BroadInstitute of MIT and Harvard and postdoctoral scholar at the Computer Science and artificial intelligence Laboratory of MIT.

ANNEXE B

Ontology-based Workflow Pattern Mining

Application to Bioinformatics Expertise Acquisition

Ahmed Halioui, Tomas Martin, Petko Valtchev, Abdoulaye Baniré Diallo*
Département d'informatique, Université du Québec à Montréal
PO BOX 8888 Downtown Station, H3C 3P8, Montréal, Québec
halioui.ahmed@uqam.ca, martin.tomas@courrier.uqam.ca, valtchev.petko@uqam.ca,
diallo.abdoulaye@uqam.ca

ABSTRACT

Workflow platforms enable the construction of solutions to complex problems as step-wise processes made of components including methods, tools, data formats, parameters, *etc.* Successful workflow solutions require a mastering of the different components paving the way to automated acquisition of problem solving expertise. Thus, process mining could be applied to discover workflow patterns. Due to the combinatorics of component instances in rich domains such as bioinformatics, generalized patterns could be a relevant way of abstraction. Here, we propose an approach for mining workflow patterns, defined on the top of a domain ontology which categorizes workflow elements and their interactions. While original workflows are doubly-labelled DAGs, the underlying problem is transformed into a mining of generalized sequential patterns with links between their items. The proposed mining method traverses the ensuing pattern space using five refinement primitives that exploit the *is-a* links from the ontology. To assess the prediction power of the approach, we applied the generated patterns as templates in a recommendation platform to complete partial workflows under construction. The analyses of recommendations vs. actual content of a real-world dataset reveals that non trivial patterns can be found and further used to provide plausible recommendations with high accuracies (fMeasure >75%).

CCS Concepts

•**Information systems** → *Data mining;*

Keywords

Sequential pattern mining; Workflows; Ontologies

1. INTRODUCTION

Workflows are increasingly designed as the efficient way of managing complex multi-step processes for the resolution of complex problems from knowledge-rich domains such as medicine, biology, chemistry and bioinformatics. This conducts to a growing popularity of workflow platforms each offering a wide spectrum of analytical services [1, 2, 3]. However, in order to build successful workflows, the user must typically master a large variety of methods, tools, data formats, *etc.* In many cases, the ultimate success of the workflows depends on the right choice of the included methods, algorithms, tool parameters, software versions, *etc.* In fact, such a demanding environment reduces the value of workflow management for non expert users. Facing these challenges, process mining [4, 5] as a branch of knowledge discovery from data (KDD), studies the discovery of reusable knowledge from databases of process records, in particular, from workflow representations such as execution logs. As workflows have an inherent graph structure, effective process mining methods face the related complexity issues. A workflow graph depicts an analytical resolution in terms of blocks and their interactions (control/data flows) drawn as nodes and links, respectively. Yet, given the large spectrum of possible inputs, it is unlikely for a recorded workflow to fully fit a new situation. A solution could be to mine higher-level of abstractions summarizing commonalities of a set of workflows. Representing abstraction levels using ontologies to assist mining of usage logs has proven popular in the last decade [6, 7, 8]. A complete framework for abstracting ontology-based patterns out of click-streams (from the logs of a semantic web portal powered by a domain ontology) provides an alternative view [7]. In this papers, we deal with a comparable problem on a different type of input format. Instead of pure sequences, we consider workflows as Directed Acyclic Graphs (DAGs). However, due to the very complex nature of the underlying domain problem, the DAGs are split into ordered layers. In the next section, we define first, the different elements of the ontology, followed by two descriptive languages (data and pattern language). We also introduce the pattern space, the instantiation relationship between workflows and patterns, and the generality order between patterns. Finally, we describe the pattern space levels and the refinement operators to generate patterns at each level. Generated pattern are then exploited by a top-down depth-first traversal method using the "support" metric as a quality criterion for patterns.

2. DEFINITIONS

DEFINITION 1. *Ontology*
Let $\Omega = \langle O, C, \mathcal{L}, \mathcal{R} \leq_\Omega, \rho \rangle$ *be an ontology where* O *is*

*Correspondence: Diallo.Abdoulaye@uqam.ca

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SAC'17, April 3-7, 2017, Marrakesh, Morocco

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM

ISBN 978-1-4503-4486-9/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3019612.3019866

the set of all objects, i.e. individual tasks and data types, C the set of all concepts, $\mathcal{L}$ the set of all links between objects, and $\mathcal{R}$ the set of all properties (relations) between concepts. $H_C = \langle C, \leq_\Omega \rangle$ and $H_\mathcal{R} = \langle \mathcal{R}, \leq_\Omega \rangle$ denote the concept and relation hierarchies of the ontology where $\leq_\Omega$ denotes the is-a relationship. The ternary relation $\rho \subseteq C \times \mathcal{R} \times C$ is made of triples $c_1 \times r \times c_2$ expressing the incidence between a relation r and its domain c_1 and range c_2 concepts. Objects $o \in O$ and links $l \in \mathcal{L}$ are instances of concepts $c \in C$ and relations $r \in \mathcal{R}$, respectively.

Given a universe of tasks and data O and links $\mathcal{L}$ defined in Ω, a workflow record represents a doubly labelled DAG with vertex labels in O and edge labels in $\mathcal{L}$. Its vertex set can be split into subsequent layers following the steps of the workflow analysis with possible further splits so that no intra-layer edges remain. In this way, the data record becomes a sequence of object bags (transactions) with a set of cross-transaction links.

DEFINITION 2. *Data language*
Let Δ_W be the data language made of pairs $w = (\zeta, \Theta)$ where $\zeta = (\{o_1, o_2, ..., o_{i_1}\}, \{o_{i_1+1}, \ldots, o_{i_2}\}, \ldots, \{o_{i_{(m-1)}+1}, \ldots, o_{i_m}\})$ is a sequence of transactions (steps). In turn, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_p\}$ is a set of triplets where $\theta_k = (i_1, l, i_2)$ represents a link of type l between the object at position i_1 in $w.\zeta$, i.e. $w.\zeta[i_1]$ and the object $w.\zeta[i_2]$.

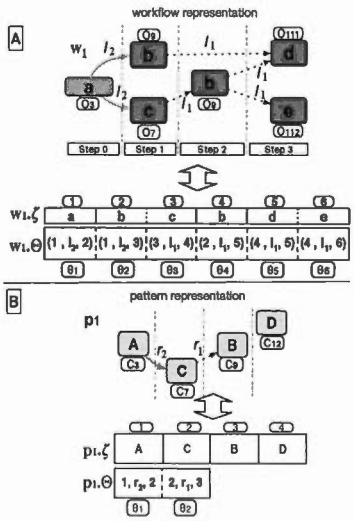


Figure 1: (A) Workflow representation of $w_1 \in \Delta_W$. (B) Pattern representation of $p_1 \in \Gamma_W$.

Example 1. In *Fig.* 1 (A), w_1 is a workflow representing tasks b, c, d, e and data type a as well as a set of links. Tasks are further split into steps, thus inducing the transaction sequence $\langle \{a\}, \{b, c\}, \{b\}, \{d, c\} \rangle$ with the following triple set $\{(1, l_2, 2), (1, l_2, 2), \ldots (4, l_1, 6)\}$. Here, l_1 embodies the execution order in the workflow and l_2 represents the input link between data type and tasks.

DEFINITION 3. *Pattern language*
Let Γ_W be set of pairs $p = (\zeta, \Theta)$ where $p.\zeta$ is the sequence of bags of concepts and $p.\Theta$ is the set of relation triplets.

Example 2. The pattern p_1 in *Fig.* 1 (B) represents the sequence $\langle \{A\}, \{C\}, \{B\}, \{D\} \rangle$ and the triplets $\theta_1 = (1, r_2, 2)$ and $\theta_2 = (2, r_1, 3)$.

For a workflow $w \in \Delta_W$ and a pattern $p \in \Gamma_W$, we assert $w \lhd p$ (said w *instantiates* p) whenever there exists a sub-structure in w isomorphic to p. The mapping function $\Phi : \{1, ..|p.\zeta|\} \rightarrow \{1, ..|w.\zeta|\}$ defines the instantiation relationship between w and p as follows:

DEFINITION 4. *Instantiation relationship*
- Φ *is an injective function ;*
- $\forall i \in \{1, .., |p.\zeta|\}$, $w.\zeta[\Phi(i)] \in_\Omega p.\zeta[i]$;
- $\forall i, i' \in [1, |p.\zeta|]$, $\tau_p(i') \leq \tau_p(i)$ *iff* $\tau_w(\Phi(i')) \leq \tau_w(\Phi(i))$
- $\forall \theta_k = (i_1, r, i_2) \in p.\Theta$, $\exists l \in \mathcal{L}$ *s.t.* $l \in_\Omega r$ *and* $(\Phi(i_1), l, \Phi(i_2)) \in w.\Theta$.

Example 3. $w_1 \lhd p_1$ in $(\{a\}, \{c\}, \{b\}, \{d\})$ and in links $\{l_1, l_2\}$ while $l_1 \in_\Omega r_1$, $l_2 \in_\Omega r_2$, $a \in_\Omega A$, $b \in_\Omega B$ etc. and $(\Phi(1), l_2, \Phi(3)) \in_\Omega p_1.\theta_1$ and $(\Phi(3), l_1, \Phi(4)) \in_\Omega p_2.\theta_2$ (see *Fig.* 1).

DEFINITION 5. *Generality relationship*
Let p_1 and p_1' two patterns from Γ_W, p_1 is more general than p_1': $p_1' \sqsubseteq p_1$ iff $\{w_i \in \Delta_W | w_i \lhd p_1'\} \subseteq \{w_j \in \Delta_W | w_j \lhd p_1\}$.

DEFINITION 6. *Levels in pattern space*
Let g be the generality degree of a pattern defining the level k of a pattern in the pattern space. g is the sum of a pattern component (concepts and relations) depths from the ontology where a depth h is the maximal path from a component to the root according to H_C and $H_\mathcal{R}$ in Ω.

According to the previous definition, the generation of $k+1$-level patterns from k-level ones amounts to the calculation of five *canonical* operations: (1) root concept insertion in a new transaction, (2) root concept insertion at the end of one transaction, (3) concept specialization, (4) root relation insertion and (5) relation specialization. Pattern elements are indexed using a topological order induced by the ontology hierarchies. Thus, refining a pattern is always done from its last element (the highest rank). Let $rank_C$ and $rank_\mathcal{R}$ be the functions ranking concepts/objects and relations/links, respectively, from Ω. Let c be the last concept in $p.\zeta$ and r be the relation in the last triple of $p.\Theta$. Refining p leads to add a concept/relation in the pattern or, alternatively, to replace c by $\hat{c}$ for whom c is the canonical parent concept i.e. $\hat{c} \prec_{can} c$. Similarly, r can only be specialized to a relation $\hat{r}$ that satisfies $\hat{r} \prec_{can} r$. Table 1 lists the precise set of conditions for each canonical operation.

Example 8. *Fig.* ?? illustrates the last three canonical operations. For instance, specializing the last concept $subC(D')$ in p_4 amounts to replace the last concept D with its descendent D' while its rank is higher than the rank of the previous concept in the transaction B. The $addR(r_5, 1)$ operation on

Table 1: Canonical operations.

Definition	Constraints
$appC(\hat{c})$ = Appends a new transaction with a root concept in it.	1. $\hat{c}$ is a root concept.
$addC(\hat{c})$ = Adds a root concept to the last transaction.	1. $\hat{c}$ is a root concept ; 2. $rank_C(\hat{c}) \geq rank_C(p.\zeta[\|p.\zeta\|])$.
$subC(\hat{c})$ = Replaces the last concept with $\hat{c}$.	1. $\hat{c} \prec_{can} p.\zeta[\|p.\zeta\|]$; 2. $rank_C(p.\zeta[\|p.\zeta\| - 1]) \leq rank_C(\hat{c})$ (2nd last concept).
$addR(\hat{r}, j)$ = Adds a relation $\hat{r}$ between position j and the last position.	1. $\hat{r}$ is a root relation ; 2. $p.\zeta[\|p.\zeta\|] \leq_\Omega ran(\hat{r})$; 3. $p.\zeta[j] \leq_\Omega dom(\hat{r})$; 4. $\forall k$ $\exists r_k$ with $(k, r_k, \|p.\zeta\|) \in p.\Theta \Rightarrow k < j$ (right most domain). 5. if $p.\zeta[\|p.\zeta\| - 1] = p.\zeta[p.\zeta\|]$ then $\forall l, \exists r_l$, $(i_l, r_l, \|p.\zeta\| - 1) \in p.\Theta \Rightarrow rank_R(\hat{r}) \geq rank_R(r_l)$, $i_l \geq j$.
$subR(\hat{r})$ = Replaces the last relation of the pattern with $\hat{r}$.	let $j_m = max(j \mid (j, r_i, \|p.\zeta\|) \in p.\Theta)$ and r' s.t. $(j_m, r'. \|p.\zeta\|) \in p.\Theta$ in: 1. $\hat{r} \prec_{can} r'$; 2. $p.\zeta[\|p.\zeta\|] \leq_\Omega ran(\hat{r})$; 3. $p.\zeta[j_m] \leq_\Omega dom(\hat{r})$. 4. if $p.\zeta[\|p.\zeta\| - 1] = p.\zeta[p.\zeta\|]$ then $\forall l, \exists r_l$, $(i_l, r_l, \|p.\zeta\| - 1) \in p.\Theta \Rightarrow rank_R(\hat{r}) \geq rank_R(r_l)$, $i_l \geq j_m$.

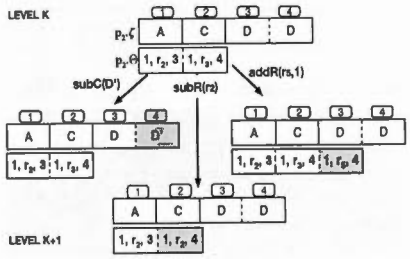


Figure 2: An example of canonical operations.

p_4 adds the relation r_5 between the first concept A and the last one D. To that end, we need at first to verify if there is at least one relation pointing to the last position. Next, we check the compatibilities between the concept A and the domain of r_5 in the ontology and between D and the range of r_5. After that, we add the concerning triple after the right most triple in $p_4.\Theta$ having 4 (last concept position) as ending point (destination) and r_5 as a relation. We remind that $p_4.\Theta$ set is sorted by its ending and starting positions and then by its relation type rank $rank_R$. The final operation $subR(r_2)$ specializes the last relation on r_3 starting from the first concept A. In order to preserve the order in $p_4.\Theta$, r_2, as the canonical descendant of r_3 from the ontology pointing to the last concept D, should start from highest starting point (source) in the subset of triples on r_3 which is $\theta_2 = (1, r_3, 4)$. This operation replaces then θ_2 with the triple $(1, r_1, 4)$.

3. MINING METHOD

Mining abstract workflows from a workflow database $\mathcal{D}_W$ on the top of an ontology Ω and a minimum support threshold σ consists of finding all the frequent patterns $\mathcal{F}_\sigma = \{p \in \Gamma_W \mid supp(p) \geq \sigma\}$ in $\mathcal{D}_W$. Here, $supp(p)$ corresponds to the number of workflows from $\mathcal{D}_W$ matching with p $|w \lhd p|$. To find all the candidate patterns from a given frequent pattern p, we perform a depth-first overall traversal in which we apply the refinement operators. The algorithm starts with the generation of the frequent 1-patterns that are made of unique root concepts from the ontology. For each pattern, the generation is applied recursively for the subsequent levels $k+1$. The candidates are generated from frequent patterns at level k by applying all the canonical operations.

Algorithm 1 Depth first pattern generation

```
1: procedure GenerateAndTestDF(f: frequent k-level pattern ; MS: matching workflows ; σ: minimum support )
2:     CP ← addC(f) ∪ appC(f) ∪ subC(f) ∪ addR(f) ∪ subR(f):
3:     for all p ∈ CP do
4:         M' ← FindMatchingWorkflows(p, MS);
5:         if size(M') >= σ * |D_W| then
6:             FP^{k+1}.add(p) ;
7:             GenerateAndTestDF(p, M', σ);
8:         end if
9:     end for
10: end procedure
```

Algorithm 1 provides the pseudo-code to expand a k-level frequent pattern to yield a set of $k+1$ candidate patterns and then filter these $w.r.t$ the set of sequences matching the key pattern. Candidate generation from f involves the application of the five canonical operations. Then, for each generated candidate, we search for its workflow matching set (M') to calculate its frequency, starting from the last matched operation position. If the pattern is frequent, we save it with its workflow matching set to calculate recursively pattern candidates of the next levels $k+2, k+3, \ldots$ until there is no frequent patterns to generate.

4. EXPERIMENTAL STUDY

We evaluated the developed[1] method on two datasets composed of concrete bioinformatic workflows that were retrieved from scientific publications. At the preparatory step, we used a dedicated text miner to extract bioinformatic workflows out of a corpus of 2 500+ articles from the PubMed PMC repository [9]. We also used the PHAGE phyloge-

[1]Source code and data are available in: https://github.com/bioinfoUQAM/tgowler.

netic ontology published in Bioportal[2]. In our experiments, the mined associations from *WFPub* were used as recommendation templates in workflows from *WFG*. To this end, out of the discovered workflows, we formed two datasets: *WFG*, which contains 500 expert-validated workflows (12 objects and 6 links, on average), constitutes our ground truth, whereas *WFPub*, with 2 783 workflows (11 objects and 7 links) is the mining target. While recommending plausible completions of unfolding user sessions (incomplete workflows), we test the predictive accuracy of the developed method's output. Thus, the patterns mined from the first dataset were split into associations that further generated recommendations for (prefixes of the) workflows from the second dataset. To examine the usefulness of our method's output (and hence of the method itself): we count a hit whenever the top k items [10] are in the workflows w' that match in the knowledge base *WFPub*. We rank items i according to the confidence of rules suggesting i. Furthermore, we compute the recall [10] based on the number of hits in the next step objects from w' whenever the total number of program objects is found[3].

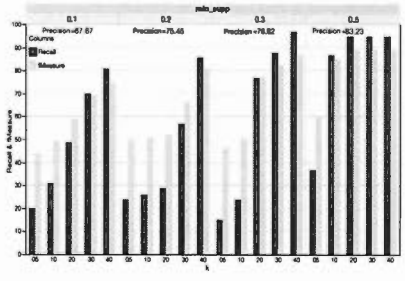


Figure 3: Top k item recommendation experiments.

5. RESULTS

In this study, we explore the complexity aspect of our method due to the limited availability of set of workflows. The experiments have been analyzed over the set of 2 783 workflows leading to output 14 416 patterns (35 concepts and 15 relations in average) in 753 seconds. To evaluate the sensitivity of our method on the value of k, we performed an experiment in which we values of k of 5, 10, 20, 30 and 40. The predictive accuracies for these experiments under different values of σ (min_supp) are shown in *Fig.* 3. The overall recall and fMeasure increases in relation with k and σ leading to the generation of more general rules. The results indicate that (i) increasing the value of k leads to significant improvements (>5%) in terms of fMeasure and (ii) for higher levels of abstraction ($\sigma > 0.3$), small values of k could be sufficient to generate a high recall but when rules tend to be more "specific"[4], small values of k decrease significantly the recall while maintaining good precisions (>75% where σ>0.1). This is due to the stringent definition of the recall, especially when the system recommends different object sub-tasks at the same time (parallel executions) for the subsequent step.

[2]http://bioportal.bioontology.org/ontologies/PHAGE

[3]Here, we ignore the object parameters while computing the number of hits.

[4]Rules using lower levels of abstraction for concepts and relations.

6. CONCLUSION

In this paper, we proposed a workflow pattern mining method for automatic acquisition of problem solving expertise that effectively exploits a domain ontology. The major challenges were the design of expressive, yet tractable pattern languages and efficient mining methods. Our approach consists of mining generalized DAG patterns (eventually reduced to sequential ones). To this end, we described the pattern space (data and pattern languages, instantiation and generality relations) and a refinement operators to move downwards in the pattern space. Finally, we designed a depth-first mining method that visits all patterns only once. Our method was assessed using a dataset of bioinformatic workflows that were retrieved from the dedicated literature. The outcome shows that workflows unseen by the mining tool were correctly guessed with a high accuracy. A challenging research direction would be the design of a domain-specific interestingness criteria exploiting the semantic knowledge in the ontology.

7. REFERENCES

[1] T. Oinn *et al.*, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, 2004.

[2] D. Blankenberg *et al.*, *Galaxy: A Web-Based Genome Analysis Tool for Experimentalists.* John Wiley and Sons, Inc., 2010.

[3] E. Lord *et al.*, "Armadillo 1.1: An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and Simulations," *PloS one*, vol. 7, no. 1, Jan. 2012.

[4] W. van der Aalst, *Process Mining: The Missing Link.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 25–52.

[5] C. A. Ellis *et al.*, "Beyond workflow mining," in *Business Process Management*, 2006, pp. 49–64.

[6] H. Dai and B. Mobasher, "Using ontologies to discover domain-level web usage profiles," *Semantic Web Mining 2nd Workshop at ECML/PKDD, Finland*, 2002.

[7] M. Adda *et al.*, "A framework for mining meaningful usage patterns within a semantically enhanced web portal," in *Proc. of the 3rd conf. CSSE.* ACM, 2010, pp. 138–147.

[8] P. Senkul and S. Salin, "Improving pattern quality in web usage mining by using semantic information," *Knowledge and Information Systems*, vol. 30, no. 3, pp. 527–541, 2012.

[9] A. Halioui *et al.*, "Acquisition of generic problem solving knowledge through information extraction and pattern mining," in *Tools with Artificial Intelligence, 2015 IEEE 27th Int. Conf. on*, Nov 2015, pp. 583–590.

[10] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

ANNEXE C

T-GOWler: Discovering Generalized Process Models within Texts

Ahmed Halioui, Petko Valtchev and Abdoulaye Baniré Diallo*
Computer Science Department, Laboratoire de Bioinformatique
Université du Québec à Montréal (UQÀM), Canada
halioui.ahmed, valtchev.petko, diallo.abdoulaye@uqam.ca

Abstract

Contemporary workflow management systems are driven by explicit process models specifying the inter-dependencies between tasks. Creating these models is a challenging and time-consuming task. Existing approaches to mining concrete workflows into models tackle design aspects related to the diverging abstraction levels of the tasks. Concrete workflow logs represent tasks and cases of concrete events –partially or totally ordered– grounding hidden multi-level (abstract) semantics and contexts. Relevant generalized events could be rediscovered within these processes. We propose in this paper an ontology-based workflow mining system to generate patterns from sequences of events which are themselves extracted from texts. Our system `T-Gowler` (Generalized Ontology-based WorkfLow minER within Texts) is based on two ontology-based modules: a workflow extractor and a pattern miner. To this end, it uses two different ontologies: a domain one (to support workflow extraction from texts) and a processual one (to mine generalized patterns from extracted workflows). `T-Gowler` implementation and data sources are available in: `https://github.com/bioinfoUQAM/tgowler`.

1 Introduction

Workflows (Taylor et al., 2014) are increasingly designed as an efficient way of managing complex multi-step processes for the resolution of complex problems from knowledge-rich domains such as medicine, biology, chemistry and bioinformatics. This conducts to a growing popularity of workflow platforms offering a wide spectrum of analytical services (Wolstencroft et al., 2013; Blankenberg et al., 2001; Lord et al., 2012). However, in order to build successful workflows, the user must typically master a large variety of methods, tools, data formats, *etc.* In many cases, the ultimate success of the workflows depends on the right choice of the included methods, software parameters and versions, *etc.* In fact,

*Correspondance author.

such a demanding environment reduces the value of Workflow Management Systems (WMS) for non expert users. Facing these challenges. *Process Mining* (Ellis et al., 2006; Van der Aalst. 2016), a branch of Knowledge Discovery from Data (KDD) (Fayyad et al., 1996), studies the discovery of reusable knowledge from databases of process records, in particular, from workflow representations such as execution logs. As workflows have an inherent graph structure, effective process mining methods face the related complexity issues. A workflow graph depicts an analytical resolution in terms of blocks and their interactions (control/data flows) drawn as nodes and links, respectively. Yet. given the large spectrum of possible inputs, it is unlikely for a recorded workflow to fully fit a new situation. A solution could be to mine higher-level of abstractions summarizing commonalities of a set of workflows. Representing abstraction levels using ontologies to assist mining of usage logs has proven popular in the last decade (Dai and Mobasher, 2002; Senkul and Salin, 2012). A complete framework for abstracting ontology-based patterns out of click-streams (from the logs of a semantic web portal powered by a domain ontology) provides an alternative view (Adda et al., 2010). Yet, here, we deal with a comparable problem on a different type of input format. Instead of pure sequences, we consider workflows as DAGs. However, due to the complex nature[1] of the underlying domain problem, the DAGs are split into ordered layers.

The rest of the paper is organized as follows: section 2 presents related work and background information on pattern mining. In section 3, we define the pipeline of our approach to extract workflows from texts and thus, mining abstract patterns from them. Section 4 presents the experimental evaluation and results on Bioinformatics workflows, *i.e.*: *Phylogenetics* and section 5 concludes the paper.

2 Background and related work

Given a universe of items O, a database $\mathcal{D} = \{W\}$ of records (*e.g.* processes) $w \in W$ combining items from O, and a frequency threshold σ, the pattern mining goal is to extract the family $\mathcal{F}_\sigma$ of patterns p present in at least σ records. Two binary relations are generally used in the pattern mining problem combining hierarchies: the *generality*[2] between patterns $\sqsubseteq_\Gamma$ (where Γ is the pattern language) and the instantiation $\propto$ between a data record and a pattern.

Inspired from the work of Bamshad Mobasher (1995) including knowledge representations in pattern mining domain, Srikant and Agrawal (1996) propose to use hierarchies in their GSP algorithm (Generalized Sequential Patterns) which searches for $\mathcal{F}_\sigma$ through the pattern space $\langle \Gamma, \sqsubseteq_\Gamma \rangle$ by exploring the monotony in frequency *w.r.t* the generality operator $\sqsubseteq_\Gamma$. Their *Apriori* miner performs a level-wise top-down traversal of the pattern space. On itemsets, it examines patterns at level k - that is, of size k - on two points: the method computes the frequency of candidate patterns by matching them against the

[1] The definition of the domain concepts and constraints is a complex task itself.
[2] Also know as the raising operation

records in $\mathcal{D}$ whereas it generates $k+1$ candidates by combining pairs of frequent k-patterns. However, their approach is expensive in terms of memory and computation. Their representation of $\mathcal{D}$ becomes huge while exploring deeper levels on the hierarchy. Han and Fu (1995) use associations in one level at a time, while Fortin and Liu (1996) propose an OO representation to generate association rules of different levels of abstraction. Furthermore, Phillips and Buchanan (2001) propose Prolog ontologies in an inference rule-based system using the first logic order language. Zhou and Geller (2007) propose a Generalization Impact measure based on support in order to estimate the sufficient level of abstraction to generate interesting generalized patterns.

Adda et al. (2010) propose a fully-blown ontology. First, the data sequences are enriched with domain properties from the domain ontology. The resulting structures are akin to labeled digraphs, in which vertices are objects O and edges represent links $\mathcal{L}$ between objects or the induced sequential order. Patterns are generalized ontological digraphs, *i.e.*, made of sets of ontology classes connected by properties (plus sequencing). Matching between data sequences and patterns is defined on-top of the standard ontological instantiation between objects and classes and represents an order preserving injective graph morphism. Their proposed `xPMiner` method applies an Apriori-like level-wise mining whereby levels in the pattern space are defined *w.r.t.* the overall depth of the pattern elements within the ontology. Candidates of level $k+1$ are generated from k-level frequent patterns by a *refinement operator* (see De Raedt, 2008). A major shortage of the proposed approach is the cost of the plain pattern-to-data matching operation. Indeed, as no optimization is proposed, the large number of labeled digraph morphism computations takes a significant toll on the method performances. In our own study, we choose to enhance that approach by saving in the memory previous matched sequences (from level k) using the last operation information from the refinement operator. In addition, data records, *i.e.* workflows (instead of sequences) that we deal with are partially ordered multisets where sets (transactions) represent workflow steps, hence, forming (double)-layered DAGs. Here, DAG nodes represent tasks, parameters, data and metadata items and edges represent the control and data flows. In addition, these elements are extracted automatically from scientific texts.

3 T-GOWler: ontology-based workflow pattern acquisition from texts

We describe, in this section, the pipeline of the methods to extract abstract patterns within texts. The proposed `T-Gowler` system contains two main engines: (A) `WfExtractor` extracts workflows W from texts and builds a workflow database $\mathcal{D}_W$ and (B) `WfMiner` mines frequent generalized patterns $\mathcal{F}$ from $\mathcal{D}_W$ (see *Fig_1*). The first module (developed on GATE platform[3] by Cunningham

[3]GATE (`www.gate.ac.uk`) is a well-known environment for annotation tasks using different NLP (Natural Language Processing) techniques, *e.g.*, the OpenNLP and Stanford POS tagger.

et al. (2009)) requires plain text articles and an RDF[4] domain ontology representing the domain terminology (*Gazetteer*) categorizing, into abstract concepts and relations, domain terms. The extraction pipeline is based on a simple NLP chain and a specific workflow tagger to classify workflow components, *i.e.* objects O and links $\mathcal{L}$, into their ontological categories, *i.e.* concepts C and relations $\mathcal{R}$. respectively. In order to preserve processes order in workflows. abstract patterns from the processual ontology are used. Once the ontology is loaded and concrete workflows are reconstructed, the `WfMiner` tool deals with the generation of frequent patterns using a level-wise pattern generation on the top of the ontology and a specific pattern-to-workflow matching algorithm to filter non frequent patterns.

3.1 WfExtractor: workflow extraction from texts

The extraction process takes place 3 different steps: (1) named entity ($o \in O$) extraction, (2) link ($l \in \mathcal{L}$) extraction and (3) workflow $w \in \mathcal{D}_W$ reconstruction from the extracted elements. Yet, before workflow component extraction, a classic NLP tool chain is executed in order to tag terms with their morphosyntactic tags, *i.e.*: grammatical classes within sentences. This chain is composed from: (1) the specialized biomedical tokenizer (Settles, 2004) and a simple sentence splitter (Bontcheva et al., 2013), (2) the MedPost/SKR POS tagger (Smith et al., 2004) and (3) the lightweight OpenNLP chunker (Buyko et al., 2006) to identify active and passive verb phrases in texts.

For both named entity and link extraction, a domain ontology is applied as a Gazetteer in order to recognize workflow elements in texts. However, this process faces some hurdles due to the syntactic and semantic ambiguities, *i.e.* *polysemy* (Stevenson and Wilks. 2003) in some domain concepts. For instance. the program term *MEGA* (Kumar et al., 2016) could be used to either validate the phylogenetics hypothesis or to visualize output data, *e.g.* phylogenetic trees. Thereof, we propose to use a supervised machine learning approach (Nadeau and Sekine, 2007) to disambiguate software contexts using specific JAPE rules[5]. For example, when the passive verb phrase (context) contains a 'program' term in its object and the root verb is a synonym to "valid" (from the *HypothesisValidation* concept), then, its probability to be a validation program is very high. Hence, for each polysemic term in the texts, we create two rules: one for the active and the other for the passive voice phrases. In such way, a learning set is constructed. Next, two sets of features are calculated to annotate ambiguous terms automatically, *i.e.*: (1) POS annotations and (2) surrounded concepts (domain concepts in the context sentence of the ambiguous word) in a window of the five next and previous concepts and tokens. In order to represent our context-based extracted features, the PAUM (Perceptron Algorithm with

[4] `https://www.w3.org/TR/rdf-primer`

[5] Java Annotation Patterns Engine (JAPE). JAPE language is a finite state transducer on the GATE platform (Cunningham et al., 2009) that operates over annotations based on regular expressions. Thus, it is useful for pattern-matching, semantic extraction, and many other operations over syntactic trees such as those produced by natural language parsers.

Uneven Margins) model (Li et al., 2002) is applied to learn Word Sense Disambiguation (WSD) (Stevenson and Wilks, 2003) and relation classifications from the extracted feature instances. PAUM is specifically designed to simulate the distance between positive and negative examples which fits the nature of our features since positive and negative contexts are represented with distance-based features (in POS tags and concepts in the same sentence).

For the relation side, we adopt the following hypothesis: A link between two related concepts exists in the same sentence evoking the instances of its domain (source term) and range (destination term). Such link is represented in the verb located between the concerned terms (depending on the sentence voice). Hence, in order to recognize links in texts, we map the couple c_1, c_2, from the triple $\rho = (c_1, r, c_2)$ in the ontology, on text sentences. For example, for the relation instances of r: *isRetrievedIn*, defined with c_1: *DataType* as the relation domain, and c_2: *Database* as the relation range, we tag the context of the link with its relation type, *e.g.* *alined in* in the text is tagged with the *isRetrievedIn* relationship. A PAUM model is then constructed using the following set of features: (1) token POS tags, (2) distance, (3) direction and (3) the list of verbs between domain and range instances in the context phrase (see table ??).

Finally, from the recognized links between named entities in texts, we build a concrete workflow knowledge database $\mathcal{D}_W$ where items are named entities ordered by the *genInputTo* properties (relations) from the processual ontology. For example, from the tuples: (ρ_1) (*16srRNA*, "*isInputOf*", *ClustalW*), (ρ_2) (*ClustalW*, "*genInputTo*", *PHYLIP*) and (ρ_3) (*PHYLIP*, "*genInputTo*", *TreeView*), we construct the sequence $\langle$ { {*16srRNA*}, {*ClustalW*}, {*PHYLIP*}, {*TreeView*} }, { (1, "*isInputOf*", 2) } $\rangle$ where 1 and 2 correspond the domain and range positions, respectively, in the workflow sequence.

3.2 WfMiner: mining generalized workflows

Here, we propose an new approach to generate generalized sub-graphs of workflows out of concrete ones. We propose two descriptive languages to represent the pattern space. The data language Δ_w is derived from sequences of objects and sets of link IDs from the processual ontology Ω. For instance, the workflow DAG w_1 in *Fig.2* is a poset $w.\zeta$ of object items $i \in \{1,..7\}$ and a set of link triples $w.\theta$ of with link types $l \in \{1,..6\}$. The second language Γ_w is used to represent a pattern p as an abstraction of $w \in \mathcal{D}_W$ (*e.g.*, $w_1 \dot{\propto} p_1$: w_1 instantiates p_1). Hence, we represent a pattern p as a sequence of concept itemsets $p.\zeta$ and a set of relation triples $p.\theta$.

Level-wise descent in the pattern space relies on the generation of $k+1$-level patterns from k-level ones, which amounts to the computation of five canonical operations in a *Depth-First* search strategy from the ontology: (1) appC - appends a root concept to a new transaction, (2) addC - adds a root concept to the last transaction, (3) addR - adds a root relation between a concept and the last one in the last transaction, (4) subC - specializes the last concept and (5) subR - specializes the last relation in the pattern. Hence, for each pattern p.

we apply all the five canonical operations. Although these operations constitute an effective generation mechanism. Their unrestricted use is redundancy-prone and hence, would harm the efficiency of the global mining process. In the previous work of Adda et al. (2007), calculating the support of a pattern p amounts to match p with all the workflow concrete sequences in $\mathcal{D}_W$. For this purpose, we introduce a new data structure to prune patterns while calculating their supports using the generality relationship between them. Calculating frequent patterns p^{k+1} begins from the last visited concept/relation in the parent pattern p^k and hence, we don't need to recalculate all the matching possibilities for each concept and relation previously matched at level k. The proposed matching structure MS is a "stack" of operation results (adding/specializing concepts/relations) on parent patterns. For example, the arrows in *Fig.3* show the way we obtain a pattern p_2 using the refinement operator. Calculating the support of p_2 is done starting from the last operation on p_1, *i.e.* addR(r_2, c_{11}). If the next operation in the $k+1^{th}$ iteration is to add another relation addR(r_3, c_{14}) to p_2, then, the matching structure of p_2 ($MS(p_2)$) puts relation blocks ('*source*' and '*destination*' relation blocks s) in their proper positions and calculates the possible matchings from the matching set of the parent pattern $MS(p_1)$.

4 Experiments and results

4.1 Case study: phylogenetics analyses

Phylogeny (Anne-Mieke, 2009) represents the evolutionary history of a group of organisms having a common ancestor. Inferred from nucleic acid (DNA or RNA) or protein sequences, phylogenetic relations are discovered with tools detecting different evolutionary phenomena such as duplications, insertions and deletions (see Anisimova et al., 2013). We downloaded more than 3 000 scientific articles (published from 2013 until April 2015) about phylogenetics studies from the PubMed Central (PMC) repository[6] in XHTML format. In addition, we constructed a terminology from different well-known databases in the phylogenetics literature such as the UniprotKB[7] (for gene and protein names), Gene Ontology (Ashburner et al., 2000), Taxonomy Browser[8] and the Felsenstein web site [9]. This gazetteer is then, reorganized in a domain ontology in order to recognize terms and relations in texts like described in the `WfExtractor` module. Next, we build manually a processual ontology with the help of experts. An example of an abstract processual knowledge is: (1) *Protein collected from the PDB database*[10] *are fed to* (2) *a homology search program and* (3) *a phylogenetic inference one.* Finally, both domain and processual ontologies are merged into

[6] `http://www.ncbi.nlm.nih.gov/pmc`
[7] `http://www.uniprot.org`
[8] `http://www.ncbi.nlm.nih.gov/Taxonomy`
[9] `http://evolution.genetics.washington.edu/programs.html`
[10] `http://www.rcsb.org/pdb`

one[11] (see *Fig.4*) in order to mine frequent patterns from the workflow database using the `WfMiner` engine.

4.2 Named entity learning results

An expert has annotated manually 100 phylogenetic sections to build a reference workflow database. Each extracted term has been compared to the proposed PAUM models results in order to validate (with 10-fold-cross-validation) their accuracies in terms of precision and recall and fMeasure (see *Table* 1). Models context features have well learned and applied several top-level domain relation and concept terms, i.e.: data types, gene ontology annotations, phylogenetic methods and phylogenetic inference programs. In total, 1 469 workflow objects have been extracted from the phylogenetic analyses sections with an average of 71% of fMeasure. In the relation side, the system annotated 325 links with 72% of precision and 84% of recall (78% of fMeasure). Hence, workflows overall fMeasure is 73%. Indeed, with only context POS tags and concept neighbours, we did recognize workflow components in texts with good accuracies.

4.3 Generalized-pattern results

For the rest of our experiments, we consider only expert curated datasets (that we call here *PhyloFlows*) in order to assess the `WfMiner` module. This database is composed from 300 workflows with an average of 11 object items and 9 links per workflow. In addition, we use the *eTP-tourism* (Electronic Tourism Platform) workflow database which is a portion of a log file describing web pages navigation of 56 different users in a Web travel site (Adda et al., 2010). This database has an average of 19 different items and 6 links per sequence and enhanced by an ontology of 157 concepts and 35 object properties.

Fig.5 (A) shows the difference in mining time between the *PhyloFlows* and *eTP-tourism* databases. Running time does eventually increases when more concepts and links are mined. Our system spends ~10 times the mining time to generate patterns with more 500 items and 100 links. While *eTP-tourism* database presents more dense graphs, matching sub-graphs and mining processes become heavier to handle. However, taking 23 seconds to do the whole process of matching and generating ~6 000 frequent graph patterns is still a reasonable time. In addition, our `WfMiner` outperforms `xPMiner` by Adda et al. (2010) (see *Fig.5* (B)) with 2 times order of magnitude in high threshold supports (>0.8). With 60% of minimum support, our approach executes ~10 times faster the mining algorithm than `xPMiner`.

Finally, we present in *Table* 2 a sample from the generated set of patterns obtained from the *PhyloFlows* database. These patterns could be used in a semantically rich recommendation framework considering the diversity of levels of abstraction in pattern representations. For instance, if we know that the 16s *rRNA* gene is used as input in the *ClustalW* alignment program (Thompson et al., 2002), then a concrete recommendation could offer for example

[11] `http://bioportal.bioontology.org/ontologies/PHAGE`

MrBayes (Ronquist and Huelsenbeck, 2003) as a phylogenetic inference tool (~1% frequent). However, offering "any" character based Inference program (see 2^{nd} pattern in *Table* 2) is a more relevant recommendation due to its (pattern) higher frequency (~10%) in the database: since it contains more "general" concepts.

5 Conclusion

Our ontology-based approach to mining generalized patterns from texts proves that different kinds of knowledge could be extracted from the phylogenetics literature. Processual and domain knowledge from articles show interesting patterns to mine. Different ontologies have been used to represent the different types of knowledge into concepts and relations. Extracted workflows are enhanced with ontological annotations to produce generalized frequent patterns. These latter, covering a wider range of values then concrete elements, could be used to assist practitioners during their problem-solving task with a richer vocabulary. The proposed system could also be applied to different domains when expertise is confronted to domain knowledge. Video games, for example, is a good domain application to our approach when gamers are constantly faced to cognitive processual gestures. Workflows of events could be then extracted from game sessions and mined in order to recommend and predict generalized patterns like: what type of event is going to happen during a game based on gamer logs or what kind of item a player is willing to buy to accomplish a mission.

Acknowledgments

This work has been partially supported by the NSERC Discovery Grants of Canada of the 2^{nd} and 3^{rd} authors.

Author disclosure statement

No competing financial interests exist.

References

Adda, M., Valtchev, P., Missaoui, R., and Djeraba, C. 2007. Toward recommendation based on ontology-powered web-usage mining. *IEEE Internet Computing* 11:45–52.

Adda, M., Valtchev, P., Missaoui, R., and Djeraba, C. 2010. A framework for mining meaningful usage patterns within a semantically enhanced web portal. *In* Proceedings of the Third C* Conference on Computer Science and Software Engineering, pp. 138–147. ACM.

Anisimova, M., Liberles, D. a., Philippe, H., Provan, J., Pupko, T., and von Haeseler, A. 2013. State-of the art methodologies dictate new standards for phylogenetic analysis. *BMC evolutionary biology* 13:161.

ANNE-MIEKE, L. P. S. M. V. 2009. The phylogenetic handbook: a practical approach to phylogenetic analysis and hypothesis testing. Cambridge University Press.

ASHBURNER, M., BALL, C. A., BLAKE, J. A., BOTSTEIN, D., BUTLER, H., CHERRY, J. M., DAVIS, A. P., DOLINSKI, K., DWIGHT, S. S., EPPIG, J. T., ET AL. 2000. Gene ontology: tool for the unification of biology. *Nature genetics* 25:25–29.

BAMSHAD MOBASHER, S. S. A. 1995. The role of domain knowledge in data mining. *In* Proceedings of the 4th int. conference on Information and knowledge management. pp. 37–43. ACM.

BLANKENBERG, D., KUSTER, G. V., CORAOR, N., ANANDA, G., LAZARUS, R., MANGAN, M., NEKRUTENKO, A., AND TAYLOR, J. 2001. Galaxy: A Web-Based Genome Analysis Tool for Experimentalists. *Current protocols in molecular biology / edited by Frederick M. Ausubel ... [et al.]* Chapter 19.

BONTCHEVA, K., DERCZYNSKI, L., FUNK, A., GREENWOOD, M. A., MAYNARD, D., AND ASWANI, N. 2013. Twitie: An open-source information extraction pipeline for microblog text. *In* RANLP, pp. 83–90.

BUYKO, E., WERMTER, J., POPRAT, M., AND HAHN, U. 2006. Automatically adapting an nlp core engine to the biology domain. *In* Proceedings of the Joint BioLINK-Bio-Ontologies Meeting. A Joint Meeting of the ISMB Special Interest Group on Bio-Ontologies and the BioLINK Special Interest Group on Text Data Mining in Association with ISMB, pp. 65–68.

CUNNINGHAM, H., MAYNARD, D., BONTCHEVA, K., TABLAN, V., URSU, C., DIMITROV, M., DOWMAN, M., ASWANI, N., ROBERTS, I., LI, Y., ET AL. 2009. Developing Language Processing Components with GATE Version 5: (a User Guide). University of Sheffield.

DAI, H. AND MOBASHER, B. 2002. Using ontologies to discover domain-level web usage profiles. *Semantic Web Mining 2nd Workshop at ECML/PKDD, Finland* .

DE RAEDT, L. 2008. Logical and relational learning. *In* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 5249 LNAI, p. 1.

ELLIS, C. A., REMBERT, A. J., KIM, K.-H., AND WAINER, J. 2006. Beyond workflow mining. *In* S. Dustdar, J. L. Fiadeiro, and A. P. Sheth (eds.), Business Process Management, volume 4102 of *Lecture Notes in Computer Science*, pp. 49–64. Springer.

FAYYAD, U. M., PIATETSKY-SHAPIRO, G., SMYTH, P., AND UTHURUSAMY, R. 1996. Advances in knowledge discovery and data mining, volume 21. AAAI press Menlo Park.

FORTIN, S. AND LIU, L. 1996. An object-oriented approach to multi-level association rule mining. *In* Proceedings of the fifth int. conf. on Information and knowledge management, pp. 65–72. ACM.

HAN, J. AND FU, Y. 1995. Discovery of multiple-level association rules from large databases. *In* VLDB, volume 95, pp. 420–431.

KUMAR, S., STECHER, G., AND TAMURA, K. 2016. Mega7: Molecular evolutionary genetics analysis version 7.0 for bigger datasets. *Molecular biology and evolution* p. msw054.

LI, Y., ZARAGOZA, H., HERBRICH, R., SHAWE-TAYLOR, J., AND KANDOLA, J. 2002. The perceptron algorithm with uneven margins. *In* ICML, volume 2, pp. 379–386.

LORD, E., LECLERCQ, M., BOC, A., DIALLO, A. B., AND MAKARENKOV, V. 2012 Armadillo 1.1: An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and Simulations. *PloS one* 7:e29903

Nadeau, D. and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30:3–26.

Phillips, J. and Buchanan, B. G. 2001. Ontology-guided knowledge discovery in databases. *In* Proceedings of the 1st int. conf. on Knowledge capture, pp. 123–130. ACM.

Ronquist, F. and Huelsenbeck, J. P. 2003. Mrbayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics* 19:1572–1574.

Senkul, P. and Salin, S. 2012. Improving pattern quality in web usage mining by using semantic information. *Knowledge and Information Systems* 30:527–541.

Settles, B. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. *Proceedings of the int. Joint Workshop on Natural Language Processing in Biomedicine and its Applications - JNLPBA '04* p. 104.

Smith, L., Rindflesch, T., Wilbur, W. J., et al. 2004. Medpost: a part-of-speech tagger for biomedical text. *Bioinformatics* 20:2320–2321.

Srikant, R. and Agrawal, R. 1996. Mining sequential patterns: Generalizations and performance improvements. *In* Proceedings of the 5th int. Conference on Extending Database Technology: Advances in Database Technology, EDBT '96, pp. 3–17, London, UK, UK. Springer-Verlag.

Stevenson, M. and Wilks, Y. 2003. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics* pp. 249–265.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. 2014. Workflows for e-Science: scientific workflows for grids. Springer Publishing Company, Incorporated.

Thompson, J. D., Gibson, T., Higgins, D. G., et al. 2002. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics* pp. 2–3.

Van der Aalst, W. 2016. Process mining software, pp. 325–352. *In* Process Mining. Springer.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S., and Goble, C. 2013. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research* 41:W557.

Zhou, X. and Geller, J. 2007. Raising, to enhance rule mining in web marketing with the use of an ontology. *Data Mining with Ontologies: Implementations, Findings and Frameworks* pp. 18–36.

Tables

Table 1: Evaluation results of the extracted workflow elements based on the expert curated dataset from the PubMed PMC articles (published from January 2013 to April 2015).

Concept/Relation	# Instances	Precision	Recall	fMeasure
DataType	597	0.68	0.75	0.71
GeneOntology	139	0.94	0.44	0.60
Taxon	202	0.72	0.58	0.64
Method	326	0.84	0.84	0.84
InferenceProgram	205	0.56	0.88	0.68
Overall (concepts)	1 469	0.74	0.69	0.71
Overall (relations)	325	0.72	0.84	0.78
Overall	**1 794**	**0.74**	**0.72**	**0.73**

The table shows the number of annotated instances of (some - from the top level of abstraction) concepts and relations and their precision, recall and fMeasure evaluation values.

Table 2: A sample from the phylogenetic extracted patterns within the texts in the PubMed PMC database (articles published from January 2013 to April 2015).

Workflow patterns		σ
p.ζ (concepts)	p.θ (relations)	
$\langle$\{*DataType*\}, \{*DataCollection*\}, \{*MultipleSequenceAlignment*\}, \{*PhylogeneticInference*\}$\rangle$	(1. *isInputOf*, 2), (1, *isOutputOf*, 4), (2 ,*genInputTo*, 3), (3, *genInputTo*, 4)	40%
$\langle$\{*RNA*, *Tree*\}, \{*HomologyS*\}, \{*CharacterBasedProgram*\}, \{*TreePlottingProgram*\}, \{*GTR*\}$\rangle$	(1, *isInputOf*, 2), (2 ,*isOutputOf*, 5). (3, *genInputTo*, 4), (3, *hasParameter*, 6), (4, *genInputTo*, 5)	10%
$\langle$\{*ProteinSequence*, *Alignments*, *Trees*\}, \{*Virus*\}, \{*GlobalMSA*\}, \{*ModelSelection*\}, \{*MaximumLikelihood*\}, \{*Bootstrapping*\}, \{*PDB*, *JTT*\}$\rangle$	(1, *hasTaxon*, 4), (1, *isInputOf*, 5), (2 , *isOutputOf*, 5), (2 , *isInputOf*, 6), (5, *genInputTo*, 6), (5, *hasParameter* , 9), (6, *genInputTo*, 7), (6, *hasParameter* , 10). (7, *genInputTo*, 8)	1%

Generated DAG patterns are sequences of concept multisets and sets of relation triples. For instance, in the 1^{st} pattern: (1, *isInputOf*, 2) represents the input relation between the 1^{st} concept (*DataType*) and the 2^{nd} one (*DataCollection*) in *p.ζ*.

ANNEXE D

RESEARCH

Workflow Pattern Acquisition out of Bioinformatic Scientific Texts

Ahmed Halioui[1], Petko Valtchev[1] and Abdoulaye Baniré Diallo[1,2*]

Abstract

Background: Contemporary workflow management systems are driven by explicit process models specifying the inter-dependencies between tasks. Creating these models is a challenging and time-consuming task. Existing approaches to mining concrete workflows into models tackle design aspects related to the diverging abstraction levels of the tasks. Concrete workflow logs represent tasks and cases of concrete events –partially or totally ordered– grounding hidden multi-level (abstract) semantics and contexts. Relevant generalized events could be rediscovered within these processes.

Results: We propose in this paper an ontology-based workflow mining system to generate patterns from sequences of events which are themselves extracted from texts. Our system `T-Gowler` (Generalized Ontology-based WorkfLow minER within Texts) is based on two ontology-based modules: a workflow extractor and a pattern miner. To this end, it uses two different ontologies: a domain one (to support workflow extraction from texts) and a processual one (to mine generalized patterns from extracted workflows). `T-Gowler` implementation and data sources are available in: `https://github.com/bioinfoUQAM/tgowler`.

Conclusions: Extracted patterns would make workflows more intuitive and easy to be reused in similar situations. This could provide a stepping-stone into the identification of best practices and pave the road to a recommender system.

Keywords: ontologies; workflow extraction; pattern mining

1 Introduction

The advances in biotechnology have revolutionized bioinformatics into a data-intensive discipline. The rapidly increasing amounts of available data, from new large scale analyses, have made data processing without automated pipelines (*a.k.a.* workflows) difficult to handle. This situation is mainly due to the complexity of integrating diverse data formats and databanks, interfacing numerous programs with various parameters, as well as to the assessment of data inputs and outputs. Current bioinformatic platforms offer a variety of computational solutions (using High-Performance and Cloud Computing) with different levels of computational abstractions [1]: two types of models are employed by Workflow Management Systems (WMS): (1) concrete ones, in which the tasks are bound to specific resources, and (2) abstract ones, presenting an abstract view of the resources used in tasks. Nowadays, service-oriented systems such as Pegasys [2], Taverna [3], Galaxy [4] or Armadillo [5] offer models based on Web services providing an abstraction layer which separates the workflow platform from programming-language details. The state-of-the-art analysis revealed that biomedical services are currently used in an ad-hoc style with a low reusability [6]. This suggests a need for techniques that help scientists better locate interesting services and workflow fragments and reuse them to attain their research purposes. Current frameworks such Taverna and Galaxy offer different approaches for Web services discovery and a semi/full automated way to workflow composition. However, none of them provide recommendations to guide users and suggest the next steps to do while taking into account the specific nature of data (complete vs partial genomes, viral proteins, plant DNA, type of the evolutionary model, *etc.*) and relevant analyses (phylogenetic analyses, function prediction, homology studies, genome wide association studies, *etc.*). For example, in phylogenetic tree reconstruction, the evolutionary history of a group of organisms is inferred using different types of data including nucleic acid (DNA or RNA) or protein sequences. Depending on tools which one

*Correspondence: Diallo.Abdoulaye@uqam.ca
[1]Département d'informatique, Université du Québec à Montréal (Québec), PO BOX 8888 Downtown Station, H3C 3P8 Montréal (Quebec), Canada
Full list of author information is available at the end of the article

would use, different types of evolutionary events could be included in the study such as point mutations, duplications, insertions and deletions, genome rearrangements, *etc.* [7]. Such a study requires several complex steps and tool parameterizations and the acquisition of raw data.

Current bioinformatics workflow repositories like myExperiment [8] offer a large but a very generic scope of workflow-based analyses over different kinds of platforms. Solving very specific bioinformatic problems such phylogenetic analyses is still limited to few[1] published and shared -non validated- workflows. Moreover, the literature of scientific texts offers a more concise and complete information about bioinformatics analyses. Phylogenetic analysis constitutes a typical bioinformatic workflows in life sciences. This is witnessed by the large number of publications on the topic. This resource could be exploited to enrich workflow solutions. However, the difficulties in the process of workflow reconstruction from scientific text publications are illustrated in the following example.

Example 1. A biologist U_1 downloads a set of 9 hemagglutinin sequences of different Influenza viruses from the NCBI database. To carry out the analysis of those sequences. U_1 chooses the Armadillo [5] platform[2]. Several algorithmic options are displayed: i.e. software parameters (*e.g.* the number of bootstraps, mutation rate, *etc.*) and data constraints (*e.g.* input/output to programs). U_1 prior *abstract* experience does not cover all the available methods and tools, therefore, the backup solution is to adapt a workflow from the literature (see Fig. 1). However, none of the workflows drawn from scientific publications could exactly match his needs. These workflows may not properly execute since a part of the analysis requires several *specific* software versions and dependencies.

The above example shows several problems which could occur while constructing bioinformatic workflows. Even simple routine projects can involve a number of different software requiring specific but also abstract problem solving knowledge. Even though software experimental conditions are related to several choices. the amount of data and the misuse of tools could also deteriorate the results of studies. This is where user skills and experience do make the difference.

[1]Less than 100 workflows representing phylogenetic analyses in the myExperiment database.

[2]Armadillo (www.bioinfo.uqam.ca/armadillo) is a drag and drop workflow platform dedicated to phylogenetic as well as general bioinformatics analysis.

Example 2. Suppose that a bioinformatician U_2 chooses to study the same protein sequences of the hemagglutinin. He runs *ClustalX* [9] to align sequences and then chooses the *JTT* evolutionary model [10] during the tree reconstruction. If we want to recommend a concrete phylogenetic program to U_2 such as the program *MEGA* [11], we propose to calculate the frequency of the sequence $s_0 : \langle \{ClustalX\}, \{JTT\}, \{MEGA\} \rangle$ in the extracted workflow database. However, such sequence is very rare due to the vast varieties of available tools for each specific task. Thus. using frequent sequences of tool categories such as the "*maximum parsimony*" or "*maximum likelihood*" in a recommendation scenario is more relevant (due to the higher support of tool categories over instances). An abstract pattern such $p_0 : \langle \{LocalAlignment\}, \{ProteinModelSelection\}, \{MaximumLikelihood\} \rangle$ is more frequent than the concrete pattern s_0.

Here, we use ontologies to represent the different categories (*concepts*) of tools and input/output data at different levels of abstractions. The constraints on combining tools and parameter values are expressed using properties (*relations*) between categories. In fact, the ontological representations are effectively exploited by two key processes: (1) the extraction of workflow terms and relations from texts (*concrete workflows*), and (2) the extraction of generalized workflow patterns (*abstract workflows*). Thereof, we distinguish *concrete* workflows made of individuals (instances) and links between them, from *abstract* workflows, comprising labels of concepts and properties at various levels of abstraction. In our approach, mining processual abstract patterns out of the concrete workflow cases is based on the hierarchies of both, concepts and relations, from the domain ontology. This could be the foundation of a semantically rich recommendation system to suggest the best next step practices.

The remainder of the paper is organized as follows: section 2 presents related work to workflow recommender systems in bioinformatics. Section 4 presents the experimental evaluation and results of the extracted workflows and patterns from texts. Finally, section 5 discusses the quality of our approach.

2 Related work

Current bioinformatics WMS have emerged as global *intelligent* infrastructures[3] by integrating new technologies to maintain a certain level of reusability through grid and cloud computing [1, 12, 13]. For example, they may offer an efficient way to abstract *services* for data transformation and searching resources

[3]For example, these systems do not suggest local configurations for tools but a global way to manage tasks through Web services.

in distributed secure databases. However, in practice, the sharing service-based capabilities may open a gateway to reuse workflows, however, the degree of reuse is still very low [14]. For example, repositories like myExperiment, CrowdLabs and SHIWA (see the review in [6]) offer a restricted number of workflows (*e.g* ∼2 500 workflows in myExperiment) for very various disciplines (chemistry, astronomy, sequence analyses, *etc.*) and in different formats (Taverna, Galaxy, Rapidminer, *etc.*) which decreases the reusability of workflows. Hence, in the literature of bioinformatics best practises, more work reported on service oriented pipelines and/or provide expert-based suggestions [3]. In this vein, [1] review the most advanced projects defining general guidelines for the methodological choices at various stages of a bioinformatics pipeline. The authors recommend dedicated practices in maintaining reproducibility and standardization in local and distributed environments. [7] criticize the outdated methodology and provides practical guidance on issues and constraints faced by each of the phylogenetic analysis steps. [15] propose a set of high-level abstract motifs (patterns) as a view of activities undertaken within workflows. They performed their manual analysis over a set of real-world scientific workflows extracted from Taverna and Wings systems. However, expert recommendations are better used in solving problems and making decisions if they are supported by the help of Knowledge Management (KM) discipline. KM is the subject that provides strategy, process and technology to share information and expertise among users. For instance, Semantic Web (SW) technologies [16], as a special case of KM, allow automatic discovery and execution of Web services that can handle the workflow tasks using ontologies to improve knowledge integration, querying, and sharing [17]. Next, we present current solutions for bioinformatics recommendation and automatic workflow composition into three categories: (1) semantic-based recommendation, (2) similarity-based recommendation and (3) pattern-based recommendation.

Semantic-based recommendation. Proteus [18] is one of the first projects to integrate an ontology-based design in bioinformatic workflows. It exploits a *DAML + OIL* ontology for the data mining domain describing resources and processes of knowledge discovery in databases. [19] describe a SW solution assuming a grid computing environment. The proposed Bio-STEER system offers SW services in OWL-S[4]. A small-size ontology is employed in the representation of both abstract workflows (for Grid resources) and concrete ones (to bind tasks to specific resources). [20] introduce the EDAM ontology for bioinformatic operations, types of data, identifiers, topics and formats. EDAM supports semantic annotations for Web services, databases, interactive tools and libraries. Offered annotations have been integrated in a set of frameworks such as EMBOSS [5] and eSysbio [6]. However, these systems do not exploit the entire richness of EDAM semantics. For example, eSysbio uses "EDAM Data and Format" to decide how to handle data by an adequate visualization and search. However this navigation, as static as it is[7], does not take benefits from the relation semantics between concepts. In the same scope, we find the Software Ontology (SWO) [21] offering an extended version of EDAM through new concepts and relations for information processing tasks (phylogenetic construction, protein structure analysis, *etc.*), algorithms (ANOVA, t-test, *etc.*), software and software versions. The ontology has been used so far in only service discovery frameworks like BioMedBridges[8] and the Gene Expression Atlas repository[9]. Yet, EDAM and SWO represent only very generic concepts/relations that still need to be specified for very specific technical domains such as phylogenetics or genome rearrangements. Moreover, GBrowse [22] and then, Seahawk [23] are two end-user oriented with ontology-based data centric approaches. Users can either specify input and/or output data and the application determines the corrected data type and proposes what to do with it instead of proposing a specific service. [24] use the BioMoby [25] services registry and construct the workflow at each step by ranking the services according to the similarity of their data type inputs and other non-functional measurements like the number of retrievals of a service in the definition metadata. Finally, we find the SADI project [26] aiming to maximize the interoperability between Web services unlike BioMoby which may contain inconsistent data [27]. SADI indexes both data and properties and offers a search engine which is designed to mimic the linguistics of

[4]OWL-S is an ontology built on top of Web Ontology Language (OWL) for describing Semantic Web Services. It enables users and software agents to automatically discover, invoke, compose, and monitor Web resources offering services.

[5]www.ebi.ac.uk/Tools/emboss

[6]www.esysbio.org

[7]The current version of eSysbio uses a limited subset of EDAM for static navigation (allowing grouping and filtering of data by their type, and sorting by type and format), without taking into account the relations other than the closure of *is format of*.

[8]www.biomedbridges.eu

[9]www.ebi.ac.uk/gxa/home

scientific queries through very specific ontological relationships. Navigating and searching for the "best" tool in categories of tasks are not enough to guide users to complete their processes. In addition, formulating queries could be a very difficult task if users do not know what is the next step and the best method/tool and configurations for that step.

Similarity-based recommendation. Similarity-based approaches aim to tackle two important problems of workflow reuse: (1) the retrieval of similar workflows form potentially large repositories and (2) the adaptation of workflows in new case studies. Here, the Process-Oriented Case-Based reasoning [28] comes to support and partially automates experience-based problem solving by creating new workflows and adapt them for new purposes and circumstances. [29] propose a graph-based approach with generalization of workflows over the concept taxonomy of a domain ontology. The similarity between a case workflow and a query one is calculated using a local similarity assessing the similarity between two nodes/edges of the same type which is, then, aggregated into a global similarity within a graph mapping process. Their similarity measures are based on an object-oriented approach [30] calculating intra and inter class similarities between abstract and concrete objects. Their approach is used later on with [31] where during adaptation, deficiencies in the retrieved case are incrementally compensated by replacing fragments of the retrieved case by appropriate workflow streams. Their work is limited to strongly structured texts like cook recipes and software manuals where sentences explicitly describe series of actions.

Pattern-based recommendation. Most of bioinformatic pattern-based recommenders use sequential pattern mining algorithms in order to discover recurring software and data usage patterns [32]. For example, [33] propose to use Natural Language Processing (NLP) tools to extract software and database terms from scientific texts. The process uses the bioNerDS terminology [10] in order to categorize terms in texts. In the pattern mining step, a straightforward mining algorithm discovers frequently occurring resource pairs. Each resource is paired with the one that immediately follows it in a text while accounting for the direction within the ensuing pairs. However, the order of the terms in the text is not always reliable. Without a context-aware extraction approach, the occurrence of such a pair could mean nothing but simple mentions. [34] propose Magallenes, a Web service discovery and automatic workflow composition tool which associates data types to connect available and compatible Web services into workflows that can process data sequentially to reach desired output given a particular input. A breadth-first with pruning algorithm is used to find the shortest path from source to target service. In the same scope [14] use the ServiceMap algorithm [35] to model/evaluate workflow "reputations" based on the number of downloads, views, credits and friends taking into account reputation time fading and popularity. They propose a service-based recommendation platform based on the paths from input to output data using the classic Apriori approach to find frequent associations between services [36] and then, find a path between two services/operations in the reputation network. In a different vein, methods for mining patterns and associations on top of an ontology have been proposed in [37]. As an example, the GSP (Generalized Sequential Patterns) algorithm [38] mines frequent patterns from sequences of user transactions. It traverses the pattern space with an Apriori-like level-wise discipline [36] and by exploring the anti-monotony property of itemset frequencies over the taxonomy. Besides, using an ontology to enrich patterns of Web usages has been researched on for at least a decade [39, 40]. However, these latter only process the class hierarchy. [41] present a complete framework for Web usage mining from the logs of a Web page clicks content where its (page resources) objects are described by an ontology. The input data are sequences of clicks further translated into IDs of content objects with their semantic links. Patterns are also sequences, yet, made of concept IDs linked with ontological properties. The proposed `xPMiner` method defines a total order between sequence items and applies a level-wise mining through the pattern space with a refinement operator [42] made of four primitives that follow the hierarchies from the ontology: addition/specialization of concepts/relations.

Inspired by the model of [41] and beside that input data is automatically extracted from texts, we offer a new pattern miner, where, instead of sequences, we deal with workflows as posets[11] (where transactions representing steps) of objets linked with each other, thus, forming (double)-layered DAGs (Direct Acyclic Graphs). Here, the DAG nodes represent task, data and metadata items and edges represent the control and data flows. In addition, on top of the workflow DAGs, an ontology provides two taxonomies, one for concepts and one for relations (*a.k.a.* generalized items). Hence, patterns are made of concepts with inter-concept relations which may belong to different generality levels in the ontology. Interested readers may refer to our publication [43] which represents

[10] www.bionerds.sourceforge.net

[11] Partially ordered sets which formalize and generalize the intuitive concept of an ordering, sequencing, or arrangement of the elements of a set.

the framework of our project and present preliminary results of the extracted concepts and relations from texts. Yet, we present in this work the full description of our approach of workflow extraction and pattern mining and offer a domain validation of the extracted patterns in the phylogenetics experience acquisition domain.

3 Methods

The proposed solution consists of reconstructing a set of workflows from the scientific literature and then mining higher-level of abstractions (generalized workflow patterns) from these (see Fig 2). A specific domain ontology is used to extract concrete workflows from texts. The included concepts and properties are further used as building blocks in the pattern generation process. Thus, the ontology is exploited for two purposes: (1) as a schema to create a terminology annotating workflows in texts and (2) as a knowledge base to mine generalized patterns from the extracted workflows.

3.1 Formalisms for enriched workflow representations

3.1.1 Workflow representation

In general, a workflow consists of a set of tasks (also called activities) combined in a *control-flow* and a *data-flow* whereby the data-flow describes interactions (relationships) between tasks and data items (task inputs/outputs corresponding to data types). Given a universe of task items and data O and links $\mathcal{L}$, a workflow record is a vertex- and edge-labeled DAG. However, by following the steps of the overall analytical process –and performing a topological sort, if needed– it can be split into layers of independent vertices (no intra-layer edges). This allows a workflow representation as: (1) a sequence of object sets (item sets), plus (2) a set of cross-itemset links.

A data language Δ_W defines workflows as pairs $w = (\zeta, \Theta)$ where $\zeta = (\{o_1, o_2, \ldots o_{i_1}\}, \{o_{i_1+1}, \ldots, o_{i_2}\}, \ldots, \{o_{i_{(m-1)+1}}, \ldots, o_{i_m}\})$ is the sequence of object *transactions* (steps). Objects, here, represent tasks and data items. In turn, $\Theta = \{\theta_1, \theta_2, \ldots, \theta_p\}$ is the set of triples where $\theta_k = (i_1, l, i_2)$ represents a link (in the control/data flows) of type l between the object at position i_1 in $w.\zeta$, *i.e.* $w.\zeta[i_1]$ and at $w.\zeta[i_2]$ where $i_1 < i_2$. Fig.3 shows an example of workflow representation in Δ_W.

3.1.2 Ontology representation

Workflow components are encoded in a specific domain ontology Ω. Formally, an ontology is defined as a six-tuple $\Omega = \langle O, \mathcal{L}, C, \mathcal{R}, \leq_\Omega, \rho \rangle$ where O is the set of all object items, *i.e.* individual tasks and data types, $\mathcal{L}$ the set of all links between objects, C the set of all concepts, and $\mathcal{R}$ the set of all relations between concepts. Moreover, concepts and relations are organized in taxonomies *w.r.t.* the generality order (*subClassOf* hierarchical relationship in RDF) of the ontology $\leq_\Omega$, *i.e.*: $H_C = \langle C, \leq_\Omega \rangle$ and $H_\mathcal{R} = \langle \mathcal{R}, \leq_\Omega \rangle$. $\rho \subseteq C \times \mathcal{R} \times C$ is a ternary relation whose triples $c_1 \times r \times c_2$ expressing the connection between a relation and its domain (c_1) and range (c_2) concepts. Objects $o \in O$ and links $l \in \mathcal{L}$ are instances of concepts $c \in C$ and relations $r \in \mathcal{R}$, respectively. *Fig.* 4 shows an example of a phylogenetic processual ontology representing workflows abstract components.

3.2 Workflow extraction from texts

Below, the sub-tasks of our workflow extraction solution are described. Following a standard Natural Language Processing (NLP) pipeline, we are going from morphological analysis of the text to the identification of grammatical classes and recognition of domain concept occurrences. The remainder addresses the known semantic ambiguity issues in a semi-automated way. Our approach, as an ontology-based annotation, is about learning how to recognize and classify workflow components (data and control flows) in texts using the ontological categories.

3.2.1 Morphosyntactic annotation

The initial step divides the text into lexical items called tokens. In fact, the tokenization operates words with white spaces between them. In our context, due to the specificity of the grammar used in bioinformatics texts, a specialized biomedical tokenizer is used relying on a dedicated pattern for bioinformatics tokens [44]. The text is also segmented into sentences by recognizing a set of known punctuation patterns (given as *regex*). The extracted tokens are further tagged with their appropriate grammatical classes, *a.k.a.* part-of-speech (POS) taggings. To this end, we use the MedPost POS tagger [45] which is based on a HMM (Hidden Markov Model) model trained over several biomedical MEDLINE texts. This tagger was able to achieve high accuracy (97.43% on 26 566 tokens) by using the contextual information in the HMM to resolve grammatical ambiguities.

3.2.2 Semantic annotation

A specific OWL ontology gazetteer (terminology) is used in order to recognize bioinformatics concept and property instances in the text. This gazetteer was generated from widely-used databases: Gene Ontology [46], UniprotKB[12], NCBI Genbank[13] and the Joe

[12] http://www.uniprot.org

[13] http://www.ncbi.nlm.nih.gov

Felsenstein Web site [14]. Gene Ontology, NBCI Genbank and UniProtKB offer structured and up-to-date databases widely popular in the context of bioinformatics. We extracted scientific terms and their synonyms (common names, short/long forms, *etc.*) using ETL (Extract, Transform and Load) tools [47]. Felsenstein's Web site offers also a very interesting categorization of 392 phylogeny packages and 54 free Web servers where each package and Web server lists a set of services and programs. Entries in this site are frequently (~6 months) updated. We crawled HTML contents from this Web site using simple XPATH extraction rules [48]. Generated terms from the different databases and HTML pages are first stored in a relational database as a common storage before transforming them into an ontology using the schema provided by an expert (see *Fig.*4) and the SQL-SPARQL mapping tool ONTOP [49]. Also, we used the Sesame ontology repository [15] as a knowledge base manager. The resulted terminology is used to recognize term tokens and N-grams in texts and support the reconstruction of workflow sequences. This process faces known hurdles rooted in the semantic ambiguities that we address in the following paragraph.

3.2.3 Disambiguation rules

For this purpose, we choose to create manually specific pattern rules to concepts grounding special semantic ambiguities *i.e.* polysemy [50]. The name of proteins, genes and species are metadata on data sequences in the workflow, hence, we are not specifically interesting in disambiguating them but only in software terms. For example, the term 'MEGA' is a polysemic term in our context: it could be either the common term (huge) or the name of a general purpose program. 'MEGA' could be used in sequence alignment, phylogenetic tree inference or/and in other phylogenetic tasks. To solve this type of ambiguities, we first created a set of disambiguation rules in JAPE [16]. These rules enabled the construction of a large set of terms already annotated (classified) to be further integrated in a machine learner (see next section). For example, to verify if 'MEGA' was used in the inference step, we search for the verb 'infer' in the context of 'MEGA'. By context here we mean the sentence in which the term appears (see Fig. 5). However, a sentence could carry a lot of information and has different forms (active or passive). Thus, we use a syntactic analyzer to construct a 2-levels parsing tree (chunk) [52] to detect the appropriate context of the term 'MEGA' in order to fire JAPE rules. For example the rule M searches if the term 'MEGA' was used to infer phylogenetic trees. *Rule M (sentence)* : { **if** the verb's root in 'MEGA' chunk is 'infer' **or** there is a phylogenetic approach in the object of the chunk **then** create 'PhylogeneticInferenceProgram' Annotation }. The chunk of 'MEGA' is the verb phrase in the sentence where 'MEGA' is its Object/Subject (depending on the active or the passive voice of the sentence). The sentence in *Fig.* 5 shows that 'MEGA' is used a phylogenetic inference program since it is preceded by the approach 'MaximumParsimony' in its context. For each polysemic 'Step Concept' (see Fig. 4), we create n disambiguation rules (where n is number of 'concept steps' in the ontology) in order to clarify the ambiguities of program classes and generate a training set to the final disambiguation model. For example, in the phylogenetic framework, 7 rules represent the 7 seven steps where program classes represent the most polysemic terms.

3.2.4 Learning features

A supervised learning approach is applied to ambiguity resolution. To this end, we created a set of features to support proper classification of bioinformatics concepts and properties in texts. For example in Fig.5. in order to classify the term 'MEGA' ($ClassC$), we generate the set of features ($featuresC$) reflecting its context. The context covers a window made of: the $win_i = 3$ concepts preceding and the win_i ones following ($Token.class$) the concerning word (*e.g.* 'MEGA') in its context sentence. It also comprises the sequence of POS tags of the selected tokens ($Token.category$). In this example $Token.class$ = { $DataType$, $MaximumParsimonyApproach$ } and $Token.category$ = { $Determiner$, $Noun$, $preposition$ }.

In addition, in order to classify a relation between the software 'MEGA' (domain) and the data 'alignments' (range) - $classR$, we define its context as the set of tokens between the domain and the range. Now, features ($featuresR$) represent token POS tags ($TokensBetween.category$), distance between domain and range, relation direction and the list of verbs between the concerned terms. The overall set of features is then used in the construction of a statistical model for automated annotation of the texts. Relation types between terms are already defined in the schema of the ontology (see Fig. 4). Tuples of concepts are then queried in order to recognize properties in text

[14]www.evolution.genetics.washington.edu

[15]www.rdf4j.org

[16]Java Annotation Patterns Engine (JAPE). JAPE language is a finite state transducer on GATE platform [51] that operates over annotations based on regular expressions. Thus, it is useful for pattern-matching, semantic extraction, and many other operations over syntactic trees such as those produced by natural language parsers.

and calculate its features. For example, the ontological property 'has_used_by_program' between datatypes and phylogenetic inference programs is used to calculate the set of features $featuresC$ representing the relation's context in the sentence.

3.2.5 Model creation and application

In this study, we choose PAUM (Perceptron Algorithm with Uneven Margins) Gate implementation [51, 53] of as a classifier over others *i.e.*: CRF (Conditional Random Fields), SVM (Support Vector machines) and Random Forest [54]. We compared the PAUM model with the mentioned algorithms in terms of F-Measure and speed, but we only present here the best classifier results. The PAUM algorithm was used to learn term and relation classifications from the extracted features. PAUM was designed especially for imbalanced data and has been successfully applied to various named entity recognition problems: [55, 56]. PAUM differs from other classifiers as it approximates the best distance between positive and negative sets. As our approach is based on concept contexts to disambiguate polysemic words, the distance between surrounded concept examples (negatives) and the concerned word is important, same for domain and range relations. PAUM uses a specific margin function to adjust weights of positive and negative example features and thus shows competitive results. This function updates weights w for each example x_i based on a fixed bias parameters τ_+ and τ_- representing the number of positive and negative examples, respectively. After creating and tuning our models to annotate the texts automatically, we applied a 10-fold cross-validation analysis [57]. Extracted terms and links are then assessed using classic evaluation metrics *i.e.*: precision, recall and F-Measure.

3.2.6 Workflow reconstruction

From recognized links between concept instances in text, we build our concrete workflow knowledge database where items are the named entities ordered by the top-level pattern in the domain ontology. Concept and property instances serve to identify respectively the data-flow and control-flow of workflows. For example, from the tuples: (ρ_1) (alignment, "has_dataSource", hemagglutinin), (ρ_2) (alignment, "used_in_SequenceAlignmentProgram", ClustalW) and (ρ_3) (alignment, "used_in_PhylogeneticInfence Program", MEGA), we construct the workflow sequence $S1$: ⟨ hemagglutinin, alignment, ClustalW, MEGA ⟩. We flatten then the workflow structure into a sequence of data and programs enriched with semantic relations (tuples) in order to represent the data and control flows.

3.2.7 Workflow quality criterion

We propose here a semantic-based approach on workflow and knowledge representation as textual descriptors, set of tags or abstract features In order to evaluate the extracted concrete workflows, we calculate the similarity measure $s_w(a,p) \in \{0..1\}$ which represents the similarity between the actual a and the predicted p workflow components. Unlike the work of [58] and [29] proposing computationally expensive workflow mapping algorithms (based on graph-edit distance or A* search), we propose a simple bag-of-words similarity measure which takes into account both objects (data, metadata and tasks) and links (data-flow and control-flow) and their ontological mappings since transactions (steps) are ordered using abstract patterns from the domain ontology (see previous subsection). Formally, given $a,p \in \Delta_W$ and $w = (\zeta, \Theta)$, where ζ is the sequence of workflow items i at each step j, Θ is the set of triples $\theta_k = (i_1, r, i_2)$ connecting relations r to the object domain at position i_1 and the object range at i_2, then the similarity s_w between a and p is given by the following formulas:

$$s_w(a,p) = \frac{s_w^{\zeta} + s_w^{\theta}}{2} \tag{1}$$

$$s_w^{\zeta}(a,p) = \frac{\sum_j^m \frac{\sum_i^n (w^{\zeta}(a.o_{i,j},\, p.\zeta))}{n}}{m} \tag{2}$$

$$s_w^{\Theta}(a,p) = \frac{\sum_i^p (w^{\Theta}(a.\theta_k,\, p.\Theta))}{p} \tag{3}$$

where m, n and p are, respectively, the total number of steps, items per step and relations in the actual workflow. Each object $o_{i,j}$ is weighted by the accuracy measure w^{ζ} and each property θ_k is weighted by the accuracy $w^{\Theta} = w^{\Theta_r} \times w^{\Theta_d}$ where the weight $w^{\Theta_r} \in \{0..1\}$ is the degree of a concept term accuracy and $w^{\Theta_d} \in \{0..1\}$ is the degree of a relation term accuracy. The similarity ratio of a workflow is then the mean of its components accuracies. Table 1 shows the weights that we propose to measure the degrees of accuracies of workflow components. If a term (concept or relation) in the actual workflow a is well annotated, then, its weight w is 1. Otherwise, if a term is partially correct, we assign a weight of 0.6. In the case where a relation type is partially correct (where the relation type term is not correct and the domain and range are correct), a much lesser weight is given (0.1). Finally, if a term/step is not correct or absent, the

weight is 0. These weights are inspired from the work of [57] where lenient precisions and recalls are calculated given a partial overlap measure.

3.3 Generalized pattern mining

Next, we generate more abstract workflows from the set of concrete workflows extracted from the texts, i.e. the workflow database $\mathcal{D}$.

3.3.1 Candidate generation

Two descriptive languages are defined to represent both concrete workflows and abstract workflows. A data language Δ_W is derived from workflow sequences W translated into IDs from the proposed ontology Ω. The second language Γ_W is used to represent patterns p in a canonic language [41]. Γ_W comprises generalized DAG patterns $p = (\zeta, \Theta)$ where $p.\zeta$ is the sequence of concepts and $p.\Theta$ is the set of relation triples.

Our Apriori-like [36] level-wise mining approach considers the global pattern space as being made of levels. A level is defined *w.r.t* the overall depth of the pattern components within the ontology. Pattern candidates of a level $k+1$ are generated from the k^{th}-level frequent pattern by a refinement operator [59]. Here we extend the work of Adda et al. [41] by applying five different canonic operations to refine a pattern p: 1) append a root concept to a new transaction (step) in $p.\zeta$. 2) add a root concept to $p.\zeta$, 3) replace a concept from $p.\zeta$ by a direct instantiation (specialization) thereof, 4) add a root property between two concepts in $p.\theta$ and 5) replace a property from $p.\theta$ by a direct instantiation. For example (see Fig. 6), a possible specialization of the concept 24-'*MultipleSequenceAlignmentProgram*' in a pattern p, would be to replace it by one of its immediate successors in the ontology, say 25-'*Progressive*'.

The starting point is the computation of frequent level 1 patterns directly from the ontology (made of ontological root concept singletons). For each of the subsequent levels $k+1$, the level candidates are generated from frequent patterns at level k by applying canonic operations to them. The resulting candidates are first tested for the known infrequent sub-patterns and then the surviving candidates are sent to the database $\mathcal{D}$ for a direct evaluation of their interestingness.

3.3.2 Pattern quality criterion

In order to characterize the "interestingness" of the generated patterns, we propose to use the depth of pattern elements in the ontology to represent their abstraction level. We remind that the abstraction level of a pattern defines how pattern elements are specific or abstract in the domain ontology. Yet, the ultimate interestingness of a pattern depends on the decision-maker (subjective criteria) and does not solely depend on the statistical strength of the pattern (objective criteria, *e.g.* support or concept depth). Albeit useful, objective measures are not enough to determine the interestingness of a pattern, particularly, in complex structured domains like phylogenetics. Our intended solution is to let users choose their levels of interest in the ontology *min_level*[17]. Then, ranking could be used to avoid checking all of generated patterns [60].

In order to motivate our ranking method, we propose to use the patterns p_1: ($\langle$ {*Protein*}, {*Bayesian*}), {(1, *isDataInAlignment*, 2) }) and p_2: ($\langle$ {*Protein*, *MuscleContraction*}, {*MSA*}, {*ML*}). {(1, isDataInAlignment, 2), (1, isDataInInference, 3)}) and the ontology from the *Fig.4*.

Depth ranking score ($rank_{depth}$). In order to rank the pattern abstraction levels based on the distance from the minimum of depth suggested by the expert *min_level*, we use, here, the nonparametric two-sample Wilcoxon Rank-Sum Test [61] where pattern depth lists are compared two-by-two. Suppose that we have a set of concept depth lists $depth(p_i) = \{maxDepth(c_j \in p_i.\zeta, H_{\mathcal{C}})\}$ and we wish to test the hypothesis that the distribution of concept depths F_1 of a pattern p_1 is the same as that (F_1) in p_2 and the means μ of both of the distributions are greater or equals to the *min_level* given by the expert. Hence, our *null* hypothesis is $H_0 : F_1 = F_2 \Rightarrow \mu_1 = \mu_2, \mu_1 \geq$ *min_level*. The two sided-alternative hypothesis would be, $H_1 : F_1 \neq F_2 \Rightarrow \mu_1 \neq \mu_2, (\mu_1, \mu_2) <$ *min_level*. We remind that the Wilcoxon rank-sum test is based upon ranking the $|depth(p_1)|+|depth(p_2)|$ observations of the combined sample. Each depth is assigned to a rank: *e.g.*, the smallest has rank 1, the 2^{nd} smallest rank has 2, and so. The rank-sum test statistic is, hence, the sum of the ranks for depth observations from one of the patterns. Next, the test finds all the possible permutations of the ranks and calculates their *p_values*. If there are depth ties, where concepts from different patterns have the same ranks, we assign, to each observation in a tie, its median rank[18].

Next, we calculate the quantity of information in a pattern in terms of the number of elements covered from the domain ontology. Patterns representing the maximum amount of phylogenetic informations are preferred, hence, we use the coverage measure as a ranking feature to sort our patterns.

[17]The expert proposes the minimum of abstraction level that interests him. This level is normalized by the maximum depth in the ontology (in both of concepts and relations). So a score of 1 means that the expert demands only the most specific patterns.

[18]We choose to use the median instead of averages since $\mu_1 - \mu_2$ is strongly influenced by some of the "*very specific*" or "*very abstract*" concepts in a pattern.

Coverage of steps (CC_S). Let $n_S(\Omega)$ be the number of all '*root concept steps*' in the ontology. The fraction of steps that a pattern covers is: $|p.\zeta|/n_S(\Omega)$. For instance, if the ontology defines 7 steps, as in the phylogenetic domain (see *Fig.* 7) and the pattern presents only 6 transactions, then, its step coverage is ~85.7%.

Coverage of data (CC_D). Let $n_D(p,\Omega)$ be the number of all data concepts[19] that could be plugged into pattern program concepts from the ontology. The data coverage of a pattern is, then, the number of data used in a pattern divided by $n_D(\Omega)$. For instance, given the pattern p_2, if the number of relations (incoming and outgoing) that could be connected to both of *MSA* and *ML* in the ontology is 4. then, $CC_D(p_2,\Omega) = 2/4 = 50\%$.

Coverage of metadata (CC_M). Let $n_M(p.\Omega)$ be the number of all metadata concepts that could be connected to pattern data concepts from the ontology. The metadata coverage of a pattern is. then, the number of its metadata divided by he number of all metadata in Ω that are connected to pattern data. For example. if the number of metadata connected to *Protein* in the ontology is 4, then. $CC_M(p_2,\Omega) = 1/4 = 25\%$.

Coverage of programs (CC_P). Let $n_P(p,\Omega)$ be the number of all program concepts that could be used in the pattern steps from the ontology. The pattern program coverage is, then, the number of program concepts in $p.\zeta$ divided by n_P. For instance, if the number of program concepts in the second and the third step of the ontology is 10, then, $n_P(p_2.\Omega) = 2/10 = 20\%$.

Coverage of relations (CC_R). Let $|\lambda_p(t)|$ be the number of elements at the t^{th} transaction, then, $n_R(p) = \sum_t^{|p.\zeta|}(|\lambda_p(t)| \times \sum_{u\neq t}^{|p.\zeta|-1}|\lambda_p(u)|)$ is the number of all possible combinations between pattern elements in $p.\zeta$. The pattern relation coverage is, hence, the number of triples in $p.\Theta$ divided by $n_R(p)$, *e.g.* $n_R(p_2) = 2/10 = 20\%$.

Finally, patterns are ranked using the weighted ranking function $rank_{pattern}$ (see 4) where α and β are the weights given, for the coverage score and the abstraction level score, respectively, and $avgC$ is the average function calculating the mean of coverage scores of a pattern p.

$$rank_{pattern} = \alpha.rank_{coverage} + \beta.rank_{depth} \quad (4)$$

[19]Data from the lowest abstraction level.

where:

$$rank_{coverage} = avgC(CC_S, CC_D, CC_M, CC_P, CC_R). \quad (5)$$

We remind that. we are ranking patterns given their abstraction level towards the expert abstraction threshold and the information quantity of a pattern in terms of the number of covered elements. Patterns with high values $avgC$ (containing more elements from the ontology) and $rank_{depth}$ are assigned with the highest ranks $rank_{pattern}$.

4 Results

4.1 Dataset

The experiments are derived from a sample of +5 000 recent articles, journals and conference papers collected from the digital repository PubMed Central (PMC)[20] in the XHTML format using the following query:

> *"open access"[filter] AND "2017/01/01"[PubDate] : "2017/03/10"[PubDate] AND ("Phylogenetic analysis"[Section Title] OR "Phylogenetic analyses"[Section Title]).*

However, phylogenetic analyses sections may not have sufficient information to reconstruct the full workflows from texts. Thus, we filtered the downloaded articles based on the smallest pattern that any phylogenetic analysis should present : ($\{DataType\}$, $\{PhylogeneticInferenceProgram\}$). A phylogenetic study might use at least one data type and one phylogenetic inference program (see Fig. 7) to produce a phylogenetic tree. Finally, we retained a set of 2 969 articles.

Our public PHAGE ontology on BioPortal [21] is composed of 37 different object properties (relations), 47 031 concepts (with 4 root concepts and 111 unique concepts[22]) and more than 1 000 000 unique individuals (instances) without counting synonyms. Fig. 4 shows a sample from the proposed PHAGE ontology. '*Root concepts*' represent data types, their provenance (data sources), steps and tool parameters. '*Step concepts*' represent the seven steps of phylogenetic analyses and *relations* between concepts represent data flows and control flows. '*Program concepts*' represent the software concepts (*e.g. Local Alignment. Distance based approaches*). '*Parameter concepts*' represent the different types method/program parameters (*e.g.* number

[20]https://www.ncbi.nlm.nih.gov/pmc/

[21]https://bioportal.bioontology.org/ontologies/PHAGE

[22]Without counting synonym concepts.

of bootstraps, DNA/protein evolutionary models) and '*Data type concepts*' represent the different data types (*DNA*, *RNA*, *distance matrices*). Finally, '*Data source concepts*' represent provenance metadata (*e.g.* data formats, databases, GO annotations, types of organisms). We remind that our ontology has been collected in a semi-automated way from well-known databases in the phylogenetics domain. From the NCBI Taxonomy browser [62], we extracted +1 000 000 (all) scientific terms and synonyms of organisms and organized them into 12 ranks (plants, bacteria, mammals, *etc.*). From the Gene Ontology [46], we extracted recent (2016/04/12) GO concepts and relations. This ontology presents 112 unique concepts and 9 object relations. More than 116 000 (all) reviewed unique gene and protein terms were extracted from the UniProtKB database [63]. Each term was then inserted as an instance for the related GO concept. Finally, the Felsenstein's Web site [64] is crawled using simple data transformation scripts in XSLT leading to create more than 500 software terms categorized into 32 categories.

4.2 Extracted workflows from texts

Using the GATE platform[23], a human expert has corrected (created, deleted, updated) manually annotations from 500 texts to build a workflow ground-truth database. This latter serves as a test set for the evaluation step. The expert annotates the given texts using our domain ontology schema on the platform GATE. He annotated only terms of concepts and relations that serve to reconstruct the phylogenetic workflows from texts. Although this task is costly in terms of time/energy (~ 80 hours), the expert validation step is important to guarantee a credible perspective of our approach.

Next, we measure whether a term is annotated correctly, partially or missing. We show in Table 2 details about the total number of discovered annotations. We calculate the classic Information Retrieval F-Measure [57] based on correct, partially correct and spurious annotations. In total, our system has annotated 2 561 concept terms while the expert has annotated 2 641. The overall strict F-Measure (overlaps discarded) is 70% and the overall lenient F-Measure (overlaps included) is 76%. We notice that 118 annotations (~20%) represent ambiguous program terms (GuessedProgram). In this case, the average F-Measure (between strict and lenient) is 84% with a precision of 73% and a recall of 98%.

[23]GATE (www.gate.ac.uk) is a well-known environment for annotation tasks using widely known NLP techniques such as the OpenNLP parser[24] and the Stanford POS tagger[65].

The manual annotation of ambiguous programs highlights all the textual mentions referring to a multiple purposes tool as indirect references. In some cases, correct results from our workflow tagger were not fully annotated, leading to false positives. The following sentences:

*A phylogenetic tree was constructed with **MEGA** 5 using the aligned sequences. The phylogenetic tree was drawn using the neighbor-joining method with p-distance and 1,000 bootstrap replications.*

show an example of such case where our tagger correctly detects *MEGA* as a phylogenetic tree inference in the first sentence but in the following sentence, *MEGA*, as a neighbor-joining method, is also used to validate trees using 1 000 bootstrap replications. This class of MEGA (*HypothesisValidation*) was not detected by our context-based approach. The reference in the second sentence is implicit (hidden in the *neighbor-joining* concept), hence, it is difficult to detect that trees generated by *MEGA* are also validated using the bootstrap method. In addition, both of the verbs in the two sentences are very generic and misleading (*aligned* and *drawn*) for an automatic tagging. *MEGA*, in the previous sentences, is not used to align or visualize sequences or trees. Consequently, the precision obtained by our pipeline for tree validation tasks are very low (~40%). In fact, 30% of the hypothesis validation programs are ambiguous programs (multiple purpose packages) Yet, the context-based pipeline supporting our system needs to be enhanced in its ability to capture more generic mentions in wider contexts (*e.g.* paragraphs).

Although strict and lenient F-Measure values are very close (~0.3%), some categories (*i.e. DataCollectionProgram*, *TreeVisualizationPrgram* and *GuessedGeneralPackagesCategories*) show the same values for strict and lenient F-Measures (see also the 'O' column). Our tagger is able to recognize (or not) the exact mentions in texts: these categories does not present non-standard program names (*e.g.* common names) or generic mentions. In addition, we note that almost the half of the presented categories show reasonably high lenient F-Measure agreement (> 80%). Yet, beside hypotheses validation software names, data collection programs, species names, protein and gene GO annotations show the worst tagging F-Measure results (with a lenient F-Measure between 60 and 67%). The recall values in data collection programs explain the weak F-Measure (for both strict and lenient agreements). Thereof, our database for this category need to be updated. In addition, since no specific JAPE rules are modelled for the metadata (GO annotations

and species), some common false negatives (*i.e.* missed resource mentions) are discovered including common names of genes (*e.g.* the *g20* gene) or species (*e.g. birds*).

Fig. 7 shows the workflow similarity distribution results based on the evaluation of the 500 expert-curated workflows. Generated workflows are 82.89% similar with the actual workflows extracted by the expert (first figure on the top). While, the best mean workflow similarity is 100%, the worst similarity is 54.68% which means that, in the worst case scenarios, our workflow extraction framework will (probably) extracts half of the "actual" workflow components. The mean of object (tasks) similarities is 78.71% (second figure in Fig.7). The best workflow task similarity is found in phylogenetic tree inference programs (100%). Generally, this task is explicitly defined in phylogenetic analysis texts. We found that phylogenetic methods are frequently co-occurred in the context of the phylogenetic inference programs which explains the high values of mean similarities of this task. However, the worst task similarity is obtained by the '*HypothesisValidationProgram*'. This is due to the context of this task which is ambiguous to be resolved in one sentence as we explained previously. Concepts and POS categories surrounding these programs in one sentence are not very conclusive to disambiguate this category of software.

The relation side of workflows represents the data and control flows (third figure in Fig.7). The mean similarity in the control flow is 89.1% and 88.92% in the data flow. While the worst similarities in the data-flow is 74.43% which represents the relations between data types and multiple sequence alignment programs, the best data-flow similarity is the relation similarity between data types and hypothesis validation programs (100%). Hence, this part of data flows is reconstructed perfectly. Verbs like "validate" and "test" are commonly and precisely used between data types and tree validation programs which explains the high similarity in this concept. The worst similarities in the control-flow are found in the links between phylogenetic inference programs and their methods with a mean of 79.84%. However, links between program parameters like the number of bootstraps and the type of evolutionary models are, generally, explicitly described in the sentence which explains also its high similarity in the control-flow with (100%).

4.3 Generalized bioinformatics patterns

In our next experiment, we apply the pattern ranking function to the generated patterns with the following parameters: $\alpha = \beta = 0.5$ and $min_level = 0$ (to generate all possible patterns) where $\sigma = 0.1$. *Fig.* 8 shows the correlation results between pattern ranks and the proposed coverage features.

The distribution of the number of steps demonstrates a good correlation with the pattern ranking function. The figure shows that our patterns cover mostly 60-70% of the total number of the phylogenetic steps which is a good coverage. These patterns contain information about most of the phylogenetic steps. In addition, generated patterns contain an average of ~25% of data (from the ontology) annotated with a maximum of 15% from all ontological metadata per pattern. The distribution of program frequencies shows that patterns use up to 20% of program categories per pattern and most of the discovered pattern concepts are 80% linked with each other. This means that generated pattern workflows are "strongly" connected digraphs.

The *Fig.* 10 shows the correlations between the pattern ranking function. the coverage and depth ranking measures. The figure shows similar correlations between the depth ranking and the coverage ranking which means that patterns covering the most (85%) of elements are very specific too (with low abstraction levels). Thereof, discovered patterns with a low support ($\sigma = 0.1$) might contain the most interesting patterns (for expert users) since they are both large (covering most of the elements from the domain ontology) and contain very specific elements.

5 Discussion

5.1 A generic framework

In order to discuss the "genericity" aspect of our approach, we need to present the abstract and domain-related techniques that we propose in (1) workflow extraction and (2) pattern mining. Although expertise is domain-related. extraction methods could use abstract heuristics to extract processual components from texts. The proposed workflow extractor classify domain concept/relation terms from texts using a word sense disambiguation method. This step is related to the domain ontology conceptualization itself. If a domain concept/relation could be represented in a clear unambiguous way, the problem become trivial. Hence, this step of disambiguation is highly depending on the processual representation of the expertise. Beside ontology schema, data provenance of tasks and data terms reveal a problem of terminology extraction itself. Yet, the popularity of ontologies as a mean to store heterogeneous data through a common vocabulary facilitate the task of data extraction and versioning[25]. In similar subdomains like functional genomics,

[25]Since domain categories are not in a very dynamic changing. the problem of versioning [66] in our case was not presented. For now, if the ontology is silently changed, the previous version is replaced by the new version without any (formal) notification.

we found the MGED ontology focusing on experiments involving microarray technology.

Another aspect touching our method genericity is the interestingness measure of generated pattern: here we use the frequency as a measure of a pattern relevancy. As shown in the previous section, our solution may produce truly relevant results, yet the traversal of the levels in the ontology structure and, to an even greater extent, the levels in the pattern space, may prove costly. Thus, if the appropriate levels in the ontology could be fixed beforehand, so that only the corresponding concepts and properties are used for pattern generation, the overall mining step will be performed much faster and the output is more likely to be of interest. The ultimate interestingness of a pattern depends on the decision-maker (subjective criteria) and does not solely depend on the statistical strength of the pattern (objective criteria, *e.g.* support) [60, 67]. Albeit useful, objective measures are not enough to determine the interestingness of a pattern particularly in complexly structured domains like phylogenetics problem solving. Our intended solution is to let users choose their levels of interest in the ontology. Then, the ranking score could be used to avoid checking too many patterns [60]. To the best of our knowledge existing solutions for semantic-based workflow mining [18, 33, 68, 69] propose domain-specific solutions while producing substantially different types of abstract models. Therefore, comparing our framework with systems like Proteus or eSysbio in terms of generated models, while conceivable, is not a trivial task. Furthermore, in order to validate the quality of our results, a human expert has already evaluated the building blocks of the generated workflows (see the above section 4).

5.2 Towards a workflow recommender system

The abstract workflows produced here could be used in a pattern-based recommender to suggest a next tool to a user who got lost after a certain number of steps with the workflow manager platform. To that end, the system has to match the partial workflow corresponding to the user's session against (the prefixes of) the abstract patterns from our pattern base. The remaining part (suffix) of a matching pattern represents a potential hint as to the way the user's session might unfold from that step on. More specifically, if we take *Example 2.* as an illustrative example, we can apply the pattern $P3$ (see section 4.3) since it matches the sequence of the concrete workflow $S2$: $(rRNA_sequences, ClustalX, MAPPS)$. $S2$ do matches with $P3$ in the data type 'Protein sequences' (same as in the source and database metadata) and in programs used: 'Global MSA Program' and 'Model Selection Program'. From $P3$ we can recommend to use a JTT model, a maximum likelihood inference approach and a bootstrap-based tree validation for the next steps. With a concept-based recommendation, the system could work on different granularity levels and hence offer more flexible recommendations. Indeed, instead of objects as concrete tools and resources from concrete workflows, we suggest a class of objects to assist users with more semantically rich recommendations. Semantic suggestions could guide to compose more complex pipelines by presenting likely compositions. Users shouldn't be worried about very specific details (software versions, technical supports, *etc.*) while constructing their workflows.

6 Conclusion

With many biologists and bioinformaticians having little experience of automating bioinformatics analyses, it is important to provide relevant recommendations in order to improve the efficiency of analyses. Concept-based pattern recommendation helps users not only to acquire contextual information but also to provide valuable suggestions. In this article, we presented the foundation of such recommendation system. We intend to extend our ontology-based mining framework in order to consider a variety of criteria, for example other method parameters (*e.g.* Gamma rates, transition ratios, *etc.*), more complex control structures (*e.g.* parallel executions, if-else statements) and resource quality in terms of time (the most recent tools/databases), author (focusing on domain's experts), journals, popularity, etc,). Updating our text corpora with the most recent articles will help to produce more relevant patterns. Moreover, the proposed generalized patterns could be used to improve even journals quality to select the most relevant articles using the abstract and concrete workflow information within texts. For instance, abstracting the proposed workflow in a submitted article is used to assist the reviewer assessment.

Declarations

List of abbreviations

DAG: Direct Acyclic Graph ; DAML: DARPA Agent Markup Language ; GATE: General Architecture for Text Engineering ; KM: Knowledge Management ; NJ: Neighbor Joining ; NLP: Natural Language Processing ; MSA: Multiple Sequence Alignment ; OIL: Ontology Interchange Language ; OWLS-S. Web Ontology Language for Services ; PAUM: Perceptron Algorithm with Uneven Margins ; POS: Part-Of-Speech ; SM: Semantic Web , WMS: Workflow Management System

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

Tools and data sources are available in:
https://github.com/bioinfoUQAM/tgowler

Competing interests
The authors declare that they have no competing interests.

Funding
This work has been supported by the NSERC Discovery Grants of Canada of P.V. and A.B.D. It has been also been supported by a PAFARC grant from the Université du Québec à Montréal.

Author's contributions
AH implemented tools of workflow extraction and mining. PV and ABD examined the related work and contributed with AH in result analyses and interpretations. All authors read and approved the final manuscript.

Acknowledgements
Not applicable

Author details
[1]Département d'informatique, Université du Québec à Montréal (Québec), PO BOX 8888 Downtown Station, H3C 3P8 Montréal (Quebec), Canada. [2]Pharmaqam Centre, Université du Québec à Montréal (Québec), PO BOX 8888 Downtown Station, H3C 3P8 Montréal (Quebec), Canada.

References

1. Spjuth, O., Bongcam-Rudloff, E., Hernández, G.C., Forer, L., Giovacchini, M., Guimera, R.V., Kallio, A., Korpelainen, E., Kańduła, M.M., Krachunov, M., Kreil, D.P., Kulev, O., Labaj, P.P., Lampa, S., Pireddu, L., Schönherr, S., Siretskiy, A., Vassilev, D.: Experiences with workflows for automating data-intensive bioinformatics. Biology Direct, 1–12 (2015)
2. Barnard, D.L.: Pegasys. Pegasys 2, 1530 (2000)
3. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M.P., Sufi, S., Goble, C.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Research 41(W1), 557 (2013). doi:10.1093/nar/gkt328
4. Blankenberg, D., Kuster, G.V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A., Taylor, J.: Galaxy: A Web-Based Genome Analysis Tool for Experimentalists. Current protocols in molecular biology / edited by Frederick M. Ausubel ... [et al.] Chapter 19 (2001). doi:10.1002/0471142727.mb1910s89
5. Lord, E., Leclercq, M., Boc, A., Diallo, A.B., Makarenkov, V.: Armadillo 1.1: An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and Simulations. PloS one 7(1), 29903 (2012)
6. Leipzig, J.: A review of bioinformatic pipeline frameworks. Briefings in bioinformatics, 020 (2016)
7. Anisimova, M., Liberles, D.a., Philippe, H., Provan, J., Pupko, T., von Haeseler, A.: State-of the art methodologies dictate new standards for phylogenetic analysis. BMC evolutionary biology 13, 161 (2013). doi:10.1186/1471-2148-13-161
8. Goble, C.A., *et al.*: myexperiment: a repository and social network for the sharing of bioinformatics workflows. Nucleic acids research 38(suppl 2), 677–682 (2010)
9. Thompson, J.D., Gibson, T., Higgins, D.G., et al.: Multiple sequence alignment using clustalw and clustalx. Current protocols in bioinformatics, 2–3 (2002)
10. Jones, D.T., Taylor, W.R., Thornton, J.M.: The rapid generation of mutation data matrices from protein sequences. Computer applications in the biosciences: CABIOS 8(3), 275–282 (1992)
11. Kumar, S., Stecher, G., Tamura, K.: Mega7: Molecular evolutionary genetics analysis version 7.0 for bigger datasets. Molecular biology and evolution, 054 (2016)
12. Yu, J., Buyya, R.: A taxonomy of scientific workflow systems for Grid computing . SIGMOD Record 34, 1–6 (2005)
13. Saltz, J., Oster, S., Hastings, S., Langella, S., Kurc, T., Sanchez, W., Kher, M., Manisundaram, A., Shanbhag, K., Covitz, P.: cagrid: design and implementation of the core architecture of the cancer biomedical informatics grid. Bioinformatics 22(15), 1910–1916 (2006)
14. Yao, L., Riedel, S., McCallum, A.: Unsupervised relation discovery with sense disambiguation. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL '12, pp. 712–720. Association for Computational Linguistics, Stroudsburg, PA, USA (2012)
15. Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y., Goble, C.: Common motifs in scientific workflows: An empirical analysis. Future Generation Computer Systems 36, 338–351 (2014)
16. Staab, S., Studer, R., Guarino, N., Oberle, D.: Handbook on Ontologies vol. 2, pp. 779–796. Springer, ??? (2010). doi:10.1007/978-3-540-92673-3. http //www.springerlink.com/index/10.1007/978-3-540-92673-3
17. Hashmi, N., Lee, S., Cummings, M.P.: Abstracting Workflows: Unifying Bioinformatics Task Conceptualization and Specification Through Semantic Web Services. In: WC Workshop on Semantic Web for Life Sciences, - October , Cambridge, Massachusetts USA, pp. 1–7 (2012)
18. Cannataro, M., Guzzo, A., Comito, C., Veltri, P.: Ontology-based Design of Bioinformatics Workflows on PROTEUS. JDIM 2(2), 87–92 (2004)
19. Lee, S., Wang, T.D., Hashmi, N., Cummings, M.P.: Bio-STEER: A Semantic Web workflow tool for Grid computing in the life sciences. Future Generation Computer Systems 23(3), 497–509 (2007)
20. Ison, J., Kalas, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., Malone, J., Lopez, R., Pettifer, S., Rice, P.: EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics 29(10), 1325–1332 (2013)
21. Malone, J., Brown, A., Lister, A.L., Ison, J., Hull, D., Parkinson, H., Stevens, R.: The software ontology (swo): a resource for reproducibility in biomedical data analysis, curation and digital preservation. Journal of biomedical semantics 5(1), 1 (2014)
22. Wilkinson, M.: Gbrowse moby: a web-based browser for biomoby services. Source Code for Biology and Medicine 1(1), 1 (2006)
23. Gordon, P.M., Sensen, C.W.: Seahawk: moving beyond html in web-based bioinformatics analysis. BMC bioinformatics 8(1), 208 (2007)
24. DiBernardo, M., Pottinger, R., Wilkinson, M.: Semi-automatic web service composition for the life sciences using the biomoby semantic web framework. Journal of biomedical informatics 41(5), 837–847 (2008)
25. Wilkinson, M.D., Links, M.: Biomoby: an open source biological web services proposal. Briefings in bioinformatics 3(4), 331–341 (2002)
26. Wilkinson, M.D., Vandervalk, B., McCarthy, L.: Sadi web-services, cause you can't always get what you want ! In: Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific, pp. 13–18 (2009). IEEE
27. Withers, D., Kawas, E., McCarthy, L., Vandervalk, B., Wilkinson, M.: Semantically-guided workflow construction in taverna: the sadi and biomoby plug-ins. In: International Symposium On Leveraging Applications of Formal Methods, Verification and Validation, pp. 301–312 (2010). Springer
28. Schumacher, P., Minor, M., Schulte-Zurhausen, E.: Extracting and enriching workflows from text. In: IEEE 14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14-16, 2013, pp. 285–292 (2013). doi:10.1109/IRI.2013.6642484. http://dx.doi.org/10.1109/IRI.2013.6642484
29. Bergmann, R., Gil, Y.: Similarity assessment and efficient retrieval of semantic workflows. Information Systems 40, 115–127 (2014)
30. Bergmann, R., Stahl, A.: Similarity measures for object-oriented case representations. In: European Workshop on Advances in Case-Based Reasoning, pp. 25–36 (1998). Springer
31. Müller, G., Bergmann, R.: Workflow streams: A means for compositional adaptation in process-oriented cbr. In: Case-Based Reasoning Research and Development, pp. 315–329. Springer, ??? (2014)
32. Aggarwal, C.C.: Recommender Systems: The Textbook, p. 498 (2016). doi:10.1007/978-3-319-29659-3. arXiv:1202.1112v1
33. Duck, G., Nenadic, G., Brass, A., Robertson, D.L., Stevens, R.: Extracting patterns of database and software usage from the bioinformatics literature. Bioinformatics 30(17), 601–608 (2014)
34. Ríos, J., Karlsson, J., Trelles, O.: Magallanes: a web services discovery and automatic workflow composition tool. BMC bioinformatics 10(1), 1 (2009)
35. Tan, W., Zhang, J., Madduri, R., Foster, I., De Roure, D.: Providing

map and gps assistance to service composition in bioinformatics. In: Services Computing (SCC), 2011 IEEE International Conference On, pp. 608–615 (2011). IEEE
36. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. VLDB '94, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994). http://dl.acm.org/citation.cfm?id=645920.672836
37. Svátek, V., Rauch, J., Ralbovský, M.: Ontology-enhanced association mining. In: Semantics, Web and Mining, pp. 163–179. Springer, ??? (2006)
38. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Proceedings 5th International Conference Extending Database Technology, EDBT, Avigon, France. LNCS, vol. 1057, pp. 3–17. Springer, ??? (1996)
39. Dai, H.K., Mobasher, B.: Using ontologies to discover domain-level web usage profiles. Semantic Web Mining. 35 (2002)
40. Senkul, P., Salin, S.: Improving pattern quality in web usage mining by using semantic information. Knowledge and Information Systems 30(3), 527–541 (2012). doi:10.1007/s10115-011-0386-4
41. Adda, M., Valtchev, P., Missaoui, R., Djeraba, C.: A framework for mining meaningful usage patterns within a semantically enhanced web portal. In: Proceedings of the Third C* Conference on Computer Science and Software Engineering, pp. 138–147 (2010). ACM
42. De Raedt, L.: Logical and relational learning. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5249 LNAI, p. 1 (2008). doi:10.1007/978-3-540-88190-2-1
43. Halioui, A., Valtchev, P., Diallo, A.B. Acquisition of generic problem solving knowledge through information extraction and pattern mining. In: Tools with Artificial Intelligence, 2015 IEEE 27th International Conference On, pp. 583–590 (2015). doi:10.1109/ICTAI.2015.90
44. Barrett, N., Weber-Jahnke, J.: Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm. BMC bioinformatics 12(Suppl 3), 1 (2011)
45. Smith, L., Rindflesch, T., Wilbur, W.J.: MedPost: a part-of-speech tagger for bioMedical text. Bioinformatics (Oxford, England) 20(14), 2320–2321 (2004). doi:10.1093/bioinformatics/bth227
46. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., *et al.*: Gene ontology: tool for the unification of biology. Nature genetics 25(1), 25–29 (2000)
47. Skoutas, D., Simitsis, A.: Ontology-based conceptual design of etl processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems (IJSWIS) 3(4), 1–24 (2007)
48. Olston, C., Najork, M.: Web crawling. Foundations and Trends in Information Retrieval 4(3), 175–246 (2010)
49. Rodriguez-Muro, M., Calvanese, D.: Quest, an owl 2 ql reasoner for ontology-based data access. OWLED 2012 (2012)
50. Bach, N., Badaskar, S.: A survey on relation extraction. Language Technologies Institute, Carnegie Mellon University (2007)
51. Cunningham, H., Tablan, V., Roberts, A., Bontcheva, K.: Getting more out of biomedical documents with gate's full lifecycle open source text analytics. PLoS Comput Biol 9(2), 1002854 (2013)
52. De Marneffe, M.-C., MacCartney, B., Manning, C.D., *et al.*: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC, vol. 6, pp. 449–454 (2006)
53. Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J.: The perceptron algorithm with uneven margins. In: ICML, vol. 2, pp. 379–386 (2002)
54. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes 30(1), 3–26 (2007)
55. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Proceedings of the Twenty-first International Conference on Machine Learning, p. 94 (2004). ACM
56. Beitzel, S.M., Jensen, E.C., Lewis, D.D., Chowdhury, A., Frieder, O.: Automatic classification of web queries using very large unlabeled query logs. ACM Transactions on Information Systems (TOIS) 25(2), 9 (2007)
57. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management 24(5), 513–523 (1988)
58. Starlinger, J., Brancotte, B., Cohen-Boulakia, S., Leser, U.: Similarity search for scientific workflows. Proceedings of the VLDB Endowment 7(12), 1143–1154 (2014)
59. Borgelt, C.: Canonical forms for frequent graph mining. In: Advances in Data Analysis, pp. 337–349. Springer, ??? (2007)
60. Collard, M., Vansnick, J.-C.: How to measure interestingness in data mining: a multiple criteria decision analysis approach. In: RCIS, pp. 395–400 (2007)
61. Wilcoxon, F., Katti, S., Wilcox, R.A.: Critical Values and Probability Levels for the Wilcoxon Rank Sum Test and the Wilcoxon Signed Rank Test. American Cyanamid Company, ??? (1963)
62. Federhen, S.: The ncbi taxonomy database. Nucleic acids research 40(D1), 136–143 (2012)
63. Kiefer, F., Arnold, K., Künzli, M., Bordoli, L., Schwede, T.: The swiss-model repository and associated resources. Nucleic acids research 37(suppl 1), 387–392 (2009)
64. Felsenstein, J., Felenstein, J.: Inferring Phylogenies vol. 2. Sinauer associates Sunderland, ??? (2004)
65. Toutanova, K., Klein, D., Manning, C.D., Singer, Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pp. 173–180 (2003). Association for Computational Linguistics
66. Klein, M., Fensel, D. Ontology versioning on the semantic web. In: Proceedings of the First International Conference on Semantic Web Working, pp. 75–91 (2001). CEUR-WS. org
67. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A survey. ACM Comput. Surv. 38(3) (2006). doi:10.1145/1132960.1132963
68. Digiampietri, L.A., Alcazar, J.d.J.P., Medeiros, C.B.: An ontology-based framework for bioinformatics workflows. International Journal of Bioinformatics Research and Applications 3(3), 1–18 (2007)
69. Soomro, K., Munir, K., McClatchey, R.. Incorporating semantics in pattern-based scientific workflow recommender systems: Improving the accuracy of recommendations. In: 2015 Science and Information Conference (SAI), pp. 565–571. IEEE, ??? (2015)
70. Liang, L., Xu, B., Chen, Y., Liu, Y., Cao, W., Fang, L., Feng, L., Goodchild, M.F., Gong, P.: Combining spatial-temporal and phylogenetic analysis approaches for improved understanding on global h5n1 transmission. PLoS One 5(10), 13575 (2010)

Figures

Tables

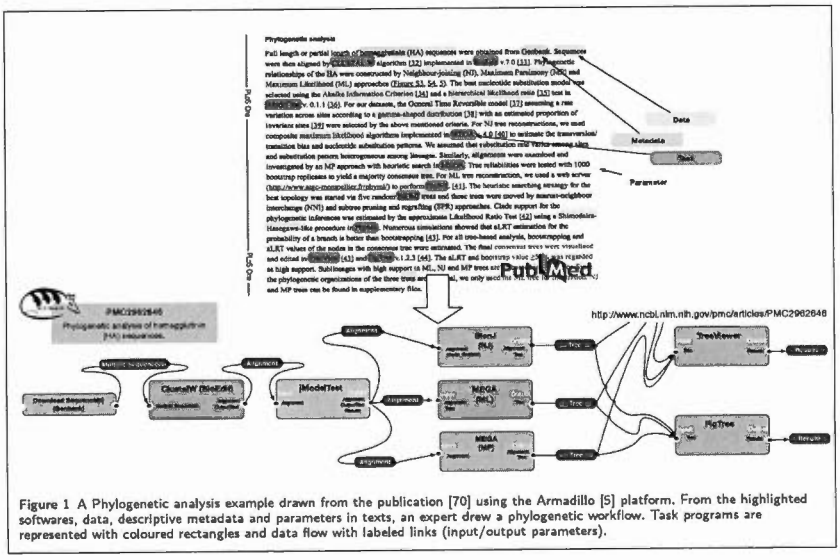


Figure 1 A Phylogenetic analysis example drawn from the publication [70] using the Armadillo [5] platform. From the highlighted softwares, data, descriptive metadata and parameters in texts, an expert drew a phylogenetic workflow. Task programs are represented with coloured rectangles and data flow with labeled links (input/output parameters).

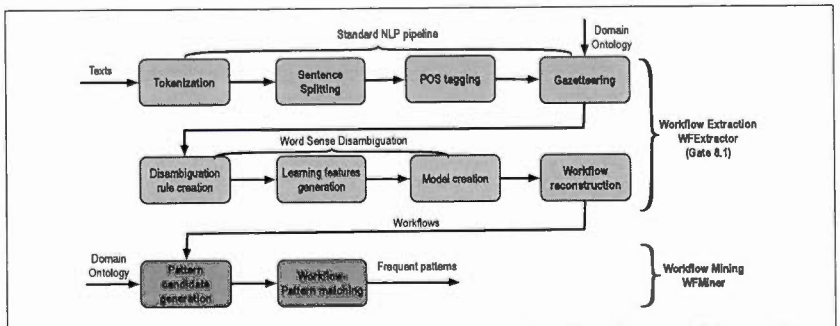


Figure 2 Overview of the pattern extraction pipeline from texts using a domain ontology. The implementation of the system is available in: https://github.com/bioinfoUQAM/tgowler. This figure shows the two main components of our system: (1) WFExtractor extracts workflows from texts using standard NLP tools and a supervised learning for word sense disambiguation and (2) WFMiner mines frequent generalized patterns out of the extracted workflows using an Apriori-like method traversing the hierarchies of workflow objects and links (from the domain ontology).

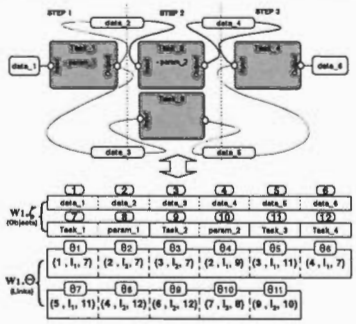


Figure 3 An example of workflow representation. Workflow items: *i.e.* tasks, data and parameters are encoded in the sequential set $w_1.\zeta$ while links between them are encoded in the triple set $w_1.\Theta$. Each triple represents a link between item positions. Here, the links l_1, l_2 and l_3 represent respectively, the $isInput$, $isOutput$ and $hasParameter$ relations.

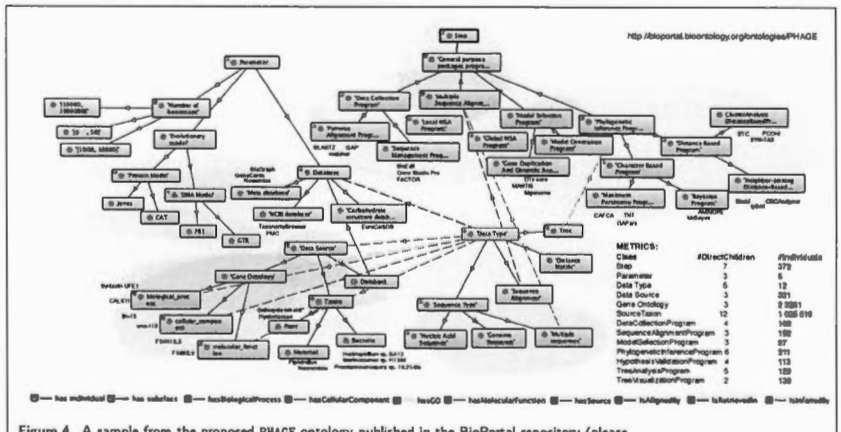


Figure 4 A sample from the proposed PHAGE ontology published in the BioPortal repository (please visit http://labo.bioinfo.uqam.ca/index.php?action=tgowler to download data). Grey rectangles represent workflow reference concepts to: steps and programs (in the blue cloud), data types (in the green cloud), sources (in the purple cloud) and parameters (in the yellow cloud). The hard arcs represent hierarchical (the inverse of *rdfs:subClassOf*) relations and the dotted arcs represent other semantic properties (domain constraints and input/output relations). Terms under concepts show some individuals (instances) for each category.

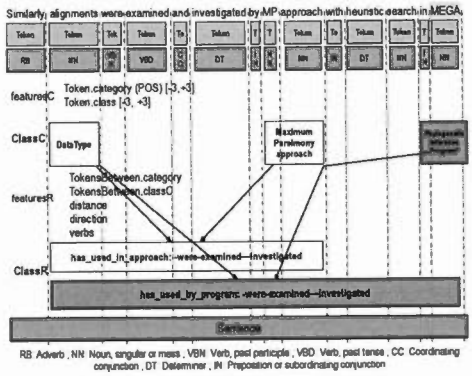


Figure 5 An example of concept and relation annotation in texts. After segmenting the text into tokens and POS tags, we construct a set of concept features *featuresC* to learn how to classify concepts *ClassC* and relation features *featuresR* to learn how to classify relations *ClassR*. Since our feature extraction method is context-based, we recognize each concept and relation in its sentence.

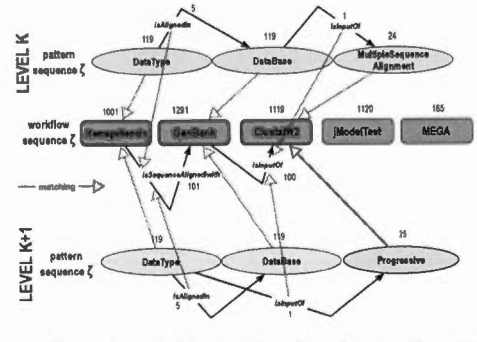


Figure 6 An example of pattern refinement from a level k to $k+1$. Let p be an abstract workflow pattern (grey ovals) and w the concrete workflow sequence (blue rectangles). After refining/replacing the concept 24: *MultipleSequenceAlignment* with its immediate ontological child concept 25: *Progressive*, we need to verify if the matching between the concepts and relations of p and w still hold from the level k to $k+1$.

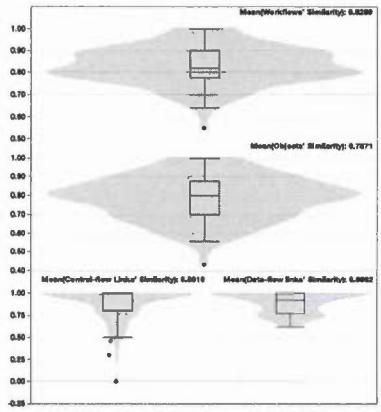


Figure 7 Workflow similarities results. The top figure represents the overall similarity between the actual a (annotated by the expert) workflows and the predicted p ones. The experiments have been performed on 100 recent articles downloaded from the PMC database. The second and third figures represent box-plots and contour distributions of concept and relation similarities between a and p. Bold points represent the interquartile ranges outliers.

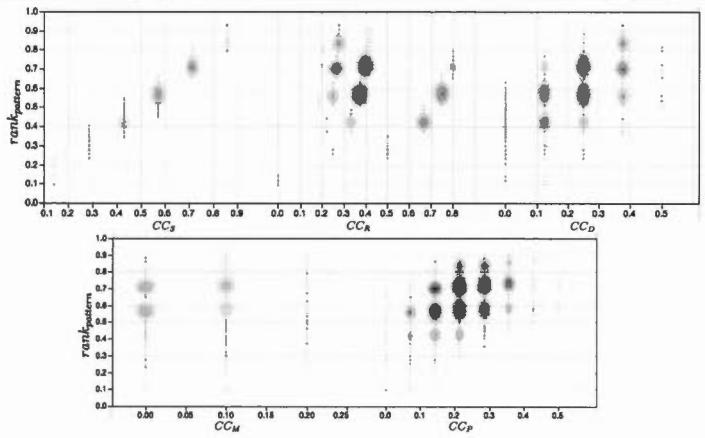


Figure 8 Pattern coverage metrics distributions Vs. pattern ranking function.

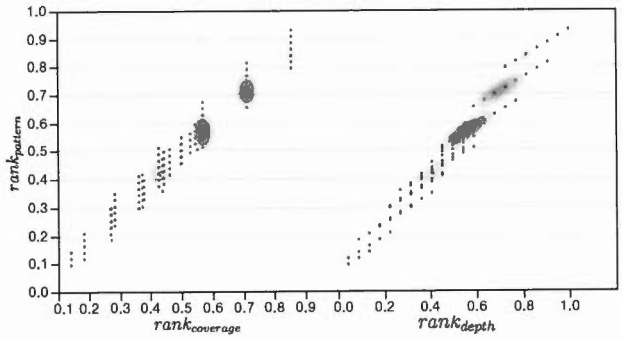


Figure 9 Pattern raking function Vs. coverage and depth rankings

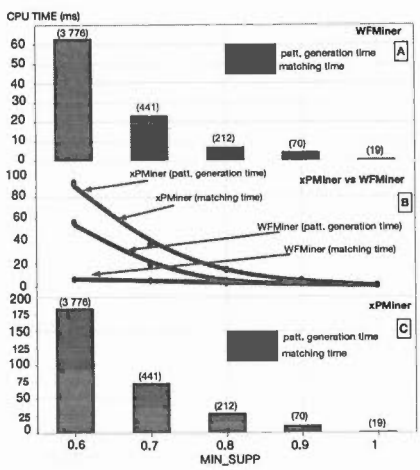


Figure 10 *WFMiner* vs *xPMiner* [41] using the expert dataset. *Fig.* A shows the matching and the overall mining CPU time of WFMiner over different values of minimum support (MIN_SUPP): 0.6, 0.7, 0.8, 0.9 and 1. Same for *Fig.* C, it shows the speed of XPMiner over the mining and matching time. *Fig.* B plots and compares the speed time of xPMiner and WFMiner. The number of generated patterns for each experiment is indicated into parentheses.

Table 1 Objects and relations weights

Weight	Description	Value
$w_1^{\zeta_t}$	The object term $o_{i,j}$ in $p.\zeta$ is correct.	1
$w_2^{\zeta_t}$	The object term $o_{i,j}$ in $p.\zeta$ is partially correct.	0.6
$w_3^{\zeta_t}$	The object term $o_{i,j}$ in $p.\zeta$ is not correct.	0
$w_1^{\Theta_t}$	The relation term of $a.\theta_k$ in $p.\Theta$ is correct.	1
$w_2^{\Theta_t}$	The relation term of $a.\theta_k$ in $p.\Theta$ is partially correct.	0.6
$w_3^{\Theta_t}$	The relation term of $a.\theta_k$ in $p.\Theta$ is not correct.	0.1
$w_1^{\Theta_s}$	The domain and range terms of $a.\theta_k$ in $p.\Theta$ are correct.	1
$w_2^{\Theta_s}$	The domain or range terms of $a.\theta_k$ in $p.\Theta$ are correct.	0.6
$w_3^{\Theta_s}$	The domain and range terms of $a.\theta_k$ in $p.\Theta$ are not correct.	0

Weights values represent strings/substrings matchings between predicted item and triple terms against the gold standard terms.

Table 2 Concepts Evaluation Distribution

	M	G	P	O	SP	SR	SFM	LP	LRs	LFM
BootstrapsParms	61	19	8	1	0.87	0.75	0.80	0.88	0.76	0.82
DataBase	43	19	13	3	0.72	0.66	0.69	0.77	0.70	0.74
DataCollectionProgram	36	29	5	0	0.87	0.55	0.67	0.87	0.55	0.67
DataType	597	103	186	89	0.68	0.75	0.71	0.78	0.86	0.82
GeneOntology	139	172	5	3	0.94	0.44	0.60	0.96	0.45	0.61
HypothesisValidationProgram	25	40	29	2	0.44	0.37	0.40	0.48	0.40	0.43
Method	326	48	46	14	0.84	0.84	0.84	0.88	0.87	0.87
ModelSelectionProgram	42	50	17	1	0.70	0.45	0.54	0.71	0.46	0.56
PhylogeneticInferenceProgram	205	16	146	10	0.56	0.88	0.69	0.59	0.93	0.72
SequenceAlignmentProgram	70	15	19	4	0.75	0.78	0.76	0.79	0.83	0.81
SourceTaxon	202	125	57	19	0.72	0.58	0.64	0.79	0.63	0.70
TreeVisualizationProgram	18	7	2	0	0.90	0.72	0.80	0.90	0.72	0.80
GuessedGeneralPackagesCategories	87	1	31	0	0.73	0.98	0.84	0.73	0.98	0.84
Sum	1851	644	564	146						
Macro summary					0.75	0.67	0.69	0.78	0.70	0.72
Micro summary					0.72	0.69	0.70	0.78	0.74	0.76

The 'M' column represents the number of successfully recognized annotations. The 'G' (only in Gold set) column represents missed annotations while the 'P' (only in Prediction set) column shows the number of unknown annotations. The 'O' (Overlap) column shows the number of lenient (partially recognized) annotations. The SFM and LFM columns shows the strict and lenient F-Measures [57], respectively. Micro averaging essentially treats the corpus as one large document while Macro averaging calculates the F-Measure on a per document basis, and then averages the results.

ANNEXE E

Tableau 5.5: Différences entre un expert et un novice.

1	Les experts se rappellent des informations pertinentes de leur domaine d'expertise mieux que les novices même que l'acquisition de connaissances chez les experts s'étend au-delà de l'accumulation des informations. Les experts regroupent inconsciemment les informations dans un nombre plus restreint d'unités (*chunks*) qui sont sauvegardées dans leur mémoire de travail.
2	Les experts sont capables d'utiliser des représentations structurelles "profondes" de résolution de problèmes tandis que les novices tendent de rester dans un niveau "superficiel" de définition de problème.
3	Les experts prennent plus de temps dans le développement d'une approche globale de résolution de problème avant de tenter de fouiller dans les détails du problème. Pourtant, les novices tendent de se "plonger" immédiatement dans la résolution du problème sans examiner le fond ou le contexte du problème.
4	Les experts résolvent plus rapidement et efficacement un problème que les novices.

ANNEXE F

Tableau 5.6: Tableau comparatif des systèmes d'extraction des entités nommées dans le domaine biomédical.

	Rèfèrence	Modèle	Domaine d'application	Approche
Apprentissage supervisé	(de Bruijn *et al.*, 2011)	semi-HMM (Witten *et al.*, 2016)	Compétition 2010 i2b2 NLP	Basée sur des attributs de type "niveau-de-discours" : token (*e.g.*, ponctuation, préfixe, stem), contexte (*e.g.*, bi/tri/quad-grammes, tokens voisins), phrase (*e.g.*, longueur, premier mot) et section (*e.g.*, titres et sous-titres).
	(Xu *et al.*, 2012)	CRF (Witten *et al.*, 2016)	Décharge clinique	En plus des attributs de niveau-de-discours, les auteurs proposent des attributs de type syntaxiques (*e.g.*, POS, phrases nominales/adjectivales) et sémantiques (appariement avec des mots dans des dictionnaires comme MeSH (Medical Subject Headings) et UMLS (Unified Medical Language System).
	(Leaman *et al.*, 2015)	CRF (Witten *et al.*, 2016)	Reconnaissance des entités chimiques	Leur système tmChem permet de reconnaitre les éléments chimiques dans les résumés de PubMed (?) en utilisant des attributs spécifiques au domaine décrivant des motifs de formules chimiques, séquences nucléiques et protéiques, *etc.*
Apprentissage semi-supervisé	(Chen *et al.*, 2015b)	CRF (Witten *et al.*, 2016)	Textes cliniques	Basée sur une approche d'apprentissage actif (via des "feedbacks" interactifs avec l'expert) visant à réduire le coût d'annotation, les auteurs proposent trois algorithmes basés sur : (1) l'incertitude, (2) la diversité et (3) un algorithme de base. Dans la première algorithme, les phrases incertaines (en se basant sur l'entropie des mots et séquences) sont considérées les plus informatives. Dans l'hypothèse de diversité, des scores de similarité deux-à-deux entre les phrases sont calculés, à savoir : (i) la similarité syntaxique basée sur les fréquences des classes *POS* dans les phrases, (ii) la similarité *cosinus* des concepts UMLS en se basant sur le plus petit ancêtre commun. L'algorithme basique calcule tout simplement le nombre de mots et de concepts dans chaque phrase et favorise les phrases courtes ayant le plus nombre de concepts du domaine.
	(Provoost et Moens, 2015)	SVM (Witten *et al.*, 2016)	Réseau de régulation de gènes (BioNLP)	Basée sur une extension de l'algorithme SVM afin de d'équilibrer les données positives et négative en re-positionnant l'hyperplan séparant les deux classes. L'algorithme ajoute ainsi deux paramètres (représentant les points positifs et négatifs) dans la fonction de maximisation de la marge. De plus, un algorithme de pré-sélection de données non-annotés est proposé pour chercher les exemples négatifs selon un seuil de fonction de masse (de probabilités) sur les vecteurs d'attributs.

Apprentissage non supervisé	**(Liu *et al.*, 2015)**	CRF (deep learning+skip-gram) (Witten *et al.*, 2016)	Reconnaissance des noms de médicaments	Les auteurs proposent d'utiliser le modèle skip-gram (Levy et Goldberg, 2014) comme une méthode de représentation dynamique (*word embedding* en anglais) de cooccurrences de mots par des vecteurs de nombres réels afin de reconnaitre les noms de médicaments dans les résumés des articles MEDLINE. Cette technique prédit le contexte d'un mot en se basant sur la représentation de ce mot (et en maximisant la probabilité softmax du mot et son contexte).
	(Choi *et al.*, 2016)	Perceptron Multi-couches (MLP) (Witten *et al.*, 2016)	Reconnaissance des codes et procédures médicales pendant des consultations	Les auteurs proposent un nouveau modèle *Med2Vec* pour apprendre les représentations des cooccurrences des codes médicales dans une large base de données de 3 millions de visites. Ce modèle de deux couches cachées de Perceptron, représente deux représentations : les cooccurrences de codes et les séquences de visites.

Tableau 5.7: Tableau comparatif des systèmes d'extraction des relations dans le domaine biomédical.

	Référence	Modèle	Domaine d'application	Approche
Apprentissage supervisé	**(?)**	SVM (Witten *et al.*, 2016)	Reconnaissances des événements cliniques	La structure sémantique d'un événement est représentée par la relation Problème, Action dans le Temps. Afin de reconnaitre les termes de la relation, un ensemble d'attributs est calculé : les occurrences des termes, les sacs-d'évènements (dans une fenêtre de 3 évennements), les fréquences de chaque classe et les occurrences des motifs d'événements comme "Médicament-Effet", "Symptôme-Traitement", etc.
	(Li *et al.*, 2015)	SVM (+noyau d'arbres de convolutions) (Witten *et al.*, 2016)	Interaction Protein-Protein	Basée sur un noyau d'arbres représentant les instances d'attributs dans arbres de convolutions (Collins et Duffy, 2002) ayant des structures syntaxiques similaires. Les attributs utilisés ici sont basés sur : les sacs-de-mots (occurrences de proteins, mots entre les proteins, *etc.*), les représentations dynamiques de mots basées sur le modèle *skip-gram* (Mikolov *et al.*, 2013) et les représentations basées sur les regroupements de vecteur de mots (K-means, n-gram, et d'autres.) (Witten *et al.*, 2016).
Apprentissage semi-supervisé	**(Sun *et al.*, 2011)**	Brown Word Clustering Brown *et al.* (1992)	Relations ACE	Les auteurs proposent d'utiliser le regroupement statistique de Brown *et al.* (1992) qui assigne à chaque mot un *cluster* puis les combine en maximisant l'entropie moyen d'information (Quinlan, 1986). En traversant les étapes de combinaison de cluster, chaque mot est représenté par une séquence binaire représentant la hiérarchie du mot. Les attributs utilisés sont basés sur le tuple <Before, Domain, Between, Range, After> qui représente la relation entre le domaine et le co-domaine. Ces attributs sont ensuite triés et filtrés par leur "importance linguistique" mesuré par le gain d'information de Quinlan (1986).
	(Poon *et al.*, 2015)	Perceptron (Witten *et al.*, 2016)	Voies biologiques (Pathways) du Cancer	L'idée clé ici est d'introduire une variable latente pour tester si une relation R entre une entité nommée E_1 et une E_2 est "bonne" pour chaque cooccurrence des E dans une phrase. L'algorithme génère d'une façon aléatoire, les exemples négatifs et calcule des attributs classiques comme : la direction, la distance et le chemin de dépendance syntaxique (Culotta et Sorensen, 2004) entre E_1 et E_2.

Apprentissage non supervisé	(?)	Open Information Extraction (?)	Sites Web d'information, Wikipedia et manuels scolaires de biologie	À partir d'un échantillon grand de phrases (18 millions), et un ensemble de tuples extraits à partir du WordNet, il s'agit d'extraire des motifs de POS représentant les relations en utilisant un analyseur de chemins de dépendances lexico-syntaxiques (dependency parse paths) afin d'analyser les contextes de phrases. Ces contextes extraient des attributs additionnels tels que des clauses de compléments et d'adverbes dans les phrases pour filtrer les tuples négatifs.
	(Alicante *et al.*, 2016)	K-means (Witten *et al.*, 2016)	Enregistrements cliniques	Dans la première étape, il s'agit d'utiliser le méta-thesaurus médical UMLS et un ensemble de motifs POS afin de reconnaitre les entités nommées dans les textes. Une méthode de clustering (regroupement) de pairs d'attributs est ensuite utiliser afin de reconnaitre les relations. Les auteurs proposent d'utiliser les attributs de bi et trigrammes de mots ainsi que des attributs de "barrières" basés sur les paires de classes POS (e, t) où t est le terme déclencheur (généralement un adjectif) et e. le premier terme situé à gauche de t.
	(Nguyen et Grishman, 2015)	Réseau neuronal convolutif (Tymoshenko *et al.*, 2016)	Détection des événements ACE	Afin de prédire les tokens déclencheurs d'événements, les auteurs proposent d'utiliser trois représentations dynamiques (embeddings) dans un réseau de neurones convolutif (Tymoshenko *et al.*, 2016). En fixant le contexte d'un token dans un phrase, il s'agit de calculer les tables dynamiques de : (i) mots en utilisant un modèle déjà entrainé afin de capturer les motifs syntaxiques, (ii) positions (distances i entre les tokens du contexte) et (iii) types d'entités nommées de l'ACE en utilisant l'annotation BIO(Begin, Inside, Outside).

ANNEXE G

Tableau 5.8: Tableau comparatif des systèmes de fouille de motifs basée sur une ontologie.

	Référence	Type d'ontologie	Stratégie de parcours de l'espace	Approche	Limites
Fouille de motifs séquentiels	**(Han et Fu, 1995)**	Taxonomie	Apriori-like (en largeur d'abord)	Pour chaque niveau k, il s'agit de remplacer tous les items des motifs (associations) d'un seul niveau conceptuel (*e.g.* lait -> pain) afin de générer des associations dans un différent niveau (*e.g.*, lait de la marque X -> pain de la marque Y).	Les motifs sont limités à des items du même niveau conceptuel.
	(Fortin et Liu, 1996)	Taxonomie	Apriori-like (en profondeur d'abord)	Étendre le travail de (Han et Fu, 1995) pour une approche Orientée Objet : transformer et encoder les items en les réorganisant selon les IDs de leurs catégories, *i.e.* en combinant les profondeurs des sous-hiérarchies avec une simple fonction agrégation (conjonction d'items).	Cette approche n'est pas applicable directement aux flux de travaux (manque de relations entre les items).
	(Srikant et Agrawal, 1996)	Taxonomie	Apriori-like (en profondeur d'abord)	S'inspirant de l'approche AprioriAll (Agrawal et Srikant, 1994), l'algorithme GSP (*Generalized Sequential Patterns*) ajoute à tous les items concrets de chaque transaction leurs ancêtres à partir de la taxonomie. Il s'agit ensuite de générer des motifs de différents niveaux hiérarchiques.	L'algorithme génère des motifs séquentiels redondants. Malgré que les auteurs proposent une mesure de pertinence (*interestingness*) basée sur le niveau d'interêt fourni par l'utilisateur pour un niveau I dans la taxonomie, cette mesure ne garantit pas malheureusement la génération des motifs uniques.
	(Šebek *et al.*, 2012)	Taxonomie	Apriori-like (en profondeur d'abord)	Étendre l'algorithme GSP (Srikant et Agrawal, 1996) en élaguant les motifs "sur-généralisés" selon la mesure de l'information de Shanon.	Pas de preuve sur l'exactitude et la complétude de l'approche.
	(Senkul et Salin, 2012)	Taxonomie	Apriori-like (en profondeur ou en largeur d'abord).	Visant une application directe à la fouille de l'usage du Web (Srivastava *et al.*, 2000), l'approche étende l'algorithme SPADE (?) afin d'inclure la taxonomie dans le processus de la fouille. L'algorithme utilise une représentation verticale des instances des concepts dans la taxonomie.	Le mapping entre les URLs des pages We bet les instances de la taxonomie est fait d'une façon manuelle. Ainsi, l'utilisateur doit choisir un seul niveau hiérarchique à la fois pour générer des motifs en naviguant dans la hiérarchie de l'ontologie.

Fouille de graphes	**(Inokuchi, 2004)**	Ontologie (DAG)	Apriori-like (en largeur d'abord)	Visant une application dans le domaine de la chimie analytique afin d'identifier des composants carcinogènes dans un ensemble de substances chimiques. L'approche permet de modéliser à la fois les relations hiérarchiques et sémantiques entre les noeuds. C'est une extension de l'algorithme AcGM (Apriori-based connected Graph Mining) (Inokuchi *et al.*, 2002) pour fouiller des sous-structures généralisées de graphes étiquetés. Les graphes sont représentés par des matrices d'adjacence en les associant avec des structures canoniques afin d'identifier (apparier) les graphes isomorphes. Les motifs sur-généralisés sont élagués s'ils existent d'autres motifs représentant la même "classe" de graphes.	L'approche de parcours d'espace de motifs en largeur d'abord est couteuse en terme de mémoire ce qui ne permet pas de gérer des taxonomies et des graphes de grandes tailles. De plus, les auteurs ne donnent pas de preuves concernant la complétude et la génération unique des motifs.
	(Cakmak et Ozsoyoglu, 2008)	Taxonomie	Apriori-like (en profondeur d'abord)	L'algorithme Taxogram décrit 3 étapes. (1) Transformer la base de données en une base de graphes avec les concepts les plus généralisés (*root concepts*) en gardant les étiquettes (*labels*) originales des instances. (2) Utiliser l'algorithme gSpan (Yan et Han, 2002) pour fouiller les motifs généralisés en créant des représentations des motifs basées sur des indexes (*noeud, occurrences*) afin faciliter la tâche d'appariement de graphes. Et (3) Spécialiser les motifs de niveau k à partir de ceux de niveau $k-1$ en remplaçant les noeuds par leurs descendants hiérarchiques.	Puisqu'il n y a pas de fonction d'ordre complet entre les items, l'algorithme peut générer des dupliquants. Ainsi, une étape de post-traitement basée sur la liste des indexes est proposée.
	(Jiang *et al.*, 2007)	Ontologie (RDF)	Apriori-like (en profondeur d'abord)	Visant une application dans la fouille de relations sémantiques (*relationset*) dans les textes du Web. L'algorithme GP-Close fouille les motifs fermés en (1) générant les motifs fermés locaux (dans chaque niveau) et (2) élaguant les candidats "enfants" (descendants de chaque relationset) redondants et leurs sous-arbres conséquents.	Les auteurs ne présentent pas la façon dont ils traitent les prédicats (relations) de triplets dans les étapes de généralisation et de appariement (isomorphisme).
	(Werth *et al.*, 2009)	Ontologie (DAG)	Apriori-like (en profondeur d'abord)	Visant une application dans le domaine d'abstractions procédurales qui cherche les codes redondants dans un programme en extrayant des nouvelles procédures substituant les motifs de codes fréquents. L'algorithme DAGMA propose une nouvelle représentation canonique basée sur l'ordre topologique des noeuds dans le DAG permettant d'accélérer le processes d'appariement. La forme canonique consiste à créer des tuples (*index du label, index du prédécesseur*) en remplaçant les noeuds/arcs par des descriptions canoniques d'un niveau d'abstraction supérieur.	Puisqu'il n'ya pas de relation d'ordre entre les arêtes des DAGs, l'approche ne garantit pas la génération unique des motifs. De plus, le temps de calcul de graphes isomorphes avec des noeuds non-connectés est supérieur à celui avec des noeuds connectés (puisque plus de structures isomorphes sont retrouvées).

	(Adda *et al.*, 2010)	Ontologie (RDF)	Apriori-like (en profondeur d'abord)	Visant une application dans la fouille de flux de *clicks* à partir des logs de pages Web annotées par des métadonnées d'une ontologie de domaine (Srivastava *et al.*, 2000). La première étape cherche à enrichir les séquences des items en y ajoutant toutes les relations ontologiques (hiérarchiques et non hiérarchiques). Quatre opérateurs canoniques sont proposés afin d'ajouter/spécialiser les items (noeuds et relations) d'un motif généralisé.	L'algorithme ne peut être applicable directement à la fouille de flux de travaux. Ces derniers présentent des multi-ensembles d'items : un item/concept peut être utilisé plusieurs fois dans la même séquence sur plusieurs transactions. De plus, malgré les contraintes proposées sur les opérateurs canoniques, les auteurs ne présentent pas de preuves sur la complétude et l'unicité des motifs générés.
	(Nguyen *et al.*, 2014)	Ontologie (OWL)	Apriori-like (en largeur d'abord)	Les DAGs de flux de travaux sont initialement transformés en des arbres (parse trees) tout en appliquant un ordre topologique puis lexicographique. Les arbres sont ensuite généralisés en suivant un ordre basé sur la distance (la longueur du chemin le plus court entre deux concepts) pour réorganiser les subsomptions dans chaque étape de généralisation. Une fois que les arbres sont généralisés, l'approche reprend l'algorithme TreeMiner (Zaki, 2005) qui cherche à trouver les sous-arbres fréquents.	La transformation des DAGs en des arbres rend le système couteux en termes de mémoire et de calcul. De plus, l'auteur ne démontre la complétude de l'approche suite à cette transformation.

ANNEXE H

Tableau 5.9: Tableau comparatif des systèmes de recommandation basés sur le contenu.

Référence	Système	Domaine d'application	Représentation de connaissances	Stratégie d'annotation	Stratégie d'appariement item-profil
(Magnini et Strapparava, 2001)	SiteIF	Sites Web d'informations (News) multilingue	MultiWordNet : une base de données lexicale multilingue alignant les significations des langues anglaise et autres.	Chaque information est associée automatiquement avec une liste de synsets utilisant une technique désambiguïsation de mots polysémiques (Stevenson et Wilks, 2003). Le profil d'utilisateur créée est un réseau sémantique dont les noeuds représentent les synsets trouvés dans les documents (textes) associés à chaque utilisateur.	Le premier système adoptant une approche basée sur le contenu des documents textes pour créer un modèle d'intérêts . Le système reçoit en entrée la représentation du synset et le modèle d'utilisateur pour estimer la pertinence du document par rapport aux co-occurrences des concepts dans différents contextes (autres documents).
(Semeraro *et al.*, 2009)	ITR (Litem Recommender)	Divers domaines (e.g., films, musiques, livre),	WordNet : ontologie de données lexicales	Comme SiteIF, ITR utilise une approche d'analyse sémantique représentant le "réseau de significations" de chaque document ainsi chaque item est représenté par un vecteur de synsets (Bag-of-synstes). Un classifieur Bayésien binaire est ensuite entrainé et appliqué afin d'identifier les items "intéressants".	L'appariement revient à calculer la probabilité d'appartenance d'un item à la classe "intéressant" selon les probabilités des synsets dans le modèle.
(?)	iTV Recommender	Télévision digitale interactive	Ontologie OWL (Web Ontology Language) du domaine	Annotation manuelle des ressources	Les auteurs utilisent une technique basée sur les associations sémantiques afin d'inférer des nouvelles connaissances : liens entre items et items partagent le même ancêtre commun dans la hiérarchie de concepts.
(Vesin *et al.*, 2012)	Protus 2.0	Apprentissage des concepts en language de programmation JAVA	Ontologie OWL du domaine et uneOntologie OWL de tâches	Annotation manuelle des ressources	Les auteurs utilisent le moteur d'inférence SWRL (Semantic Web Rule Language (SWRL) et les règles Jess (Java Expert System Shell) de la logique du premier ordre afin de recommander des nouvelles ressources aux utilisateurs.

(Cobos *et al.*, 2013)	RSPP (Recommendation System of Pedagogical Patterns)	Stratégies pédagogiques d'enseignement	Ontologie RDF de tâches	Annotation manuelle des ressources	Les données du contexte pédagogique (caractéristiques sur le cours à enseigner, objectifs, profils d'étudiants, professeurs, *etc.*) sont d'abord transformées en une matrice de valeurs normalisées (entre 0 et 1) qui sera réduite en une représentation plus compacte via une Décomposition en Valeurs Singulières. Enfin, il s'agit de calculer les voisins les plus proches à la requête de l'enseignant en utilisant la similarité *cosinus* par rapport aux connaissances du système.
(Pukkhem, 2013)	-	Apprentissage en ligne	3 ontologies OWL représentant : l'apprenant (appelée SLS), l'objet d'apprentissage (appelée LOS), les règles de mapping (entre le style et l'objet d'apprentissage)	Les règles de mapping, décrites par un expert dans une ontologie, sont utilisées pour extraire les caractéristiques des objets à apprendre. Ces dernières sont validées par des experts manuellement.	Une fois que le profil d'apprenant et le modèle d'objets associé sont construits, le système applique un algorithme de recommandation multi-agents qui apprend à calculer un score de préférence basé sur la fréquence des caractéristiques de chaque objet/apprenant en interagissant avec l'environnement (objets, application, etc.).

ANNEXE I

```
<?xml version="1.0" encoding="utf-8"?>
<Sequences>
<Sequence>
   <individual>dataStructureID_2</individual>
   <individual>dataStructureID_12</individual>
   <individual>sequence</individual>
   <individual>dataStructureID_1</individual>
   <individual>databaseID_3</individual>
   <individual>programID_3237</individual>
   <individual>alignmentProgramID_96</individual>
   <individual>alignmentProgramID_8</individual>
   <individual>programID_3381</individual>
   <individual>Boots_50_1000</individual>
</Sequence>
<Sequence>
   <individual>sequence</individual>
   <individual>dataStructureID_1</individual>
   <individual>dataStructureID_8</individual>
   <individual>databaseID_3</individual>
   <individual>programID_3613</individual>
   <individual>programID_3373</individual>
</Sequence>
<Sequence>
   <individual>sequence</individual>
   <individual>dataStructureID_1</individual>
   <individual>databaseID_98</individual>
```

FIGURE 5.8: Extrait du résultat de l'extraction de flux de travaux concrets.

ANNEXE J

```
[BaseSequence(10.025152% (279/2783)) [concepts=[GenomeSequence, SequenceType, Multipl
[BaseSequence(10.025152% (279/2783)) [concepts=[GTR, GeneralPurposePackagesProgram, G
[BaseSequence(10.025152% (279/2783)) [concepts=[InternationalNucleotideSequenceDataba
[BaseSequence(10.025152% (279/2783)) [concepts=[NucleicAcidSequence, Tree, DataBase,
[BaseSequence(10.025152% (279/2783)) [concepts=[ProteinSequence, DataBase, GeneralMSA
[BaseSequence(10.025152% (279/2783)) [concepts=[ProteinSequence, DataBase, MultipleSe
[BaseSequence(10.025152% (279/2783)) [concepts=[ProteinSequence, DataBase, MultipleSe
[BaseSequence(10.025152% (279/2783)) [concepts=[ProteinSequence, SequenceAlignment, G
[BaseSequence(10.025152% (279/2783)) [concepts=[ProteinSequence, SequenceAlignment, G
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, CharacterBasedMethod, Chara
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, DataBase, GeneralMSAProgram
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, DataBase, GeneralMSAProgram
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, DataBase, GeneralPurposePac
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, GeneralMSAProgram, DNAModel
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, GeneralPurposePackagesProgr
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, MakingDistancesProgram, ],
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, PairwiseAlignmentProgram, G
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, ProteinSequence, bootstraps
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, ProteinSequence, Parameters
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceAlignment, Multiple
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, GeneralPurpos
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, GeneralPurpos
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, International
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, International
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, International
[BaseSequence(10.025152% (279/2783)) [concepts=[Sequence, SequenceType, International
```

FIGURE 5.9: Extrait du résultat de la fouille de motifs généralisés (1).

```
[BaseSequence(56.16242% (1563/2783)) [concepts=[ParametersBootstrappingProgram, ], p
[BaseSequence(56.30614% (1567/2783)) [concepts=[SequenceType, BootstrappingProgram,
[BaseSequence(57.204456% (1592/2783)) [concepts=[GeneralPurposePackagesProgram, Boot
[BaseSequence(58.965145% (1641/2783)) [concepts=[BootstrappingProgram, ], properties=
[BaseSequence(60.366512% (1680/2783)) [concepts=[Tree, GeneralPurposePackagesProgram,
[BaseSequence(61.157024% (1702/2783)) [concepts=[SequenceType, GeneralPurposePackage
[BaseSequence(61.80381% (1720/2783)) [concepts=[Tree, GeneralPurposePackagesProgram,
[BaseSequence(62.1272% (1729/2783)) [concepts=[Tree, CharacterBasedMethod, ], proper
[BaseSequence(62.88178% (1750/2783)) [concepts=[SequenceType, CharacterBasedMethod,
[BaseSequence(63.061447% (1755/2783)) [concepts=[SequenceType, GeneralPurposePackage
[BaseSequence(64.13942% (1785/2783)) [concepts=[GeneralPurposePackagesProgram, Chara
[BaseSequence(65.72044% (1829/2783)) [concepts=[GeneralPurposePackagesProgram, Gener
[BaseSequence(66.04384% (1838/2783)) [concepts=[CharacterBasedMethod, ], properties=
[BaseSequence(66.870285% (1861/2783)) [concepts=[Tree, GeneralPurposePackagesProgram,
[BaseSequence(68.02013% (1893/2783)) [concepts=[SequenceType, Tree, GeneralPurposePa
[BaseSequence(70.894714% (1973/2783)) [concepts=[SequenceType, Tree, ], properties=],
[BaseSequence(71.43371% (1988/2783)) [concepts=[SequenceType, SequenceType, GeneralPu
[BaseSequence(74.77542% (2081/2783)) [concepts=[Sequence, GeneralPurposePackagesProg
[BaseSequence(74.84729% (2083/2783)) [concepts=[SequenceType, SequenceType, ], prope
[BaseSequence(78.4046% (2182/2783)) [concepts=[Sequence, ], properties=],
[BaseSequence(88.03449% (2450/2783)) [concepts=[Tree, GeneralPurposePackagesProgram,
[BaseSequence(90.72943% (2525/2783)) [concepts=[SequenceType, GeneralPurposePackages
[BaseSequence(91.9152% (2558/2783)) [concepts=[Tree, ], properties=],
[BaseSequence(95.04132% (2645/2783)) [concepts=[SequenceType, ], properties=],
[BaseSequence(95.29285% (2652/2783)) [concepts=[GeneralPurposePackagesProgram, ], pr
```

FIGURE 5.10: Extrait du résultat de la fouille de motifs généralisés (2).

ANNEXE K

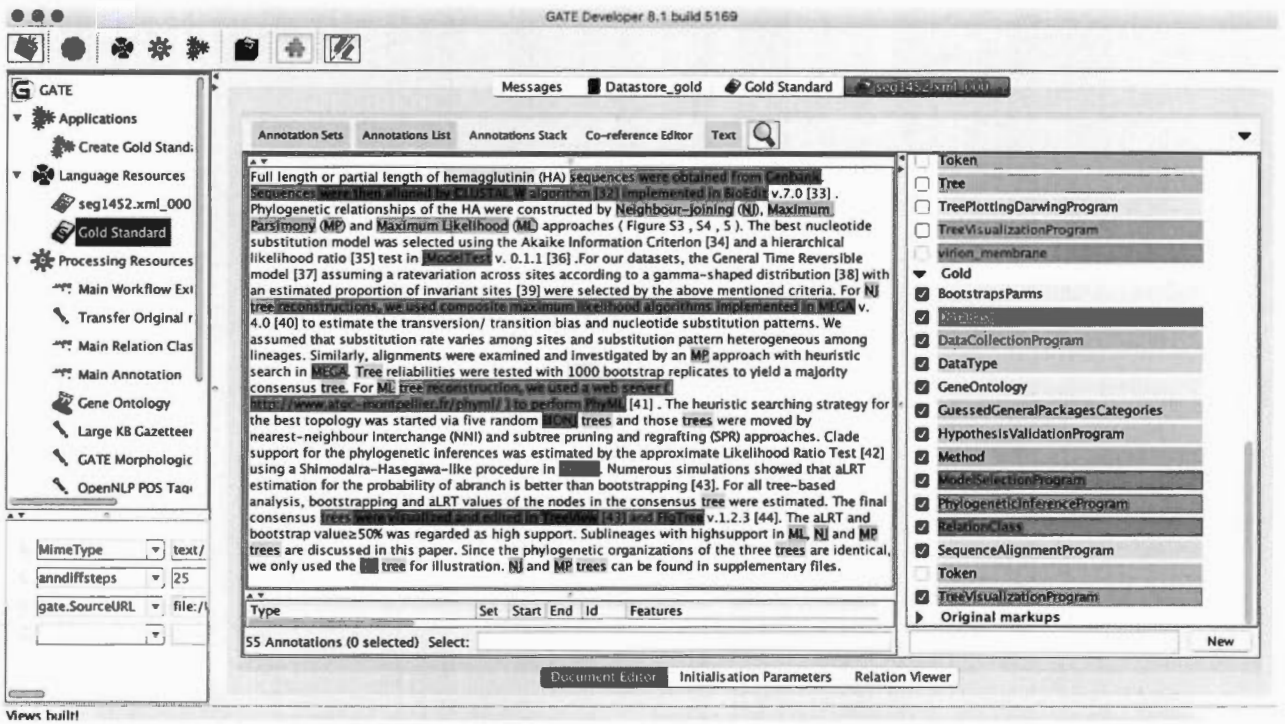


FIGURE 5.11: Plateforme GATE 8.1.

ANNEXE L

```
/**
 *
 * @author Tomas Martin
 */

package ontopatternmatching;

import ontologyrep2.OntoRepresentation;

public final class ValideDomaineRelation implements Job{
    /**
     *
     * @param block
     * @param sequence
     * @param appariement
     * @param modifications
     * @param ontology
     * @param m
     * @param sm
     * @return
     */
    @Override
    public final int doJob(JobBlock block, final Sequence sequence, int[] appariement,
    boolean[] modifications, final OntoRepresentation ontology, final Motif m) {
        int size=sequence.relations.size();
        RelationJobBlock rjob = ((RelationJobBlock)block);
        int solution = appariement[block.position];
        int prev_solution = 0;
        int prev_prev_domaine=0;

        //si on a un concept (normalement toujours vrai)
        if(block.prev!=null) {
            //alors on recupere la solution du concept
            prev_solution = appariement[block.prev.position];

            //si on a un concept precedent le concept domaine et qu'il est domaine
            d'au moins une relation
            if(block.prev.prev!=null && block.prev.prev.last_domain_added!=null){
                //alors on recupere sa position dans la sequence
                prev_prev_domaine = appariement[block.prev.prev.
                last_domain_added.position];
            }
```

FIGURE 5.12: Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (1).

```
int solution_domaine = 0;
if(rjob.domaine!=null)
    solution_domaine = appariement[rjob.domaine.position];

int j =0;
Integer[] curr_rel = rjob.curr_rel;
if(SequenceMatcher.debug) System.out.println("Searching domain..."+
block.prev.position);

//si notre concept est domaine d'une autre relation precedente
if(rjob.domaine!=null){
    //on place notre curseur a droite
    j = solution_domaine;
    if(SequenceMatcher.debug) System.out.println("(D) Starting from "+j+".");
}

//si on a deja matche cette case au moins une fois avec une solution et
que notre concept a une solution
//mais que la solution du concept a ete modifee et ne correpond plus a la
position utilisee lors de notre derniere solution
//pour la case actuelle
else if(solution!= 0 && curr_rel!=null && prev_solution != curr_rel[1]){
    //si on est pas dans le premier concept de la liste et qu'il possede
    au moisn un domaine
    if(block.prev.prev!=null && block.prev.prev.last_domain_added!=null){
        //on place le curseur a droite de cette solution
        j = prev_prev_domaine;
        if(SequenceMatcher.debug) System.out.println("(D) Setting j to "+j+".");
    }
    else{
        //sinon, on repart au debut
        j = 0;if(SequenceMatcher.debug) System.out.println("(D) Reseting j
        to "+j+".");
    }
}
//on est dans le premier concept de la case, on n'a jamais matche correctement
// la case
else{
    //que doit on faire ici ? normalement rien du tout...
    if(SequenceMatcher.debug) System.out.println("Cas inconnu!!!!");
}
```

FIGURE 5.13: Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (2).

```
            if(SequenceMatcher.debug) System.out.println("(D) Starting j at "+j+".");


            for(int i=j;i<size;i++){
                //on recupere la relation courrante
                curr_rel = sequence.relations.get(i);

                //si le nom de la relation correspond
                if(curr_rel[1] == prev_solution){
                    //si le codomaine correspond aussi
                    if(ontology.isPropertyEqualOrDescendant(block.item, curr_rel[0])){
                        if(SequenceMatcher.debug) System.out.println("(D)Found potential
                        +"relation:"+i+" => "+block.item+" is "
                                + "equal or D to "+curr_rel[0]);
                        rjob.curr_rel = curr_rel;

                        return i+1;
                    }
                }
                //si > alors on sort.
            }
            if(SequenceMatcher.debug) System.out.println("No more solutions for Domain....
            +"GOING BACK");
            return 0;
        }

    }
```

FIGURE 5.14: Portion du code JAVA implémentant la structure d'appariement de motifs et de flux de travaux (3).

ANNEXE M

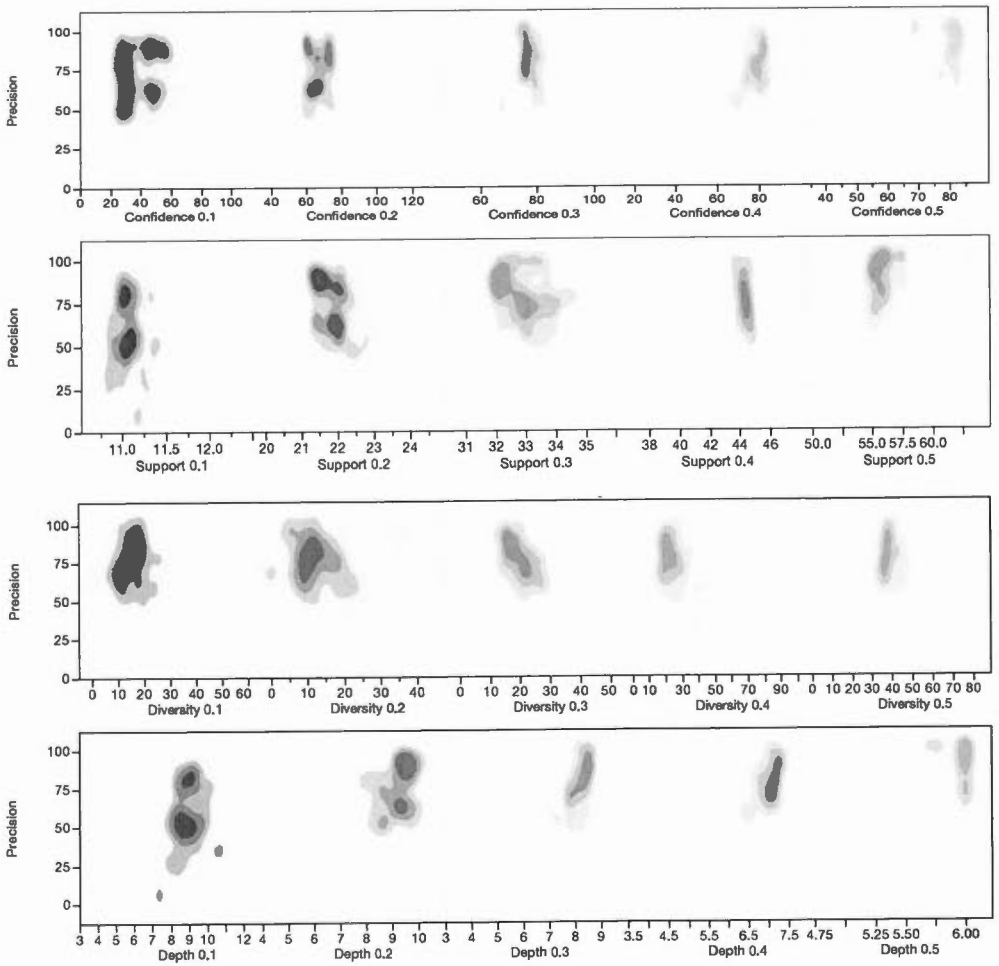


FIGURE 5.15: Distribution des métriques de tri de motifs.

ANNEXE N

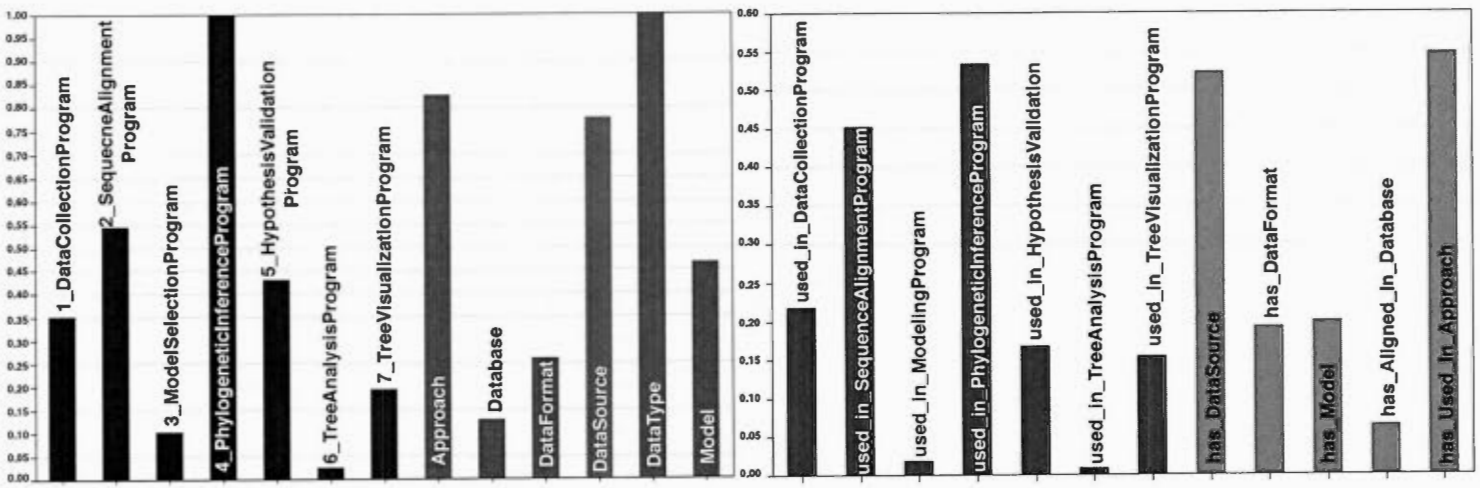


FIGURE 5.16: Extraits des supports relatifs de concepts et relations de l'ontologie dans la base d'articles extraits depuis PubMed PMC 2013-2015.

ANNEXE O

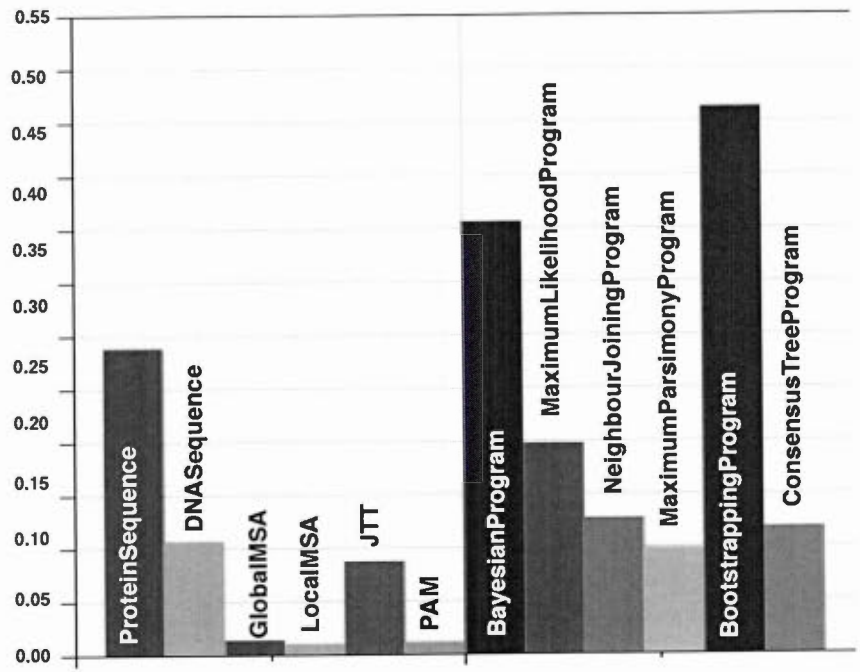


FIGURE 5.17: Extrait des supports relatifs des objets les plus fréquents extraits depuis PubMed PMC 2013-2015.

BIBLIOGRAPHIE

Abascal, F., Zardoya, R. et Posada, D. (2005). Prottest : selection of best-fit models of protein evolution. *Bioinformatics*, *21*(9), 2104–2105.

Adda, M., Valtchev, P., Missaoui, R. et Djeraba, C. (2007). Toward recommendation based on ontology-powered web-usage mining. *Internet Computing, IEEE*, *11*(4), 45–52.

Adda, M., Valtchev, P., Missaoui, R. et Djeraba, C. (2010). A framework for mining meaningful usage patterns within a semantically enhanced web portal. Dans *Proceedings of the Third C* Conference on Computer Science and Software Engineering*, 138–147. ACM.

Addis, A. et Borrajo, D. (2010). From unstructured web knowledge to plan descriptions. In *Information Retrieval and Mining in Distributed Environments* 41–59. Springer.

Aggarwal, C. C. (2016). *Recommender systems : The Textbook.* `http://dx.doi.org/10.1007/978-3-319-29659-3`

Aggarwal, C. C. et Han, J. (2014). *Frequent Pattern Mining.* Springer Publishing Company, Incorporated.

Agrawal, R. et Srikant, R. (1994). Fast algorithms for mining association rules in large databases. Dans *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, 487–499., San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. Récupéré de `http://dl.acm.org/citation.cfm?id=645920.672836`

Alicante, A., Corazza, A., Isgrò, F. et Silvestri, S. (2016). Unsupervised entity and relation extraction from clinical records in italian. *Computers in Biology and Medicine*, *72*, 263 – 275. `http://dx.doi.org/http://dx.doi.org/10.1016/j.compbiomed.2016.01.014`. Récupéré de `http://www.sciencedirect.com/science/article/pii/S001048251630004X`

Angles, R. et Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, *40*(1), 1.

Anisimova, M., Liberles, D. a., Philippe, H., Provan, J., Pupko, T. et von Haeseler, A. (2013). State-of the art methodologies dictate new standards for phylogenetic analysis. *BMC evolutionary biology*, *13*, 161. http://dx.doi.org/10.1186/1471-2148-13-161

Aparna, M. et Ku, T. S. (2016). A Survey on Novel Approach for Biomedical Named Entity Recognition Using Db-Scan. *International Journal of Engineering Science*, *6*(11), 3074–3077. http://dx.doi.org/100:55-61

Arias, J. J. P., Vilas, A. F. et Redondo, R. P. D. (2012). *Recommender Systems for the Social Web*. Springer Science & Business Media. Récupéré de http://books.google.ca/books?id=DA5p81v6EQ4C&pg=PA100&dq=recommender+system+ontology&hl=&cd=2&source=gbs_api

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. *et al.* (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T. *et al.* (2000). Gene ontology : tool for the unification of biology. *Nature genetics*, *25*(1), 25–29.

Baader, F. (2007). *The description logic handbook*. theory, implementation, and applications. Cambridge University Press. Récupéré de http://books.google.ca/books?id=-pEoAQAAMAAJ&q=description+logic&dq=description+logic&hl=&cd=3&source=gbs_api

Bach, N. et Badaskar, S. (2007). A survey on relation extraction. *Language Technologies Institute, Carnegie Mellon University*.

Back, R.-J. (1987). *Procedural abstraction in the refinement calculus*. Åbo akademi, Departments of Information Processing & Mathematics.

Bafna, V. et Pevzner, P. A. (1996). Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, *25*(2), 272–289.

Barnard, D. L. (2000). Pegasys. *Pegasys 2*, *2*, 1530.

Barrett, N. et Weber-Jahnke, J. (2011). Building a biomedical tokenizer using the token lattice design pattern and the adapted viterbi algorithm. *BMC bioinformatics*, *12*(Suppl 3), S1.

Becker, B. E., Huselid, M. A., Becker, B. et Huselid, M. A. (1998). High performance work systems and firm performance : A synthesis of research and managerial implications. Dans *Research in personnel and human resource management*. Citeseer.

Bergmann, R. et Gil, Y. (2014). Similarity assessment and efficient retrieval of semantic workflows. *Information Systems*, *40*, 115–127.

Bergmann, R. et Stahl, A. (1998). Similarity measures for object-oriented case representations. Dans *European Workshop on Advances in Case-Based Reasoning*, 25–36. Springer.

Blankenberg, D., Kuster, G. V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A. et Taylor, J. (2001). Galaxy : A web-based genome analysis tool for experimentalists. *Current protocols in molecular biology, Chapter 19.* `http://dx.doi.org/10.1002/0471142727.mb1910s89`. Récupéré de `http://dx.doi.org/10.1002/0471142727.mb1910s89`

Bogers, T. et Van Den Bosch, A. (2011). *Recommender Systems Handbook.* Boston, MA : Springer US. `http://dx.doi.org/10.1007/978-0-387-85820-3`. Récupéré de `http://link.springer.com/10.1007/978-0-387-85820-3`

Bogost, I. (2010). Process vs. procedure. Dans *Fourth International Conference of the Whitehead Research Project.*

Bollacker, K., Evans, C., Paritosh, P., Sturge, T. et Taylor, J. (2008). Freebase : a collaboratively created graph database for structuring human knowledge. Dans *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. AcM.

Bontcheva, K., Derczynski, L., Funk, A., Greenwood, M. A., Maynard, D. et Aswani, N. (2013). Twitie : An open-source information extraction pipeline for microblog text. Dans *Recent Advances in Natural Language Processing (RANLP)*, 83–90.

Brown, P. F., DeSouza, P. V., Mercer, R. L., Della Pietra, V. J. et Lai, J. C. (1992). Class-Based n-gram Models of Natural Language. *Computational Linquistics*, *18*(1950), 467–479.

Buyko, E., Wermter, J., Poprat, M. et Hahn, U. (2006). Automatically adapting an nlp core engine to the biology domain. Dans *Proceedings of the Joint BioLINK-Bio-Ontologies Meeting. A Joint Meeting of the ISMB Special Interest Group on Bio-Ontologies and the BioLINK Special Interest Group on Text Data Mining in Association with ISMB*, 65–68.

Cakmak, A. et Ozsoyoglu, G. (2008). Taxonomy-superimposed graph mining. Dans *EDBT '08 : Proceedings of the 11th Intl. Conference on Extending database technology*, 217–228., New York, NY, USA. ACM. `http://dx.doi.org/http://doi.acm.org/10.1145/1353343.1353372`

Cannataro, M., Guzzo, A., Comito, C. et Veltri, P. (2004). Ontology-based Design of Bioinformatics Workflows on PROTEUS. *Journal of Digital Information Management (JDIM)*, *2*(2), 87–92.

Cardie, C. (1997). Empirical methods in information extraction. *AI magazine*, *18*, 65–79.

Cavalli-Sforza, L. L. et Edwards, A. W. (1967). Phylogenetic analysis. models and estimation procedures. *American journal of human genetics*, *19*(3 Pt 1), 233.

Chen, L.-C., Kuo, P.-J. et Liao, I.-E. (2015a). Ontology-based library recommender system using mapreduce. *Cluster Computing*, *18*(1), 113–121.

Chen, X. et Chang, J. T. (2017). Planning bioinformatics workflows using an expert system. *Bioinformatics*, *33*(8), 1210–1215.

Chen, Y., Lasko, T. A., Mei, Q., Denny, J. C. et Xu, H. (2015b). A study of active learning methods for named entity recognition in clinical text. *Journal of Biomedical Informatics*, *58*, 11–18. `http://dx.doi.org/10.1016/j.jbi.2015.09.010`

Chieu, H. et Ng, H. (2002a). Probabilistic reasoning for entity and relation recognition. Dans *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2002)*.

Chieu, H. L. et Ng, H. T. (2002b). Named entity recognition : a maximum entropy approach using global information. Dans *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, 1–7. Association for Computational Linguistics.

Choi, E., Bahadori, M. T., Searles, E., Coffey, C., Thompson, M., Bost, J., Tejedor-Sojo, J. et Sun, J. (2016). Multi-layer Representation Learning for Medical Concepts. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 1495–1504. `http://dx.doi.org/10.1145/2939672.2939823`. Récupéré de `http://arxiv.org/abs/1602.05568{%}5Cnhttp://dl.acm.org/citation.cfm?doid=2939672.2939823`

Cobos, C., Rodriguez, O., Rivera, J., Betancourt, J., Mendoza, M., León, E. et Herrera-Viedma, E. (2013). A hybrid system of pedagogical pattern recommendations based on singular value decomposition and variable data attributes. *Information Processing and Management*, *49*(3), 607–625. `http://dx.doi.org/10.1016/j.ipm.2012.12.002`. Récupéré de `http://dx.doi.org/10.1016/j.ipm.2012.12.002`

Cohen, G. (1997). Align : a program to superimpose protein coordinates, accounting for insertions and deletions. *Journal of applied crystallography*, *30*(6), 1160–1161.

Collard, M. et Vansnick, J.-C. (2007). How to measure interestingness in data mining : a multiple criteria decision analysis approach. Dans *In Proceedings of the IEEE International Conference on Research Challenges in Information Science*, 395–400.

Collins, M. et Duffy, N. (2002). Convolution kernels for natural language. Dans *Advances in neural information processing systems*, 625–632.

Corbett, A. T. et Anderson, J. R. (1994). Knowledge tracing : Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, *4*(4), 253–278.

Culotta, A. et Sorensen, J. (2004). Dependency tree kernels for relation extraction. Dans *Proceedings of the 42nd annual meeting on association for computational linguistics*, p. 423. Association for Computational Linguistics.

Cunningham, H., Maynard, D., Bontcheva, K. et Tablan, V. (2002a). A framework and graphical development environment for robust nlp tools and applications. Dans *Annual Meeting of the Association for Computational Linguistics (ACL)*, 168–175.

Cunningham, H., Maynard, D., Bontcheva, K. et Tablan, V. (2002b). Gate : an architecture for development of robust hlt applications. Dans *Proceedings of the 40th annual meeting on association for computational linguistics*, 168–175. Association for Computational Linguistics.

Cunningham, H., Tablan, V., Roberts, A. et Bontcheva, K. (2013). Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLoS Computional Biology*, *9*(2), e1002854.

Dai, H. et Mobasher, B. (2002). Using ontologies to discover domain-level web usage profiles. *Semantic Web Mining 2nd Workshop at ECML/PKDD, Finland.*

Daley, B. J. (1999). Novice to expert : An exploration of how professionals learn. *Adult education quarterly*, *49*(4), 133–147.

Dane, E. (2010). Reconsidering the trade-off between expertise and flexibility : A cognitive entrenchment perspective. *Academy of Management Review*, *35*(4), 579–603.

Day, W. H. et Edelsbrunner, H. (1984). Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, *1*(1), 7–24.

de Bruijn, B., Cherry, C., Kiritchenko, S., Martin, J. et Zhu, X. (2011). Machine-learned solutions for three stages of clinical information extraction : the state of the art at i2b2 2010. *Journal of the American Medical Informatics Association*, *18*(5), 557–562. http://dx.doi.org/10.1136/amiajnl-2011-000150. Récupéré de http://www.ncbi.nlm.nih.gov/pubmed/21565856{%}5Cnhttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3168309

De Raedt, L. (2008). Logical and relational learning. Dans *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5249 LNAI, p. 1. http://dx.doi.org/10.1007/978-3-540-88190-2-1

Dědek, J. (2010). Towards semantic annotation supported by dependency linguistics and ilp. In *The Semantic Web (ISWC)* 297–304. Springer.

Deshpande, M. et Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 143–177.

DiBernardo, M., Pottinger, R. et Wilkinson, M. (2008). Semi-automatic web service composition for the life sciences using the biomoby semantic web framework. *Journal of Biomedical Informatics*, *41*(5), 837–847.

Diehl, V. A. et Mills, C. B. (2002). Procedural text structure and reader perceptions and performance. *The Journal of general psychology*, *129*(1), 18–35.

Dietterich, T. G. (2000). Ensemble methods in machine learning. Dans *International workshop on multiple classifier systems*, 1–15. Springer.

Doak, J. (1992). *An evaluation of feature selection methods and their application to computer security.* University of California, Computer Science.

dos Reis, M., Donoghue, P. C. et Yang, Z. (2016). Bayesian molecular clock dating of species divergences in the genomics era. *Nature Reviews Genetics*, *17*(2), 71–80.

Duck, G., Nenadic, G., Brass, A., Robertson, D. L. et Stevens, R. (2014). Extracting patterns of database and software usage from the bioinformatics literature. *Bioinformatics*, *30*(17), i601–i608.

Dufour-Lussier, V., Le Ber, F., Lieber, J. et Nauer, E. (2014). Automatic case acquisition from texts for process-oriented case-based reasoning. *Information Systems*, *40*, 153–167.

Durbin, R., Eddy, S. R., Krogh, A. et Mitchison, G. (1998). *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press. Récupéré de `http://www.amazon.com/Biological-Sequence-Analysis-Probabilistic-Proteins/dp/0521629713`

Ebeling, W. et Nicolis, G. (1991). Entropy of symbolic sequences : the role of correlations. *Europhysics Letters (EPL)*, *14*(3), 191.

Edgar, R. C. (2004). Muscle : multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, *32*(5), 1792–1797.

Ellis, C. A., Rembert, A. J., Kim, K.-H. et Wainer, J. (2006). Beyond workflow mining. Dans *Business Process Management*, 49–64.

Faatz, A. et Steinmetz, R. (2002). Ontology enrichment with texts from the www. *Semantic Web Mining*, p. 20.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. et Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park.

Federhen, S. (2012). The ncbi taxonomy database. *Nucleic acids research*, *40*(D1), D136–D143.

Fellbaum, C. (1998). *WordNet*. Wiley Online Library.

Felsenstein, J. (1985). Confidence limits on phylogenies : an approach using the bootstrap. *Evolution*, 783–791.

Felsenstein, J. et Felenstein, J. (2004). *Inferring phylogenies*, volume 2. Sinauer associates Sunderland.

Finn, R. D., Clements, J., Arndt, W., Miller, B. L., Wheeler, T. J., Schreiber, F., Bateman, A. et Eddy, S. R. (2015). Hmmer web server : 2015 update. *Nucleic acids research*, p. gkv397.

Forneris, S. G. et Peden-McAlpine, C. (2007). Evaluation of a reflective learning intervention to improve critical thinking in novice nurses. *Journal of advanced nursing*, *57*(4), 410–421.

Fortin, S. et Liu, L. (1996). An object-oriented approach to multi-level association rule mining. Dans *Proceedings of the fifth International Conference on Information and knowledge management*, 65–72. ACM.

Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gil, Y. et Goble, C. (2014). Common motifs in scientific workflows : An empirical analysis. *Future Generation Computer Systems*, *36*, 338–351.

Gascuel, O. (1997). Bionj : an improved version of the nj algorithm based on a simple model of sequence data. *Molecular biology and evolution*, *14*(7), 685–695.

Geng, L. et Hamilton, H. J. (2006). Interestingness measures for data mining : A survey. *ACM Computing Surveys (CSUR)*, *38*(3). `http://dx.doi.org/10.1145/1132960.1132963`. Récupéré de `http://doi.acm.org/10.1145/1132960.1132963`

Gil, Y., Ratnakar, V., Deelman, E., Mehta, G. et Kim, J. (2007). Wings for pegasus : Creating large-scale scientific applications using semantic representations of computational workflows. Dans *Proceedings of the National Conference on Artificial Intelligence*, volume 22, p. 1767. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999.

Goble, C. A., Bhagat, J., Aleksejevs, S., Cruickshank, D., Michaelides, D., Newman, D., Borkum, M., Bechhofer, S., Roos, M., Li, P. *et al.* (2010). myexperiment : a repository and social network for the sharing of bioinformatics workflows. *Nucleic acids research*, *38*(suppl_2), W677–W682.

Golub, G. H. et Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische mathematik*, *14*(5), 403–420.

Gonzalez, G. H., Tahsin, T., Goodale, B. C., Greene, A. C. et Greene, C. S. (2016). Recent advances and emerging applications in text and data mining for biomedical discovery. *Briefings in Bioinformatics*, *17*(1), 33–42. `http://dx.doi.org/10.1093/bib/bbv087`

Gordon, A. S. (1998). *The design of knowledge-rich browsing interfaces for retrieval in digital libraries.* (Thèse de doctorat). Northwestern University.

Gordon, P. M. et Sensen, C. W. (2007). Seahawk : moving beyond html in web-based bioinformatics analysis. *BMC bioinformatics*, *8*(1), 208.

Grishman, R. (2012). Information extraction. In *The Oxford handbook of computational linguistics.* Oxford University Press.

Guindon, S., Dufayard, J.-F., Lefort, V., Anisimova, M., Hordijk, W. et Gascuel, O. (2010). New algorithms and methods to estimate maximum-likelihood phylogenies : assessing the performance of phyml 3.0. *Systematic biology*, *59*(3), 307–321.

Guler, A. T., Waaijer, C. J. et Palmblad, M. (2016). Scientific workflows for bibliometrics. *Scientometrics*, *107*(2), 385–398.

Halioui, A., Martin, T., Valtchev, P. et Diallo, A. B. (2017a). Ontology-based workflow pattern mining : Application to bioinformatics expertise acquisition. Dans *Proceedings of the Symposium on Applied Computing*, SAC '17, 824–827., New York, NY, USA. ACM. http://dx.doi.org/10.1145/3019612.3019866. Récupéré de http://doi.acm.org/10.1145/3019612.3019866

Halioui, A., Valtchev, P. et Diallo, A. B. (2015). Acquisition of generic problem solving knowledge through information extraction and pattern mining. Dans *Tools with Artificial Intelligence, 2015 IEEE 27th International Conference on*, 583–590. http://dx.doi.org/10.1109/ICTAI.2015.90

Halioui, A., Valtchev, P. et Diallo, A. B. (2017b). T-gowler : Discovering generalized process models within texts. *Journal of Computational Biology*, *24*(8), 799–808.

Han, J. et Fu, A. (1999). Mining multiple-level association rules in large databases. *Knowledge and Data Engineering, IEEE Transactions on*, *11*(5), 798–805.

Han, J. et Fu, Y. (1995). Discovery of multiple-level association rules from large databases. Dans *Very Large Data Bases (VLDB)*, volume 95, 420–431.

Hart, A. (1988). *Knowledge acquisition for expert systems*. Springer.

Hashmi, N., Lee, S. et Cummings, M. P. (2012). Abstracting Workflows : Unifying Bioinformatics Task Conceptualization and Specification Through Semantic Web Services. Dans *WC Workshop on Semantic Web for Life Sciences*, 1–7., Cambridge, Massachusetts USA.

Hilario, M., Nguyen, P., Do, H., Woznica, A. et Kalousis, A. (2011). Ontology-Based Meta-Mining of Knowledge Discovery Workflows. In N. Jankowski, W. Duch, et K. Grabczewski (dir.), *Meta-Learning in Computational Intelligence*, volume 358 de *Studies in Computational Intelligence* 273–315. Springer.

Hu, Y., Koren, Y. et Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. Dans *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, 263–272., Washington, DC, USA. IEEE Computer Society. http://dx.doi.org/10.1109/ICDM.2008.22. Récupéré de http://dx.doi.org/10.1109/ICDM.2008.22

Inokuchi, A. (2004). Mining generalized substructures from a set of labeled graphs. Dans *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04)*, 415–418. http://dx.doi.org/10.1109/ICDM.2004.10041

Inokuchi, A., Washio, T., Nishimura, K. et Motoda, H. (2002). *Research Report.* Rapport technique, IBM Research, Tokyo Research Laboratory.

Ison, J., Kalas, M., Jonassen, I., Bolser, D., Uludag, M., McWilliam, H., Malone, J., Lopez, R., Pettifer, S. et Rice, P. (2013). EDAM : an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics*, *29*(10), 1325–1332.

Jiang, T., Tan, A.-H., Senior Member, I. et Wang, K. (2007). Mining Generalized Associations of Semantic Relations from Textual Web Content. *IEEE Transactions on Knowledge and Data Engineering*, *19*(2), 164–179. `http://dx.doi.org/http://dx.doi.org/10.1109/TKDE.2007.36`

Jones, D. T., Taylor, W. R. et Thornton, J. M. (1992). The rapid generation of mutation data matrices from protein sequences. *Computer applications in the biosciences : CABIOS*, *8*(3), 275–282.

Kahney, H. (1986). *Problem solving : A cognitive approach.* Open University Press.

Kambhatla, N. (2004). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. Dans *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, p. 22. Association for Computational Linguistics.

Kazama, J., Makino, T., Ohta, Y. et Tsujii, J. (2002). Tuning support vector machines for biomedical named entity recognition. Dans *Proceedings of the ACL-02 workshop on Natural language processing in the biomedical domain-Volume 3*, 1–8. Association for Computational Linguistics.

Kiefer, F., Arnold, K., Künzli, M., Bordoli, L. et Schwede, T. (2009). The swiss-model repository and associated resources. *Nucleic acids research*, *37*(suppl 1), D387–D392.

Kim, J.-D., Ohta, T., Tateisi, Y. et Tsujii, J. (2003). Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, *19*(suppl 1), i180–i182.

Kumar, S., Stecher, G. et Tamura, K. (2016). Mega7 : Molecular evolutionary genetics analysis version 7.0 for bigger datasets. *Molecular biology and evolution*, p. msw054.

Kuramochi, M. et Karypis, G. (2001). Frequent Subgraph Discovery. Dans *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM)*, 313–320. IEEE Computer Society. Récupéré de `http://dl.acm.org/citation.cfm?id=645496.658027`

Kuutti, K. (1996). Activity theory as a potential framework for human-computer interaction research. *Context and consciousness : Activity theory and human-computer interaction*, 17–44.

Lau, T. A., Drews, C. et Nichols, J. (2009). Interpreting written how-to instructions. Dans *International Joint Conference on Artificial Intelligence (IJCAI)*, 1433–1438.

Leaman, R., Wei, C. H. et Lu, Z. (2015). TmChem : A high performance approach for chemical named entity recognition and normalization. *Journal of Cheminformatics*, *7*(Suppl 1), S3. `http://dx.doi.org/10.1186/1758-2946-7-S1-S3`. Récupéré de `http://www.jcheminf.com/content/7/S1/S3`

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, S., Wang, T. D., Hashmi, N. et Cummings, M. P. (2007). Bio-STEER : A Semantic Web workflow tool for Grid computing in the life sciences. *Future Generation Computer Systems*, *23*(3), 497–509.

Leigh, J. W., Lapointe, F. J., Lopez, P. et Bapteste, E. (2011). Evaluating phylogenetic congruence in the post-genomic era. *Genome Biology and Evolution*, *3*, 571–587. `http://dx.doi.org/10.1093/gbe/evr050`

Leipzig, J. (2016). A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, p. bbw020.

Lemey Philippe, Salemi Marco, V. A.-M. (2009). *The phylogenetic handbook : a practical approach to phylogenetic analysis and hypothesis testing*. Cambridge University Press. `http://dx.doi.org/10.1002/ajhb.20017`. Récupéré de `www.cambridge.org/9780521877107`

Letovsky, S. I. (2006). *Bioinformatics : databases and systems*. Springer Science & Business Media.

Levy, O. et Goldberg, Y. (2014). Dependency-Based Word Embeddings. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 302–308. `http://dx.doi.org/10.3115/v1/P14-2050`. Récupéré de `http://aclWeb.org/anthology/P14-2050`

Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A. et Lee, B.-S. (2012). Twiner : named entity recognition in targeted twitter stream. Dans *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, 721–730. ACM.

Li, L., Guo, R., Jiang, Z. et Huang, D. (2015). An approach to improve kernel-based protein–protein interaction extraction by learning from large-scale network data. *Methods*, *83*, 44 – 50. Network-based Approaches to the Analysis of Omics Data, http://dx.doi.org/http://dx.doi.org/10.1016/j.ymeth.2015.03.026. Récupéré de http://www.sciencedirect.com/science/article/pii/S1046202315001437

Li, Y. et Bontcheva, K. (2007). Hierarchical, perceptron-like learning for ontology-based information extraction. Dans *Proceedings of the 16th International Conference on World Wide Web*, 777–786. ACM.

Li, Y., Bontcheva, K. et Cunningham, H. (2005). Using uneven margins svm and perceptron for information extraction. Dans *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, 72–79., Stroudsburg, PA, USA. Association for Computational Linguistics.

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J. et Kandola, J. (2002). The perceptron algorithm with uneven margins. Dans *In In Proceedings of the Conference International Conference on Machine Learning (ICML)*, volume 2, 379–386.

Liang, L., Xu, B., Chen, Y., Liu, Y., Cao, W., Fang, L., Feng, L., Goodchild, M. F. et Gong, P. (2010). Combining spatial-temporal and phylogenetic analysis approaches for improved understanding on global h5n1 transmission. *PLoS One*, *5*(10), e13575.

Lin, D. et Pantel, P. (2001). Discovery of inference rules for question answering. *Natural Language Engineering*, *7*, 343–360.

Liu, S., Tang, B., Chen, Q. et Wang, X. (2015). Effects of semantic features on machine learning-based drug name recognition systems : Word embeddings vs. manually constructed dictionaries. *Information*, *6*(4), 848–865. http://dx.doi.org/10.3390/info6040848

Liu, Y. et Agah, A. (2009). Crawling and extracting process data from the web. Dans *International Conference on Advanced Data Mining and Applications*, 545–552. Springer.

Lord, E., Diallo, A. B. et Makarenkov, V. (2015). Classification of bioinformatics workflows using weighted versions of partitioning and hierarchical clustering algorithms. *BMC bioinformatics*, *16*(1), 68.

Lord, E., Leclercq, M., Boc, A., Diallo, A. B. et Makarenkov, V. (2012). Armadillo 1.1 : An Original Workflow Platform for Designing and Conducting Phylogenetic Analysis and Simulations. *PloS One*, *7*(1), e29903.

Magnini, B. et Strapparava, C. (2001). *Improving user modelling with content-based techniques*, volume 2109. http://dx.doi.org/10.1007/3-540-44566-8_8. Récupéré de http://link.springer.com/10.1007/3-540-44566-8{_}8

Malone, J., Brown, A., Lister, A. L., Ison, J., Hull, D., Parkinson, H. et Stevens, R. (2014). The software ontology (swo) : a resource for reproducibility in biomedical data analysis, curation and digital preservation. *Journal of Biomedical Semantics*, *5*(1), 1.

Mikolov, T., Chen, K., Corrado, G. et Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv :1301.3781*, 1–12. http://dx.doi.org/10.1162/153244303322533223. Récupéré de http://arxiv.org/abs/1301.3781

Miller, G. A. (1995). Wordnet : a lexical database for english. *Communications of the ACM*, *38*(11), 39–41.

Minor, M. et Montani, S. (2012). Process-oriented case-based reasoning. *Information Systems*, p. 9.

Mitchell, K. J., Becich, M. J., Berman, J. J., Chapman, W. W., Gilbertson, J., Gupta, D., Harrison, J., Legowski, E. et Crowley, R. S. (2004). Implementation and evaluation of a negation tagger in a pipeline-based system for information extraction from pathology reports. *Medinfo*, *2004*, 663–7.

Mitkov, R. (2014). *Anaphora resolution*. Routledge.

Müller, G. et Bergmann, R. (2014). Workflow streams : A means for compositional adaptation in process-oriented cbr. In *Case-Based Reasoning Research and Development* 315–329. Springer.

Myaeng, S.-H., Jeong, Y., Jung, Y. *et al.* (2013). Experiential knowledge mining. *Foundations and Trends in Web Science*, *4*(1), 1–102.

Nadeau, D. et Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Najafabadi, M. K. et Mahrin, M. N. (2016). A systematic literature review on the state of research and practice of collaborative filtering technique and implicit feedback. *Artificial Intelligence Review*, *45*(2), 167–201. http://dx.doi.org/10.1007/s10462-015-9443-9. Récupéré de http://dx.doi.org/10.1007/s10462-015-9443-9

Nguyen, P., Hilario, M. et Kalousis, A. (2014). Using meta-mining to support data mining workflow planning and optimization. *Journal of Artificial Intelligence Research*, *51*, 605–644.

Nguyen, T. H. et Grishman, R. (2015). Event Detection and Domain Adaptation with Convolutional Neural Networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2 : Short Papers)*, 365–371. Récupéré de http://www.aclWeb.org/anthology/P15-2060

Notredame, C., Higgins, D. G. et Heringa, J. (2000). T-coffee : A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, *302*(1), 205–217.

Peterson, J. L. (1981). Petri net theory and the modeling of systems.

Philippe, H., Brinkmann, H., Lavrov, D. V., Littlewood, D. T. J., Manuel, M., Worheide, G. et Baurain, D. (2011). Resolving difficult phylogenetic questions : Why more sequences are not enough. *PLoS Biology*, *9*. http://dx.doi.org/10.1371/journal.pbio.1000602

Plantevit, M., Laurent, A. et Teisseire, M. (2006). HYPE : mining hierarchical sequential patterns. Dans *Proceedings of the 9th ACM International workshop on Data warehousing and OLAP (DOLAP)*. ACM Request Permissions.

Plotree, D. et Plotgram, D. (1989). Phylip-phylogeny inference package (version 3.2). *Cladistics*, *5*(163), 6.

Poesio, M., Barbu, E., Giuliano, C. et Romano, L. (2008). Supervised relation extraction for ontology learning from text based on a cognitively plausible model of relations. Dans *ECAI 2008 3rd Workshop on Ontology Learning and Population*, 1–5.

Poon, H., Toutanova, K. et Quirk, C. (2015). Distant supervision for cancer pathway extraction from text. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, *20*, 120–31. http://dx.doi.org/10.1142/9789814644730_0013. Récupéré de http://www.ncbi.nlm.nih.gov/pubmed/25592574

Posada, D. (2008). jmodeltest : phylogenetic model averaging. *Molecular biology and evolution*, *25*(7), 1253–1256.

Posada, D., Crandall, K. et Templeton, A. (2000). Geodis : a program for the cladistic nested analysis of the geographical distribution of genetic haplotypes. *Molecular Ecology*, *9*(4), 487–488.

Provoost, T. et Moens, M.-F. (2015). Semi-supervised Learning for the BioNLP Gene Regulation Network. *BMC bioinformatics*, *16 Suppl 1*(Suppl 10), S4.

http://dx.doi.org/10.1186/1471-2105-16-S10-S4. Récupéré de http://www.biomedcentral.com/1471-2105/16/S10/S4

Pukkhem, N. (2013). Ontology-based Semantic Approach for Learning Object Recommendation. *International Journal on Information Technology*, *3*(4), 12. Récupéré de http://hal.archives-ouvertes.fr/hal-00942526/

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Ríos, J., Karlsson, J. et Trelles, O. (2009). Magallanes : a web services discovery and automatic workflow composition tool. *BMC bioinformatics*, *10*(1), 1.

Robinson, D. F. et Foulds, L. R. (1981). Comparison of phylogenetic trees. *Mathematical biosciences*, *53*(1-2), 131–147.

Rodríguez-Muro, M. et Calvanese, D. (2012). Quest, an owl 2 ql reasoner for ontology-based data access. Dans *In Proceedings of Experiences and Directions Workshop (OWLED)*. Citeseer.

Ronquist, F. (2003). Parsimony analysis of coevolving species associations. *Tangled trees : phylogeny, cospeciation and coevolution*, 22–64.

Ronquist, F. et Huelsenbeck, J. P. (2003). Mrbayes 3 : Bayesian phylogenetic inference under mixed models. *Bioinformatics*, *19*(12), 1572–1574.

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Rousse, J. T. (2013). From novice to expert : Problem solving in icd-10-pcs procedural coding. *Perspectives in health information management/AHIMA, American Health Information Management Association*, *10*(Summer).

Ryu, J., Ihm, H. et Myaeng, S.-H. (2013). Automatic organization of human task goals for web-scale problem solving knowledge. Dans *Proceedings of the seventh International Conference on Knowledge capture*, 81–88. ACM.

Sahdra, B. et Thagard, P. (2003). Procedural knowledge in molecular biology. *Philosophical Psychology*, *16*(4), 477–498.

Saldanha, A. J. (2004). Java treeview—extensible visualization of microarray data. *Bioinformatics*, *20*(17), 3246–3248.

Salton, G. et Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Saltz, J., Oster, S., Hastings, S., Langella, S., Kurc, T., Sanchez, W., Kher, M., Manisundaram, A., Shanbhag, K. et Covitz, P. (2006). cagrid : design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, *22*(15), 1910–1916.

Schank, R. C. et Abelson, R. P. (2013). *Scripts, plans, goals, and understanding : An inquiry into human knowledge structures*. Psychology Press.

Schumacher, P., Minor, M. et Schulte-Zurhausen, E. (2013). Extracting and enriching workflows from text. Dans *14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14-16, 2013*, 285–292. http://dx.doi.org/10.1109/IRI.2013.6642484. Récupéré de http://dx.doi.org/10.1109/IRI.2013.6642484

Schütze, H. (2008). Introduction to information retrieval. Dans *Proceedings of the International communication of association for computing machinery Conference*.

Šebek, M., Hlosta, M., Zendulka, J. et Hruška, T. (2012). Multi-level sequence mining based on gsp. *Acta Electrotechnica et Informatica*, *12*(2), 31–38. http://dx.doi.org/10.2478/v10198-012-0012-8. Récupéré de http://www.degruyter.com/view/j/aeei.2012.12.issue-2/v10198-012-0012-8/v10198-012-0012-8.xml

Sekine, S. et Nobata, C. (2004). Definition, dictionaries and tagger for extended named entity hierarchy. Dans *The International Conference on Language Resources and Evaluation (LREC)*, 1977–1980.

Semeraro, G., Basile, P., de Gemmis, M. et Lops, P. (2009). User profiles for personalizing digital libraries. In *Handbook of Research on Digital Libraries : Design, Development, and Impact* 149–158. IGI Global.

Senkul, P. et Salin, S. (2012). Improving pattern quality in web usage mining by using semantic information. *Knowledge and Information Systems*, *30*(3), 527–541. http://dx.doi.org/10.1007/s10115-011-0386-4

Settles, B. (2004). Biomedical named entity recognition using conditional random fields and rich feature sets. *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications - JNLPBA '04*, p. 104. http://dx.doi.org/10.3115/1567594.1567618. Récupéré de http://portal.acm.org/citation.cfm?doid=1567594.1567618

Shani, G. et Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook* 257–297. Springer.

Shao, B., Wang, H. et Li, Y. (2013). Trinity : A distributed graph engine on a memory cloud. Dans *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 505–516. ACM.

Singh, K. (2016). Graph Pattern Matching : A Brief Survey of Challenges and Research Directions. 204–209. http://dx.doi.org/10.13140/RG.2.1.1402.6644

Smith, L., Rindflesch, T. et Wilbur, W. J. (2004). MedPost : a part-of-speech tagger for bioMedical text. *Bioinformatics*, *20*(14), 2320–2321. http://dx.doi.org/10.1093/bioinformatics/bth227. Récupéré de http://dx.doi.org/10.1093/bioinformatics/bth227

Song, S.-k., Oh, H.-S., Myaeng, S. H., Choi, S.-P., Chun, H.-W., Choi, Y.-S. et Jeong, C.-H. (2011). Procedural knowledge extraction on medline abstracts. Dans *International Conference on Active Media Technology*, 345–354. Springer.

Spjuth, O., Bongcam-Rudloff, E., Hernández, G. C., Forer, L., Giovacchini, M., Guimera, R. V., Kallio, A., Korpelainen, E., Kańduła, M. M., Krachunov, M., Kreil, D. P., Kulev, O., Łabaj, P. P., Lampa, S., Pireddu, L., Schönherr, S., Siretskiy, A. et Vassilev, D. (2015). Experiences with workflows for automating data-intensive bioinformatics. *Biology Direct*, 1–12.

Srikant, R. et Agrawal, R. (1995). Mining generalized association rules. Dans *Proc. of the 21th International Conference on Very Large (VLDB)*.

Srikant, R. et Agrawal, R. (1996). Mining sequential patterns : Generalizations and performance improvements. Dans *Proceedings 5th International Conference Extending Database Technology, EDBT, Avigon, France*, volume 1057 de *LNCS*, 3–17. Springer.

Srivastava, J., Cooley, R., Deshpande, M. et Tan, P. N. (2000). Web usage Mining : Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, *1*(2), 12–23. http://dx.doi.org/10.1145/846183.846188. Récupéré de http://portal.acm.org/citation.cfm?id=846188{&}dl=GUIDE,{&}referer=www.clickfind.com.au

Staab, S., Studer, R., Guarino, N. et Oberle, D. (2010). *Handbook on Ontologies*, volume 2. Springer. http://dx.doi.org/10.1007/978-3-540-92673-3. Récupéré de http://www.springerlink.com/index/10.1007/978-3-540-92673-3

Starlinger, J., Brancotte, B., Cohen-Boulakia, S. et Leser, U. (2014). Similarity search for scientific workflows. *Proceedings of the VLDB Endowment*, *7*(12), 1143–1154.

Sternberg, R. J. et Ben-Zeev, T. (2001). *Complex cognition : The psychology of human thought.* Oxford University Press.

Stevens, R., Zhao, J. et Goble, C. (2007). Using provenance to manage knowledge of In Silico experiments. *Briefings in Bioinformatics*, *8*, 183–194. http://dx.doi.org/10.1093/bib/bbm015

Stevens, R. D., Robinson, A. J. et Goble, C. A. (2003). mygrid : personalised bioinformatics on the information grid. *Bioinformatics*, *19*(suppl 1), i302–i304.

Stevenson, M. et Wilks, Y. (2003). Word sense disambiguation. *Oxford Handbook of Computational Linguistics*, 249–265.

Stumme, G., Hotho, A. et Berendt, B. (2006). Semantic web mining : State of the art and future directions. *Web Semantics : Science, Services and Agents on the World Wide Web*, *4*(2), 124–143.

Sun, a., Grishman, R. et Sekine, S. (2011). Semi-supervised relation extraction with large-scale word clustering. *Proceedings of the 49th Annual Meeting ...*, 521–529. Récupéré de http://www.aaai.org/Papers/AAAI/2007/AAAI07-224.pdf{%}5Cnhttp://dl.acm.org/citation.cfm?id=2002539

Svatek, V. *et al.* (2006). Ontology-enhanced association mining. In M. e. a. Ackermann (dir.), *Semantics, Web and Mining*, volume 4289 163–179. Springer Berlin / Heidelberg

Tamura, K., Peterson, D., Peterson, N., Stecher, G., Nei, M. et Kumar, S. (2011). Mega5 : molecular evolutionary genetics analysis using maximum likelihood, evolutionary distance, and maximum parsimony methods. *Molecular biology and evolution*, *28*(10), 2731–2739.

Tan, W., Zhang, J., Madduri, R., Foster, I. et De Roure, D. (2011). Providing map and gps assistance to service composition in bioinformatics. Dans *Services Computing (SCC), 2011 IEEE International Conference on*, 608–615. IEEE.

Tarus, J. K., Niu, Z. et Mustafa, G. (2017). Knowledge-based recommendation : a review of ontology-based recommender systems for e-learning. *Artificial Intelligence Review*. http://dx.doi.org/10.1007/s10462-017-9539-5. Récupéré de http://link.springer.com/10.1007/s10462-017-9539-5

Tatusova, T. A. et Madden, T. L. (1999). Blast 2 sequences, a new tool for comparing protein and nucleotide sequences. *FEMS microbiology letters*, *174*(2), 247–250.

Taylor, I. J., Deelman, E., Gannon, D. B. et Shields, M. (2014). *Workflows for e-Science : scientific workflows for grids*. Springer Publishing Company, Incorporated.

Tenorth, M., Nyga, D. et Beetz, M. (2010). Understanding and executing instructions for everyday manipulation tasks from the world wide web. Dans *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 1486–1491. IEEE.

Thompson, J. D., Gibson, T., Higgins, D. G. *et al.* (2002). Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, 2–3.

Toutanova, K., Klein, D., Manning, C. D. et Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. Dans *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, 173–180. Association for Computational Linguistics.

Tsoumakas, G., Katakis, I. et Vlahavas, I. (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook* 667–685. Springer.

Tymoshenko, K., Bonadiman, D. et Moschitti, A. (2016). Convolutional neural networks vs. convolution kernels : Feature engineering for answer sentence reranking. Dans *Proceedings of NAACL-HLT*, 1268–1278.

Van der Aalst, W. M. (2011). *Process Mining : Discovery, Conformance and Enhancement of Business Processes* (1st éd.). Springer Publishing Company, Incorporated.

Van der Aalst, W. M. (2016). Process mining software. In *Process Mining* 325–352. Springer.

Vesin, B., Ivanović, M., Klašnja-Milićević, A. et Budimac, Z. (2012). Protus 2.0 : Ontology-based semantic recommendation in programming tutoring system. *Expert Systems with Applications*, *39*(15), 12229–12246. `http://dx.doi.org/10.1016/j.eswa.2012.04.052`

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, *13*(2), 260–269.

Wang, J., Han, J., Lu, Y. et Tzvetkov, P. (2005). Tfp : an efficient algorithm for mining top-k frequent closed itemsets. *Knowledge and Data Engineering, IEEE Transactions on*, *17*(5), 652–663. `http://dx.doi.org/10.1109/TKDE.2005.81`

Werth, T., Wörlein, M., Dreweke, A., Fischer, I. et Philippsen, M. (2009). DAG Mining for Code Compaction. In *Data Mining for Business Applications* 209–223. Boston, MA : Springer US

Wilcoxon, F., Katti, S. et Wilcox, R. A. (1963). *Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test.* American Cyanamid Company.

Wilkinson, M. (2006). Gbrowse moby : a web-based browser for biomoby services. *Source Code for Biology and Medicine*, *1*(1), 1.

Wilkinson, M. D. et Links, M. (2002). Biomoby : an open source biological web services proposal. *Briefings in bioinformatics*, *3*(4), 331–341.

Wilkinson, M. D., Vandervalk, B. et McCarthy, L. (2009). Sadi web-services, cause you can't always get what you want ! Dans *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, 13–18. IEEE.

Withers, D., Kawas, E., McCarthy, L., Vandervalk, B. et Wilkinson, M. (2010). Semantically-guided workflow construction in taverna : the sadi and biomoby plug-ins. Dans *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, 301–312. Springer.

Witten, I. H., Frank, E., Hall, M. A. et Pal, C. J. (2016). *Data Mining : Practical machine learning tools and techniques.* Morgan Kaufmann.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S. et Goble, C. (2013). The taverna workflow suite : designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, *41*(W1), W557. `http://dx.doi.org/10.1093/nar/gkt328`. Récupéré de `+http://dx.doi.org/10.1093/nar/gkt328`

Xia, X. et Xie, Z. (2001). Dambe : software package for data analysis in molecular biology and evolution. *Journal of Heredity*, *92*(4), 371–373.

Xu, Y., Hong, K., Tsujii, J. et Chang, E. I.-C. (2012). Feature engineering combined with machine learning and rule-based methods for structured information extraction from narrative clinical discharge summaries. *Journal of the American Medical Informatics Association*, *19*(5), 824–32. `http://dx.doi.org/10.1136/amiajnl-2011-000776`

Yan, X. et Han, J. (2002). gSpan : Graph-Based Substructure Pattern Mining. Dans *Proceedings of IEEE International Conference on Data Mining*,

volume 1, 721–724. http://dx.doi.org/10.1109/ICDM.2002.1184038. Récupéré de http://dl.acm.org/citation.cfm?id=844380.844811{%}5Cnhttp://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1184038

Yan, X., Han, J. et Afshar, R. (2003). Clospan : Mining closed sequential patterns in large datasets. Dans *Proceedings of the 2003 SIAM international conference on data mining*, 166–177. SIAM.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information retrieval*, *1*(1-2), 69–90.

Yang, Z. (1994). Estimating the pattern of nucleotide substitution. *Journal of molecular evolution*, *39*(1), 105–111.

Yao, J., Tan, W., Nepal, S., Chen, S., Zhang, J., De Roure, D. et Goble, C. (2012a). Reputationnet : A reputation engine to enhance servicemap by recommending trusted services. Dans *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, 454–461. IEEE.

Yao, L., Riedel, S. et McCallum, A. (2012b). Unsupervised relation discovery with sense disambiguation. Dans *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics : Long Papers - Volume 1*, ACL '12, 712–720., Stroudsburg, PA, USA. Association for Computational Linguistics.

Yu, J. et Buyya, R. (2005). A taxonomy of scientific workflow systems for Grid computing . *SIGMOD Record*, *34*, 1–6.

Zaki, M. J. (2005). Efficiently Mining Frequent Trees in a Forest : Algorithms and Applications. *IEEE Transactions on Knowledge and Data Engineering*, *17*(8), 1021–1035. http://dx.doi.org/10.1109/TKDE.2005.125

Zaki, M. J. et Hsiao, C.-J. (2002). Charm : An efficient algorithm for closed itemset mining. Dans *Proceedings of the 2002 SIAM international conference on data mining*, volume 2, 457–473. SIAM.

Zelenko, D., Aone, C. et Richardella, A. (2003). Kernel methods for relation extraction. *Journal of machine learning research*, *3*(Feb), 1083–1106.

Zempleni, M.-Z., Renken, R., Hoeks, J. C., Hoogduin, J. M. et Stowe, L. A. (2007). Semantic ambiguity processing in sentence context : Evidence from event-related fmri. *Neuroimage*, *34*(3), 1270–1279.

Zhang, K. et Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, *18*(6), 1245–1262.

Zhou, G. et Su, J. (2002). Named entity recognition using an hmm-based chunk tagger. Dans *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 473–480. Association for Computational Linguistics.

Zhou, G., Su, J., Zhang, J. et Zhang, M. (2005). Exploring various knowledge in relation extraction. Dans K. Knight, H. T. Ng, et K. Oflazer (dir.). *Proceedings of the 43rd annual meeting on association for computational linguistics*. The Association for Computer Linguistics. Récupéré de `http://dblp.uni-trier.de/db/conf/acl/acl2005.html#ZhouSZZ05`

Zhu, H. (2005). *Software Design Methodology : From Principles to Architectural Styles*. Elsevier Science. Récupéré de `https://books.google.ca/books?id=rqRVbb0SKjEC`