UNIVERSITE DU QUEBEC A MONTREAL

CONCEPTION ET DEVELOPPEMENT D'UN MODELE D'EVALUATION DES CONTRIBUTIONS DES CHERCHEURS DANS LES APPLICATIONS WIKI

MEMOIRE

PRESENTE

COMME EXIGENCE PARTIELLE

DE LA MAITRISE EN INFORMATIQUES

PAR

ABDERRAZAK LANDOULSI

SEPTEMBRE 2015

REMERCIEMENTS

Je remercie vivement Monsieur Normand Séguin, Vice doyen aux études à la faculté des sciences et professeurs à l'UQAM, qui m'a offert cette opportunité de réaliser mon projet de mémoire. Pour ses orientations, conseils et remarques, et de la qualité de nos rapports professionnels. Je suis reconnaissant de tout ce qu'il m'a appris. Qu'il trouve ici l'expression de mes profondes gratitudes et reconnaissance.

Je tiens à remercier mes parents et ma grande sœur, pour leur soutien morale et bien sûr financier. Ils m'ont sans cesse soutenu. Je tiens à rendre hommage à mes amis Caroline Bérubé, Haythem Mansour et Mohamed Yassine Riahi pour leurs qualités humaines et leur soutien.

Je remercie aussi toute l'équipe du groupe de recherche GRISOU, pour l'ambiance chaleureuse et conviviale qu'ils ont toujours maintenue pendant les réunions.

Enfin, je tiens à remercier vivement toutes les personnes qui de près ou de loin ont contribué au bon déroulement de ce projet. Qu'elles trouvent ici l'expression de mes profondes gratitudes.

MERCI à TOUS.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUME

L'objectif principal des environnements de travail collaboratif est d'assurer une bonne maintenance du contenu de façon à permettre son évolution d'une façon cohérente. L'utilisation de ces environnements est de plus en plus répandue. En effet, l'avancement technologique et le progrès réalisé dans plusieurs domaines touchent le travail collaboratif et la vision que nous avons sur ce dernier. Un grand nombre de chercheurs se sont intéressés au problème de mesure de la contribution des utilisateurs dans ce genre d'environnement de travail. Par exemple, la mesure de la contribution peut être utilisée pour diviser les revenus, ainsi que pour reconnaître les mérites, pour attribuer des promotions, ou bien pour le choix de l'ordre des auteurs quand on emploie des citations. Parmi les environnements de travail collaboratifs les plus connus, citons Wikipédia. Ce dernier représente un environnement de travail collaboratif ouvert pour un très grand nombre d'utilisateurs (auteurs). Il offre aussi plusieurs outils de maintenance et de suivi des versions des articles créés.

Dans ce travail, nous proposerons un modèle d'évaluation des contributions basé sur l'historique des changements que subit l'article pendant son cycle de vie. Cette approche est inspirée de plusieurs modèles identifiés dans les travaux précédents. Ainsi, le principe de notre modèle se construit autour de l'identification des opérations des auteurs. Cette identification nous permet de voir leurs actions. Nous incluons un mécanisme de suivi de la propriété des mots nous permettant d'identifier ce qui a été retenu dans la dernière version d'un article. Nous effectuons des mesures de quantité et de la qualité afin de donner un score global aux contributions des auteurs.

Mots clés : Wikipédia, contribution, algorithme, environnement collaboratif.

ABSTRACT

The main aim of collaborative working environments is to ensure proper maintenance of content to allow its development in a consistent manner. The use of these environments is becoming increasingly widespread. Indeed, technological advancement and progress in several areas of the collaborative work is affecting the vision we have on it. Many researchers have studied the problem of measuring the contribution of users in this type of work environment. For example, the extent of the contribution can be used to divide revenues and to recognize the merits, to award promotions, or to the choice of the order of authors when it uses quotes. Among the collaborative work environments, one of the best known is Wikipedia. This is a collaborative work environment open to a very large number of users (authors). It also offers several maintenance tools and versioning articles created.

In this work, we propose a contribution assessment model based on historical changes undergone by the article during its life cycle. This approach is inspired by several models identified in previous work. Thus, the principle of our model is built around identifying the perpetrators of operations. This identification allows us to see their actions. We include a monitoring mechanism of the property words to allow us to identify what has been retained in the final version of an article. We conduct quantitative and qualitative measures in order to give an overall score for authors of contributions.

Keywords: Wikipedia, contribution algorithm, collaborative environment.

CHAPITRE I

INTRODUCTION

Le progrès du web nous a fourni de nouvelles formes de collaboration et d'interaction facilitant la manipulation de l'information (ainsi que les connaissances). En effet, la collaboration en ligne est rapidement devenue l'une des méthodes primaires dans laquelle le contenu et l'information sont créés et partagés. Parmi les plateformes de collaboration en ligne, on trouve les applications wikis. Les wikis sont adoptés dans différents contextes, tels que l'éducation, la recherche et les affaires. Ces applications ont un potentiel déterminant pour simplifier la création, le partage, l'intégration et l'utilisation des connaissances. La plus connue de ces applications wikis est Wikipédia. C'est la destination première des *surfeurs* à la recherche de connaissances et l'un des sites web les plus visités sur le réseau mondial. Elle représente une encyclopédie en ligne, tous ces articles étant créés par un grand nombre d'auteurs bénévoles. Aussi, elle offre la possibilité à tous de modifier ou de créer des articles. Les applications wikis sont supportées par un mécanisme de révision qui permet le suivi des changements et incidemment, les contributions positives.

Les chercheurs se sont de plus en plus intéressés au perfectionnement des composantes de ce genre d'application pour deux raisons principales. D'une part, pour assurer la qualité du contenu et d'autre part, pour identifier le résultat des efforts des auteurs. L'identification de cet effort revient à évaluer leur contribution. Avoir un

outil permettant l'évaluation de la contribution peut être utilisé comme un outil d'aide à la décision afin d'attribuer les promotions et les honneurs (comme barnstar pour Wikipédia). De surcroît, il peut aider pour partager les revenus, pour reconnaître les mérites ou pour choisir l'ordre des auteurs dans une citation.

Également, la gestion des privilèges des auteurs par rapport à l'édition d'un article permet de protéger des articles spécifiques, et ceci grâce à un système d'alerte. Par exemple, s'il y a eu une intervention d'un auteur ayant un faible score, il s'en suivra un signalement pour les auteurs privilégiés afin qu'ils puissent valider cette intervention.

La présence d'un outil d'évaluation de la contribution dans un environnement de travail collaboratif permet d'une façon indirecte à promouvoir l'utilisation de ce genre d'environnement, ainsi qu'à encourager les auteurs à fournir des contributions utiles afin que le contenu de qualité soit prédominant.

Toutefois, ces utilités dépendront toujours des approches employées pour évaluer les contributions des auteurs. Il existe une variété d'approches, chacune ayant une vision différente sur la modélisation de la contribution, qui elle-même est fortement liée à l'identification du score. De ce fait, nous avons décidé de concevoir un modèle d'évaluation permettant d'attribuer les scores mérités aux auteurs.

1.1 Problématique

Les wikis ont été créés pour supporter le travail collaboratif sans se concentrer sur les contributions individuelles. En effet, dans ce genre d'application, toutes les versions des articles sont sauvegardées, mais l'information sur la nature des opérations réalisées par les utilisateurs est dissimulée. C'est en analysant toutes les versions que

nous pouvons identifier ces opérations et par la suite, essayer d'évaluer ces contributions.

Le problème de l'évaluation de la contribution dans les applications wikis n'est pas facile à résoudre. En effet, le nombre conséquent de révisions implique une grande quantité d'information à traiter et par la suite la tâche d'analyse est très lourde en termes de temps de calcul. De plus, l'identification des contributions est très complexe à programmer. Autrement dit, il est difficile pour une machine à étiqueter des actions telles que (la mise en forme, la correction des fautes de grammaire et l'amélioration de la navigation, etc.). D'où le défi d'avoir un outil permettant d'extraire ce genre d'actions abstraites et non pas basiques tel que (ajout, suppression et déplacement).

Il existe plusieurs modèles différents que l'on présentera dans le Chapitre II. D'abord, ces différentes approches sont liées aux domaines d'applications, qui peuvent être public, tel que Wikipédia, ou privé, tels que les entreprises ayant des départements de recherches. Ensuite, ils sont liés aux intérêts des évaluateurs, qui influencent les choix de mesures. Par exemple, certains s'intéressent à l'analyse du réseau social engendré par ce genre d'applications, alors que d'autres analysent les pages de discussions, ou analysent l'historique des articles afin de traquer toutes les contributions des auteurs.

Notre choix s'est porté sur l'analyse de l'historique des articles, car nous voulons identifier les actions des auteurs et les décortiquer pour pouvoir mieux évaluer leur contribution.

Dans la phase d'identification des actions, nous analysons une grande quantité d'information, ce qui rend la charge de calcul énorme. En effet, la catégorisation des actions n'est pas évidente, car il n'existe pas de règles permettant d'identifier des actions abstraites telles que les corrections des fautes de grammaire, ou bien

l'amélioration de la navigation. Il est donc important de bien définir ce genre de règle pour préciser et faciliter le calcul.

Une fois les actions identifiées, il faut leur attribuer des pondérations afin d'avoir un score global modélisant la contribution de l'auteur. Certes, cette attribution semble arbitraire, car il n'existe pas de règles générales pour modéliser la contribution globale. Il s'agit du défi ultime.

1.2 Objectifs

Face au problème soulevé dans la problématique, nous voulons proposer un modèle pour l'évaluation de la contribution des auteurs dans les applications wikis. L'objectif étant d'attribuer un score à l'auteur. Ce score représente notre estimation de la valeur sa contribution globale. Ainsi, l'unité de mesure est proposée dans un but d'amélioration de l'environnement collectif, puisqu'il permet de comprendre la contribution d'un auteur, par et pour ses pairs (auteurs).

Notre méthode s'inspire de l'approche de (Fong et Biuk-Aghai, 2010). Dans cette approche, ils arrivent à extraire non seulement les opérations primitives, mais aussi un autre type d'opérations qu'ils appellent « opérations abstraites». On trouve une description de cette approche dans notre revue de littérature. Dans ce qui suit, nous allons énoncer notre démarche de recherche.

1.3 Approche

Tout d'abord, nous avons tenté d'étudier quelques concepts liés à notre sujet de recherche, tel que le concept Wiki. Ce dernier représente la base de toutes applications wiki. De plus, nous avons approfondi l'utilisation des techniques de comparaison de texte. En effet, celles-ci seront employées pour pouvoir extraire les

différences entre deux versions d'un même article. Parmi ces techniques, on trouve la distance d'édition (autrement, distance de Levenshtein), le cosinus de similarité entre deux documents et la méthode de pondération TF-IDF (*Term Frequency-Inverse Document Frequency*) qui sont nécessaires au calcul du cosinus de similarité.

Dans notre revue de littérature, nous présenterons les différentes approches pour l'évaluation de la contribution des auteurs. Une fois qu'elles sont analysées, nous voulons exploiter les paramètres et les mesures identifiés afin de créer un modèle plus complet que ses prédécesseurs.

Puis, nous allons définir les entités à mesurer, les modules à implémenter, ainsi qu'une proposition de la formule globale de la contribution. Parmi les modules, nous retrouverons le module de différenciation entre deux versions. Ce dernier permettra de donner une présentation des changements appliqués à une version pour en générer une autre. L'évaluation de la contribution dépend fortement de ce module.

Ensuite, nous allons créer l'environnement de test et de validation. Il inclura l'implémentation de notre approche basée sur l'analyse des changements.

Enfin, nous discuterons des résultats obtenus.

1.4 Plan du mémoire

En plus des chapitres d'introduction et de conclusion, notre mémoire est organisée comme suit :

Dans le chapitre II, nous présentons le cadre conceptuel de notre sujet de recherche. Il inclut des définitions de base pour l'élaboration et la compréhension de notre modèle.

Dans le chapitre III, nous décrivons les approches d'évaluation des contributions identifiées dans notre revue de littérature. Pour chaque approche, nous expliquons son fonctionnement et nous identifions les paramètres et les mesures utilisés.

Dans le chapitre IV, nous définissons notre propre modèle d'évaluation de la contribution.

Dans le chapitre V, nous exposons les résultats obtenus lors de la phase de test et de validation.

CHAPITRE II

CADRE CONCEPTUEL

2.1 Contexte

Les wikis sont des plateformes interactives de diffusion de connaissances qui prennent chaque jour de l'ampleur sur le net. Son concept se veut démocratique et autonome, et le résultat est très probant. L'exemple de Wikipédia et de ses millions de contributions volontaires démontre à lui seul le potentiel immense de ces outils qui sont ouverts à tous.

Un wiki est une application web qui se veut ouverte, parce qu'elle est conçue avec des principes de simplicité et de transparence, et contributive, puisque même les utilisateurs les moins expérimentés ont accès au contenu et peuvent interagir facilement grâce à son interface intuitive, telle que la zone d'édition ou les modules explicatifs. Ces derniers facilitent l'utilisation au grand public, mais valorisent également la contribution des utilisateurs, car ils peuvent créer, ajouter, modifier, améliorer toutes les pages des wikis. Les wikis dépendent en grande partie de ses contributeurs, largement volontaires, pour se construire une base de données qui peut devenir significative socialement (Ebersbach *et al*, 2008).

L'idée générale du wiki est sans ambiguïté: une structure simple et accessible qui permet aux utilisateurs de modifier le contenu et d'y ajouter des informations manquantes tout en profitant de la *rapidité* des opérations sur le résultat final. En

effet, le mot « wikiwiki » provient de la langue hawaiienne. Il signifie « vite » ou « dépêche » (Ebersbach *et al*, 2008).

Les wikis sont devenus un phénomène adopté dans différents organismes à l'échelle planétaire. En effet, ce qui fait la popularité et l'explication générale de l'utilisation et de la contribution au contenu des wikis, c'est tout le concept sous-jacent : la simplicité, l'accessibilité et le partage.

Encore une fois, la simplicité et l'accessibilité sont en grande partie les catalyseurs du succès des wikis. Sa conception intuitive et son ouverture au public ont permis aux wikis de toucher un public plus large que celui des connaisseurs en informatique. Elle est aussi transparente, au sens où les contributeurs savent que leur version ne sera pas perdue, même si quelqu'un la modifie, car elle est restée enregistrée dans les versions précédentes. La contribution des différents auteurs est basée sur la facilité à participer à un tout plus grand et plus important que son simple ajout. Effectivement, la fonction « edit » (modifier) nous permet de changer n'importe quelle page que l'on lit présentement. Cette liberté est ce qui propulse la créativité et la popularité des wikis. La contribution d'une telle diversité d'utilisateurs est bien plus compréhensible quand on se rend compte que beaucoup de gens retirent du plaisir à s'investir dans les réseaux wiki.

Depuis les premiers wikis, les communautés utilisant ce concept n'ont cessé de se multiplier. Aujourd'hui, beaucoup d'applications web s'inspirent du concept de wiki et y ont ajouté de nouvelles fonctions. À titre d'exemple de wiki on trouve (Ebersbach *et al*, 2008):

- UseModWiki;
- MediaWiki;
- PmWiki;
- ProWiki.

2.1.1 Technologie

Il s'agit d'une application web ayant une architecture standard Client/Serveur. La figure suivante illustre la stratification d'un tel environnement (Ebersbach *et al*, 2008)

<table>
<tr><td rowspan="3">Wiki</td><td>Contenu Lecteurs</td><td rowspan="2">Client</td></tr>
<tr><td>Interface Wiki Auteurs / Administrateurs Wiki</td></tr>
<tr><td>Scripts Wiki Administrateurs Web</td><td rowspan="2">Serveur</td></tr>
<tr><td></td><td>Infrastructure Administrateur Système
(Serveur Web, Base de données)</td></tr>
</table>

Figure 2.1 Environnement Wiki

Du côté client, on trouve les lecteurs. Ces derniers ont un accès direct aux pages wiki pour la consultation. Ils peuvent se transformer en auteurs (administrateurs wiki) dans le cas où ils veulent contribuer à une page. Leurs tâches sont d'administrer les pages wiki. Il s'agit de simples auteurs ayant des privilèges d'actions. Puis, côté serveur, les administrateurs web et systèmes s'occupent particulièrement de l'infrastructure de l'application wiki.

2.1.2 Fonctionnalité

Les utilisations des wikis peuvent être multiples et très diverses. Leur fonction, qu'elle est comme outil administratif, pour compiler de la documentation, comme CMS, pour de la planification, ou comme forum, va être déterminée par les besoins de la communauté, plus précisément, par la vision des fondateurs du wiki. En fait, les

possibilités d'utilisation des wikis sont quasi infinies et de nouveaux types de wiki voient le jour ; les wikis clones, qui affichent des fonctionnalités additionnelles et spécifiques aux besoins desdits wikis.

Voici les fonctionnalités communes aux wikis (Ebersbach *et al*, 2008) :

- Edition : Interface simple et intuitive qui facilite la contribution.
- Historique/Versions précédentes : Permets l'accès aux versions précédentes du document pour archive ou vérification du contenu actuel.
- Liens hypertextes : Des structures intégrées en hypertexte redirigent l'utilisateur vers des pages dont le contenu est relié à la page consultée. La technique du CamelCase permet de créer ses liens en utilisant un format simple pour une création de liens rapide.
- Changements récents : Donne les modifications récentes du document.
- SandBox (ou module d'instruction) : Aide l'utilisateur à comprendre à utiliser le wiki.
- Recherche : Un moteur de recherche qui permet de trouver un sujet en particulier.

2.1.3 Types d'utilisateurs

Tel qu'abordé précédemment, le phénomène wiki s'est développé, modifié, amélioré, grâce à sa popularité. De tout côté positif, son revers : différents types d'utilisation, aussi bénéfiques que problématiques. Au 21st Chaos Community Congress de 2004 à Berlin, Jimmy Whales établissait différents types d'utilisateurs : abeilles, marionnettes, juges, papillons de nuit, vandales, et les externes (Ebersbach *et al*, 2008). Le but d'une telle catégorisation est d'enrichir l'étude des wikis par une compréhension des principaux acteurs, soient les auteurs. Ainsi, les contributions sont

affectées par la qualité et la constance de certains auteurs, ou de leur bienveillante surveillance, mais aussi par les saboteurs, ou les auteurs aux styles agressifs.

2.1.4 Obstacles

En effet, il y a plusieurs utilisateurs des wikis qui ont une attitude peu constructive. Certains d'entre eux agissent de manière destructive pour le contenu du wiki. Le vandalisme et les réactions agressives sont des problèmes de premières heures des wikis. Malgré tout, plusieurs solutions, d'abord vues comme imposantes, furent proposées et appliquées à la communauté. En effet, des wikis tels que Wikipédia a des règles de médiation et essaie d'éviter les conflits en restant neutre et en essayant de rendre le contenu neutre lui aussi. Finalement, le futur des wikis va dépendre surtout de la communauté des contributeurs, car le pire pour un wiki n'est pas de faire face à beaucoup de vandales, mais c'est d'enregistrer un manque d'intérêt envers le contenu du wiki en particulier (Ebersbach *et al*, 2008).

2.2 Comparaison de texte

Dans cette section, nous nous intéressons à quelques techniques de comparaison de texte. D'une part, parce que la majorité des applications wikis utilise ce genre d'outils afin de visualiser l'évolution des articles en termes de quantité de changements, et d'autre part, parce que ces techniques seront utilisées dans notre modèle d'évaluation de la contribution.

2.2.1 La distance de Levenshtein

La distance d'édition est une mesure de similarité entre deux chaînes de caractère. Il s'agit du nombre minimum d'opérations primitives nécessaires pour transformer une chaîne de caractères ch1 en ch2. On identifie trois types d'opérations : insertion,

suppression et substitution de caractères. Le coût de chaque opération vaut 1 (Negre, 2013).

En annexe A.1, nous retrouvons le pseudo-code pour le calcul de la distance de Levenshtein.

2.2.2 TF-IDF

TF-IDF[1](*Term Frequency-Inverse Document Frequency*)[1] est une méthode de pondération de termes figurant dans un ensemble de documents. Cet ensemble est appelé « Corpus ». Elle permet de mesurer l'importance d'un terme par rapport à son corpus (Negre, 2013).

Pour le calcul du poids d'un terme, il faut tout d'abord, déterminer sa fréquence tf. En réalité, il s'agit du nombre d'occurrences. Ensuite, il faut déterminer la fréquence inverse du document idf. Celle-ci se calcule comme suit :

$$log\frac{|D|}{|\{d_j : t_i \in d_j\}|} \quad (2.1)$$

Avec :

- $|D|$: Le nombre total de document formant le corpus.
- $|\{d_j : t_i \in d_j\}|$: Le nombre de document auquel où le terme t_i y figure.

Dans la section 4.3.1.2, vous trouverez un exemple simplifié pour le calcul des vecteurs TF-IDF.

[1] http://fr.wikipedia.org/wiki/TF-IDF [En ligne]

2.2.3 Similarité cosinus

Il s'agit d'une mesure de similarité entre deux documents (Negre, 2013). On calcule le cosinus de l'angle formé par deux vecteurs représentant les documents. Afin de modéliser les documents sous forme de vecteurs, on utilise la plupart du temps la méthode TF-IDF. Le cosinus similarité[2] s'obtient en divisant le produit scalaire des deux vecteurs modélisant les documents par le produit des normes. Ci-dessous la formule :

$$Cos(Vi, Vj) = \frac{\overrightarrow{vi}.\overrightarrow{vj}}{\left\|\overrightarrow{vi}\right\| \cdot \left\|\overrightarrow{vj}\right\|} \qquad (2.2)$$

Avec : V_i et V_j qui représente les deux vecteurs modélisant les deux versions i et j d'un document. Dans le cas où les deux versions sont presque similaires, on obtiendra une valeur qui converge vers 1 sinon dans le cas contraire ça sera une valeur qui converge vers -1.

Comme les versions d'un document ne sont pas toujours de même taille, il y aura, avant de faire le calcul, une phase dans laquelle nous normalisons les vecteurs pour qu'ils aient la même taille.

Dans ce chapitre, nous avons abordé le cadre conceptuel de notre sujet de recherche. Tout d'abord, en présentant le concept des applications wikis et ses différentes variantes. Ensuite, nous avons défini quelques techniques de comparaison de texte utiles pour notre futur outil. Dans ce qui suit, nous allons explorer les différentes approches identifiées dans notre revue de littérature.

---

[2] http://fr.wikipedia.org/wiki/Similarit%C3%A9_cosinus [En ligne]

CHAPITRE III

REVUE DE LITERATURE

Depuis quelques années, les chercheurs s'intéressent de plus en plus à perfectionner les composantes des environnements de travail collaboratif afin d'assurer la qualité du contenu. L'analyse des contributions est une tâche obligatoire pour garantir la qualité, d'où la nécessité d'avoir un bon modèle d'évaluation permettant d'attribuer les notes méritées aux utilisateurs.

Dans les sections suivantes, nous exposerons quelques modèles d'évaluation de la contribution, nous décrirons leur solution, nous soulignerons les mesures utilisées et nous discuterons de leurs avantages et inconvénients. De surcroît, nous évoquerons la notion de système de réputation qui représente un outil complémentaire pour la mesure de la contribution.

3.1 Approche basée sur des mesures de quantité et de la qualité

Les wikis sont parmi les plateformes de collaboration en ligne les plus populaires. La grande quantité d'utilisateurs présents sur cette plateforme produit un nombre conséquent de révisions, d'où la complexité d'analyser leurs contributions. Dans cette section, nous présenterons quelques modèles d'évaluation de la contribution basés sur

des mesures de quantité et de la qualité, ainsi que des modèles basés sur d'autres visions de la contribution.

3.1.1 Modèle de Adler *et al*

3.1.1.1 Description

Dans ce modèle, les chercheurs ont adopté comme mesure de la contribution, la longévité totale d'édition. En effet, dans de précédents travaux sur la contribution des auteurs dans Wikipédia (Adler *et al*, 2008), les chercheurs se sont intéressés à deux critères : le *texte total créé* et le *nombre d'éditions*. Il s'agissait de leur point de départ, mais ils ont dû apporter quelques modifications à cause des failles qu'ils ont découvertes à travers ces deux mesures. Ils ne pouvaient point considérer la quantité de texte créée comme critère d'évaluation de la contribution au vu de la présence d'utilisateurs malveillants qui exploitent ce mécanisme en faisant des actes de vandalisme (Adler *et al*, 2008) (Exemple : ajout de texte qui détériore le contenu, etc.). Par la suite, ce genre d'action requiert une intervention humaine pour apporter les corrections nécessaires. Après avoir comparé les différentes mesures et les paramètres associés, puis en les combinant, les chercheurs ont obtenu l'unité de mesure retenue, à savoir la longévité totale d'édition.

3.1.1.2 Paramètres et mesures

Dans ce modèle, Adler et ses collègues ont identifié deux classes de mesures :

1re Classe : Mesures de quantités

Les mesures de contribution quantitatives sont les suivantes :

La quantité de texte ajoutée dans une révision. Elle est notée par (Adler *et al*, 2008):

$$txt(v_i, v_i) = txt(r_i) \qquad (3.1)$$

La distance d'édition : cette mesure représente le nombre de changements qui ont eu lieu entre deux versions, comme le nombre de mots ajoutés, le nombre de mots supprimés et le nombre de mots déplacés. Elle est notée par (Adler *et al*, 2008):

$$d(r_i) = d(V_{i-}, V_i) \qquad (3.2)$$

Dans le modèle (Adler *et al*, 2008), cette mesure utilise la formule suivante:

$$d(V_i, V_j) = max(I, D) - \frac{1}{2} min(I, D) + M \qquad (3.3)$$

Avec, $D(V_i, V_j)$ le nombre de mots supprimés, $I(V_i, V_j)$ le nombre de mots ajoutés et M le nombre de mots déplacés.

La quantité de changement : Cette mesure représente la quantité de changement qui a eu lieu entre deux versions consécutives pour toutes les interventions de l'auteur. Il s'agit de la somme des distances d'édition propres à toutes les révisions de l'auteur. Elle utilise la fonction pour la mesure de la distance d'édition, soit la formule suivante (Adler *et al*, 2008) :

$$Pour\ tout\ a \in \mathbb{A}, EditOnly(a) = \sum_{p \in P} \sum_{r \in E(a,p)} 1.d(r) \quad (3.4)$$

Le nombre de modifications (ou le nombre d'éditions) : Cette mesure représente le nombre d'éditions faites par un auteur sur tous les articles sur lesquels il a contribué. Elle utilise la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a\ \in\ \mathbb{A}, NumEdit(a) = \sum_{p \in P} \sum_{r \in E(a,p)} 1.1 \qquad (3.5)$$

Avec P étant l'ensemble des pages de Wikipédia, et E(a,p) étant l'ensemble des révisions faites par un auteur « a » sur une page « p ».

Le nombre de mots ajoutés : Cette mesure représente le nombre de mots ajoutés par un auteur dans tous les articles sur lesquels il a contribué. Elle utilise la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a\ \in \mathbb{A}, TextOnly(a) = \sum_{p \in P} \sum_{r \in E(a,p)} 1.txt(r) \qquad (3.6)$$

2e Classe : Mesures de qualité

La qualité d'édition : Cette mesure permet d'associer une valeur à la qualité d'édition pour une révision. Elle utilise la formule suivante (Adler *et al*, 2008):

$$\alpha_{edit}(V_i, V_j) = \frac{d(V_{i-1}, V_j) - d(V_i, V_j)}{d(V_{i-1}, V_i)} \qquad (3.7)$$

Elle a une valeur proche de 1 quand la contribution est gardée en totalité dans les versions suivantes et proches de -1 quand la contribution est effacée.

La moyenne de la qualité d'édition : Cette mesure permet de juger la qualité d'édition d'une révision à partir des dix (10) révisions consécutives faites par des auteurs différents. Elle utilise la formule suivante (Adler *et al*, 2008):

$$\bar{\alpha}_{edit}(r_i) = \frac{1}{|J(r_i)|} \cdot \left(\sum_{r_j \in J(r_i)} \alpha_{edit}(V_i, V_j) \right) \quad (3.8)$$

La qualité d'un texte : Dans leur modèle (Adler *et al*, 2008), les chercheurs ont présenté deux méthodes permettant de calculer la qualité d'un texte.

La 1re méthode consiste à résoudre une équation. Celle-ci mesure le taux de changement appliqué à un texte inséré dans une révision donnée par rapport aux dix (10) révisions suivantes. La valeur de la qualité d'un texte représente la solution de cette équation. Celle-ci utilise la formule suivante (Adler *et al*, 2008):

$$\sum_{r_j \in J(r_i)} txt(i,j) = txt(i,i) \cdot \left(1 + \sum_{r_j \in J(r_i)} (\alpha_{text}(r_i))^{j-i} \right) \quad (3.9)$$

La 2e méthode consiste à diviser la quantité de texte qui survit par rapport aux dix (10) révisions suivantes par la quantité de texte initialement ajoutée. Voici la formule :

$$\beta_{text}(r_i) = \frac{1}{txt(i,i)} \cdot \left(\sum_{r_j \in J(r_i)} txt(i,j) \right) \quad (3.10)$$

La quantité de mots qui survivent après dix (10) révisions : Il s'agit d'une manière de mesurer l'utilité d'un texte ajouté et de calculer le nombre de mots qui survivent

après dix (10) révisions consécutives d'un article. En combinant cette valeur avec la qualité du texte, ils ont obtenu la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a\ \in \mathbb{A}, TenRevisions(a) = \sum_{p \in P} \sum_{r \in E(a,p)} \beta_{text}(r).txt(r) \qquad (3.11)$$

La longévité du texte : elle mesure l'affaiblissement d'un texte suite aux révisions. Pour le calcul, ils ont utilisé la qualité du texte. Elle utilise la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a\ \in \mathbb{A}, TextLongevity(a) = \sum_{p \in P} \sum_{r \in E(a,p)} \alpha_{text}(r).txt(r) \qquad (3.12)$$

La longévité d'édition : il s'agit d'une mesure similaire à la longévité du texte. Elle décrit la façon dont les modifications faites par un auteur perdurent dans les révisions suivantes. Elle utilise la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a \in \mathbb{A}, EditLongevity(a) = \sum_{p \in P} \sum_{r \in E(a,p)} \bar{\alpha}_{edit}(r).d(r) \qquad (3.13)$$

La longévité du texte avec pénalité : Cette mesure combine les deux mesures (Longévité du texte et longévité d'édition). Le but est de remédier aux éditions faites par les utilisateurs malveillants (Vandales). Elle utilise la formule suivante (Adler *et al*, 2008):

$$Pour\ tout\ a\ \in \mathbb{A}, TextLongevityWithPenality(a)$$
$$= TextLongevity(a) + \sum_{p \in P} \sum_{r \in E(a,p)} min\,(0, \bar{\alpha}_{edit}(r)).d(r) \qquad (3.14)$$

3.1.1.3 Discussion

Ce modèle d'Adler *et al* effectue des calculs selon des opérations primitives. Ces opérations incluent seulement l'ajout et la suppression de texte. Les opérations de déplacement de texte sont en fait une suppression suivie d'un ajout. Nous soulignons par conséquent un inconvénient majeur pour ce modèle d'évaluation, car à des niveaux plus abstraits, d'autres formes de contributions pourront être identifiées et devront faire l'objet d'une évaluation différente (par exemple : correction de la grammaire, ajout de liens, amélioration de la navigation).

En ce qui concerne la longévité d'édition, nous soulignons la pertinence de l'outil de mesure privilégié par les chercheurs. C'est une mesure qui associe à la fois la quantité d'éléments changés et ceux qui restent inchangés. Aussi, elle résiste aux simples attaques par sa méthode d'attribution de valeurs à la contribution. En d'autres termes, elle permet d'attribuer une valeur à la contribution si et seulement si elle est acceptée par d'autres auteurs tout au long du cycle de vie de l'article. De fait, les fausses manœuvres seront supprimées par les « bons » contributeurs. Un autre point positif à mentionner est la sensibilité au volume de la contribution. Enfin, elle représente un indicateur de choix contre le vandalisme.

À partir des mesures effectuées sur les 25 millions de révisions, les chercheurs ont pu identifier des comportements inhabituels de certains auteurs. En effet, à partir des mesures relatives à l'édition, les auteurs ont constaté que les plus grands contributeurs étaient les robots. Ces derniers sont également les seconds plus grands contributeurs, issus des deux mesures que sont la longévité du texte et le texte avec pénalité. Parmi ces robots, nombre d'entre eux sont à l'origine de vandalisme.

En résumé, ce modèle représente un bon point de départ pour la conception d'une solution robuste pour contrer les attaques des utilisateurs malveillants en exploitant la longévité d'édition comme mesure. Aussi, une amélioration demeure possible au niveau des formes de contribution, en y incluant d'autres types de contributions dans le calcul de sa valeur.

3.1.2 Modèle de Muller-birn *et al*

3.1.2.1 Description

Dans ce modèle (Muller-birn *et al*, 2009), les chercheurs proposent un calcul composite de l'activité de l'auteur dans Wikipédia afin d'en déduire son classement et ses contributions. Pour cela, ils ont défini trois approches. Toujours dans le domaine des réseaux collaboratifs, la première approche est une analyse des actions. Elle emploie comme mesure le compteur d'édition. La deuxième approche est l'analyse du contenu. Elle identifie comme mesure l'importance du contenu. La troisième approche est l'analyse de l'influence de l'activité de l'auteur. Elle utilise comme mesure la centralité d'intermédiarité. Enfin, ces trois approches ont été combinées pour pouvoir mesurer l'activité de chaque auteur.

Dans la section suivante, nous décrirons plus en détail ces mesures.

3.1.2.2 Paramètres et mesures

Le compteur d'édition

Il s'agit du nombre d'interventions sur tous les articles pour chaque auteur. Elle permet d'évaluer l'activité des auteurs dans les wikis. Cette mesure utilise la formule suivante (Muller-birn *et al*, 2009):

$$m_{EC}(a_i) = \sum_{p=1}^{p_m} \sum_{t_0}^{t_n} V_{t_n} \qquad (3.15)$$

Avec $P = \{p_1, \dots p_m\}$ l'ensemble des pages qui forment le contenu de Wikipédia. Chaque page possède différentes versions notées par l'ensemble$P_i = \{V_{t_0}, \dots V_{t_n}\}$. Et chaque version V_{t_i} est crée par un auteur a_i à l'instant t_i.

L'importance du contenu

Cette mesure évalue la pertinence du contenu ajouté par un auteur. Les concepteurs du modèle ont proposé la formule suivante (Muller-birn *et al*, 2009):

$$m_{CS}(a_i) = \sum_{i=1}^{i_{max}} f_{i,n,a_i} . w_{i,n} \qquad (3.16)$$

Avec f_{i,n,a_i} la fréquence d'ajout du terme tm_i dans une page p_n par l'auteura_i. $w_{i,n}$ Est l'importance d'un terme spécifique pour cette page. Celle-ci est obtenue par le produit de la fréquence normalisée du terme tm_i dans une page p_n et la fréquence de l'inverse du terme. Voici leurs formules :

$$nf_{i,n} = \frac{f_{i,n}}{\sum_{tm_i \in C} f tm_{j,m}}, idf_i = log\left(\frac{|C|}{|C: f_{tm_i} \in C|}\right), w_{i,n} = nf_{i,n} . idf_i \qquad (3.17)$$

Centralité d'intermédiarité

La centralité d'intermédiarité est une mesure qui indique le niveau d'implication d'un auteur contribuant à un réseau de collaboration dynamique. Il s'agit du degré de connectivité d'un auteur avec d'autres auteurs, mais ce, de façon indirecte. Plus le nombre de liens est élevé, plus il est centralisé. Elle est calculée à l'aide de cette formule (Muller-birn *et al*, 2009):

$$m_{C_B}(a_i) = \frac{\sum_{j<k} g_{jk}(a_i)/g_{jk}}{[(g-1)(g-2)/2]} \qquad (3.18)$$

Avec : g est le nombre de liens totaux. g_{jk} est le plus court chemin entre deux auteurs a_j et a_k sachant que a_i est sur le chemin.

Activité d'auteur

Il s'agit de l'intégration des trois précédentes mesures dans une seule. Elle inclut à la fois le nombre d'éditions, la pertinence du contenu, ainsi que la connectivité. Elle est représentée par la formule suivante (Muller-birn *et al*, 2009):

$$m_{AC}(a_{i_t}) = \frac{\widetilde{m_{CS}}(a_{i_t})^{0.5}}{\widetilde{m_{EC}}(a_{i_t})^{0.5}} \cdot m_{C_B}(a_{i_t})^{0.5} \qquad (3.19)$$

Avec :

$$\widetilde{m_{CS}}(a_{i_t}) = 1 - \frac{1}{1 + m_{CS}(a_{i_t})^{\propto}}, \widetilde{m_{EC}}(a_{i_t}) = 1 - \frac{1}{m_{EC}(a_{i_t})^{\propto}} \qquad (3.20)$$

D'où l'activité cumulative de l'auteur est la suivante (Muller-birn *et al*, 2009):

$$m_{AC}(a_i) = \sqrt{\frac{1}{t_{max}} \cdot \sum_{t_0}^{t_{max}} (m_{AC}(a_{i_t}))^2} \qquad (3.21)$$

3.1.2.3 Discussion

Dans ce modèle (Muller-birn *et al*, 2009), les chercheurs ont analysé la contribution des auteurs selon trois perspectives. En premier lieu, ils ont utilisé le compteur d'édition, celui-ci étant un indicateur de contribution pour les auteurs si et seulement si ces derniers sont de bonne foi, a contrario du vandalisme créé par les utilisateurs malveillants. En effet, cet outil de mesure représente un inconvénient majeur, à savoir son possible détournement par les utilisateurs pour leur propre compte. En définitive,

cette mesure, considérée de manière isolée, ne permet pas d'identifier les utilisateurs malveillants.

En second lieu, ils ont utilisé l'importance du contenu utilisé comme un indicateur de pertinence du contenu. Cette mesure est basée sur des techniques de fouille de texte. Dans cette perspective, les chercheurs ne prennent pas en considération tout le contenu des pages ; ils analysent seulement les termes ayant de l'importance par rapport au sujet de l'article. On considère ainsi que cet outil de mesure reste faible, car il ne traite qu'une partie du contenu ajouté.

En troisième et dernier lieu, ils ont utilisé la centralité d'intermédiarité. C'est un indicateur de collaboration permettant d'avoir une idée de l'influence de l'activité des auteurs dans ce genre d'environnement. Nous considérons cette mesure très utile pour déterminer le taux d'implication des auteurs dans l'élaboration et la rédaction des articles.

Le résultat global de ces trois perspectives a abouti à l'élaboration d'une nouvelle mesure. Il s'agit de la mesure de l'activité des auteurs. Très intéressante, elle prend en considération plusieurs mesures différentes en les intégrant dans une seule formule, ce qui permet de mesurer la contribution des auteurs de manière plus exacte.

Dans la section suivante, nous nous intéresserons aux modèles basés sur la catégorisation des opérations de contribution.

3.2 Approche basée sur la catégorisation des opérations

La catégorisation des opérations est une action de classification des opérations réalisées sur un article sous forme de plusieurs catégories. Le but est de décrire le processus de révision avec une suite d'opérations.

Notre recherche nous a permis de relever deux modèles s'intéressant à l'identification des opérations faites sur un article. Toujours dans l'optique de s'interroger sur l'état de l'art en la matière, il nous semble judicieux d'identifier les actions entreprises afin d'en ressortir la réelle contribution des auteurs.

3.2.1 Modèle de Arazy *et al*

3.2.1.1 Description

Dans ce modèle (Arazy *et al*, 2010), les chercheurs ont tenté d'attribuer une valeur à la contribution d'un auteur selon différentes catégories d'action. Les principales catégories de tâches proposées dans (Arazy *et al*, 2010) sont les suivantes :

- Ajout de contenu ;
- Amélioration de la navigation dans la page ;
- Suppression du contenu ;
- Ajout de liens pour d'autres pages ;
- Relectures / Corrections et raffinement du texte.

Un algorithme spécifique est utilisé pour la capture de chaque type de contribution. En appliquant ces algorithmes, les chercheurs obtiennent, dans un premier temps, une note de contribution pour chaque catégorie, puis, dans un deuxième temps, un score global de la contribution. En ce qui concerne l'évaluation, ils ont comparé leurs algorithmes de capture de contribution à des évaluations faites manuellement. Pour une combinaison optimale de ces mesures, ils ont employé une analyse régressive en utilisant le score du meilleur contributeur identifié manuellement.

Dans ce qui suit, nous allons décrire les mesures utilisées (Arazy *et al*, 2010), ainsi que les algorithmes permettant leurs identifications.

3.2.1.2 Paramètres et mesures

Quantité de contenu ajouté

Comme son nom l'indique, cette mesure donne une information sur la quantité de texte ajouté par un auteur. La phrase représente l'unité de mesure. Avant le calcul, une phase de préparation est nécessaire. Tout d'abord, ils ont déterminé les limites des phrases en utilisant un outil de segmentation. Ensuite, ils ont identifié les ressemblances entre les deux versions en utilisant l'algorithme Munkers (Arazy *et al*, 2010) pour des matrices rectangulaires, avec comme variable les phrases des deux versions. Enfin, pour le calcul de la quantité de contenu nouvellement ajouté, ils ont utilisé trois approches (Arazy *et al*, 2010):

- 1re approche : Elle est basée sur le calcul de l'ajout cumulatif d'un auteur sur toutes les révisions.
- 2e approche : Ils ont considéré le contenu restant sur la révision de la page la plus récente, et ceci selon l'hypothèse : « c'est le contenu de haute qualité qui persiste ».
- 3e approche : Ils ont considéré tous les ajouts de l'auteur et les longévités de ses contributions. Ainsi, le score de l'utilisateur sera basé sur le nombre de révisions pour lequel ses ajouts persistent.

Le nombre de liens hypertextes internes

Afin d'améliorer la navigation, certains auteurs rajoutent des liens hypertextes internes. Pour identifier cette mesure, ils ont employé deux approches (Arazy *et al*, 2010) :

- 1re approche : basée sur le nombre total de liens internes ajoutés.
- 2e approche : basée sur le nombre de liens restant dans la version la plus récente de la page.

Le nombre de changements

Cette mesure est reliée à la catégorie des corrections (Arazy *et al*, 2010). Pour le calcul, les chercheurs ont identifié le nombre de changements au niveau des mots.

Le nombre de phrases supprimées / le nombre de révision

Ces deux mesures sont reliées à la catégorie de suppression. Les chercheurs ont calculé le nombre de phrases supprimées par l'utilisateur, ainsi que le nombre de révisions incluant l'action de suppression (Arazy *et al*, 2010).

Nombre de liens externes

Cette mesure est reliée à la catégorie des liens (Arazy *et al*, 2010). Les chercheurs ont calculé le nombre de liens externes insérés par un utilisateur. Ils ont utilisé deux méthodes :

- 1re méthode : Elle est basée sur le nombre total de liens ajoutés.
- 2e méthode : Elle est basée sur le nombre de liens restés.

3.2.1.3 Discussion

Dans ce modèle (Arazy *et al*, 2010), les chercheurs n'ont malheureusement pas étudié la qualité de la contribution. En effet, ils ont seulement focalisé leurs mesures sur celles des quantités. Par ailleurs, ils n'ont pas détaillé toutes les formules ni présenté toutes les unités de mesure. D'où le questionnement qui demeure quant à l'unité de mesure globale et sa formule.

Toutefois, l'idée de combiner plusieurs perspectives nous mène, *a fortiori*, vers un modèle plus exact.

3.2.2 Modèle de Fong et Biuk-Aghai

3.2.2.1 Description

Dans ce modèle (Fong et Biuk-Aghai, 2010), les chercheurs ont défini une méthode pour catégoriser et présenter les contributions des auteurs afin de pouvoir mesurer l'importance d'édition. Dès lors, cette mesure est très flexible dans le sens où la formule est une somme pondérée par des coefficients définis par les concepteurs de ce modèle. Pour des raisons d'exactitude, les coefficients peuvent être modifiés. Cette méthode est basée sur un analyseur d'historique d'édition (Fong et Biuk-Aghai, 2010). Ce dernier effectue une comparaison entre deux versions d'un article pour extraire une liste d'actions classées dans des ensembles appelés catégories.

L'analyseur d'historique d'édition (Fong et Biuk-Aghai, 2010) est divisé en quatre modules :

Analyseur lexical (Lexical Analyser) (Fong et Biuk-Aghai, 2010): Son rôle est de transformer une suite de caractère en une liste de jetons (Tokens). Ce traitement est facile à réaliser puisque MediaWiki emploie un langage à balise pour la structuration de ses pages. Après la génération de la liste des jetons, l'analyseur construit une liste de phrases. Pour la construction des phrases, il exploite les règles de ponctuation et celles du langage à balise de MediaWiki (Ebersbach *et al*, 2008).

Moteur de différence textuelle (Text Difference Engine) (Fong et Biuk-Aghai, 2010): Son rôle est de déterminer les différences et les similitudes par rapport à la révision précédente. Les phrases existantes dans la révision antérieure et n'ayant pas de liens

avec la nouvelle révision sont étiquetées « suppression ». Les phrases existantes dans la nouvelle révision et n'ayant pas de liens avec la révision antérieure sont étiquetées « insertion ». Pour déterminer le lien entre les phrases des deux versions, ils ont calculé le taux de correspondance. Ce dernier est comparé à un seuil pour décider s'il s'agit de la même phrase ou pas.

Module pour la catégorisation des actions (Action Categorizer) (Fong et Biuk-Aghai, 2010): Selon des règles bien définies (Fong et Biuk-Aghai, 2010), ce module parcourt la liste des phrases pour extraire la liste des actions.

Module de synthèse de l'historique (History Summarizer) (Fong et Biuk-Aghai, 2010): Après la catégorisation des actions, ce module permet de les réunir dans des groupes plus abstraits.

Dans la partie suivante, nous détaillerons la formule de l'importance d'édition, ainsi que les paramètres reliés aux modules de l'analyseur.

3.2.2.2 Paramètres et mesures

L'importance d'édition

L'importance d'édition d'un auteur représente la pertinence de sa contribution (Fong et Biuk-Aghai, 2010). Sa valeur est une somme pondérée. Le calcul se fait après avoir quantifié tous les types d'actions. Chaque type d'actions sera multiplié par son poids respectif. Voici sa formule (Fong et Biuk-Aghai, 2010):

$$s = s_{high} + s_{basic} \qquad (3.22)$$

Sachant que :

$$s_{high} = \sum_{x=1}^{m}\sum_{i=1}^{n} W_{x,i} \cdot c_{x,i} \qquad (3.23)$$

Et

$$s_{basic} = \sum_{i=1}^{n} W_{ins,i} \cdot c_{ins,i} + W_{del,i} \cdot c_{del,i} + W_{repl,i} \cdot c_{repl,i} + W_{mov,i} \cdot c_{mov,i} \qquad (3.24)$$

Avec : $W_{x,i}$ est le poids de la catégorie du contenu i par rapport à l'action d'édition x. et $c_{x,i}$ est une quantité mesurée qui dépend de la nature de l'action d'édition et de la nature du contenu (Fong et Biuk-Aghai, 2010).

Le taux de correspondance

Ce paramètre est utilisé dans l'algorithme de différenciation entre les phrases. Le principe est de calculer le taux de correspondance entre chacune des phrases de la version précédente avec celles de la version courante. Il est déterminé de cette façon (Fong et Biuk-Aghai, 2010):

$$m_{i,j} = 2 \times \frac{lc_{i.j}}{lo_i + ln_j} \qquad (3.25)$$

Avec :

lo_i est le nombre de jetons de la $i^{ème}$ phrase dans la version précédente.
ln_j est le nombre de jetons de la $j^{ème}$ phrase dans la version courante.
$lc_{i.j}$ est le nombre de jetons communs aux deux phrases (précédente et courante).
Le seuil utilisé pour valider la correspondance est de 40%.

Concernant la séparation et la fusion des phrases, l'équation utilisée est la suivante :

$$\frac{2 \times (lc_{i,j} + lc_{i+1.j})}{lo_i + lo_{i+1} + ln_j} > m_{i,j} \qquad (3.26)$$

Cette comparaison leur permet de dire s'il s'agit d'une séparation ou d'une fusion de phrases.

3.2.2.3 Discussion

À nouveau dans ce modèle (Fong et Biuk-Aghai, 2010), les chercheurs ont mis en avant l'importance de la méthode de calcul, mais non sa contribution. Toutefois, il va sans dire que ce modèle présente un potentiel considérable, car il essaie de résoudre une variante importante du problème de la contribution, à savoir l'historique d'édition. Autrement dit, il s'agit de pouvoir détailler les actions après chacune des révisions. Ainsi, être capable de les quantifier nous aide quant à l'évaluation de la contribution. Aussi, étant normalisé par la syntaxe des éditeurs de Wikipédia, l'avantage de ce modèle est qu'il peut être appliqué à n'importe quel langage.

Pour évaluer leur modèle, Fong et Biuk-Aghai ont comparé les résultats obtenus avec ceux d'un groupe de volontaires. Ce dernier devait effectuer manuellement les mêmes opérations que l'outil. Le résultat de l'évaluation montre que leur prototype produit un historique d'édition en concordance avec l'interprétation des volontaires par rapport aux changements.

Dans la section suivante, nous focaliserons notre attention sur des modèles qui analysent les liens entre les utilisateurs afin de fournir un classement des contributeurs. Il s'agit d'approches basées sur les réseaux sociaux.

3.3 Approche basée sur les réseaux sociaux

L'analyse des réseaux sociaux est un paradigme de recherche qui tente de démêler les modèles de relations sociales entre divers individus dans un contexte social (Tang *et al*, 2008). En effet, l'article produit est le résultat d'un processus communautaire impliquant un certain nombre d'interactions sociales intégrées dans la modification du contenu. En ce sens, il existe un processus de négociation lors de l'écriture et la structuration de l'article. Par la suite, l'analyse de ce processus représente une nouvelle perspective pour l'évaluation des contributions. Dans ce genre d'approche, on se focalise sur l'analyse des liens d'interaction entre plusieurs contributeurs.

3.3.1 Modèle de Tang *et al*

3.3.1.1 Description

Dans ce modèle (Tang *et al*, 2008), les chercheurs se sont intéressés à la mesure des liens qui existent entre les utilisateurs. Lorsque ces derniers contribuent à l'écriture d'un article, il se crée entre eux un lien de coauteurs. La méthode et l'algorithme développés calculent le degré du lien entre les coauteurs pour une paire d'auteurs donnée. Cette méthode permet aussi de déterminer des groupes d'expertise pour un sujet choisi. Cependant, la puissance du lien qui relie les coauteurs n'est pas égale parmi tous les utilisateurs d'un même groupe. Par conséquent, il faut analyser la relation des coauteurs non pas par le niveau du groupe, mais bien par la qualité des paires de coauteurs. Il faut aussi prendre en considération le facteur du temps, c'est-à-dire savoir si les révisions faites pas les coauteurs sont simultanées ou séparées. En effet, connaître les relations implicites de coauteurs permet de découvrir dans le Wiki les groupes d'expertise pour un domaine particulier. Ainsi, il sera possible d'extraire une métrique d'expertise (mesure d'expérience par domaine). Leur implémentation est sous forme d'une extension pour MediaWiki.

Il existe trois catégories d'évaluation des liens de coauteurs (Tang *et al*, 2008): le réseau de coauteurs, le réseau social, et le petit réseau mondial « small-world network ».

Les chercheurs ont énoncé d'autres méthodes implémentées dans des travaux reliés :

Biuk-Aghai (réseau de coauteurs) (Tang *et al*, 2008)
Se base plutôt sur la visualisation du réseau de liens. Ces visualisations « graphiques » exposent les relations entre les entités, les catégories et les résultats.

Huang (réseau social) (Tang *et al*, 2008)
Utilise un algorithme de réseau social afin de comptabiliser les informations des coauteurs dans un programme de visualisation (InterRing) personnalisé qui permet aux utilisateurs de comprendre la collaboration académique.

Liu (Tang *et al*, 2008)
Modèle basé sur le « poids » des révisions, c'est-à-dire que des collaborations fréquentes reçoivent un plus grand poids et permettant ainsi aux utilisateurs d'analyser des graphiques sur les liens entre coauteurs à l'aide d'un outil de visualisation.

Ils ont utilisé ces dernières méthodes comme un indicateur des avancements en la matière, mais aussi pour se distancer de celles-ci. En fait, un algorithme qui extrait les utilisateurs interreliés par le réseau de coauteurs doit être rapide dans son calcul. Puisque les utilisateurs du web et les bases de données ne sont guère réputés pour leur patience, la nouvelle méthode présentée en l'espèce permet d'augmenter la vitesse de réponse de l'algorithme, satisfaisant ainsi à la demande des utilisateurs web, ainsi qu'aux exigences de la qualité du renseignement obtenu.

L'introduction des quelques concepts liés aux coauteurs présentés ci-dessous permet d'expliciter leur méthode.

Révision

D'après MediaWiki (Ebersbach *et al*, 2008), il y a deux types de révision :

-Révision mineure (Minor Edit) : Correction de la grammaire, de la syntaxe et de l'orthographe.

-Révision non mineure (Non-minor Edit) : Ajout de paragraphes, par exemple.

Auteur

Dans ce modèle (Tang *et al*, 2008), les chercheurs se concentrent surtout sur les utilisateurs inscrits, soit les auteurs, mais ils prennent aussi en compte les utilisateurs anonymes. Si deux auteurs différents travaillent sur l'article « X », alors ils ont au moins une révision chacun sur le même article, ce qui crée une relation de coauteurs entre eux. L'article, la révision et l'auteur sont les trois éléments primordiaux de leur méthode. Mais la révision est la pierre angulaire.

Méthodologie

L'algorithme calcule le degré du lien entre des coauteurs pour une paire d'auteurs donnée. Les paramètres saisis sont les identifiants de deux auteurs enregistrés, et le paramètre de sortie est le degré des liens entre les coauteurs.

Pour un auteur donné, les autres auteurs seront qualifiés de hors-sujet comme coauteurs s'ils n'ont fait que des révisions mineures sur des articles où « l'auteur donné » a aussi contribué, et des articles seront qualifiés de hors-sujet si « l'auteur donné » n'y a fait que des révisions mineures. Quant au degré du lien entre les coauteurs, plus la valeur du degré est large, plus le lien entre les coauteurs sera fort.

Voici les étapes suivies pour le calcul du degré des liens coauteurs (Tang *et al*, 2008):

1-Récupérer la liste des pages éditées par l'auteur *a* ;

2-Éliminer les pages sur lesquelles l'auteur a effectué des révisions dites mineures ;

3-Pour chaque page, déterminer la liste des autres auteurs participants ;

4-Éliminer les auteurs qui ont effectué des révisions mineures ;

5-Pour l'ensemble des auteurs de chaque page, calculer un degré de la page ;
6-Calculer le degré coauteur à partir de tous les degrés de page.

Dans la partie suivante, nous présenterons les paramètres utilisés dans cet algorithme.

3.3.1.2 Paramètres et mesures

Degré de page

Le degré de page est le degré de coauteur pour une page entre deux auteurs. Elle est définie comme suit (Tang *et al*, 2008):

$$p(a,b)_i = \left(\frac{min(n_{ia}, n_{ib})}{n_i} + k\frac{min(m_{ia}, m_{ib})}{m_i}\right) \times \frac{L_{ia} \cap L_{ib}}{L_{ia}} \qquad (3.27)$$

Avec :

n_i est le nombre des éditions majeures de la page i.

n_{ia} (Respectivement n_{ib}) est le nombre des éditions majeures réalisées par l'auteur a (respectivement b) sur la page i.

m_i est le nombre des éditions mineures de la page i.

m_{ia} (Respectivement m_{ib}) est le nombre des éditions mineures réalisées par l'auteur a (respectivement b) sur la page i.

Degré coauteur

Le degré de coauteur pour deux contributeurs est la somme des degrés de page pour ces derniers. Il est défini par la formule suivante (Tang *et al*, 2008):

$$d(a,b) = s \times \sum_{i=1}^{t} p(a,b)_i \qquad (3.28)$$

La constante « s » est utilisée pour affiner le résultat et faire une représentation sur une échelle appropriée.

3.3.1.3 Discussion

À l'aide de ce modèle (Tang *et al*, 2008), les chercheurs ne distinguent que les éditions mineures des éditions majeures. Par ailleurs, il est impossible de faire la distinction entre un ajout de texte ou une correction de la grammaire. Par conséquent, on considère que leurs mesures ne fournissent pas une évaluation exacte. En effet, l'approche des réseaux sociaux est pertinente pour estimer la distribution d'un seul effort à travers les pages wikis, mais ne fournit pas une estimation représentative de l'ampleur des contributions apportées à une page wiki spécifique.

3.3.2 Modèle de Korfiatis et Naeve

3.3.2.1 Description

Dans ce modèle (Korfiatis et Naeve, 2005), les chercheurs ont essayé d'analyser les contributions wiki en utilisant l'approche des réseaux sociaux. D'après la structure de Wikipédia, ils ont défini deux niveaux : le 1er niveau est le réseau des articles (Korfiatis et Naeve, 2005) et le 2^{e} niveau est le réseau des contributeurs (Korfiatis et Naeve, 2005). Le lien entre deux contributeurs est défini lorsqu'un auteur modifie une version soumise par un autre. Par conséquent, avec une ou plusieurs contributions à un article, le contributeur établit des liens avec les autres du même article. Aussi,

les liens entre les articles sont définis par les références existantes entre eux. Pour les poids des liens, ils ont utilisé la fonction « diff » de wiki.

3.3.2.2 Paramètres et mesures

Les mesures utilisées par Korfiatis et Naeve afin d'analyser le réseau social sont les suivantes:

CDC : degré de centralité contributeur

Le degré de centralité contributeur est la somme des liens que le contributeur possède avec les autres contributeurs, divisés par le plus grand degré (d'après la théorie des graphes, le plus grand degré est le nombre total de nœuds moins un). Ci-dessous sa formule (Korfiatis et Naeve, 2005):

$$C_D'(n_i) = \frac{d(n_i)}{g-1} \qquad (3.29)$$

Avec : $C_D(n_i) = d(n_i) = \sum_j x_{ij}$

x_{ij} est une variable booléenne.

ADC : degré de centralité article

Le degré de centralité article représente la variabilité des degrés de centralité contributeur. Ci-dessous sa formule (Korfiatis et Naeve, 2005):

$$C_{DM} = \frac{\sum_i^g [C_D(n^*) - C_D(n_i)]}{(g-1)(g-2)} \qquad (3.30)$$

$C_D(n^*)$ est la plus grande valeur observée du degré de centralité contributeur.

3.3.2.3 Discussion

À partir de ce modèle (Korfiatis et Naeve, 2005), il est difficile d'évaluer ou d'estimer directement les contributions. Celui-ci permet seulement d'analyser le comportement des utilisateurs par rapport à leurs implications et leurs taux de connectivité. En effet, d'après (Korfiatis et Naeve, 2005), quand un article possède un faible degré de centralité, cela signifie qu'il a été élaboré par des contributeurs individuels ayant des intérêts pour d'autres domaines. L'inverse est également valable ; quand un article possède un degré de centralité élevé, cela indique qu'il a été rédigé par des utilisateurs ayant de l'intérêt pour le domaine. De plus, l'utilisation d'une variable booléenne présente une faiblesse de mesure, car une contribution ne sera jamais limitée à deux valeurs (0 et 1).

D'autre part, l'étude des liens dans les deux niveaux (réseau de contributeurs et réseau d'articles), permettra d'avoir une classification de contributeurs par domaine d'expertise, et de se rapprocher davantage de l'évaluation de la pertinence des contributions.

Dans la section suivante, d'autres types d'analyse des aspects liés indirectement aux contributions seront évoqués.

3.4 Autres modèles

3.4.1 Modèle d'Anthony *et al* (Motivation et qualité)

3.4.1.1 Description

Dans ce modèle (Anthony *et al*, 2007), les chercheurs ont examiné les effets de la motivation des contributeurs sur la qualité des contributions en se focalisant sur deux axes reliés. En effet, les contributeurs sont motivés par la réputation et/ou

l'engagement à l'identité du groupe de la communauté Wikipédia ; en excluant évidemment les utilisateurs malveillants. Leur modèle est basé sur l'analyse du taux de rétention des contributeurs.

3.4.1.2 Paramètres et mesures

Le taux de rétention

Cette mesure permet de calculer le pourcentage de caractère retenu par contribution pour chaque contributeur. Il s'agit d'une autre forme d'évaluation de la qualité des contributeurs. Ci-dessous sa formule (Anthony *et al*, 2007) :

$$R_i = \frac{\sum_{j=1}^{K} C_{ij}}{\sum_{j=1}^{K} T_{ij}} \qquad (3.31)$$

Avec :

C_{ij} est le nombre de caractères retenu des contributions du contributeur i dans l'article j.

T_{ij} est le nombre total de caractère constituant l'article j auquel a contribué le contributeur i.

Pour simplifier l'analyse, ils ont appliqué un logarithme sur ces variables.

3.4.1.3 Discussion

L'utilisation du taux de rétention comme indicateur de la qualité du contributeur est une bonne initiative. Le bémol est que cette mesure ne prend pas en compte tous les aspects importants de la qualité du contenu, tel que le temps écoulé entre les modifications ou le statut du contenu. Anthony et ces collègues ont constaté que la qualité des contributeurs augmente avec le nombre de contributions (Anthony *et al*,

2007). Cette constatation sonne comme un contresens de la définition même de la qualité. En effet, la qualité des contributeurs devrait dépendre de la qualité de la contribution et non pas de leur nombre.

Concernant la motivation des utilisateurs, leur hypothèse semble confirmée. Sous couvert des résultats obtenus, les chercheurs ont constaté que les utilisateurs anonymes sont susceptibles de contribuer autant sur des articles bien établis que pour de nouveaux sujets avec peu de contenu. Autrement dit, la plupart des utilisateurs anonymes enrichissent les articles avec un contenu considérable.

Dans les faibles niveaux d'engagement (en nombre de contributions), la qualité de la contribution des utilisateurs anonymes est plus élevée et diminue à mesure que la participation augmente. Pour les utilisateurs enregistrés, la qualité augmente avec la participation. Enfin d'après leurs analyses, le nombre ainsi que la qualité des contributeurs affectent positivement la qualité du contenu.

3.4.2 Modèle de Sesia *et al*

3.4.2.1 Description

Toujours dans la recherche des facteurs qui influencent la valeur de la contribution (Sesia *et al*, 2013), les chercheurs ont défini la contribution comme étant la perception des utilisateurs individuels sur l'utilité, la serviabilité et la valeur de leurs contributions à Wikipédia (Sesia *et al*, 2013). En effet, il est important de connaître les activités d'édition des utilisateurs pour estimer leurs performances de contribution.

D'après les chercheurs, il existe deux types de contributeurs :

- Contributeur apportant des changements substantiels : Ajout de contenu / d'information, ajout de liens, ajout d'images, suppression de contenu inexact,

qui apporte des modifications de fond / substantiels aux articles (Sesia *et al*, 2013). Appelé aussi contributeur de connaissance

- Contributeur apportant des changements non substantiels : Réorganiser le contenu existant, des révisions non substantielles (correction des erreurs de grammaire, mise en forme du texte pour une meilleure présentation). Appelé aussi contributeur de forme.

Pour mesurer le type de contributeur, Sesia et ses collègues se sont basés sur l'identification des fréquences de contributions substantielles. Pour confirmer leurs hypothèses concernant les liens entre les intérêts et les ressources avec la valeur des contributions, ils ont utilisé un questionnaire envoyé à un nombre d'utilisateurs enregistrés.

Les paramètres prélevés sur les réponses des utilisateurs sont les suivantes :

- Ampleur des intérêts : Il s'agit du nombre des différents intérêts que l'utilisateur possède. C'est le nombre de catégories d'intérêts qui sont supérieurs à 4 (Sesia *et al*, 2013).
- Profondeur des intérêts : Il s'agit de l'étendue du plus grand intérêt que l'utilisateur possède. C'est la valeur la plus élevée parmi toutes les catégories d'intérêt (Sesia *et al*, 2013).
- Ampleur des ressources : Il s'agit du nombre de compétences différentes (dans l'éducation, profession, et passe-temps) que les utilisateurs possèdent (Sesia *et al*, 2013).
- Profondeur des ressources : La mesure de la plus haute expertise (dans l'éducation, professeur, et passe-temps) que les utilisateurs possèdent (Sesia *et al*, 2013).

Finalement, il s'agit d'un modèle basé sur une investigation restreinte par rapport au nombre d'utilisateurs qui ont répondu au questionnaire.

Dans la section suivante, nous étudierons le système de réputation.

3.5 Système de réputation

Un système de réputation est très utile pour déterminer l'expérience des auteurs dans une plateforme de travail collaboratif telle que Wikipédia. Dans (Adler et Alfaro, 2007), on trouve un système de réputation basé sur le contenu. Dans un tel système, les auteurs augmentent leur réputation quand leurs modifications sont gardées par les auteurs effectuant les révisions subséquentes et voient leur réputation diminuer lorsque les auteurs-réviseurs suppriment lesdites modifications.

La réputation des auteurs qui contribuent et révisent minutieusement les textes peut être utilisée comme un guide approximatif quant à la véracité et la confiance accordées aux articles. La réputation peut aussi être utilisée pour la gestion des auteurs (classification selon leur expérience). De surcroît, elle peut être utilisée comme un moyen d'alerter les auteurs lorsque leur article a été modifié par des utilisateurs ayant une faible réputation. De plus, la réputation incite les auteurs à produire des contributions de meilleure qualité.

Leur système (Adler et Alfaro, 2007) mesure deux quantités :
-«Text life» : C'est la quantité de texte écrit par un utilisateur X et qui a été gardée par un utilisateur Y.
-« Edit life» : C'est le nombre d'éditions faites par un utilisateur X et qui a été gardé par un utilisateur Y.
Un système basé sur la réputation du contenu amène une plus grande objectivité de celui-ci qu'un système basé sur la réputation de l'utilisateur, mais le premier est

moins précis que le second. En effet, la réputation est déduite de manière uniforme et automatique à partir de toutes les modifications.

Leur système (Adler et Alfaro, 2007) peut percevoir la différence entre une suppression et une modification sur une contribution. Il s'agit d'un des éléments les plus pertinents, car l'auteur dont la contribution est supprimée verra sa réputation diminuer, alors que celui dont la contribution est modifiée partiellement ne verra pas d'effets sur sa réputation. Aussi, il peut être utilisé pour prédire la qualité des futures contributions des auteurs.

La pierre angulaire de la méthodologie des auteurs est qu'ils ont basé leur méthode sur trois axes de développement de la réputation :

Valeur prescriptive : Le système spécifie la manière dont les utilisateurs peuvent acquérir une bonne réputation; il permet donc de définir les bonnes pratiques.

Valeur descriptive : Le système permet de catégoriser les utilisateurs et leurs contributions, basées, évidemment, selon leur réputation.

Valeur prédictive: La réputation d'un auteur doit être statistiquement reliée avec la qualité de ses futures contributions.

Enfin, la réputation des auteurs est un indicateur de qualité et de confiance entre l'utilisateur et son lecteur.

Amélioration

Un système de réputation est sujet à des attaques par des utilisateurs qui veulent accroître leur réputation sans fournir de contributions utiles. Ces utilisateurs se servent d'identités secondaires pour arriver à leurs fins. Ce genre d'action est

interprété comme une attaque. Dans leur article (Chatterjee *et al*, 2008), les chercheurs en ont défini quelques types :

-Attaque de suppression-restauration : En utilisant une identité secondaire, l'utilisateur peut supprimer le contenu d'une version, puis le restaurer avec son identité principale.

-Attaque d'ajout-restauration : En utilisant une identité secondaire, l'utilisateur peut ajouter du contenu non approprié dans une version, puis supprimer cet ajout avec son identité principale.

-Attaque des faux utilisateurs : En utilisant son identité principale, l'utilisateur procède à un ajout dans une version (pertinent ou pas), puis il le fait approuver par un nombre de faux utilisateurs.

-Attaque Zig-Zag : Dans ce genre d'attaque, l'utilisateur divise sa contribution en plusieurs itérations. Au lieu d'avoir une seule contribution, il aura un nombre de contributions consécutives (ceci augmente sa réputation).

En effectuant ce genre d'attaque, l'utilisateur diminuera la réputation des identités secondaires (ce qui n'est pas très important) et augmentera la sienne.

Les algorithmes présentés dans (Chatterjee *et al*, 2008) sont :

Algorithme de base : il est basé sur les modifications puisqu'il englobe l'ajout du texte. Il existe un bloc initial qui représente l'algorithme pour le calcul de la distance d'édition. Tous les auteurs possèdent initialement une réputation nulle. Dès qu'une nouvelle version d'un article est insérée, l'algorithme met à jour les valeurs des réputations des auteurs liés à cet article. La valeur de la réputation dépend de la qualité des modifications et des réputations des intervenants.

L'algorithme REPUTATION-CAP : C'est une amélioration de l'algorithme de base pour empêcher les utilisateurs d'augmenter leur réputation sans produire un travail utile. Un auteur augmente sa réputation seulement si sa version est comparée à d'autres versions produites par des utilisateurs ayant une bonne réputation (supérieure ou égale à la sienne).

L'algorithme REPUTATION-CAP-NIX : En appliquant l'algorithme REPUTATION-CAP au début du cycle de vie du wiki, les auteurs ne verront pas leurs réputations augmenter puisque l'algorithme initialise leurs réputations à zéro. Cet algorithme combine les deux premiers algorithmes selon des conditions (Chatterjee *et al*, 2008) bien définies pour le calcul de la réputation.

3.6 Synthèse

Cette revue de littérature nous présente les quelques définitions et quelques modèles (méthodologies) entourant notre sujet de recherche, ainsi que les axes de recherche tels que la réputation des auteurs et les degrés de lien entre auteurs. L'idée est de tenter de prédire la solution proposée ultérieurement. Une première proposition serait de concevoir un modèle d'évaluation de la contribution (basé sur un analyseur d'opérations) paramétrée par des variables permettant de calculer une valeur presque exacte de la contribution. Les paramètres seront déduits par l'analyse de l'activité des utilisateurs. Et ce, que ça soit d'un point de vue qualitatif ou quantitatif.

Pour notre solution, nous nous sommes inspirés du modèle identifié dans (Fong et Biuk-Aghai, 2010). En effet, dans ce modèle, les chercheurs ont essayé de résoudre une variante du problème de contribution, l'historique d'édition. Ils ont pris en considération non seulement les actions de base (l'ajout, la suppression et le déplacement), mais aussi les actions abstraites auparavant identifiables qu'à l'aide

d'une intervention humaine. En d'autres termes, cette technique permet de mieux nous renseigner sur la nature de la contribution de l'auteur. Ceci représente un bon progrès pour les modèles d'évaluation de celle-ci.

De plus, il y a le modèle identifié dans (Adler *et al*, 2008). Ce dernier détaille un ensemble de mesures utiles pour le calcul, ainsi que le classement des contributions des auteurs. À partir de (Adler *et al*, 2008), nous avons adopté la mesure de la longévité d'édition, qui permet d'attribuer une valeur à la contribution si elle est acceptée par d'autres auteurs tout au long du cycle de vie de l'article. L'avantage d'utiliser cette mesure, c'est qu'elle est sensible au volume de la contribution, ainsi qu'elle représente un indicateur de choix pour contrer le vandalisme.

Dans ce qui suit, nous présenterons notre méthodologie pour la conception de notre modèle d'évaluation, inspirée des approches précédentes.

CHAPITRE IV

MÉTHODOLOGIE PROPOSÉE

L'évaluation de la contribution dans les environnements de travail collaboratif est devenue un élément très important pour la maintenance du contenu, assurant ainsi sa qualité par la suite. Dans notre sujet de recherche, nous allons nous intéresser aux wikis comme environnement de travail collaboratif. Dans ce document, nous allons proposer un modèle d'évaluation de la contribution des auteurs en nous appuyant sur les travaux identifiés dans notre revue de littérature.

Le chapitre est organisé comme suit : tout d'abord, nous définirons les objectifs de notre modèle. Ensuite, nous examinerons les entités à mesurer pour répondre à la question « Quoi mesurer ? ». Enfin, nous décrirons les méthodes utilisées pour l'évaluation de ces mesures, le tout pour répondre à la question « Comment les mesurer ? ».

4.1 Objectifs

L'objectif général de cette recherche est de concevoir et développer un modèle permettant l'évaluation de la contribution des auteurs dans les wikis. En effet, un tel modèle doit être capable :

- D'identifier les opérations faites lors d'une révision sur un article. Cette identification doit être précise et non approximative. En effet, dans quelques

travaux précédents, les chercheurs se sont intéressés aux opérations de base afin de simplifier le problème. Les opérations identifiées sont l'ajout, la suppression et le déplacement de texte. Toutefois, en considérant seulement ces opérations, la valeur de la contribution ne sera pas significative, puisque le contenu peut être sous plusieurs formes (texte, image, liens internes ou externes, etc.). Autrement dit, un contributeur qui ajoute seulement des images ou des liens de référence ne pourra et ne devra pas avoir la même valeur de contribution qu'un auteur qui ajoute du texte pour enrichir l'article édité. Par la suite, dans notre modèle nous devons pouvoir reconnaître tous les types d'opérations selon la nature du contenu.

- De dénombrer ces opérations, ainsi que les informations que l'on peut en retirer. Par exemple, pour une opération d'ajout de texte, nous pouvons avoir comme renseignement la quantité d'information ajoutée. Aussi, pour l'opération d'ajout de liens (internes ou externes), nous pouvons obtenir comme données le nombre de liens ajoutés. En effet, pour chaque phase d'évaluation, il y a toujours une étape de quantification.
- De définir un indicateur de qualité sur les opérations réalisées. Ce point est très important pour le *score* des contributeurs, car une quantification des actions ne représente pas la valeur réelle de la contribution. Une action peut être de bonne qualité, mais le contraire est valable aussi. De plus, il ne faut pas oublier qu'il existe des wikis ouverts au public, donc on ne peut pas confirmer l'expertise des utilisateurs sans avoir analysé leurs contributions.

Concernant les attaques de vandalisme, il existe déjà des mécanismes propres aux wikis pour les contrer.

Pour récapituler, notre modèle inclura des mesures de quantité et de la qualité, pour donner un *score* à chaque auteur. Ce *score* représente la valeur de sa contribution globale.

4.2 Les entités mesurées

Toujours dans le contexte des wikis, notre modèle analyse un ensemble d'articles. Chaque article possède un nombre fini de versions. Donc, les données analysées sont sous formes textuelles et graphiques (image, tableau, schéma, etc.). La contribution globale d'un auteur peut être identifiée par ces trois éléments :

- La quantité d'information ajoutée à l'article : cet élément garantit l'aspect matériel de la contribution. Autrement dit, cette quantification reflète le produit de l'effort de l'auteur.
- Le nombre d'opérations utiles faites sur un article : cet élément montre l'intérêt et l'implication des auteurs dans le processus de construction d'un article donné. Par exemple, il y a des opérations de mise en forme : un auteur faisant ce genre d'opération montre ses intentions de bien présenter les connaissances au public.
- La qualité du contenu ajoutée, ainsi que la qualité des opérations : comme son nom l'indique, il s'agit de l'élément qui va calibrer et attribuer une valeur à la qualité de la contribution.

Ces trois éléments représentent la base des mesures choisies. Pour la conception de notre modèle, nous nous sommes inspirés du domaine de la construction. En effet, sur chantier, on peut voir un processus de collaboration entre les ouvriers, qui réalisent chacun une tâche. Cette dernière laisse une trace dans la structure. Donc, on peut voir les opérations faites par les ouvriers, et ainsi vérifier la qualité de la réalisation. De plus, la diversité des actions dans la construction et l'utilisation de plusieurs outils

impliquent une évaluation précise de la contribution. D'où l'idée de développer un modèle permettant l'identification des opérations réalisées dans un article, pour pouvoir mieux évaluer la contribution et lui attribuer une valeur quasi précise.
Par analogie avec le domaine de construction, nous avons classé les contributions qu'un auteur peut réaliser sur un article. En effet, lors de l'écriture de ce dernier, on peut trouver deux types de contribution :

4.2.1 Contribution de connaissance

Une contribution est dite de connaissance lorsque l'utilisateur effectue une opération d'ajout de contenu. Dans le cas des wikis, les contributions de connaissance sont extraites à partir des opérations d'ajout de contenu (texte, images, liens externes ou références, tableaux). Pour ce type de contribution, on mesure le nombre d'opérations d'ajout, la quantité d'information ajoutée et leur longévité. Cette dernière représentera le facteur de qualité de la contribution.

4.2.2 Contribution de forme

Une contribution est dite de forme lorsque l'utilisateur effectue des modifications de forme sur le contenu. Dans le cas des wikis, les contributions de forme sont extraites à partir des opérations de correction apportée à l'article pour améliorer sa présentation, telles que les corrections au niveau des mots, ainsi qu'à partir de l'ajout de liens internes pour améliorer la navigation et la suppression des actions de vandalisme. Pour ce type de contribution, on mesure le nombre d'opérations de correction de contenu, le nombre de liens internes ajoutés, et évidemment leur longévité.

Cette classification a pour but de détailler en profondeur la contribution globale d'un utilisateur. Elle serait représentée sous une forme intelligible, à la façon d'une liste de

tâches pour les ouvriers de construction. Ainsi, cette description permettra d'identifier les opérations, les quantifier et leur attribuer une note de qualité.

Dans la section suivante, nous exposerons notre modèle d'évaluation de la contribution.

4.3 Approche

Dans notre modèle, nous avons essayé d'extraire le maximum de mesures permettant de décrire les contributions des auteurs tout au long du cycle de vie d'un article donné. Une fois qu'on obtient la contribution de chaque auteur par article, nous pourrons déduire sa contribution au wiki.

Nous identifions les étapes suivantes :

1. Étape d'initialisation : Nous y transformons les versions de l'article en liste de mots afin de générer les vecteurs nécessaires pour le calcul.
2. Étape d'analyse : Nous y parcourons la liste des versions pour déterminer et suivre les changements.
3. Étape de validation : Il s'agit de la dernière étape. Nous attribuons un score de contribution à chaque auteur selon la formule que nous allons présenter par la suite.

4.3.1 Phase d'initialisation

Il s'agit de la préparation des données pour notre module de calcul. Premièrement, nous réduisons la liste des versions pour obtenir une liste de versions dont les auteurs sont distincts. Autrement dit, quand nous trouvons une succession de versions produites par même auteur, nous ne considérons que sa dernière dans liste. La figure (4.1) donne un exemple sur une liste de versions réduites. Ensuite, pour chaque

version, nous produisons sa liste de mots. Finalement, nous générons trois vecteurs à partir de ces listes.

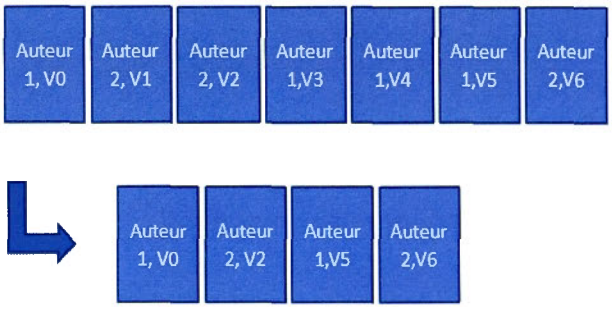


Figure 4.1 Liste de versions réduites

4.3.1.1 Les vecteurs TF (*Term Frequency*)

Il s'agit de la fréquence des termes dans une version. En effet, chaque version sera modélisée sous forme de liste de mots. Pour chaque mot, nous calculons son nombre d'occurrences, puis nous le divisons par le nombre total de mots formant la version. Nous obtenons un vecteur, dont chaque élément représente la fréquence nominale d'un mot. La figure suivante montre un exemple d'un vecteur TF propre à un texte donné.

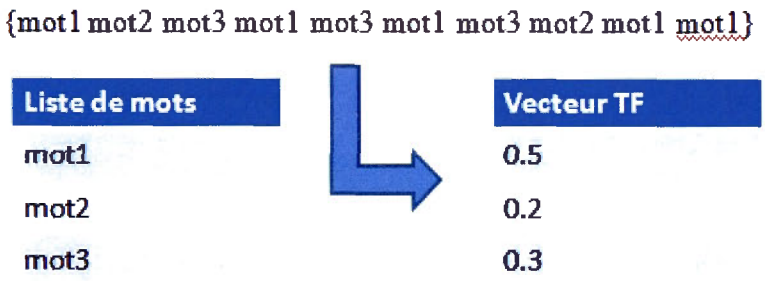


Figure 4.2 Exemple d'un vecteur TF

4.3.1.1 Les vecteurs IDF (Inverse Document Frequency)

Pour chaque version, on génère son vecteur *IDF*. C'est le même principe que le calcul des vecteurs *TF*. La figure suivante illustre un exemple d'un vecteur IDF pour un ensemble de versions (formule 2.1)

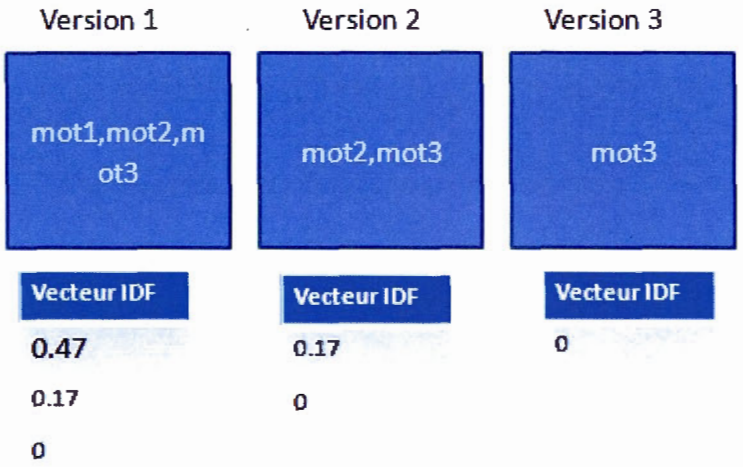


Figure 4.3 Exemple de vecteurs IDF

4.3.1.2 Les vecteurs TF-IDF

Il s'agit tout simplement du produit entre le vecteur TF et le vecteur IDF de chaque version. La figure suivante montre le résultat obtenu par rapport à l'exemple précédent :

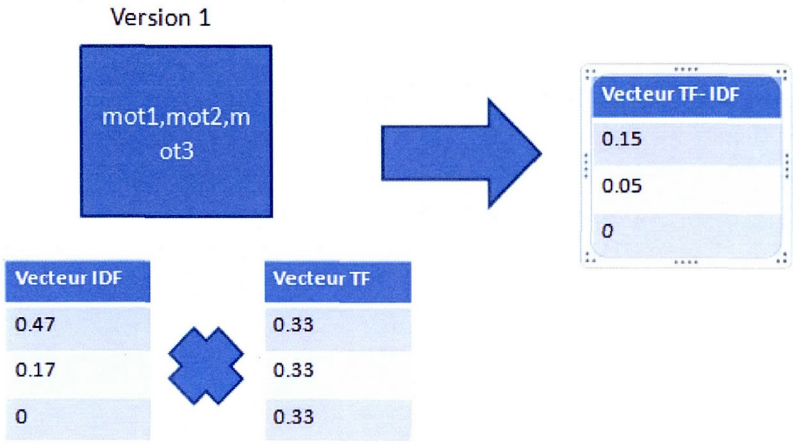


Figure 4.4 Exemple de vecteur TF-IDF

4.3.2 Phase d'analyse

Dans cette phase, nous parcourons la liste de versions d'un article. Tout d'abord, la première version est traitée pour extraire la contribution du premier auteur. Puisqu'il s'agit de la première version, nous aurons comme mesures de contribution le nombre de mots ajoutés, le nombre de liens externes et le nombre de liens internes.

Ensuite, nous comparons chaque version avec celle qui la précède pour extraire les opérations réalisées, afin de mettre à jour les mesures que nous avons choisies et que nous présenterons par la suite.

À ce niveau de calcul, il y a deux types de traitement :

- (1^{er} cas) Traitement de la première contribution : lorsqu'il s'agit de la première contribution de l'auteur sélectionné.
- (2^{e} cas) Traitement du reste des contributions : lorsqu'il s'agit d'une nouvelle contribution de l'auteur sélectionné.

Dans le premier cas, on effectue seulement une quantification de la contribution, puisqu'il s'agit de sa première. Dans notre modèle, nous ne mesurons la qualité de cette contribution qu'à la prochaine intervention de l'auteur.

Dans le deuxième cas, nous quantifions toujours la nouvelle contribution. De plus, nous mesurons la qualité de la contribution précédente. Nous avons choisi les mesures de qualité suivantes :

- Le cosinus similarité ;
- Longévité d'édition ;
- Le taux de rétention.

Les opérations que nous pouvons identifier au niveau textuel sont :

- Insertion de mots : si le contenu existe dans la version courante et non pas la précédente.
- Modification partielle de mots : si le mot a été modifié partiellement.
- Déplacement de mots: si le mot change de position dans l'article.
- Suppression de mots : si le mot existe dans la version précédente et non dans la courante.
- Substitution de mots : si on change un mot pour un autre mot dans la même position.

Ces opérations représentent des actions de bas niveau. À partir de celle-ci, nous extrayons les opérations de haut niveau telles que l'action de correction des fautes d'orthographe. Le but de cette extraction est de décortiquer les actions de bas niveau et de cibler les tâches respectives de chaque utilisateur. En identifiant leurs tâches, nous pourrons par la suite évaluer la contribution globale de chacun des auteurs. Cette dernière étant la combinaison des contributions de connaissance et de forme.

Chaque opération d'insertion correspond à une contribution de connaissance, sauf dans le cas d'ajout d'un lien interne ; celle-ci correspond à une contribution de forme. En effet, l'utilisation des liens internes sert à améliorer la navigation et ne correspond pas à une connaissance ajoutée. Toutefois, il ne faut pas oublier qu'un utilisateur malveillant peut effectuer des opérations d'insertion, alors le modèle doit prendre en compte ce genre d'activité. Le reste des actions seront considérées comme des contributions de forme.

Après l'identification des actions de bas niveau, nous extrayons les actions de haut niveau. Cette étape permet de modéliser les intentions des utilisateurs. Par exemple, lorsqu'un utilisateur effectue des insertions et des suppressions de caractères aux mots déjà existants dans l'article, cela signifie que son intention est de corriger l'écriture de ces mots ; il s'agit de la correction des fautes d'orthographe. Aussi, l'ajout de liens internes nous apprend que l'utilisateur essaie d'améliorer la navigation sur l'article. Voici la liste des actions de haut niveau que nous voulons considérer dans notre modèle :

- Correction des fautes d'orthographe ;
- Amélioration de la navigation à l'aide des liens internes ;
- Déplacement ;
- Indentation : il s'agit de l'utilisation des « : » propre au langage à balise de MediaWiki. Ceci est très utile pour avoir un texte indenté, et par la suite, améliorer la présentation du texte.

➔ Ces actions représentent des contributions de forme.

4.3.3 Phase de validation

L'étape de validation consiste à calculer le score de chaque auteur après avoir fini l'analyse de toutes les versions. Le score dépendra de la valeur du taux de rétention

calculer au niveau de la dernière version. Dans la section suivante, nous décrirons les mesures choisies, ainsi que la méthode utilisée pour l'évaluation de la contribution.

Notation utilisée

Soit A l'ensemble des articles dans un wiki. On désigne l'article par $a_i \epsilon A, i \epsilon [1..I]$.
Soit V_{a_i} l'ensemble des versions d'un article a_i . Chaque version est générée par une révision d'un seul utilisateur, elle noté par V_k.
Soit U l'ensemble des utilisateurs de Wikipédia. On désigne un utilisateur par u_i.

4.4 Paramètres et mesures

Le choix des mesures est l'élément clé de la fiabilité du modèle. Nous avons opté pour deux classes de mesures, de quantité et de la qualité. Le modèle étant à la base un analyseur d'historique d'édition, le premier objectif est implicitement atteint, car notre modèle nous permettra d'identifier les opérations ainsi que les informations utiles pour le calcul. Dans ce qui suit, nous allons définir ces mesures, justifier notre choix et faire le lien avec les objectifs que notre modèle d'évaluation doit atteindre.

4.4.1 Mesures de quantité

Pour pouvoir évaluer la contribution globale d'un utilisateur, nous avons besoin en premier lieu de quantifier les actions, ainsi que les informations qui les suivent. Ces données serviront à visualiser le produit de l'effort des utilisateurs en termes de quantité. Par exemple, le compteur d'édition existant déjà dans Wikipédia.
Toutefois, mesurer les quantités d'information qui circulent tout au long du cycle de vie d'un article ne reflète pas réellement les contributions des auteurs; ces mesures peuvent être utilisées pour déterminer la qualité d'une contribution. Par la suite, nous avons essayé d'obtenir une autre vision par rapport à la quantification des

contributions, en effectuant plusieurs mesures sur les opérations réalisées, ainsi que les quantités d'informations manipulées.

L'ensemble des mesures de quantité permet de satisfaire notre objectif de dénombrer les opérations, ainsi que les données sur l'information qui transite lors de la création d'un article.

Pour chaque utilisateur intervenant sur un article donné, nous déterminons :

4.4.1.1 La quantité de texte

Nous mesurons la quantité de texte ajoutée à l'article tout au long de son cycle de vie. Nous avons besoin de cette mesure, car elle représente une preuve matérielle de l'effort produit par l'utilisateur. Cet effort peut être positif lorsque les ajouts persistent dans l'article et négatif dans le cas contraire. Cette mesure est importante pour déduire la valeur de la contribution de connaissance. Elle est notée par :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ txt(u_i, a_j) = \sum_{v_k} ftxt()_k \qquad (4.1)$$

Avec $ftxt()_k$ est la fonction qui permettra de déterminer tous les ajouts faits par un utilisateur u_i dans sa dernière version v_k. On ne considère que les ajouts qui persistent dans la dernière version de l'article.

4.4.1.2 Le nombre de liens internes

Nous mesurons le nombre de liens internes ajouté à un article. Cette mesure sert à identifier les utilisateurs faisant des améliorations de navigation. Elle est notée par :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ lienInt(u_i, a_j) = \sum_{v_k} flienInt()_k \qquad (4.2)$$

Avec $flienInt()_k$ est la fonction qui permet de calculer le nombre de liens internes ajoutés par un utilisateur u_i dans la version v_k. On ne considère que les liens qui persistent dans la dernière version.

4.4.1.3 Le nombre de liens externes

Nous mesurons le nombre de liens externes ajouté à un article. Cette mesure sert à identifier les utilisateurs qui rajoutent des références vers d'autres pages de Wikipédia ou bien vers d'autres sites web. Il s'agit d'une contribution de connaissance. Elle est notée par :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ lienExt(u_i, a_j) = \sum_{v_k} flienExt()_k \qquad (4.3)$$

Avec $flienExt()_k$ est la fonction qui permet de calculer le nombre de liens externes ajoutés par un utilisateur u_i dans la version v_k. On ne considère que les liens qui persistent dans la dernière version.

4.4.1.4 Le nombre d'images

Nous mesurons le nombre d'images ajouté dans un article. En effet, comme les blocs textuels, l'image représente une autre forme de connaissance. Cette mesure est utilisée pour le calcul de la contribution de connaissance d'un utilisateur donné. Elle notée par :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ nbImg(u_i, a_j) = \sum_{v_k} nbImg()_k \qquad (4.4)$$

Avec $nbImg()_k$ est la fonction qui permet de calculer le nombre d'images ajoutées par un utilisateur u_i dans la version v_k. On ne considère que les images qui persistent dans la dernière version.

4.4.1.5 Le nombre d'opérations de forme

Nous mesurons le nombre d'opérations de forme effectué par l'utilisateur sur un article. En effet, il existe des utilisateurs qui effectuent plusieurs types de modifications pour rendre un article présentable. D'où le besoin de leurs attribuer un *score* pour leurs contributions. Elle est notée par :

$$Pour\ u_i \in U\ ;\ a_j \in A\ ;\ nbOpForme(u_i, a_j) = \sum_{v_k} \left[nbOpCO(u_i, a_j)_k + nbOpAN(u_i, a_j)_k \& + nbOpInd(u_i, a_j)_k + nbOpDep(u_i, a_j)_k \right] \quad (4.5)$$

Avec :

$nbOpCO()_k$ est la fonction permettant d'extraire le nombre de fois que l'utilisateur u_i a effectué une opération de correction de fautes de grammaire.

$nbOpAN()_k$ est la fonction permettant d'extraire le nombre de fois que l'utilisateur u_i a effectué une opération d'amélioration de la navigation.

$nbOpInd()_k$ est la fonction permettant d'extraire le nombre de fois que l'utilisateur u_i a effectué une opération d'indentation.

$nbOpDep()_k$ est la fonction permettant d'extraire le nombre de fois que l'utilisateur u_i a effectué une opération de déplacement.

4.4.2 Mesures de la qualité

Ces mesures sont utilisées comme indicateur de qualité pour les contributions. En effet, la qualité des révisions dépendra de leur persistance tout au long du cycle de vie de l'article. Autrement dit, une révision faite par un utilisateur u_i sont de bonnes qualités si tous les autres utilisateurs n'effectuent pas des modifications dessus et la

gardent après leurs interventions sur l'article. Certes, il ne s'agit pas de l'évaluation de la qualité du contenu. Nous avons choisi comme mesure :

4.4.2.1 La longévité d'édition

Cette mesure permet d'indiquer comment les changements appliqués à une version d'un article persistent dans les versions ultérieures. Elle représente une mesure de la qualité des modifications apportées par la révision r_i générant la version V_i. En effet, elle sert à attribuer une valeur de qualité aux opérations de forme. Elle est notée par :

$$LE = \frac{d(V_{i-1}, V_j) - d(V_i, V_j)}{d(V_{i-1}, V_i)} \qquad (4.6)$$

Avec d() étant la fonction pour le calcul de la distance d'édition entre deux versions.

4.4.2.2 Les taux de rétention

Pour chaque type de contenu, nous calculons son taux de rétention.
Le taux de rétention est égal à la quantité qui persiste sur la dernière version, divisée par la quantité totale. Cette mesure est très pertinente. Elle permet de voir les portions des contributions de connaissance par utilisateur, en ne considérant que la version finale, validée par tous les participants.

Le taux de rétention (texte)

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ; R_{txt} = \frac{txt(u_i, a_j)}{Taille\ de\ l'article} \qquad (4.7)$$

Le taux de rétention (lien interne)

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ; R_{lienInt} = \frac{lienInt(u_i, a_j)}{nombre\ total\ de\ liens\ internes} \qquad (4.8)$$

Le taux de rétention (lien externe)

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ; R_{lienExt} = \frac{lienExt(u_i, a_j)}{nombre\ total\ de\ liens\ externes} \qquad (4.9)$$

Le taux de rétention (image)

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ; R_{img} = \frac{nbImg(u_i, a_j)}{nombre\ total\ d'images} \qquad (4.10)$$

4.4.2.3 Le cosinus de similarité

Comme nous l'avons défini au chapitre deux, il s'agit d'une mesure permettant de nous indiquer le taux de ressemblance entre deux versions, en ayant deux révisions [ri] et [rj] d'un même auteur. Nous calculons le cosinus de similarité entre la version Vi et la version Vj-1. Le résultat obtenu nous indique de combien la version Vi ressemble à la version Vj-1 en pourcentage.

$$cosSim(Vi, Vj) = \frac{\overrightarrow{vi}.\overrightarrow{vj}}{\left\|\overrightarrow{vi}\right\| \cdot \left\|\overrightarrow{vj}\right\|} \qquad (4.11)$$

4.4.3 Contribution globale

Il s'agit du *score* attribué à chaque utilisateur après l'évaluation de sa contribution. Elle est la somme de la contribution de connaissance et la contribution de forme. Elle notée par :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ; ContributionGlobale = CConnaissance + CForme \qquad (4.12)$$

Avec :

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ CConnaissance = \sum_{te\{txt,img,lint,lext\}} R_t \qquad (4.13)$$

$$Pour\ u_i \epsilon\ U\ ;\ a_j \epsilon\ A\ ;\ CForme\ = (LE|cosSim * nbOpForme(u_i, a_j) \qquad (4.14)$$

Pour déterminer la contribution d'un utilisateur, nous calculons tout d'abord les taux de rétention pour chaque type de contenu. Ensuite, nous calculons la longévité d'édition basée sur la distance d'édition pour les opérations de forme ainsi que leur nombre. Enfin, nous combinons ces valeurs pour obtenir une note finale que nous appelons contribution globale. L'analyse traite toutes les versions des articles. Par ailleurs, nous ne nous sommes pas intéressés à l'analyse des pages de discussions, mais ce point pourra représenter une information supplémentaire pour l'évaluation des contributions.

Dans le chapitre suivant, nous allons décrire les modules implémentés, ainsi que les algorithmes utilisés pour extraire les informations utiles au calcul des contributions.

CHAPITRE V

RÉALISATION

Dans ce chapitre, nous présentons l'implémentation de notre modèle de manière détaillée, plus particulièrement sous la forme d'exemple concret basé sur des auteurs wiki fictifs. Ces cas non réels furent mis sur pied afin de pallier aux lacunes décrites plus en détail à la section 5.1.5. La première section est dédiée à la description de ses composantes, telles que les diverses fonctionnalités, les lacunes et les améliorations possibles. Ensuite, la deuxième section s'oriente vers notre analyse des résultats d'une exécution appliquée à un test manuel. Nous terminons en interprétant les divers résultats.

5.1 Implémentation technologique

5.1.1 Environnement et architecture logiciels

En maitrisant le langage de programmation JAVA et en ayant beaucoup d'expérience avec l'IDE Netbeans. Nous avons choisi ce dernier comme environnement de développement. La figure suivante présente l'architecture et l'environnement de notre outil d'évaluation :

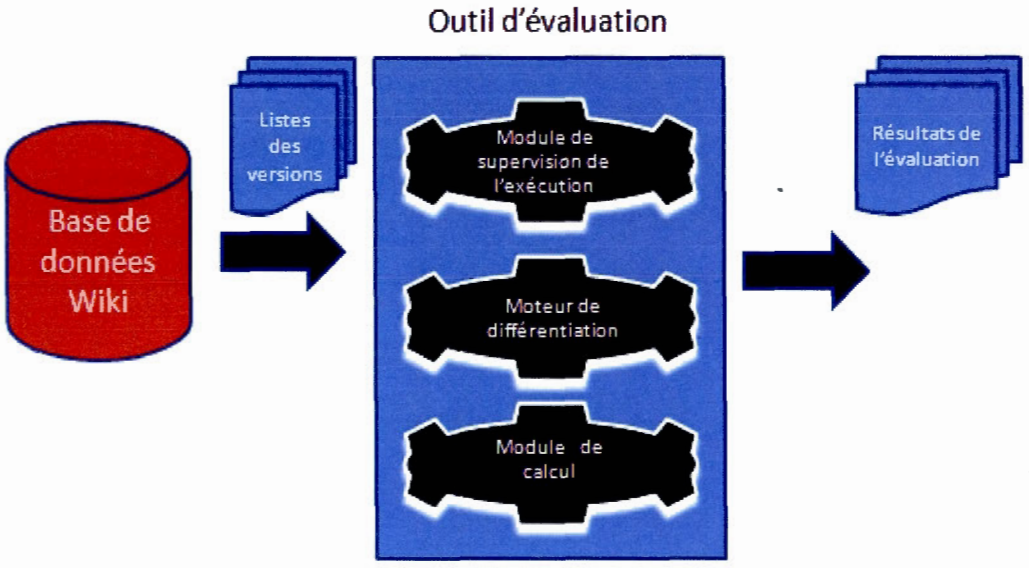


Figure 5.1 Architecture et environnement

Notre objectif est d'avoir un outil qui a un accès direct à la base de données du wiki étudié. Cet accès comprend les versions antérieures et courantes.

Il prend en entrée le contenu textuel des versions d'un article. Le premier module supervise le déroulement de l'évaluation. Le deuxième module s'occupe de l'extraction des différences entre deux versions, en termes d'opérations ou d'actions. Il permet de générer l'historique d'édition. Le troisième module effectue les calculs nécessaires pour déterminer les valeurs des mesures choisies. En sortie, on obtient pour chaque auteur le score que nous avons calculé, ainsi que son taux de rétention.

La division de l'outil en plusieurs modules facilite sa maintenance. Aussi, elle permettra de l'enrichir avec d'autres fonctionnalités.

5.1.2 Classes principales

Pour notre modèle, nous avons défini deux classes principales. La première classe est la classe MoteurDiff. Elle est responsable de la comparaison de deux versions d'un article et elle offre plusieurs autres fonctionnalités qui seront détaillées par la suite. La deuxième classe est la classe MdlContribution. Elle permet de préparer les données pour les traitements de notre algorithme et c'est une implémentation de notre modèle. L'appendice A.6 présente la liste des fonctions implémentées pour chacune de ces deux classes.

5.1.2.1 Classe MoteurDiff

Comme son nom l'indique, il s'agit d'un moteur de différentiation appliqué à des textes. C'est le noyau de notre modèle, puisqu'il permet d'extraire les informations utiles pour notre calcul. L'extraction s'applique sur les diverses versions d'articles. Parmi les fonctionnalités implémentées, nous avons :

- La détection des opérations ;
- La génération du script des changements ;
- La génération des vecteurs (TF, IDF, TF-IDF) ;
- La modélisation des textes des versions sous formes vectorielles ;
- Le calcul du taux de ressemblance à l'aide du cosinus de similarité.

5.1.2.2 Classe MdlContribution

Il s'agit d'une implémentation de l'algorithme représentant notre modèle pour l'évaluation de la contribution. Elle permet les fonctionnalités suivantes :

- Préparation des données pour le calcul ;
- Suivi des propriétaires des mots présents dans l'article ;
- Calcul des taux de rétention ;

- Exécution des étapes de l'algorithme.

5.1.3 Fonctionnalités

5.1.3.1 Préparation des données

La première étape consiste à préparer les données pour l'exécution du programme. Par conséquent, les textes des versions sont transformés en liste de mots. Par la suite, nous déterminons les vecteurs TF-IDF de toutes les versions, car nous en aurons besoin pour le calcul du taux de similarité entre deux versions. L'appendice A.1 détaille une portion du code montrant l'implémentation de cette fonctionnalité.

5.1.3.2 Détection des opérations

La détection des opérations se déroule en deux phases. La première phase consiste à une intersection entre les deux versions afin d'identifier les similitudes directes et proches. En premier lieu, les mots identiques qui se trouvent dans les deux versions sont marqués. Une fois les similitudes directes identifiées, nous passons aux similitudes proches. Une similitude proche se qualifie par deux mots présentant un taux de ressemblance supérieur à 0.8.

Une fois l'intersection définie, la deuxième phase commence. Au cours de celle-ci, nous dégageons les opérations réalisées. Le modèle réussit à identifier les opérations de base suivantes :

- Insertion de mot ;
- Suppression de mot ;
- Déplacement de mot ;
- Substitution de mot ;
- Correction partielle d'un mot.

Dans cette technique, nous utilisons deux tableaux d'entiers. Le premier tableau représente l'ancienne version, et sa taille, définie par des cases, est donc égale au nombre de mots dans l'ancienne version. Idem, le second tableau représente la nouvelle version, et sa taille, définie par des cases, est égale au nombre de mots dans la nouvelle version. Chaque élément du tableau indique le statut du mot. Ce dernier peut avoir différentes valeurs selon l'opération identifiée, la position et la source. Cette dernière définit l'ancienne ou la nouvelle version.

Le tableau ci-dessous présente une description des différents codes utilisés.

Tableau 5.1 Description des codes d'opération

Code	Ancienne version	Nouvelle version
-2	*Les mots sont identiques et à la même position.*	*Les mots sont identiques et à la même position.*
-3	Non présent	Insertion d'un mot à cette position
-4	Suppression d'un mot à cette position	Non présent
-6	Correction partielle du mot	Correction partielle du mot
Nombre positif	Déplacement et le nombre indique la position du déplacement	Il indique la position du mot initial dans le cas où c'est identique

L'appendice A.2 montre un exemple de tableau décrivant les opérations détectées entre deux versions.

5.1.3.3 Génération du script des changements

La génération du script des changements vient juste après l'étape de détection des opérations. Celle-ci effectue une détection au niveau des mots. Cependant, ce qui nous intéresse réellement, c'est une détection par blocs de texte. Dans le but d'éviter la redondance des opérations, tel le déplacement d'une phrase d'un paragraphe à un autre, chaque mot changera de position. Ainsi, cette étape prévoit de comptabiliser un seul déplacement au lieu de comptabiliser n déplacements ; n étant le nombre de mots formant la phrase.

À ce niveau d'exécution, nous obtenons la liste des opérations, ainsi que leur nombre. Les mesures calculées sont :

- Nombre de mots ajoutés
- Nombre de mots supprimés
- Nombre de mots corrigés
- Nombre de déplacements (en nombre de bloc ou de mots)
- Nombre de mots substitués

5.1.3.4 Modélisation vectorielle des textes

Afin de calculer le cosinus similarité, nous avons besoin de deux vecteurs avec des composantes réelles. Donc, il faut pouvoir modéliser les deux textes sous forme de deux vecteurs et les représenter dans un espace donné. Ayant déterminé les vecteurs TF-IDF des deux versions, nous réunissons ces vecteurs en enlevant les mots qui se répètent. Le but est d'avoir deux vecteurs de même dimension pour pouvoir calculer le taux de similarité.

5.1.3.5 Suivi des propriétaires

L'un des problèmes rencontrés dans notre implémentation fut de connaître les propriétaires du contenu. Notre idée consiste à avoir, pendant le parcours des versions, deux listes de mots globaux.

La première liste est celle des mots survivants. Cette liste contiendra, pendant le parcours, les mots qui forment la version courante. Ces mots sont accompagnés du numéro de la version dans laquelle ils ont été ajoutés la première fois.

La deuxième liste est celle des mots morts. Cette liste contiendra, pendant le parcours, les mots qui ont été supprimés. De la même manière que la liste précédente,

ils sont accompagnés du numéro de la version dans laquelle elles ont été ajoutées la première fois.

Ainsi, pour chaque nouveau mot ajouté, on vérifie s'il existe déjà dans la liste des mots morts, ce qui permet de s'assurer que le nouveau mot ajouté n'a pas été récupéré d'une version précédente.

Autrement dit, ces deux listes jouent le rôle d'un annuaire qui définit les origines des mots ; c'est-à-dire, de quelles versions les mots sont originaires. Par la suite, nous pourrons définir pour chaque mot son propriétaire puisque nous aurons l'information sur l'auteur de la version. À chaque itération, nous mettons à jour les deux listes. La figure suivante décrit une évolution des deux listes pour quatre versions données.

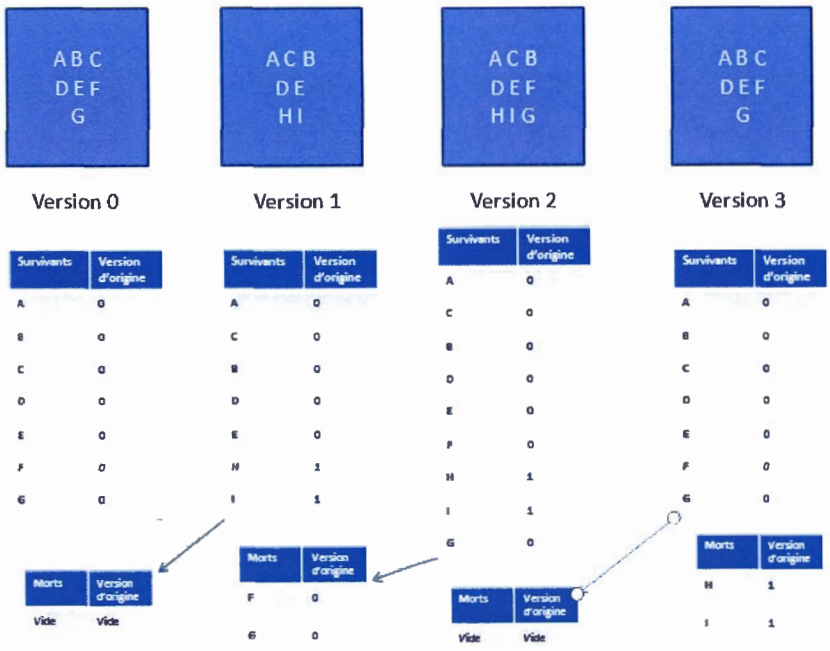


Figure 5.2 Principe de suivi de la propriété

Il est primordial de mentionner que la liste des mots survivants aura toujours les mêmes éléments que la version courante.

5.1.3.6 Calcul des taux de rétention

Le calcul des taux de rétention se fait à partir des listes de suivi. En effet, la liste des mots survivants nous informe sur les origines des mots. Donc, nous pouvons connaître les propriétaires à partir des numéros de version. Prenons l'exemple de la version suivante accompagnée de la liste des mots survivants.

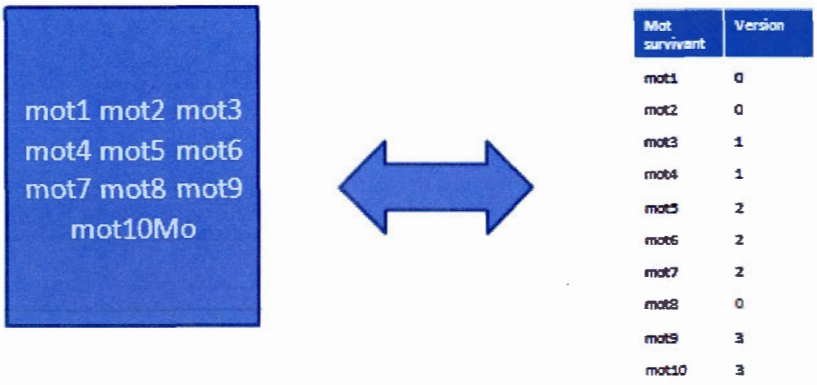


Figure 5.3 Exemple d'une version et sa liste de mots survivants

Supposons que nous connaissons les auteurs de chaque version. Le tableau suivant présente, pour chaque auteur, les versions qu'il a produites.

Tableau 5.2 Auteurs / Versions produits

Auteur	Versions produites
Auteur X	3
Auteur Y	1

Auteur Z	2

Nous obtenons alors les taux suivants :

Tableau 5.3 Exemple de taux de rétention

Auteur	**Taux de rétention(%)**
Auteur x	50
Auteur y	20
Auteur z	30

5.1.4 L'algorithme

La conception et le développement du modèle se sont déroulés en trois itérations. Pour étudier les contributions sur un ensemble d'articles, il nous fallait un module qui permet d'abord de voir la différence entre deux versions. D'où le besoin à la première itération d'implémenter un moteur de différentiation. Après chaque révision, il y a de nouveaux mots ajoutés ou partiellement modifiés ou supprimés. Nous nécessitions un mécanisme de suivi des propriétaires, pour pouvoir calculer les taux de rétentions. C'est ce qui a été implémenté à la deuxième itération. À la dernière itération, nous avons implémenté l'algorithme global de notre modèle en incorporant les mesures à prélever.

Nous avons appliqué deux types de raffinement. Pour le premier, il s'agit d'un raffinement au niveau des données afin d'alléger les calculs, tel expliquer précédemment. Pour le deuxième, il s'agit d'un raffinement au niveau des algorithmes. En effet, à chaque itération de développement, nous effectuons des tests locaux du module implémenté afin de s'assurer de son bon fonctionnement.

Pour finir, voici une description des étapes définissant notre algorithme :

- 1re étape : Initialisation ;
- 2e étape : Parcours des versions. Pour chaque version appartenant au corpus, effectuer ceci :
  - Récupérer la version précédente ;
  - Détecter les opérations et générations du script des changements ;
  - Vérifier s'il s'agit d'une première contribution pour un auteur ou sa nouvelle ;
  - S'il s'agit de la première contribution de l'auteur, nous saisissons les mesures calculées. Sinon, nous déterminons la longévité d'édition, ainsi que son taux de rétention par rapport à sa dernière contribution ;
  - Mise à jour des mesures ;
- 3es étapes : Calcul des scores.

À l'appendice A.4, vous trouverez le code pour notre fonction d'exécution de l'algorithme.

5.1.5 Lacune et alternative

Pendant la phase d'implémentation de notre modèle, nous avons rencontré des problèmes en utilisant l'api JWPL (**Java Wikipédia Library**). Celle-ci nous permet d'interroger une base de données wiki et offre des fonctions utiles pour récupérer le contenu des versions sous format WikiText, ce qui nous permet de pouvoir identifier l'indentation, les liens internes et externes et les actions d'amélioration de la navigation. Cependant, puisqu'il perdure encore un problème d'exécution, où la commande est exécutée, mais aucune information n'est récupérée, les parties concernant l'identification de l'indentation, l'extraction des opérations

d'améliorations de la navigation, ainsi que les types de contenu, n'ont pas été implémentés.

Nous avons donc utilisé la formule du score suivante afin d'y palier :

$$score = \frac{Nombre\ de\ révision}{\sum cosSim*(I+S+Dep+Sub+Corr)} + Taux\ de\ rétention \qquad (5.1)$$

Tout au long de l'exécution, nous avons prélevé plusieurs autres mesures afin de voir le comportement de notre outil et d'interpréter les résultats obtenus à la fin de l'exécution.

5.2 Analyse des résultats

Dans cette section, nous allons présenter les résultats obtenus par notre programme. Afin de juger de l'exactitude de notre modèle, nous avons choisi de comparer les résultats d'analyse de notre programme avec une analyse faite par un être humain.

5.2.1 Présentation des données

Nous avons appliqué le test sur un texte extrait d'un article sur Wikipédia, soit l'article Wiki. Nous avons considéré dix versions de celui-ci. En ce qui concerne les auteurs, nous avons généré une distribution aléatoire de trois auteurs. Nous supposons que chaque auteur a produit au minimum une version. Les versions sont définies dans le code. Le tableau ci-dessous présente cette distribution, en sachant que chaque case désigne une version :

Tableau 5.4 Distribution aléatoire d'auteurs

Aut0	Aut0	Aut1	Aut2	Aut2	Aut2	Aut0	Aut1	Aut0	Aut0
V0	V1	V2	V3	V4	V5	V6	V7	V8	V9

Après l'initialisation, nous ne considérons pas les versions successives d'un même auteur. Nous obtenons alors la liste de version suivante :

Tableau 5.5 Liste raffinée

Aut0	Aut1	Aut2	Aut0	Aut1	Aut0
V1	V2	V5	V6	V7	V9

En appendice A.5, nous présentons les dix versions prises en compte pour le test de l'outil.

5.2.2 Mesures implémentées

Les mesures implémentées sont divisées en deux ensembles. Dans le premier ensemble, nous trouvons les mesures de quantité :

- Le nombre de mots insérés ;
- Le nombre de mots supprimés ;
- Le nombre de mots déplacés ;
- Le nombre de mots substitués ;
- Le nombre de mots qui ont subi des corrections partielles ;

Dans le deuxième ensemble, nous trouvons les mesures de la qualité :

- La distance d'édition (utilisée pour calculer la longévité d'édition d'après la formule (4.6));
- La longévité d'édition ;
- Le cosinus de similarité (utilisé comme indicateur de ressemblance entre deux interventions successives d'un même auteur) ;
- Le taux de rétention.

5.2.3 Résultats obtenus

Tout au long de l'exécution, nous avons prélevé différentes mesures afin de modéliser les contributions, et voir leur évolution tout au long du cycle de vie de l'article. Nous avons choisi d'avoir des versions réduites de taille dans le but de faciliter l'analyse manuelle, et par la suite comparer les résultats avec le programme.

Pour pouvoir tester l'outil, il suffit d'exécuter le fichier principal du programme. Comme l'exemple de test est introduit manuellement dans le code, nous aurons toujours le même résultat. Donc, en apportant des changements dans les versions ou dans la distribution des auteurs, nous obtenons des résultats différents. Ceci permettra d'analyser encore plus le comportement de l'outil ainsi que son efficacité. Les versions se trouvent dans le fichier principal du programme, contenant la fonction Main().

En appendice A.7, nous trouvons une analyse manuelle des versions. Il s'agit de la progression des interventions des trois (3) auteurs sur les six (6) versions formant le corpus raffiné.

Nous décrivons ci-dessous les tableaux présentant les mesures propres à chaque auteur, ainsi qu'un résumé des opérations détectées sur chaque version produite.

Pour l'auteur «Aut0» on obtient :

Tableau 5.6 Mesures de l'auteur «Aut0»

Mesure	Version 1	Version 6	Version 9
Insertion (mots)	11	10	2
Suppression (mots)	0	0	1
Déplacement (opérations)	0	0	0
Substitution (mots)	0	0	1
Correction (mots)	0	0	4
Distance d'édition (caractères)	65	145	11
Longévité d'édition	0.25	13.18	X
Cosinus de similarité	0.14	0.81	X
Taux de rétention	0.21	0.39	X

L'auteur «Aut0» a produit trois (3) révisions. Pour la première, l'outil ne détecte que des insertions. Puisqu'il s'agit de la première version produite et que nous la comparons à une version vide. La même chose pour la distance d'édition, puisqu'il s'agit de la distance entre la première version et une version vide. Le reste des opérations détectées sont de valeur nulle. Dans les interventions suivantes, nous calculons les trois autres mesures de la qualité propres aux révisions précédentes et nous déterminons les opérations réalisées dans chaque version. Il y a une exception pour la troisième révision, puisqu'il s'agit de sa dernière. En effet, dans notre modèle nous ne prenons pas en considération la dernière version d'un auteur. Nous n'identifions que les opérations réalisées. En ce qui concerne les mesures de la qualité, elles sont déterminées une fois que l'auteur réalise une nouvelle intervention. Le calcul de ces mesures se fait toujours entre deux révisions. Le but de ce choix est d'alléger les calculs, car nous n'aurons pas à effectuer des calculs à chaque production d'une version.

Pour l'auteur «Aut1», nous obtenons :

Tableau 5.7 Mesure de l'auteur «Aut1»

Mesure	Version 2	Version 7
Insertion (mots)	9	13
Suppression (mots)	0	0
Déplacements (opérations)	0	0
Substitution (mots)	0	0
Correction (mots)	1	0
Distance d'édition (caractères)	46	76
Longévité d'édition	0.84	X
Cosinus de similarité	0.64	X
Taux de rétention	0.13	X

L'auteur «Aut1» a produit deux (2) révisions. Tel que mentionné précédemment, les mesures de la qualité sont celles de sa première révision.

Pour l'auteur «Aut2», nous obtenons :

Tableau 5.8 Mesure de l'auteur «Aut2»

Mesure	Version 5
Insertion(mots)	9
Suppression(mots)	14
Déplacements(opérations)	6
Substitution(mots)	16
Correction(mots)	0
Distance d'édition (caractères)	64
Longévité d'édition	X
Cosinus de similarité	X
Taux de rétention	X

L'auteur «Aut2» a produit une seule version. En ce qui concerne ses statistiques, nous avons déterminé que les opérations de base et son taux de rétention finale.

Le tableau ci-dessous présente les scores, ainsi que les taux de rétentions finaux pour chaque auteur :

Tableau 5.9 Score, taux rétention / Auteur

	Auteur 0	**Auteur 1**	**Auteur 2**
Taux de rétention	0.49	0.20	0.30
Score	0.66	0.30	0.32

5.2.4 Interprétation

En analysant les résultats et en les comparant à notre analyse manuelle, nous constatons que notre modèle a atteint ses objectifs. Ce malgré le manque d'information à analyser, puisque nous n'avons pas réussi à résoudre le problème avec l'API JWPL (Java Wikipedia Library), pour pouvoir le tester sur un cas réel. En effet, avec notre moteur de différentiation, nous avons pu identifier avec précision les opérations de base, les dénombrer, ainsi que leurs attribuer des indicateurs de qualité. D'après les scores obtenus, l'auteur «Aut0» a le plus contribué, ce qui concorde avec notre analyse manuelle. En ce qui concerne l'auteur «Aut2», il a un taux de rétention plus élevé que celui de l'auteur «Aut1», mais les deux auteurs ont deux scores très proches. Ceci s'explique par le fait que dans la formule implémentée, nous prenons en considération le nombre de révisions et la somme des opérations, multipliés par son taux de changement, qui est en réalité le cosinus de similarité. Or, pour l'auteur «Aut2», nous n'avons pas calculé le cosinus de similarité pour sa révision, puisque pour notre modèle, il en a fait qu'une seule (révision). Celui-ci sera déterminé que s'il effectue une révision supplémentaire future. Donc, même si l'auteur «Aut2» a effectué plus d'opérations que l'auteur «Aut1», il a obtenu un score proche de ce dernier.

De surcroît, notre module de suivi de la propriété nous a fourni des valeurs presque parfaites. Les taux de rétention sont très proches de l'analyse manuelle, ce qui nous montre l'efficacité de ce module.

Dans ce chapitre, nous avons présenté les résultats obtenus par notre programme, appliqués à un seul exemple manuel. Il s'agit d'une partie de notre modèle, puisque les mesures sont prises seulement sur les opérations de base. Un autre module complémentaire sera développé par la suite pour rajouter les fonctionnalités d'analyse requise à l'identification des opérations abstraites. Réussir à modéliser les contributions avec un nombre élevé de mesures diversifiées pourrait permettre une meilleure évaluation, de même qu'améliorer les traitements internes des modules afin de réduire le temps d'exécution. Finalement, il est important de compléter le tout en greffant un tel outil sur une application wiki afin de pouvoir le tester spécifiquement et d'en améliorer ses fonctionnalités encore plus.

CHAPITRE VI

CONCLUSION

Dans ce mémoire, nous avons présenté une approche pour l'évaluation de la contribution des chercheurs dans un environnement wiki. Tout d'abord, nous nous sommes intéressés à plusieurs autres approches en identifiant les mesures utilisées dans celles-ci. Nous avons remarqué une grande différence d'intérêt dans les approches. En effet, il y a ceux qui étudient les réseaux sociaux formés dans ce genre d'environnement pour modéliser les contributions des utilisateurs, puis il y a ceux qui analysent le contenu afin d'extraire les natures des opérations. À partir de ces opérations, ils essaient de modéliser les contributions. D'ailleurs, notre intérêt s'est porté pour le deuxième type d'approche.

Nous avons implémenté un programme qui permet d'analyser une liste de versions et d'extraire les opérations de base telles que les insertions, les suppressions et les déplacements. Nous avons inclus un mécanisme de suivi des propriétaires des mots. Ce dernier permet de connaître ce que possède chaque auteur dans une version donnée, et par la suite, calculer les taux de rétention.

Notre objectif principal étant de concevoir et de développer un modèle permettant l'évaluation de la contribution des auteurs dans un environnement collaboratif, nous avons été capables d'identifier les opérations faites lors d'une révision sur un article. Il s'agit des opérations de base telles que l'ajout, la suppression, le déplacement, la

correction et la substitution de mots. Ainsi, nous avons répondu à la question clé de notre projet de recherche : Quelles sont les opérations réalisées ?

De plus, nous sommes parvenus à dénombrer ces opérations. Par la suite, nous leur avons attribué des indicateurs de qualité. Il s'agit de longévité d'édition et du cosinus de similarité entre deux versions. Par ailleurs, nous n'avons pas réussi à exploiter les opérations identifiées afin d'en extraire les deux types de contributions, qui sont les contributions de formes et de connaissances. À cause du problème d'accès à notre application wiki de test, nous n'avions pas accès aux données sous forme de wikitext, qui apporte des indications supplémentaires pour identifier ces opérations. Cependant, le modèle implémenté est très satisfaisant pour la recherche, car il inclut des mesures de quantité et de la qualité, il attribue un score global reflétant la contribution de chaque auteur.

En ce qui concerne les résultats de notre programme, nous avons obtenu des valeurs proches de l'analyse manuelle. Nous constatons que l'écart entre les valeurs manuelles et les valeurs calculées par notre programme est dû à la phase de correspondance entre les versions. En effet, il s'agit d'une étape initiale lors de la différentiation. Elle correspond à trouver les ressemblances ou les mots identiques qui se trouvent dans les deux versions.

D'où l'idée d'améliorer le module de différentiation, utilisé par l'approche d'Adler *et al* (2008), puisque tout repose sur ce module. Il permettra de bien identifier les opérations de base et leurs types de contenu. Notre modèle pourra donc servir de fondation à d'autres chercheurs afin de développer un compilateur d'opérations, qui permettront d'extraire les opérations abstraites. Il offrira une source de données supplémentaire afin d'enrichir la formule de la contribution globale, pour obtenir une valeur du score plus précise. Nous sommes sûrs que de tels ajouts permettraient de perfectionner notre algorithme et d'amener à un niveau supérieur l'analyse des

contributions. La popularité des wikis étant à son apogée, nul doute est que l'approfondissement de ces algorithmes permettra d'améliorer le concept des environnements collaboratifs, au point où la diffusion du savoir et son partage sera d'office d'une qualité nécessaire à l'amélioration connaissance collective.

BIBLIOGRAPHIE

Adler, B.T., et de Alfaro, L. 2007. A content-driven reputation system for the wikipedia. *Proceedings of the 16th international conference on World Wide Web*, (May 08-12, 2007, Banff, Alberta, Canada) ACM.

Adler, B.T., de Alfaro, L., et Pye, I., 2010. Detecting Wikipedia Vandalism Using WikiTrust. *Rapport de laboratoire pour le PAN au CLEF*.

Adler, B.T., de Alfaro, L., Pye, I., et Raman, V. 2008. Measuring author contributions to the Wikipedia. *Proceedings of the 2008 International Symposium on Wikis* (Porto, Portugal, 2008). *WikiSym 08 ACM*.

Anthony D, Smith S, Williamson T. The quality of open source production: Zealots and Good Samaritans in the case of Wikipedia. *Technical Report TR2007-06*. Dartmouth College 2007.

Arazy, O. *et al.* (2010). Recognizing contributions on wikis: Authorship, Categories, Algorithms, and Visualizations. *Journal of the American Society for Information Science and Technology*, 61, pages 1166–1179.

Burns R. et Long D. A linear time, constant space differencing algorithm. *Performance, Computing, and Communication Conference (IPCCC), IEEE International, (1997)*. pages *429–436*.

Chatterjee, K., de Alfaro, L., et Pye, I. Robust Content-Driven Reputation. *Proceedings of the 1st ACM workshop on Workshop on AISec*. AISec 08.

Ebersbach A, Glaser M, Heigl R et Warta A. (2008). Wiki: Web Collaboration. *Springer Science and Business Media*, ISBN 3-540-35150-7, pages 11-161.

Fong P.K.-F et Biuk-Aghai, R. P. Visualizing author contribution statistics in Wikis using an Edit Significance Metric. *Proceedings of the 7th International Symposium on Wikis and Open Collaboration, WikiSym '11.*

Fong P.K.-F et Biuk-Aghai, R. P. What did they do? Deriving high-level edit histories in wikis. *Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10, ACM 2010.*

Gohr, A., Hüttemann, D., Faust, D., et Fuchs-Kittowski, F. Quality check with DokuWiki for instant user feedback". *Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10.*

Javanmardi, S *et al.* User contribution and trust in Wikipédia. *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on.*

Korfiatis N, Naeve A. (2005). Evaluating wiki contributions using social networks: A case study on wikipedia. *Proceedings of the First International Conference on Metatadata and Semantics Research.*

Moturu, S. T., et Liu, H. Evaluating the trustworthiness of Wikipedia articles through quality and credibility. *Proceedings of the 5th International Symposium on Wikis and Open Collaboration, WikiSym '09, ACM 2009.*

Muller-Birn, C *et al.* A Composite Calculation for Author Activity in Wikis: Accuracy Needed. *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences on (Volume:1).*

Myers, E. An o(ND) difference algorithm and its variations. *Algorithmica, 1(2),* (1986),pages 251–266.

Negre, E. (2013). Comparaison de textes: quelques approches. *Cahier du LAMSAD.*

Sesia J. Zhao *et al.* Investigating the determinants of contribution value in Wikipedia. *International Journal of information Management 33 (2013) pages 83-92.*

Sheppard, S. A., et Terveen, L. Quality is a Verb: The operationalization of data quality in a citizen science community. *Proceedings of the 7th Annual International Symposium on Wikis and Open Collaboration. WikiSym 2011.*

Tang, L.V. S, Biuk-Aghai, R. P., et Fong, S. 2008. A Method for Measuring Co-authorship Relationships in MediaWiki. *Proceedings of the 2008 International Symposium on Wikis, (Porto, Portugal, 2008).*

Tichy W. The string-to-string correction problem with block move. *ACM Transactions on Computer Systems, 2(4), (1984).*

Wierzbicki, A, Turek P, et Nielek, R. Learning about team collaboration from Wikipedia edit history. *Proceedings of the 6th International Symposium on Wikis and Open Collaboration, WikiSym '10.*

APPENDICE A

A.1 Préparation des données.

```
for(int i=0;i<this.taille;i++){
    StringTokenizer st = new StringTokenizer(listesversionTxt[i]);
    this.listeVersionCorpus.add(listesversionTxt[i]);
    ArrayList<String> list = new ArrayList<String>();
    while (st.hasMoreTokens()){
    tmp=this.mtrDiff.retirerPonctuation(st.nextToken());
    for(int j=0;j<tmp.length;j++){
        list.add(tmp[j]);
    }
    }
this.corpusTxt.add(list);
this.corpusWikiTxt.add(listesversionWiki[i]);
this.contributeurs.add(listContrib[i]);
```

A.2 Exemple de tableau d'entier pour la détection des opérations.

En ayant ces deux versions :

V1 = {une applicstion web qui permet la création de pages à l'intérieur}

V2 = {Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur}

On obtient :

Pour V1 : 3 | 4 | 5 | 6 | 7 | 8 | 9 | 16 | 17 | 18 | 19 |

Pour V2 : -3 | -3 | -3 | 0 | -6 | 2 | 3 | 4 | 5 | 6 | -3 | -3 | -3 | -3 | -3 | -3 | 7 | 8 | 9 | 10 |

A.3 Distance de Levenshtein

```
entier DistanceDeLevenshtein(caractere chaine1[1..longueurChaine1],
                             caractere chaine2[1..longueurChaine2])
  // d est un tableau de longueurChaine1+1 rangées et longueurChaine2+1 colonnes
  declarer entier d[0..longueurChaine1, 0..longueurChaine2]
  // i et j itèrent sur chaine1 et chaine2
  declarer entier i, j, coût

  pour i de 0 à longueurChaine1
      d[i, 0] := i
  pour j de 0 à longueurChaine2
      d[0, j] := j

  pour i de 1 à longueurChaine1
      pour j de 1 à longueurChaine2
          si chaine1[i] = chaine2[j] alors coût := 0
                                     sinon coût := 1
          d[i, j] := minimum(
                                     d[i-1, j  ] + 1,     // effacement
                                     d[i,   j-1] + 1,     // insertion
                                     d[i-1, j-1] + coût   // substitution
                                   )

  renvoyer d[longueurChaine1, longueurChaine2]
```

http://fr.wikipedia.org/wiki/Distance_de_Levenshtein

A.4 Fonction d'exécution

```
public void executer(){
    initialisation();
    System.out.println("Initialisation  ... OK");
    calculDesContributions();
    System.out.println("Calul des contributions  ... OK");
    System.out.println();
    ArrayList<Double> trAll=this.tauxRetentionAll();

    for(int i=0;i<this.listeContribution.size();i++){
        this.listeContribution.get(i).calculerScore();
        System.out.println("Stats de l'auteur "+this.listeContribution.get(i).id+" :");
        affichageStatUsr(i);
        System.out.println();
        affichageDE(i);
        System.out.println();
        affichageLE(i);
        System.out.println();
        affichageTR(i);
        System.out.println();
        System.out.println("SCORE = "+this.listeContribution.get(i).score+" , Taux de "
                + "rétention Finale = "+trAll.get(i));
        System.out.println();
    }
}
```

A.5 Versions utilisées pour le teste

Version 0 = "une applicstion web";

Version 1 = "une applicstion web qui permet la création de pages à l'intérieur ";

Version 2 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur ";

Version 3 = "Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web.";

Version 4 = "Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu";

Version 5 = "Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale";

Version 6 = "Un wiki est une application web qui permet la création la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale";

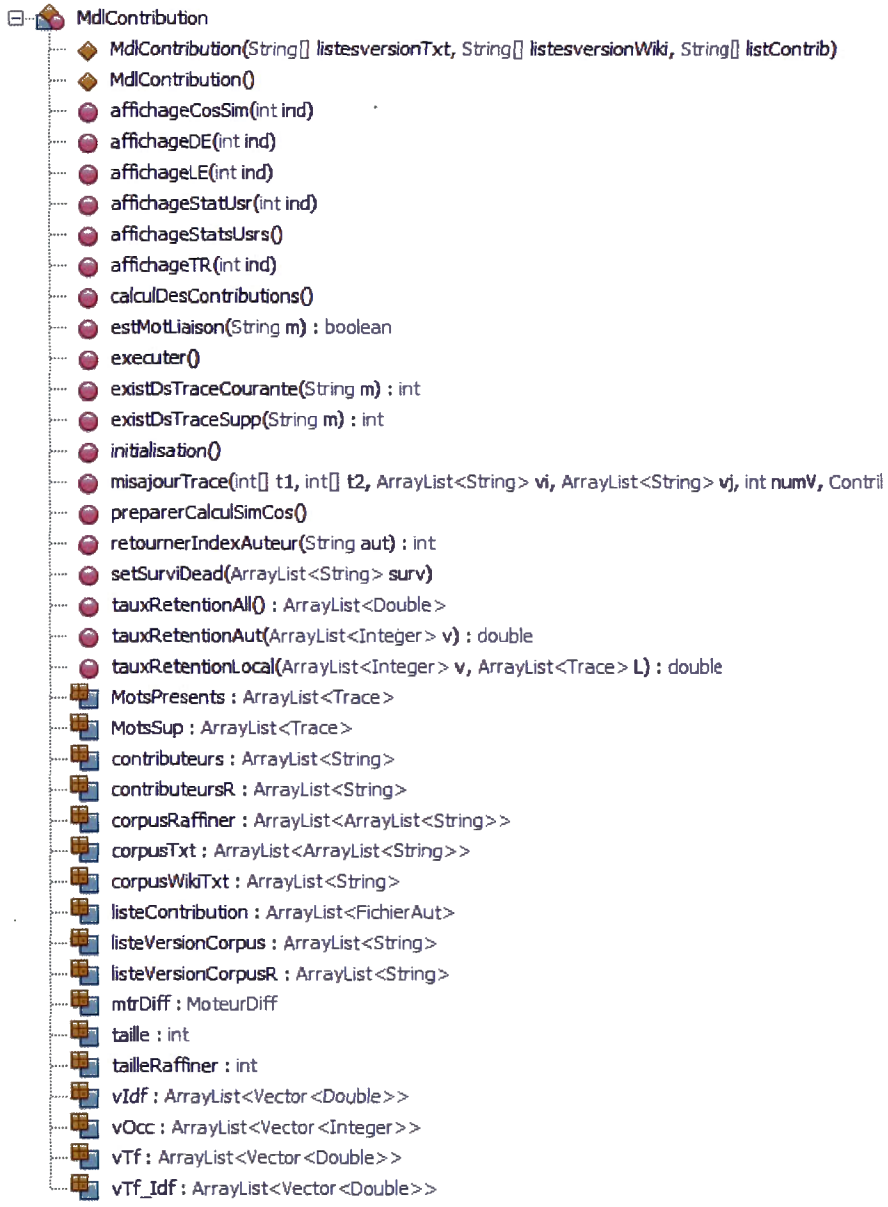
Version 7 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale, tandi que la structure explicite emerge en fonction du besoin d'un usager";

Version 8 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale, tandis que la structure explicite émerge en fonction du besoin d'un usager";

Version 9 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale, tandis que la structure explicite émerge en fonction des besoins des usagers.";

A.6 Classes principales

A.6.1 MdlContribution



Code source :

```
package javaapplication2;

import java.util.ArrayList;
```

```
import java.util.LinkedList;
import java.util.Random;
import java.util.StringTokenizer;
import java.util.Vector;
import javaapplication2.diff_match_patch.Diff;

public class MdlContribution {

   ArrayList<String> listeVersionCorpus;
   ArrayList<String> listeVersionCorpusR;
   ArrayList<ArrayList<String>> corpusTxt;
   String[] listeVersionTxt;
   ArrayList<String> corpusWikiTxt;
   ArrayList<String> contributeurs;
   ArrayList<String> contributeursR;
   //ArrayList<String> proprios;//Liste des contributeurs qui posséde des contribution exisstante
                        //dans la derniere version c'est à dire la version courante
   int taille;
   int tailleRaffiner; //Lors qu'on trouve des versions successives d'un même contributeur
                //on considére que la derniere de cette liste
   ArrayList<ArrayList<String>> corpusRaffiner;
   MoteurDiff mtrDiff; //Moteur de différenciation  entre les versions , implémente des méthodes utiles
                //à notre calcul
   ArrayList<FichierAut> listeContribution;

   ArrayList<Trace> MotsPresents;
   ArrayList<Trace> MotsSup;
   ArrayList<Vector<Integer>> vOcc;
   ArrayList<Vector<Double>> vTf;
   ArrayList<Vector<Double>> vIdf;
   ArrayList<Vector<Double>> vTf_Idf;

   Vector<Integer> indiceVersionR;


   public int retournerIndexAuteur(String aut){
      int i=0;

      while(i<this.listeContribution.size() && this.listeContribution.get(i).id.compareTo(aut)!=0 )
         i++;

      if(i==this.listeContribution.size())
         return -1;

      return i;
   }

   public  MdlContribution(String[] listesversionTxt,String[] listesversionWiki,String[] listContrib){
```

```
//Création du corpus
this.corpusTxt=new ArrayList<ArrayList<String>>();
this.corpusWikiTxt=new ArrayList<String>();
this.contributeurs=new ArrayList<String>();
this.contributeursR=new ArrayList<String>();
//this.proprios=new ArrayList<String>();
this.corpusRaffiner=new ArrayList<ArrayList<String>>();
this.mtrDiff=new MoteurDiff();
this.taille=listesversionTxt.length;
this.listeContribution=new ArrayList<FichierAut>();
this.listeVersionCorpus=new ArrayList<String>();
this.listeVersionCorpusR=new ArrayList<String>();
this.listeVersionTxt=listesversionTxt;
this.indiceVersionR=new Vector<Integer>();

String[] tmp;

for(int i=0;i<this.taille;i++){
   tmp=this.mtrDiff.retirerPonctuation(listesversionTxt[i]);
   this.listeVersionCorpus.add(this.mtrDiff.lsttoString(tmp));

   StringTokenizer st = new StringTokenizer(this.listeVersionCorpus.get(i));

   ArrayList<String> list = new ArrayList<String>();
   while (st.hasMoreTokens()){
   //tmp=this.mtrDiff.retirerPonctuation();
      list.add(st.nextToken());

   }
this.corpusTxt.add(list);
this.corpusWikiTxt.add(listesversionWiki[i]);
this.contributeurs.add(listContrib[i]);

}

 this.MotsPresents=new ArrayList<Trace>();
 this.MotsSup=new ArrayList<Trace>();
this.tailleRaffiner=0;
}

public MdlContribution(){
   this.MotsPresents=new ArrayList<Trace>();
  this.MotsSup=new ArrayList<Trace>();
}

public void setSurviDead(ArrayList<String> surv){
  for(int i=0;i<surv.size();i++){
     this.MotsPresents.add(new Trace(surv.get(i),0));
```

```
    }
  }
  public void initialisation(){
    int i=0,j;
    while(i<this.taille){
      j=i+1;
      while(j<this.taille && this.contributeurs.get(i)==this.contributeurs.get(j)){
        j++;
      }
      //System.out.println(listeAuteur[j-1]);
      this.corpusRaffiner.add(this.corpusTxt.get(j-1));
      this.indiceVersionR.add(j-1);
      this.contributeursR.add(this.contributeurs.get(j-1));
      this.listeVersionCorpusR.add(this.listeVersionCorpus.get(j-1));

      i=j;

      this.tailleRaffiner++;
    }

    /*(i=0;i<this.corpusRaffiner.get(0).size();i++){
      this.proprios.add(this.contributeurs.get(0));
    }*/

    //Initialisation de la premiere contribution
    for(int k=0;k<this.corpusRaffiner.get(0).size();k++){
      this.MotsPresents.add(new Trace(this.corpusRaffiner.get(0).get(k),0));
    }
    this.listeContribution.add(new FichierAut(this.contributeursR.get(0)));
    this.listeContribution.get(0).contribution.nbOperationI=this.MotsPresents.size();
    this.listeContribution.get(0).derniereVersionL=this.corpusRaffiner.get(0);
    this.listeContribution.get(0).derniereVersionS=this.listeVersionCorpusR.get(0);
    this.listeContribution.get(0).avantDerniereVersionS="";
    this.listeContribution.get(0).listeVersionAut.add(0);
    DamerauLevenshteinAlgorithm defonc=new DamerauLevenshteinAlgorithm(1, 1, 1, 1);


this.listeContribution.get(0).distanceEditionList.add(defonc.execute("",this.listeVersionCorpusR.get(0)
));
    this.listeContribution.get(0).lstnbOperationI.add(Double.valueOf(this.MotsPresents.size()));
        this.listeContribution.get(0).lstnbOperationS.add(0.0);
        this.listeContribution.get(0).lstnbOperationD.add(0.0);
        this.listeContribution.get(0).lstnbOperationSub.add(0.0);
        this.listeContribution.get(0).lstnbOperationCorrectionMot.add(0.0);

    /*for(int k=1;k<this.tailleRaffiner;k++){
      this.listeContribution.add(new FichierAut(this.contributeursR.get(k)));
    }*/
```

```
    this.vOcc=new ArrayList<Vector<Integer>>();
    this.vTf=new ArrayList<Vector<Double>>();
    this.vIdf=new ArrayList<Vector<Double>>();
    this.vTf_Idf=new ArrayList<Vector<Double>>();
    this.preparerCalculSimCos();
    //System.out.println(this.tailleRaffiner);
  }

  public int existDsTraceCourante(String m){
    int res=-1;
    boolean cont=false;
    for(int i=0;i<this.MotsPresents.size()&& !cont;i++){
      if(this.MotsPresents.get(i).mot.compareTo(m)==0)
      {cont=true;
      res=i;
      }
    }
    return res;
  }
  public int existDsTraceSupp(String m){
    int res=-1;
    boolean cont=false;
    for(int i=0;i<this.MotsSup.size()&& !cont;i++){
      if(this.MotsSup.get(i).mot.compareTo(m)==0)
      {cont=true;
      res=i;
      }
    }
    return res;
  }

  public int getIndMP(String m){

    int i=0;
    while(i<this.MotsPresents.size()&&!this.MotsPresents.get(i).mot.equals(m)){
      i++;
    }
    if(i==this.MotsPresents.size()){
      return -1;
    }
    else

    return i;
  }

  public void misajourTrace(int[] t1,int[] t2,ArrayList<String> vi,ArrayList<String> vj,int
numV,Contributions c){

    int ind;
```

```
ArrayList<Trace> tmpMotsPresents=new ArrayList<Trace>();
//ArrayList<Trace> tmpMotsSup=new ArrayList<Trace>();

for(int i=0;i<t2.length;i++){
  switch(t2[i]){
    case -7:{ind=existDsTraceSupp(vj.get(i));
      if(ind==-1){
      tmpMotsPresents.add(new Trace(vj.get(i),numV));
      }
      else{
        c.nbOperationSub--;
      tmpMotsPresents.add(this.MotsSup.get(ind));
      this.MotsSup.remove(ind);
      }};break;
    case -2:{
      tmpMotsPresents.add(this.MotsPresents.get(getIndMP(vj.get(i))));
    };break;
    case -3:{
      ind=existDsTraceSupp(vj.get(i));
      if(ind==-1){
      tmpMotsPresents.add(new Trace(vj.get(i),numV));
      }
      else{
        c.nbOperationI--;
      tmpMotsPresents.add(this.MotsSup.get(ind));
      this.MotsSup.remove(ind);
      }
    };break;
    case -6:{

      //
      /*if(i<t1.length){
        if(t1[i]==t2[i]){
          tmpMotsPresents.add(new Trace(vj.get(i),this.MotsPresents.get(i).versionNum));
        }
        else{
          int k=0;
          while(k<vi.size()&& t1[k]!=i){
          k++;
          }
          tmpMotsPresents.add(new Trace(vj.get(i),this.MotsPresents.get(k).versionNum));
        }
      }
      else{
        int k=0;
      while(k<vi.size()&& t1[k]!=i){
        k++;
      }
      tmpMotsPresents.add(new Trace(vj.get(i),this.MotsPresents.get(k).versionNum));
```

```
            }*/
            ind=existDsTraceSupp(vj.get(i));
            if(ind==-1){
              tmpMotsPresents.add(new Trace(vj.get(i),numV));
              }
              else{
                 c.nbOperationCorrectionMot--;
              tmpMotsPresents.add(this.MotsSup.get(ind));
              this.MotsSup.remove(ind);
              }
           };break;
           default :{
              tmpMotsPresents.add(new Trace(vj.get(i),this.MotsPresents.get(t2[i]).versionNum));
           };break;
        }

     }
     for(int i=0;i<t1.length;i++){
        if(t1[i]==-4){
           this.MotsSup.add(new Trace(vi.get(i),this.MotsPresents.get(i).versionNum));
        }
     }

     this.MotsPresents=tmpMotsPresents;
  }

  public void preparerCalculSimCos(){
     for(int i=0;i<this.corpusRaffiner.size();i++){
        //Vecteur TF
        Vector<Integer> v=this.mtrDiff.genererVecteurOccurence(this.corpusRaffiner.get(i));
        this.vOcc.add(v);
        Vector<Double> v1=this.mtrDiff.genererTfVect(v,this.corpusRaffiner.get(i).size());
        this.vTf.add(v1);

        //Vecteur IDF
        ArrayList<String> list = this.mtrDiff.enleverReptition(this.corpusRaffiner.get(i));
        Vector<Double> v2=this.mtrDiff.genererIdfVect(this.corpusRaffiner,list);
        this.vIdf.add(v2);

        //Vecteur Tf-Idf
        Vector<Double> v3=this.mtrDiff.calculerTf_Idf(v1,v2);
        this.vTf_Idf.add(v3);

     }



  }
  public boolean estMotLiaison(String m){
```

```
if((m.compareToIgnoreCase("et")==0)||(m.compareToIgnoreCase("la")==0)||(m.compareToIgnoreCas
e("est")==0)||(m.compareToIgnoreCase("sont")==0)||(m.compareToIgnoreCase("le")==0)||(m.compare
ToIgnoreCase("les")==0)||(m.compareToIgnoreCase("un")==0)||(m.compareToIgnoreCase("une")==0)
||(m.compareToIgnoreCase("des")==0)||(!Character.isLetter(m.charAt(0)))){
        return true;
      }
      else
        return false;
    }
    public void affichageStatUsr(int ind){

      //Pour chaque version (i,s,d,c,sub)
      System.out.println("Opérations réalisé par l'auteur "+this.listeContribution.get(ind).id+" sont :");
      for(int i=0;i<this.listeContribution.get(ind).listeVersionAut.size();i++){
                System.out.println("Pour la version :
"+this.listeContribution.get(ind).listeVersionAut.get(i));
                System.out.println("----------");
                System.out.println("Insertion : "+this.listeContribution.get(ind).lstnbOperationI.get(i));
                System.out.println("Suppression :
"+this.listeContribution.get(ind).lstnbOperationS.get(i));
                System.out.println("Déplacement :
"+this.listeContribution.get(ind).lstnbOperationD.get(i));
                System.out.println("Substitution
"+this.listeContribution.get(ind).lstnbOperationSub.get(i));
                System.out.println("Correction de mots
"+this.listeContribution.get(ind).lstnbOperationCorrectionMot.get(i));
                System.out.println();
      }

    }
    public void affichageStatsUsrs(){
    }


    public void calculDesContributions(){
      int i;
      for(i=1;i<this.tailleRaffiner;i++){
        System.out.println("iteration "+i);

        ArrayList<String> tmpL1=this.corpusRaffiner.get(i-1);
         ArrayList<String> tmpL2=this.corpusRaffiner.get(i);

        //création des listes des liens interne externe
        //Comparaison des listes des liens internes et externe
        //pour extraire Nav Ref
        diff_match_patch dmp=new diff_match_patch();
        //
```

```
        LinkedList<diff_match_patch.Diff>       lstDiff=dmp.diff_main(this.listeVersionCorpusR.get(i-
1),this.listeVersionCorpusR.get(i));
           dmp.diff_cleanupSemantic(lstDiff);
           for (Diff aDiff : lstDiff){
              System.out.println(aDiff);
           }

           ArrayList<int[]>      opsDiff=this.mtrDiff.diffToTabc(lstDiff,this.listeVersionCorpusR.get(i-
1),this.listeVersionCorpusR.get(i));

opsDiff=this.mtrDiff.diffToTabM(opsDiff.get(0),opsDiff.get(1),this.listeVersionCorpusR.get(i-
1),this.listeVersionCorpusR.get(i));
           this.mtrDiff.etapeRecheDep(opsDiff.get(0),opsDiff.get(1),tmpL1,tmpL2);
           //ArrayList<int[]> ops=this.mtrDiff.detecterOperations(tmpL1,tmpL2);
        //System.out.println("etape 1");
        //int nbSup=0;
        /*if(i==2){

           for(int k=0;k<ops.get(0).length;k++){
              if(ops.get(0)[k]>0){
                 if(this.estMotLiaison(tmpL1.get(k))){
                 ops.get(1)[ops.get(0)[k]]=-3;
                 ops.get(0)[k]=-4;
              }
              }
           }
        }
        */
        Contributions                    contt=this.mtrDiff.genererScriptChangement(opsDiff.get(0),
opsDiff.get(1),tmpL1,tmpL2);

        //System.out.println("etape 2");

        this.misajourTrace(opsDiff.get(0), opsDiff.get(1), tmpL1,tmpL2, i,contt);

        //System.out.println("etape 3");
         for(int k=0;k<opsDiff.get(0).length;k++){
        System.out.print(opsDiff.get(0)[k]+" | ");
        }
     System.out.println();
     for(int k=0;k<opsDiff.get(1).length;k++){
        System.out.print(opsDiff.get(1)[k]+" | ");
      }
     System.out.println();
     System.out.println("**********************");
        int ind=this.retournerIndexAuteur(this.contributeursR.get(i));
        if(ind==-1){//Premiere  contribution de l'auteur
           //System.out.println("etape 4A");
           this.listeContribution.add(new FichierAut(this.contributeursR.get(i)));
```

```
            this.listeContribution.get(this.listeContribution.size()-
1).contribution.nbOperationI=contt.nbOperationI;
            this.listeContribution.get(this.listeContribution.size()-
1).contribution.nbOperationS=contt.nbOperationS;
            this.listeContribution.get(this.listeContribution.size()-
1).contribution.nbOperationD=contt.nbOperationD;
            this.listeContribution.get(this.listeContribution.size()-
1).contribution.nbOperationSub=contt.nbOperationSub;
            this.listeContribution.get(this.listeContribution.size()-
1).contribution.nbOperationCorrectionMot=contt.nbOperationCorrectionMot;
            this.listeContribution.get(this.listeContribution.size()-
1).dernieresionL=this.corpusRaffiner.get(i);
            this.listeContribution.get(this.listeContribution.size()-
1).dernieresionS=this.listeVersionCorpusR.get(i);
            this.listeContribution.get(this.listeContribution.size()-1).listeVersionAut.add(i);
            DamerauLevenshteinAlgorithm defonc=new DamerauLevenshteinAlgorithm(1, 1, 1, 1);
            this.listeContribution.get(this.listeContribution.size()-
1).distanceEditionList.add(defonc.execute(this.listeVersionCorpus.get(i-
1),this.listeVersionCorpus.get(i)));
            this.listeContribution.get(this.listeContribution.size()-
1).avantDerniereVersionS=this.listeVersionCorpusR.get(i-1);
            this.listeContribution.get(this.listeContribution.size()-
1).lstnbOperationI.add(contt.nbOperationI);
            this.listeContribution.get(this.listeContribution.size()-
1).lstnbOperationS.add(contt.nbOperationS);
            this.listeContribution.get(this.listeContribution.size()-
1).lstnbOperationD.add(contt.nbOperationD);
            this.listeContribution.get(this.listeContribution.size()-
1).lstnbOperationSub.add(contt.nbOperationSub);
            this.listeContribution.get(this.listeContribution.size()-
1).lstnbOperationCorrectionMot.add(contt.nbOperationCorrectionMot);
            }
          else{//Nouvelle contribution

            //System.out.println("etape 4B");

            int index=0;
            ArrayList<String>                                    list11                                =
this.mtrDiff.enleverReptition(this.listeContribution.get(ind).dernieresionL);
            ArrayList<String> list22 = this.mtrDiff.enleverReptition(this.corpusRaffiner.get(i-1));
            ArrayList<String> vectSc=this.mtrDiff.genererVecteurSc(list11,list22);
            ArrayList<Double> vectSc1=new ArrayList<Double>();
            ArrayList<Double> vectSc2=new ArrayList<Double>();
            for(int x=0;x<vectSc.size();x++){
              index=this.mtrDiff.retournerIndex(list11,vectSc.get(x));
              if(index==-1){
                vectSc1.add(Double.valueOf(0));
              }
              else{
```

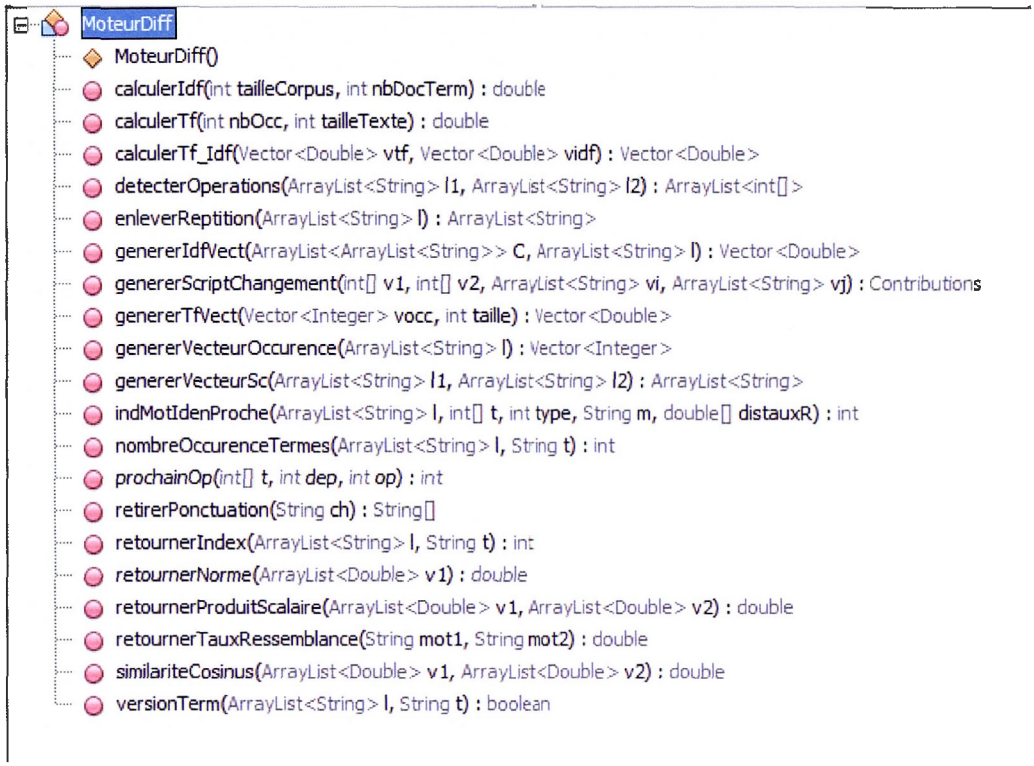
```
vectSc1.add(this.vTf_Idf.get(this.listeContribution.get(ind).listeVersionAut.get(this.listeContribution.g
et(ind).listeVersionAut.size()-1)).elementAt(index));
                //vectSc1.add(v7.elementAt(index));
            }
            index=this.mtrDiff.retournerIndex(list22,vectSc.get(x));
            if(index==-1){
               vectSc2.add(Double.valueOf(0));
            }
            else{
               vectSc2.add(this.vTf_Idf.get(i).elementAt(index));
            }
         }
         double coSim=this.mtrDiff.similariteCosinus(vectSc1,vectSc2);

this.listeContribution.get(ind).contribution.nbOperationI=(coSim*this.listeContribution.get(ind).contri
bution.nbOperationI)+contt.nbOperationI;

this.listeContribution.get(ind).contribution.nbOperationS=(coSim*this.listeContribution.get(ind).contri
bution.nbOperationS)+contt.nbOperationS;

this.listeContribution.get(ind).contribution.nbOperationD=(coSim*this.listeContribution.get(ind).contr
ibution.nbOperationD)+contt.nbOperationD;

this.listeContribution.get(ind).contribution.nbOperationSub=(coSim*this.listeContribution.get(ind).co
ntribution.nbOperationSub)+contt.nbOperationSub;

this.listeContribution.get(ind).contribution.nbOperationCorrectionMot=(coSim*this.listeContribution.
get(ind).contribution.nbOperationCorrectionMot)+contt.nbOperationCorrectionMot;

         this.listeContribution.get(ind).lstnbOperationI.add(contt.nbOperationI);
         this.listeContribution.get(ind).lstnbOperationS.add(contt.nbOperationS);
         this.listeContribution.get(ind).lstnbOperationD.add(contt.nbOperationD);
         this.listeContribution.get(ind).lstnbOperationSub.add(contt.nbOperationSub);

this.listeContribution.get(ind).lstnbOperationCorrectionMot.add(contt.nbOperationCorrectionMot);
         this.listeContribution.get(ind).lstcosSim.add(coSim);


         DamerauLevenshteinAlgorithm defonc=new DamerauLevenshteinAlgorithm(1, 1, 1, 1);
         int de=defonc.execute(this.listeVersionCorpusR.get(i-1),this.listeVersionCorpusR.get(i));
         this.listeContribution.get(ind).distanceEditionList.add(de);

         double          trl=this.tauxRetentionLocal(this.listeContribution.get(ind).listeVersionAut,
this.MotsPresents);
         this.listeContribution.get(ind).tauxRetentionList.add(trl);

         String m1,m2,m3;
```

```
//m1=this.listeVersionCorpusR.get(this.listeContribution.get(ind).listeVersionAut.get(this.listeContrib
ution.get(ind).listeVersionAut.size()-1));
            m1=this.listeContribution.get(ind).avantDerniereVersionS;
            m2=this.listeVersionCorpusR.get(i-1);
            m3=this.listeContribution.get(ind).derniereVersionS;
            double
ds=Double.valueOf(this.listeContribution.get(ind).distanceEditionList.get(this.listeContribution.get(in
d).distanceEditionList.size()-1));
            //System.out.println("******"+ds);
            double d1=Double.valueOf(defonc.execute(m1,m2));
            double d2=Double.valueOf(defonc.execute(m3,m2));
            double le=(d1-d2)/ds;
            this.listeContribution.get(ind).longeviteEditionList.add(le);
            this.listeContribution.get(ind).listeVersionAut.add(i);
            this.listeContribution.get(ind).derniereVersionL=this.corpusRaffiner.get(i);
            this.listeContribution.get(ind).derniereVersionS=this.listeVersionCorpusR.get(i);
            this.listeContribution.get(ind).avantDerniereVersionS=this.listeVersionCorpusR.get(i-1);

         }


         //Longevite edition : pour l'avant dernière contribution qualité * distance
         System.out.println("Liste mots supprimé : ");
         for(int p=0;p<this.MotsSup.size();p++){
            System.out.print(this.MotsSup.get(p).mot+" | ");
         }
         System.out.println();
      }
   }

   public double tauxRetentionAut(ArrayList<Integer> v){
      double res=0;
      int nbMot=0;

      for(int i=0;i<this.MotsPresents.size();i++){
         for(int j=0;j<v.size();j++){
            if(this.MotsPresents.get(i).versionNum==v.get(j))
               nbMot++;
         }
      }
      res=Double.valueOf(nbMot)/Double.valueOf(this.MotsPresents.size());

      return res;
   }
   public double tauxRetentionLocal(ArrayList<Integer> v,ArrayList<Trace>  L){
      double res=0;
      int nbMot=0;
```

```
    for(int i=0;i<L.size();i++){
      for(int j=0;j<v.size();j++){
        if(L.get(i).versionNum==v.get(j))
          nbMot++;
      }
    }
    res=Double.valueOf(nbMot)/Double.valueOf(L.size());

    return res;
}

public ArrayList<Double> tauxRetentionAll(){

    ArrayList<Double>  res =new ArrayList<Double>();
    for(int i=0;i<this.listeContribution.size();i++){
      res.add(this.tauxRetentionAut(this.listeContribution.get(i).listeVersionAut));
    }
    return res;
}



public void executer(){
    initialisation();
    System.out.println("Initialisation  ... OK");
    calculDesContributions();
    System.out.println("Calul des contributions  ... OK");
    System.out.println();
    ArrayList<Double> trAll=this.tauxRetentionAll();

    for(int i=0;i<this.listeContribution.size();i++){
      this.listeContribution.get(i).calculerScore();
      System.out.println("Stats de l'auteur "+this.listeContribution.get(i).id+" :");
      affichageStatUsr(i);
      affichageDE(i);
      affichageLE(i);
      affichageCosSim(i);
      affichageTR(i);
      System.out.println("SCORE = "+this.listeContribution.get(i).score+" , Taux de "
          + "rétention Finale = "+trAll.get(i));
      System.out.println();
    }
}

public void affichageTR(int ind){
    if(this.listeContribution.get(ind).tauxRetentionList.size()!=0){
       System.out.println("Les Taux de rétentions :");
      for(int j=0;j<this.listeContribution.get(ind).tauxRetentionList.size();j++){
```

```
            System.out.print(this.listeContribution.get(ind).tauxRetentionList.get(j)+" , ");
        }
        System.out.println();
    }


}
public void affichageDE(int ind){
    if(this.listeContribution.get(ind).distanceEditionList.size()!=0){
        System.out.println("Les distances d'éditions :");
        for(int j=0;j<this.listeContribution.get(ind).distanceEditionList.size();j++){
            System.out.print(this.listeContribution.get(ind).distanceEditionList.get(j)+" , ");
        }
        System.out.println();
    }

}
public void affichageLE(int ind){
    if(this.listeContribution.get(ind).longeviteEditionList.size()!=0){
        System.out.println("Les longévités d'éditions :");
        for(int j=0;j<this.listeContribution.get(ind).longeviteEditionList.size();j++){
            System.out.print(this.listeContribution.get(ind).longeviteEditionList.get(j)+" , ");
        }
        System.out.println();
    }

}
public void affichageCosSim(int ind){
    if(this.listeContribution.get(ind).lstcosSim.size()!=0){
        System.out.println("Les CosSim :");
        for(int j=0;j<this.listeContribution.get(ind).lstcosSim.size();j++){
            System.out.print(this.listeContribution.get(ind).lstcosSim.get(j)+" , ");
        }
        System.out.println();
    }

}

}
```

A.6.2 MoteurDiff



Code source :

```
package javaapplication2;

import difflib.Chunk;
import difflib.Delta;
import difflib.Patch;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.TreeMap;
import java.util.Vector;
import javaapplication2.diff_match_patch.Diff;

public class MoteurDiff {
```

```
  public MoteurDiff(){

  }


public double retournerProduitScalaire(ArrayList<Double> v1,ArrayList<Double> v2){
   double res=0;
   for(int x=0;x<v1.size();x++){
      res+=v1.get(x)*v2.get(x);
   }
   return res;
}
public   double retournerNorme(ArrayList<Double> v1){
   double res=0;
   for(int x=0;x<v1.size();x++){
      res+=Math.pow(v1.get(x),2);
   }
   return Math.sqrt(res);
}
public  double similariteCosinus(ArrayList<Double> v1,ArrayList<Double> v2){

   double produitScalaire=retournerProduitScalaire(v1,v2);
   double normeV1=retournerNorme(v1);
   double normeV2=retournerNorme(v2);

   return (produitScalaire/(normeV1*normeV2));
}
public  int retournerIndex(ArrayList<String> l,String t){
   int res=-1;
   boolean suivant=false;
   int i=0;
   while(!suivant && i<l.size()){

      if((l.get(i).compareTo(t)==0)){
         res=i;
         suivant=true;
      }

      i++;
   }

   return res;
}

public  ArrayList<String> genererVecteurSc(ArrayList<String> l1,ArrayList<String> l2){
   ArrayList<String> res = new ArrayList<String>();
   for(int i=0;i<l1.size();i++){
```

```
        res.add(l1.get(i));
    }
    for(int i=0;i<l2.size();i++){

        res.add(l2.get(i));
    }

    return enleverReptition(res);
}

public   Vector<Double> calculerTf_Idf(Vector<Double> vtf,Vector<Double> vidf){
    Vector<Double> res=new Vector<Double>();
    for(int i=0;i<vtf.size();i++){
        res.add(vtf.elementAt(i)*vidf.elementAt(i));
    }

    return res;
}
public  double calculerTf(int nbOcc,int tailleTexte){
    return (Double.valueOf(nbOcc)) / (Double.valueOf(tailleTexte));

}

public  double calculerIdf(int tailleCorpus,int nbDocTerm){

    return 1+Math.log((Double.valueOf(tailleCorpus)) / (Double.valueOf(nbDocTerm)));
}

public   Vector<Double> genererIdfVect(ArrayList<ArrayList<String>> C,ArrayList<String> l){

    Vector<Double> v = new Vector<>();
    int dt=0;
    //vérifier les versions qui contienne le termes


    for(int i=0;i<l.size();i++){
        for(int j=0;j<C.size();j++){
            if(versionTerm(C.get(j),l.get(i)))
                dt++;
        }
        //System.out.println("log("+C.size()+"/"+dt+")");
        v.add(calculerIdf(C.size(),dt));
        dt=0;
    }
    return v;
}
```

```
public   Vector<Double> genererTfVect(Vector<Integer> vocc, int taille){
   Vector<Double> v = new Vector<>();
   for(int i=0;i<vocc.size();i++){
      v.add(calculerTf(vocc.elementAt(i),taille));
      //System.out.println(vocc.elementAt(i)+"|"+taille+"|"+calculerTf(vocc.elementAt(i),taille));
   }
   return v;
}


public  boolean versionTerm(ArrayList<String> l, String t){
   boolean res=false;
   int i=0;
   while(!res && i<l.size()){

      if(l.get(i).compareTo(t)==0)
         res=true;

      i++;
   }
   return res;
}


public  ArrayList<String> enleverReptition(ArrayList<String> l){

   ArrayList<String> res = new ArrayList<String>();
   for(int i=0;i<l.size();i++){
      if(nombreOccurenceTermes(res,l.get(i))==0){
         res.add(l.get(i));
      }
   }
   return res;
}
public  int nombreOccurenceTermes(ArrayList<String> l,String t){
   int res=0;
   for(int i=0;i<l.size();i++){
      if (l.get(i).compareTo(t)==0) {
         res++;
      }
   }
   return res;
}
public  Vector<Integer> genererVecteurOccurence(ArrayList<String> l){
   Vector<Integer> v = new Vector<>();
   ArrayList<String> ltmp=enleverReptition(l);


   for(int i=0;i<ltmp.size();i++){
```

```
        v.add(nombreOccurenceTermes(l,ltmp.get(i)));
    }
    return v;

}

public  String[] retirerPonctuation(String ch){


    String tmp="";

    if (Character.isLetter(ch.charAt(0))){
        tmp=""+ch.charAt(0);
    }
    else{
        tmp=""+ch.charAt(0)+" ";
    }

    for(int i=1;i<ch.length();i++){
        if(Character.isLetter(ch.charAt(i))||(ch.charAt(i)=='\')) {
            tmp=tmp+ch.charAt(i);
        }
        else {
            tmp=tmp+" "+ch.charAt(i)+" ";
        }
    }

    StringTokenizer st = new StringTokenizer(tmp);
    String[] liste=new String[st.countTokens()];

    int z=0;
        while (st.hasMoreTokens()){

        liste[z]=st.nextToken();
        z++;

    }


    return liste;
}

public double retournerTauxRessemblance(String mot1,String mot2){
    double res=0;
    String com="";
    boolean cont;
    int[] tab=new int[mot2.length()];
    for(int i=0;i<mot2.length();i++){
        tab[i]=0;
```

```
    }
    for(int i=0;i<mot1.length();i++){
      cont=true;
      for(int j=0;j<mot2.length()&&cont;j++){
        if(mot1.charAt(i)==mot2.charAt(j)&&tab[j]==0){
          cont=false;

            com+=mot1.charAt(i);
          tab[j]=1;
        }
      }

    }
    res=2*Double.valueOf(com.length())/(mot1.length()+mot2.length());
    return res;

}
//index du mot le plus proche selon Le TR et DDL et contien
public int indMotIdenProche(ArrayList<String> l,int[] t,int type,String m,double[] distauxR){

    //type = 1 : Pour trouver le mot identique
    //type = 2 : Pour trouver le mot le plus proche
    int res=-1;
    DamerauLevenshteinAlgorithm defonc=new DamerauLevenshteinAlgorithm(1, 1, 1, 1);
    int dtmp;
    double trtmp;


    switch (type) {
      case 1: {

        for(int i=0;i<l.size();i++){
        //dtmp=defonc.execute(m,l.get(i));
        //trtmp=this.retournerTauxRessemblance(m, l.get(i));
        if((l.get(i).compareTo(m)==0) &&(t[i]==-1)){

          distauxR[0]=0;
          distauxR[1]=1;
          return i;
        }
        }

      } ;
           break;
      case 2: {
        int c=0;
        int d=0;
        double tr=0;
        while((c<l.size())&&(t[c])!=-1){
```

```
            c++;
          }
          if(c==l.size()){
            return -1;
          }
          else{
            d=defonc.execute(m,l.get(c));
            tr=this.retournerTauxRessemblance(m, l.get(c));
            res=c;
          }

          for(int i=c+1;i<l.size();i++){
            dtmp=defonc.execute(m,l.get(i));
            trtmp=this.retournerTauxRessemblance(m, l.get(i));
            if(((dtmp<d)&&(trtmp>tr))&&(t[i]==-1)){
              d=dtmp;
              tr=trtmp;
              res=i;
            }
          }
          distauxR[0]=d;
          distauxR[1]=tr;
        } ;
             break;

    }


    return res;
}

public  ArrayList<int[]> detecterOperations(ArrayList<String> l1,ArrayList<String> l2){

    /*
     Non lié=-1;
     identique=-2;
     insertion=-3;
     suppression=-4;
     deplacement= index du mot dans le texte cible;
     correction=-6;
     CD=-7;
         */

     //System.out.println(l1.size()+" , "+l2.size()+" detect op");
     ArrayList<int[]> res = new ArrayList<int[]> ();

     //Deux vecteurs pour marquer les changements
     int[] v1=new int[l1.size()];
     int[] v2=new int[l2.size()];
```

```
int index;
double[] tabTrDis=new double[2];
for(int i=0;i<l1.size();i++){
   v1[i]=-1;
}
for(int i=0;i<l2.size();i++){
   v2[i]=-1;
}

//1ere phase d'analyse : Similitude Directe
for(int i=0;i<l1.size();i++){

  index=this.indMotIdenProche(l2, v2, 1, l1.get(i), tabTrDis);
  if (index!=-1)
  {

     if(index==i){
        v1[i]=-2;
        v2[i]=-2;
     }
     else{
        v1[i]=index;
        v2[index]=i;
     }
  }



}
//Correction
for(int i=0;i<v1.length-1;i++){

   if(v1[i]>0){
      if((v1[i]+1)!=v1[i+1]){
         if((v1[i]+1)<l2.size()){
            if (l1.get(i+1).equals(l2.get(v1[i]+1))){
               v2[v1[i+1]]=-1;
               v1[i+1]=v1[i]+1;
               v2[v1[i+1]]=i+1;

            }
         }
      }
   }
}

//2eme phase d'analyse : Similitude proche
for(int i=0;i<l1.size();i++){
```

```
        if(v1[i]==-1){
          index=this.indMotIdenProche(l2, v2, 2, l1.get(i), tabTrDis);
        if (index!=-1)
        {
          if (tabTrDis[1]>=0.8) {
            if(index==i){
            v1[i]=-6;
            v2[i]=-6;
            }
            else{
            v1[i]=index;
            v2[index]=-6;
            }
          }

        }
        }

      }
      //System.out.println("----------------------------------------");
      for(int i=0;i<l1.size();i++){
        if (v1[i]==-1)
          v1[i]=-4;
      }
      for(int i=0;i<l2.size();i++){
        if (v2[i]==-1)
          v2[i]=-3;
      }
      res.add(v1);
      res.add(v2);

      //System.out.println(l1.size()+" , "+l2.size()+" detect op");

      return res;
    }

    public int prochainOp(int[] t,int dep,int op){

      int j;

        j=dep+1;
        while(j<t.length && t[j]==op)
          j++;

      if(j==t.length)
        return t.length-1;

      return j;
    }
```

```
public void etapeRecheDep(int[] vo,int[] vn,ArrayList<String> lstOld,ArrayList<String> lstNew){
  //ArrayList<int[]> res = new ArrayList<int[]> ();
  //HashMap map = new HashMap();
  TreeMap<Integer, String> lstInsert=new TreeMap();
  //HashMap<String, Integer> lstInsertInv=new HashMap();
  TreeMap<Integer, String> lstDelete=new TreeMap();
  //HashMap<String, Integer> lstDeleteInv=new HashMap();

  for(int i=0;i<lstOld.size();i++){
    if(vo[i]==-4){
      lstDelete.put(i,lstOld.get(i));
      //lstDeleteInv.put(lstOld.get(i), i);
    }
  }

  for(int i=0;i<lstNew.size();i++){
    if(vn[i]==-3){
      lstInsert.put(i,lstNew.get(i));
      //lstInsertInv.put(lstNew.get(i), i);
    }
  }


  //System.out.println("Liste des mots supprimés : ");
  Set setD = lstDelete.entrySet();
  Iterator iterD = setD.iterator();
  while(iterD.hasNext()) {
  Map.Entry me = (Map.Entry)iterD.next();
  //recherche
  int r=getkeyMapIS(lstInsert,(String)me.getValue());

  if(r!=-1){
    vo[(int)me.getKey()]=r;
    vn[r]=(int)me.getKey();
    lstInsert.remove(r);
    //lstDelete.remove((int)me.getKey());
  }

  //validation des deplacements
  int nbid=0;
  int j=0;
  for(int i=0;i<vo.length;i++){
    if(vo[i]!=-4) {
      nbid++;
    }
    else{
      //if()()
```

```
            while((j<vn.length)&&(nbid!=0)){
              if((vn[j]==-2)||(vn[j]==-6)) {
                nbid--;
              }
              j++;
            }
            if((j<vn.length)&&(vn[j]==-3)){
              vn[j]=-7;
              j++;
            }

          }
        }
        //System.out.print(me.getKey() + ": ");
        //System.out.println(me.getValue());
        }

        //return res;
      }


      //,ArrayList<Trace> pres,ArrayList<Trace> supp
      public Contributions genererScriptChangement(int[] vold,int[] vnew,ArrayList<String>
vi,ArrayList<String> vj){

        Contributions statChangement=new Contributions();
        for(int i=0;i<vold.length;i++){

          switch(vold[i]){

            case -2:{};break;
            case -6:{};break;
            case -4:{ statChangement.nbOperationS++; } ;break;
            default:
            {

              if (i==(vold.length-1)){
                statChangement.nbOperationD++;
              }
              else{
                while(i<vold.length-1 && ((vold[i]+1)==vold[i+1]))
                  i++;
                statChangement.nbOperationD++;
              }
            }
            ;break;
          }
        }
        //System.out.println(vi.size()+" , "+vj.size()+" generer script 2/3");
```

```
  for(int i=0;i<vnew.length;i++){
    switch(vnew[i]){
      case -7:{
        statChangement.nbOperationSub++;
      };break;

      case -3:{
        statChangement.nbOperationI++;

      };break;
      case -6 :{
        statChangement.nbOperationCorrectionMot++;
      };
      break;
      default : {
      };
      break;

    }


  }

  //System.out.println(vi.size()+" , "+vj.size()+" genere Script 3/3");

  return statChangement;
}
public ArrayList<int[]> patchTotab(Patch p,ArrayList<String> lo,ArrayList<String> ln){
  ArrayList<int[]> res = new ArrayList<int[]> ();

  int[] vo=new int[lo.size()];
  int[] vn=new int[ln.size()];

  //ici : construction du patch au lieu de le passer en parametre

  for(int i=0;i<lo.size();i++){
    vo[i]=-1;
  }
  for(int i=0;i<ln.size();i++){
    vn[i]=-1;
  }

  for (Delta delta: p.getDeltas()) {
    Chunk chO=delta.getOriginal();
    //List<String> elemO=(List<String>) chO.getLines();
    Chunk chR=delta.getRevised();
    //List<String> elemR=(List<String>) chR.getLines();

    if(delta.getType().compareTo(Delta.TYPE.INSERT)==0){
```

```
            //System.out.println("Insertion");
            for(int k=chR.getPosition();k<(chR.size()+chR.getPosition());k++){
              vn[k]=-3;
            }
          }
          if(delta.getType().compareTo(Delta.TYPE.DELETE)==0){
            //System.out.println("Suppression");
            for(int k=chO.getPosition();k<(chO.size()+chO.getPosition());k++){
              vo[k]=-4;
            }

          }
          if(delta.getType().compareTo(Delta.TYPE.CHANGE)==0){
            //System.out.println("Changement");
            for(int k=chR.getPosition();k<(chR.size()+chR.getPosition());k++){
              vn[k]=-6;
            }
            for(int k=chO.getPosition();k<(chO.size()+chO.getPosition());k++){
              vo[k]=-6;
            }
          }
        }
        res.add(vo);
        res.add(vn);

        return res;
      }


      public ArrayList<int[]> diffToTabc(LinkedList<diff_match_patch.Diff> diffs, String vOld, String
vNew){
        ArrayList<int[]> res = new ArrayList<int[]> ();

        int[] vo=new int[vOld.length()];
        int[] vn=new int[vNew.length()];

        for(int i=0;i<vOld.length();i++){
          vo[i]=-1;
        }

        for(int i=0;i<vNew.length();i++){
          vn[i]=-1;
        }

        int cptO=0;
        int cptN=0;

        for (Diff aDiff : diffs) {
```

```
        switch (aDiff.operation) {
        case INSERT:
          {
            for(int j=0;j<aDiff.text.length();j++){
              vn[cptN]=-3;
              cptN++;
            }
          }
         break;
        case DELETE:
          {
            for(int j=0;j<aDiff.text.length();j++){
              vo[cptO]=-4;
              cptO++;
            }
          }
         break;
        case EQUAL:
          {
            for(int j=0;j<aDiff.text.length();j++){
              vo[cptO]=-2;
              vn[cptN]=-2;
              cptO++;
              cptN++;
            }
          }
         break;
        }
      }
      res.add(vo);
      res.add(vn);
      return res;
}

public String lsttoString(String[] tmp){
      String res="";
      for(int i=0;i<tmp.length-1;i++){
        res+=tmp[i]+" ";
      }
      res+=tmp[tmp.length-1];
      return res;
}

public int ValidOperation(int I,int D,int E,String m,int t){
      int res=0;
      //System.out.println(m+" : "+I+" : "+D+" : "+E);
      switch(t){
        case 0:{
          if(E==m.length()){
```

```
            res=-2;
          }
          else if(D==m.length()) {
            res=-4;
          }
          else {
            //System.out.println("0-I: "+I);
            if(E>=D){
              res=-6;
            }else {
              res=-4;
            }

          }
        };break;
        case 1:{
          if(E==m.length()){
            res=-2;
          }
          else if(I==m.length()) {
            res=-3;
          }
          else {
            //System.out.println("1-D: "+D);
            if(E>=I){
              res=-6;
            }else {
              res=-3;
            }
          }
        };break;
      }
      return res;
  }

  public static int getkeyMapIS(TreeMap<Integer, String> hm,String m){
      int res=-1;
      Set s = hm.entrySet();
      Iterator iter = s.iterator();
      boolean find=false;
      while(iter.hasNext()&& !find) {
      Map.Entry me = (Map.Entry)iter.next();

      if(me.getValue().equals(m)){
        res=(int)me.getKey();
        find=true;
      }
      //System.out.print(me.getKey() + ": ");
      //System.out.println(me.getValue());
```

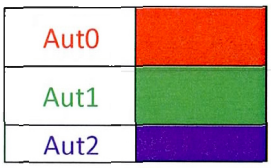
```
    }
    return res;
}

public ArrayList<int[]> diffToTabM(int[] vo,int[] vn,String vOld, String vNew){
    ArrayList<int[]> res = new ArrayList<int[]> ();

    StringTokenizer stO = new StringTokenizer(vOld);
    ArrayList<String> listO = new ArrayList<String>();

    StringTokenizer stN = new StringTokenizer(vNew);
    ArrayList<String> listN = new ArrayList<String>();

    while (stO.hasMoreTokens()){
      listO.add(stO.nextToken());
    }
    while (stN.hasMoreTokens()){
      listN.add(stN.nextToken());
    }
    int[] vvo=new int[listO.size()];
    int[] vvn=new int[listN.size()];


    int cptI=0;
    int cptE=0;
    int cptD=0;

    int cpt=0;
    for(int i=0;i<vOld.length();i++){

      while((i<vOld.length())&&(!Character.isSpaceChar(vOld.charAt(i)))){
        switch(vo[i]){
          case -4:cptD++;break;
          case -2:cptE++;break;
        }
        i++;
      }
      vvo[cpt]=ValidOperation(cptI,cptD,cptE,listO.get(cpt),0);
      cpt++;
      cptI=0;
      cptE=0;
      cptD=0;
    }
    cpt=0;
    cptI=0;
    cptE=0;
    cptD=0;
    //System.out.println("XXXXXXXXXXXXXXXXXXXXXXXXX");
    for(int i=0;i<vNew.length();i++){
```

```
        while((i<vNew.length())&&(!Character.isSpaceChar(vNew.charAt(i)))){
          switch(vn[i]){
             case -3:cptI++;break;
             case -2:cptE++;break;
          }
          i++;
        }
        vvn[cpt]=ValidOperation(cptI,cptD,cptE,listN.get(cpt),1);
        cpt++;
        cptI=0;
        cptE=0;
        cptD=0;

      }
      res.add(vvo);
      res.add(vvn);

      return res;
  }

  public int getDeleteNB(int[] t){
      int res=0;
      for(int i=0;i<t.length;i++){
        if(t[i]==-4)
          res++;
      }

      return res;
  }
  public static int getInsertNB(int[] t){
      int res=0;
      for(int i=0;i<t.length;i++){
        if(t[i]==-3)
          res++;
      }
      return res;
  }
  public static int getEqualNB(int[] t){
      int res=0;
      for(int i=0;i<t.length;i++){
        if(t[i]==-2)
          res++;
      }
      return res;
  }
  public static int getChangeNB(int[] t){
      int res=0;
      for(int i=0;i<t.length;i++){
        if(t[i]==-7)
```

```
            res++;
        }
        return res;
    }
    public static int getCorrectNB(int[] t){
        int res=0;
        for(int i=0;i<t.length;i++){
            if(t[i]==-6)
                res++;
        }
        return res;
    }
}
```

A.7 Résultat des versions analysées

Pour chaque auteur correspond une couleur afin de voir la progression de leurs interventions sur la séquence des versions.



Version 1 = "une applicstion web qui permet la création de pages à l'intérieur ";

Version 2 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur ";

Version 5 = "Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale";

Version 6 = "Un wiki est une application web qui permet la création la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale";

Version 7 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale, tandi que la structure explicite emerge en fonction du besoin d'un usager";

Version 9 = "Un wiki est une application web qui permet la création, la modification et l'illustration collaboratives de pages à l'intérieur d'un site web. Il utilise un langage de balisage et son contenu est modifiable au moyen d'un navigateur web. C'est un outil de gestion de contenu, dont la structure implicite est minimale, tandis que la structure explicite émerge en fonction des besoins des usagers.";