UNIVERSITÉ DU QUÉBEC À MONTRÉAL

CLASSIFICATION DE COURRIELS AU MOYEN DE DIVERSES MÉTHODES D'APPRENTISSAGE ET CONCEPTION D'UN OUTIL DE PRÉPARATION DES DONNÉES TEXTUELLES BASÉ SUR LA PROGRAMMATION MODULAIRE : PDTPM

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR

OUSSAMA ZIRI

JANVIER 2013

UNIVERSITÉ DU QUÉBEC À MONTRÉAL
Service des bibliothèques

Avertissement

La diffusion de ce mémoire se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.01-2006). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

À ma mère.

REMERCIEMENTS

L'accomplissement de ce mémoire de recherche s'appuie évidemment sur la collaboration de nombreuses personnes. C'est pourquoi je tiens à remercier personnellement les gens qui ont participé à cette recherche ainsi qu'au comité d'évaluation de cette étude.

Je voudrais tout d'abord adresser toute ma gratitude à Hakim Lounis, professeur à l'université du Québec à Montréal et directeur de mon mémoire de recherche, pour sa patience, sa disponibilité, son encouragement et surtout pour sa collaboration que ce soit par ses judicieux conseils ou par son soutien financier. Merci de votre confiance.

Plus près de moi, j'aimerais insister sur le support inconditionnel de mon entourage dans cette aventure. D'abord mon père et mes grands parents qui sont toujours dans mon cœur et grâce à qui que je suis en maîtrise. Merci pour votre support inconditionnel! Vous êtes tous une source d'inspiration. Je tiens à remercier également ma grande famille qui m'a supporté, aidé et qui a été à son tour une source d'inspiration. Merci à Mohamed Anis Ziri, grâce à qui je suis à Montréal, Aymen Ziri, Imen Ziri, Iheb Ziri, Sihem Hajjem, Hssan Hajjem, Amel Hajjem, Abdelwaheb Hajjem, Najet Ziri, Mahrziya Ziri, mes cousins et enfin à Imed Jandoubi qui m'a soutenu tout au long de cette période de recherche. Je ne peux compter les soirées et les week-ends où vous m'avez appuyé, encouragé, motivé et pour tous les repas que vous m'avez préparés.

Par la même occasion, un merci infini à mes amis pour leur soutien moral et leurs encouragements tout au long de ce processus; ils étaient souvent à l'écoute lors des moments difficiles. Merci à Rana Bousslama, Tamer Fares, Romdhane Ben Younes, Rami Hadfi, Sabeh Caid Sebssi, Walid Achouri, Talel Amari, Marouene Hammami, Marouen Sinkèz, Karima Somri, Kalthoum Soua, Asma Rihani, Najla Jaziri, Imen Achour, Khaled Zitouni et mon éternel ami Mohamed Fezai.

Enfin, un merci tout particulier à mon oncle Chawki Hajjem qui n'a pas hésité à me prodiguer de judicieux conseils au niveau de la recherche et que je considère comme mon co-

directeur honoraire. Un grand merci à ma tante Ahlem Hajjem pour m'avoir soutenu moralement durant toutes ces années de recherche. Ils m'ont toujours dressé la barre haute pour avoir un travail rigoureux et de qualité. Je leurs serai toujours reconnaissant.

Merci de m'avoir soutenu jusqu'à la ligne d'arrivée.

TABLE DES MATIÈRES

LISTE DES FIGURES IX
LISTE DES TABLEAUX XIII
RESUMÉ XV
INTRODUCTION 1
0.1 Introduction générale 1
0.2 Objectifs 2
CHAPITRE I
1.1 Introduction 3
1.2 La préparation des données : de l'information textuelle vers des vecteurs numériques 4
1.2.1 La collecte des documents et leur standardisation 4
1.2.2 Segmentation de l'ensemble du texte en mots 6
1.2.3 La réduction de l'espace des caractéristiques 7
1.2.4 La génération des vecteurs pour la prédiction 10
1.3 La prédiction 10
1.3.1 Les méthodes d'apprentissage supervisées 11
1.3.2 Les méthodes d'apprentissage non supervisées 26
1.3.3 Comparaison entre les différentes méthodes d'apprentissage 32
1.4 Conclusion 34
CHAPITRE II
2.1 Introduction 37
2.2 Évaluation de performance 37
2.3 Aperçu sur la classification des courriels au moyen des méthodes d'apprentissage automatique 40
2.4 Conclusion 46
CHAPITRE III
3.1 Introduction 47

3.2 Description des utilisateurs et des acteurs 50
3.3 Exigence logique de la base de données 51
3.4 Spécification fonctionnelle de PDTPM 52
3.4.1 Consultation ou suppression d'un module exécutable ou d'une chaîne 53
3.4.2 Ajouter un nouveau module 54
3.4.3 Créer une chaîne 55
3.4.4 Exécuter une chaîne/un module 56
3.4.5 Convertir un fichier textuel (*.txt*) en format Weka (*.arff*) 57
3.5 Principaux scénarios des cas d'utilisation 57
3.5.1 Ajouter module 59
3.5.2 Créer une chaîne de préparation de données 61
3.5.3 Exécuter une chaîne de préparation de données 62
3.6 Conclusion 65
CHAPITRE IV
4.1 Introduction 68
4.2 Base de données publique : LingSpam 68
4.3 Préparation de données 69
4.3.1 Chaîne 1 : Chaîne pour former la matrice binaire 70
4.3.2 Chaîne 2 : Chaîne pour former la matrice fréquence 76
4.3.3 Chaîne 3 : Chaîne pour former la matrice TfIdf 78
4.4 Expérimentation de plusieurs méthodes d'apprentissage sur les données prétraitées extraites de la base de données LINGSPAM 80
4.4.1 KNN 80
4.4.2 Naive Bayes 81
4.4.3 J48 82
4.4.4 JRip 83
4.4.5 SVM 84
4.4.6 Réseaux de neurones (NN) 85
4.5 Discussion des résultats obtenus 88
4.5.1 Discussion quantitative 88
4.5.2 Discussion qualitative 89
CONCLUSION 94

5.1 Introduction 94
5.2 Survol de la recherche 94
5.3 Contribution de la recherche 98
5.4 Limites de la recherche 99
5.5 Recherches futures 99
BIBLIOGRAPHIE 102

LISTE DES FIGURES

Figure		page
Figure 1.1	Mots ignorés par les moteurs de recherche	7
Figure 1.2	Un arbre de décision pour une classification binaire : éligibilité à un emprunt.	14
Figure 1.3	Les règles de décision qui forment la classe « loan».	15
Figure 1.4	Algorithme de génération d'un arbre de décision.	16
Figure 1.5	Exemple de réseau Bayésien avec la table des valeurs des probabilités conditionnelles (CPT) pour la variable LungCancer (LC) [15]	19
Figure 1.6	Exemple d'un perceptron multicouche (PMC) [15].	21
Figure 1.7	Apprentissage d'un réseau de neurone pour la classification, en utilisant l'algorithme de rétro-propagation [15].	23
Figure 1.8	Classification binaire par SVM.	24
Figure 1.9	Comparaison entre la prédiction supervisée et la recherche d'information (MI) : la prédiction à partir des documents recherchés et sélectionnés.	27
Figure 1.10	Calcul de la similarité-cosinus.	28
Figure 1.11	L'algorithme k-means [6].	29
Figure 1.12	Exemple d'exécution de K-means pour $k=2$ [6].	30
Figure 1.13	L'algorithme de regroupement hiérarchique [6].	31
Figure 1.14	Un dendrogramme représentatif du regroupement hiérarchique des données a, b, c et d.	31
Figure 2.1	La partition de l'ensemble de données pour la prédiction et l'évaluation [6].	38
Figure 2.2	Comparaison des performances de quelques méthodes d'apprentissage sur l'ensemble de données textuelles publique REUTERS-21578.	45
Figure 3.1	Position de PDTPM vis à vis de la cybercriminalité et de l'apprentissage automatique.	48

Figure 3.2	Exemple de chaînes exécutables que l'utilisateur peut former au moyen de PDTPM.	49
Figure 3.3	Matrice Documents-Mots obtenue en exécutant une chaîne de préparation de données sur un corpus de données.	51
Figure 3.4	Schéma relationnel de la base de données BDplateforme de PDTPM.	52
Figure 3.5	C-U «Mise à jour à un module exécutable ou à une chaîne».	53
Figure 3.6	GUI-Fenêtre principale de PDTPM.	54
Figure 3.7	C-U «Ajouter un nouveau module».	54
Figure 3.8	C-U «Créer une chaîne de préparation de données».	55
Figure 3.9	C-U «Exécuter une chaîne/un module».	56
Figure 3.10	C-U «Convertir un fichier matriciel textuel (.txt) en format Weka (.arff)».	57
Figure 3.11	GUI-Ajouter un module exécutable.	59
Figure 3.12	GUI-Ajouter les options d'un module exécutable.	60
Figure 3.13	GUI-Ajouter les paramètres des options saisies pour un module.	61
Figure 3.14	GUI-Créer une chaîne de préparation de données.	62
Figure 3.15	GUI-Exécuter une chaîne de préparation de données sur un corpus de texte.	63
Figure 3.16	GUI-Sélection du corpus de texte qui sera exécuté par la chaîne de préparation de données.	64
Figure 3.17	GUI-Convertion une matrice textuelle *(.txt)* en format *Weka(.arff)*.	65
Figure 4.1	Chaîne de préparation de données textuelles pour former des matrices binaires par l'intermédiaire de l'outil PDTPM.	70
Figure 4.2	Organisation des données dans un fichier textuel qui a subit une tokenisation par le biais du module Tokenization.exe intégré dans PDTPM.	72
Figure 4.3	Organisation des données sous la forme d'une matrice textuelle après l'exécution du module *ComposerMatrice.exe* intégré dans PDTPM.	73
Figure 4.4	Chaîne de préparation de données textuelles pour former des matrices fréquences par l'intermédiaire de l'outil PDTPM.	77
Figure 4.5	Chaîne de préparation de données textuelles pour former des matrices Tfldf par l'intermédiaire de l'outil PDTPM.	78

Figure 4.6 Taux de précision de la méthode IBK sur les trois matrices textuelles (binaire, fréquence et TfIdf)..... 81

Figure 4.7 Taux de précision de la méthode Naive Bayes sur les trois matrices textuelles (binaire, fréquence et Tf Idf)..... 82

Figure 4.8 Taux de précision de la méthode J48 sur les trois matrices textuelles (binaire, fréquence et Tf Idf)..... 83

Figure 4.9 Taux de précision de la méthode JRip sur les trois matrices textuelles (binaire, fréquence et Tf Idf)..... 84

Figure 4.10 Taux de précision de la méthode SVM sur la matrice textuelle binaire..... 85

Figure 4.11 Construction du réseau de neurones : architecture proposée..... 86

Figure 4.12 Taux de précision de la méthode réseau de neurones sur la matrice textuelle binaire..... 87

Figure 4.13 Arbre de décision formé lors de l'application de J48 sur la matrice fréquence..... 90

Figure 4.14 Les règles de décision obtenues lors de l'application de Jrip sur la matrice fréquence..... 91

LISTE DES TABLEAUX

Tableau		page
Tableau 1.1	Matrice binaire Documents-mots pour l'apprentissage (Training).	12
Tableau 4.1	Tableau récapitulatif de tous les résultats obtenus.	87

RESUMÉ

Les technologies numériques de l'information et de la communication sont de plus en plus utilisées. Certes, ces technologies offrent des moyens de communication pratiques, mais elles soulèvent des préoccupations concernant la protection de la vie privée et le respect en ligne. Les utilisations malveillantes des courriels ne cessent de croitre et la quantité de spams a rendu pratiquement impossible d'analyser les courriels manuellement. Vu ces problématiques, le besoin de techniques automatisées, capables d'analyser des données est devenu primordial. Plusieurs applications se basant sur l'apprentissage automatique se sont établies pour analyser des données textuelles de grand volume.

Les techniques du «Text Mining» analysent des données textuelles en utilisant des méthodes d'apprentissage automatique afin d'extraire les principales tendances. Cependant, les techniques de «Text Mining» ne sont capables d'être utilisées que si les données sont déjà préparées et bien structurées. Cette recherche a deux objectifs : 1) Concevoir un outil original de préparation de données qui offre et regroupe des fonctionnalités primordiales dans l'étape de préparation de données textuelles. Nous n'avons pas trouvé ces fonctionnalités dans d'autres outils de préparation de données (Sato, Alceste, WordStat, NLTK) ce qui a crée un besoin de concevoir notre logiciel. Nous l'avons appelé outil de Préparation de Données Textuelles basé sur la Programmation Modulaire, PDTPM. 2) Application des méthodes d'apprentissage automatiques dédiées au «Text Mining» pour classifier un ensemble de courriels et détecter les spams. Pour le premier objectif, après avoir étudié le processus de préparation de données, l'étude propose un outil de Préparation de Données Textuelles. Cette plateforme permet de considérer en entrée un ensemble de données textuelles brutes et de générer en sortie ces données sous une forme bien structurée qui consiste en une matrice de données documents-mots. En ce qui concerne le deuxième objectif de notre recherche, nous explorons des courriels d'une base de données publique, Lingspam, pour les classifier au moyen de méthodes d'apprentissage automatique en spams et courriels légitimes.

Mots-clés: text mining, apprentissage automatique, préparation des données textuelles, détection de spams, filtrage de courriels.

INTRODUCTION

0.1 Introduction générale

Qui ne possède pas actuellement une boite courriels personnelle? Qui n'a pas reçu des courriers électroniques non sollicités (spams) dans sa boite de réception? Beaucoup même se sentent dérangés par la quantité de spams reçus par jour. Ces questions nous motivent déjà.

En effet, les technologies numériques de l'information et de la communication sont de plus en plus utilisées. Ces technologies offrent des moyens de communication pratiques et rapides. Avec la propagation exponentielle de l'utilisation d'internet, outre les vidéos, les images et le son, le contenu textuel est considéré comme un des moyens les plus utilisés en ligne. Plus précisément, les courriels jouent un rôle central dans nos communications sur internet. Cependant, les courriels non désirables ont commencé à se propager avec l'arrivée des services de courriels au début des années 90 [1]. Définissons tout d'abord le courriel indésirable. Un courriel est dit indésirable s'il s'avère non pertinent, inapproprié et non sollicité et s'il est envoyé à plusieurs personnes [1]. Le but du spam est souvent la publicité et la promotion. Plus menaçant, il peut être utilisé pour la diffusion de programmes malveillants. Actuellement, l'hameçonnage est également considéré comme l'un des principaux objectifs des spammeurs lors de l'utilisation des spams par courrier électronique [1]. Selon Microsoft, pour la période allant de juillet jusqu'à décembre 2010, la proportion de messages indésirables est de 95,3 % [2]. Le numéro un de la sécurité des terminaux, Symantec, annonce dans son rapport MessageLabs Intelligence que le ratio global de spams dans le trafic de courriels était de 90 % (1 par 1,11 courriels) [3]. Étant donné que les utilisations malveillantes des courriels sont devenues massives, le besoin en techniques et en systèmes informatiques automatisés, capable d'analyser des données et de les examiner afin de détecter les sources malveillantes, est devenu primordial. Au sein de ces problématiques, plusieurs applications se basant sur l'apprentissage automatique se sont établies pour détecter

les spams dans les boites de courriels clientes. L'application des techniques d'apprentissage automatique dans le contexte de la cybercriminalité (cyberforensic) est très prometteuse et commence à donner des résultats en termes d'applications conçues et d'articles publiés. C'est un domaine de recherche très actif impliquant l'apprentissage automatique d'une part et les systèmes de communication d'autre part.

0.2 Objectifs

Dans ce travail de recherche, nous évoquons les applications et les études les plus récentes dans le domaine de l'apprentissage automatique appliqué pour la classification des courriels.

Nous présentons dans le premier chapitre le domaine de l'apprentissage automatique en étalant tout le processus qui va de la préparation des données jusqu'à la prédiction. Ensuite, nous mettons de l'avant un état de l'art sur la classification des courriels au moyen de diverses méthodes d'apprentissage automatique ; nous discutons chacune d'elles, et nous montrons dans cette partie le rôle important que joue, la tâche de préparation de données et la sélection des caractéristiques, sur la précision et la qualité de la prédiction. Après avoir mentionné les limites des outils de préparation de données textuelles présents sur le marché, nous présentons dans le chapitre trois, notre outil de préparation de données basé sur la programmation modulaire que nous appelons PDTPM (Plateforme de Préparation de Données Textuelles basée sur la Programmation Modulaire). Cet outil sera utilisé pour traiter l'ensemble de données que nous utilisons pour expérimenter les méthodes d'apprentissage automatique. Il permet de nettoyer des données textuelles brutes en input et de sélectionner les caractéristiques jugées les plus pertinentes. Enfin, dans le quatrième chapitre et à la lumière des nombreuses expérimentations sur les méthodes de catégorisation du texte appliquées sur une base de données publique, nous discuterons la pertinence de chacune de ces approches. Nous discutons aussi l'importance de l'outil PDTPM sur la précision de prédiction de ces méthodes.

CHAPITRE I

APPRENTISSAGE AUTOMATIQUE : D'UN ENSEMBLE DE TEXTES NON STRUCTURÉS VERS UN VECTEUR NUMÉRIQUE ET LA PRÉDICTION

1.1 Introduction

La tâche de classification des courriels est un cas particulier du champ de catégorisation du texte ou classification du texte (TC, Text Classification, Text Categorisation) par laquelle nous allons classer un ensemble de documents dans des catégories prédéfinies, où chaque catégorie implique un sujet unique. C'est typiquement le fait de trouver de nouvelles réponses pour un nouvel ensemble de données textuelles. La classification des courriels n'est donc qu'une application de ce domaine où les données en entrée seront classifiées par exemple, soit en courriels légitimes soit en courriels non désirables [4], appelés aussi pourriels au Canada, ou spams. La classification du texte se partage entre la recherche d'information (IR, information retrival) et le domaine de l'apprentissage automatique (ML, machine learning) [5]. Certes, les méthodes de «Data Mining» et de «Text Mining» interviennent fortement dans la classification des courriels, mais cette tâche coïncide plus avec les méthodes spécifiques au «Text Mining» pour les raisons suivantes : les données utilisées dans cette tâche correspondent à un ensemble de données textuelles non structurées. Les méthodes de «Data Mining» attendent les données sous le format de tableaux (comme les tables individus*variables) alors que les méthodes de «Text Mining» attendent des données sous le format de documents (documents-mots), ce qui coïncide avec l'ensemble des courriels. Les méthodes qui procèdent avec les documents texte doivent s'attendre à des matrices creuses qui contiennent plusieurs cellules contenant des 0. Au contraire, les méthodes spécifiques au «Data Mining» ne supportent pas, en général, des données avec des valeurs manquantes. Enfin, les méthodes classiques de fouille de données supportent à la fois les données positives et négatives. Cependant, les données de «Text

Mining» sont toujours positives [6]. Vus ces différences. La classification des courriels touche plus les méthodes conçues pour le «Text Mining» que celles faites pour le «Data Mining».

Avant de classifier ou regrouper du texte, nous devons obligatoirement passer par une étape primordiale qui consiste au prétraitement des données textuelles. Dans ce qui suit, nous présentons ce concept de prétraitement des données textuelles, qui représente la première étape pour former des données bien structurées sur lesquelles nous serons capables d'appliquer les méthodes d'apprentissage automatique. Nous présentons ensuite les méthodes d'apprentissage automatique utilisées pour fouiller le texte. Enfin, nous montrons comment ces méthodes nous permettent de faire des prédictions.

1.2 La préparation des données : de l'information textuelle vers des vecteurs numériques

Les méthodes d'apprentissage automatique analysent les documents pour trouver quelques règles générales qui donneront des réponses correctes pour de nouveaux échantillons de données. Pour fouiller le texte, on doit tout d'abord traiter le texte sous une forme avec laquelle les procédures d'apprentissage automatique peuvent être utilisées. La transformation des données du texte en matrice documents-mots est largement utilisée. Cette transformation de données pourra être très méthodique, et on a des procédures soigneusement organisées pour remplir les cellules de la matrice [6].

1.2.1 La collecte des documents et leur standardisation

La question principale durant cette étape est comment obtenir un ensemble de données de bonne qualité. Une manière de faire est d'utiliser des ensembles de données génériques. Pour la recherche et le développement de la classification du texte (TC). Les données génériques sont indispensables [6]. Il existe plusieurs corpus de données textuelles publiques sur lesquels se basent plusieurs recherches. Nous citons à titre d'exemple le corpus des courriels LingSpam utilisé dans [20] [17] [24] qui est un ensemble de données contenant des spams et des courriels légitimes, la collection PU1 utilisée dans [20] et l'ensemble de données Reuter's-21578 Dataset [22] compilé par David Lewis et recueilli à l'origine par le groupe de Carnegie et WebKB Dataset [22].

Dans ces cas, on s'attend à ce que le nettoyage de données ait été fait avant le dépôt et on peut avoir confiance en la qualité des données. Mais ces corpus de textes restent génériques et parfois ils ne sont pas mis à jour.

La deuxième alternative est d'avoir un processus de collection des données (application web contenant des sites web autonomes et utiliser un outil logiciel comme un moteur de recherche qui collecte les documents). On utilise des ensembles de données non génériques qui proviennent directement des utilisateurs ou des forums sur le net. Nous pouvons même avoir un processus de sauvegarde, attaché à l'entrée de flux de données [6] ; nous citons le travail dans [10] où on obtient tous les courriels des citoyens reçus au service à la clientèle d'un organisme gouvernemental. Dans ce cas, un effort supplémentaire doit être accompli pour nettoyer et standardiser les données. L'avantage dans ce cas, est que les données sont mises à jour et reflètent la réalité.

L'avantage d'utiliser des données génériques est qu'elles sont déjà nettoyées est standardisées. Un autre avantage, est qu'il y a déjà d'autres recherches qui se sont faites sur ces corpus, ce qui permet de faire des comparaisons entre les résultats obtenus en utilisant le même corpus de textes. Le seul inconvénient est qu'elles ne sont pas mises à jour et ne représentent pas des données récentes, ce qui peut entrainer une baisse quant à la qualité de la prédiction. Prenons l'exemple de la classification des courriels : nous avons besoin des caractéristiques qui reflètent bien les attributs ou les mots que se répètent beaucoup dans les courriels indésirables pour former un bon classificateur. Ce n'est pas le cas si nous utilisons des caractéristiques prises à partir d'un ensemble de données génériques et ancien parce que le contenu des messages indésirables évolue avec le temps.

L'autre démarche qui consiste à collecter des données non génériques permet d'avoir des données mises a jour mais entraîne un effort additionnel et parfois coûteux pour nettoyer et standardiser les données. L'avantage principal de la standardisation des données est que les outils logiciel ont besoin de lire les données juste en un seul format et non pas dans différents format [6]. Le nettoyage de données permet d'améliorer la précision de la prédiction en supprimant les mots fonctionnels et les données bruitées et en ne sélectionnant que les caractéristiques importantes pour la formation du classificateur.

1.2.2 Segmentation de l'ensemble du texte en mots

La première étape dans le traitement de texte est de séparer le flux de caractères en jetons. C'est la «Tokenisation» (Tokenization) qui se base sur la ponctuation et les espaces pour séparer les jetons. Cette étape est fondamentale pour d'autres analyses ultérieures. Chaque jeton est une instance d'un type donné, ainsi, le nombre de jetons est supérieur au nombre de type [6]. Par exemple, les jetons «joue», «joué» et «jouant», appartiennent au type «jouer».

Après la séparation de flux de données en jetons, chaque jeton sera converti en un format standard. Une technique permettant de réduire l'espace de mots ou le nombre de jetons en augmentant le nombre d'occurrences dans chaque type s'appelle la lemmatisation. La lemmatisation se réfère à la conversion des mots en leur forme de base morphologique [7]. Au lieu de placer tous les mots possibles dans le dictionnaire, on peut éviter de stocker toutes les variantes d'un même mot. La logique de ce choix est que toutes les variantes référent réellement au même concept et à la même famille [6]. Il existe des variantes de la lemmatisation. On évoque la lemmatisation flexionnelle qui consiste à limiter la normalisation des jetons en des variantes grammaticales (singulier/pluriel...). Cette technique se base sur un dictionnaire de standardisation et sur des règles [6].

Il existe aussi la racinisation (ou désuffixation, ou stemming en anglais) qui est plus agressive que la lemmatisation. Le but de la racinisation, évoquée dans le livre de Sholom M. Weiss et al [6] est d'éliminer les préfixes flexionnels et les suffixes. Le besoin d'un type donné de racinisation dépend toujours de l'application. Par exemple dans le travail de Mehrnoush F. Saeedian et al. 7, les auteurs ont utilisé un algorithme de désuffixation à la racine appelé «Porter Stemmer algorithm».

À cette étape, nous avons préparé l'ensemble des caractéristiques des documents qui forme le dictionnaire de mots. Ce dictionnaire sera utilisé pour former le concept formel, la matrice documents-mots.

1.2.3 La réduction de l'espace des caractéristiques

Le dictionnaire regroupe l'espace des mots de tous les documents. Ces attributs seront utilisés pour former la matrice mots-documents. Une des principales caractéristiques de cette représentation est la haute dimensionnalité de l'espace des attributs due à un grand nombre d'attributs ; cela pose un problème quant à la performance des algorithmes de «Text Mining». Une particularité très connue dans la représentation d'un ensemble textuel sous la forme de matrice documents-mots est que cette dernière est souvent creuse. Les algorithmes de catégorisation du texte ne peuvent pas fonctionner efficacement à cause de la faible densité intrinsèque entre les documents, dans un espace de grande dimension [9]. Plusieurs recherches récentes ont mentionné que les concepts sur lesquels se basent les méthodes de regroupement et de catégorisation, comme la similarité, peuvent ne pas êtres significatifs face à une grande dimensionnalité d'attributs [9]. En effet, les recherches récentes montrent qu'une grande partie de l'espace de mots n'est pas si importante pour la catégorisation du texte. Des attributs inutiles jouent un rôle d'obstacle à la bonne performance des méthodes de «Text Mining» [8][11][13][25][26] [27] et certaines de ces caractéristiques peuvent être redondantes. Dans certains cas, on peut essayer de réduire la taille du dictionnaire par plusieurs techniques. Une des techniques utilisées est la liste des mots vides (StopWord) qui consiste à enlever des éléments redondants et inutiles de l'espace des caractéristiques. En effet, l'ensemble des mots vides regroupe tous les mots fonctionnels qui sont inutiles pour une catégorisation future du texte. La liste suivante donne un aperçu de quelques mots vides qu'utiliseraient quelques moteurs de recherche sur le web (http://www.link-assistant.com/seo-stop-words.html).

Able, about, above, abroad, according, accordingly, across, actually, adj, after, afterwards, again, against, ago, ahead ...

Figure 1.1 Mots ignorés par les moteurs de recherche.

Selon S. M. Weiss et al. [6], la façon la plus efficace consiste à générer les caractéristiques (attributs), et appliquer toutes les autres transformations et dans une dernière étape, on écarte les mots vides. Les principaux avantages de l'application de «StopWord» est la réduction de l'espace des caractéristiques et l'amélioration possible sur la précision de la prédiction des classificateurs. Un autre moyen de réduire l'espace des caractéristiques repose

sur le concept de N-Gram. En effet, ce concept permet la division d'un texte ou de segments de texte en unités comptées par leurs longueurs. Nous pouvons utiliser ce concept pour supprimer ou garder des mots d'une longueur donnée. Par exemple, on peut supprimer les mots de taille inferieure à quatre ou supérieure à dix. Cette technique peut être efficace pour capter une partie des pseudo-mots (NonWords) [10]. Le problème majeur dans l'application de cette technique réside dans le choix et le paramétrage de la valeur de n. cette valeur diffère selon le contexte d'application et se base beaucoup sur l'expérience et l'expertise. H. Dalianis et al [10] ont utilisé le concept N-gram pour enlever les mots de taille plus petite que 3 et supérieure à 20.

Le «StopWord» et le N-gram sont des techniques de réduction de l'espace des attributs simples. Des techniques plus complexes qui se basent sur des théorèmes mathématiques sont de plus en plus utilisées [14] [15] [17] [19]. Ces méthodes sont appelées "sélection des caractéristiques" (Features Selection). La sélection des caractéristiques est le processus de sélection d'un sous-ensemble des attributs pour la classification de texte [11]. La sélection des caractéristiques est appliquée pour supprimer les attributs redondants et inutiles, de l'espace des caractéristiques de tous les documents ; l'ensemble des caractéristiques sélectionnées doit contenir des informations suffisantes et fiables sur le document original [12]. La sélection des fonctionnalités vise deux objectifs principaux : (1) réduire la taille de l'espace des mots afin de construire un classificateur du texte efficace, (2) augmenter la précision de la classification par l'élimination des caractéristiques bruits (inutiles) [11]. La valeur d'une caractéristique peut être évaluée par plusieurs techniques. Nous allons évoquer dans ce qui suit, deux techniques simples et deux techniques complexes, basées sur des théorèmes mathématiques et statistiques. La première technique simple est celle basée sur la fréquence. Elle permet d'éliminer ou de garder chaque mot qui dépasse un seuil donné. La sélection des caractéristiques basée sur la fréquence peut être une bonne alternative aux méthodes plus complexes. Deuxièmement, la technique de TF_IDF (Fréquence terme-fréquence inverse document) est souvent utilisée dans le «Text Mining» et spécifiquement dans la sélection de caractéristiques. Cette méthode est utilisée pour évaluer l'importance d'un mot dans un document, dans une collection ou dans un corpus [11]. L'idée principale de cet algorithme est que quand un mot existe d'une façon massive dans un document X et apparait peu dans les

autres documents, alors il est considéré important. Par contre, quand le mot est présent d'une façon importante à la fois dans le document X et dans les autres documents, alors le mot est considéré sans importance [6]. Si la sélection des caractéristiques, basée sur la fréquence et la méthode *TF-IDF* semblent simplistes, il existe d'autres méthodes plus complexes qui se basent sur la théorie des probabilités et la théorie de l'information. L'information mutuelle (*MI*) est une mesure statistique qui calcule la dépendance entre deux variables. Cette méthode est très utilisée dans la tâche de sélection des caractéristiques [17][8][11]. Dans le processus de sélection de caractéristiques, *MI* mesure la dépendance entre une caractéristique f_i et une classe donnée C_k. *MI* mesure combien d'information contribue à la présence ou à l'absence d'une caractéristique f_i pour prendre une décision de classification correcte dans la classe C_k [11]. La formule donnant *MI* est la suivante :

$$MI(f_i, c_k) = \log \frac{P_r(f_i \wedge c_k)}{P_r(f_i) \times p_r(c_k)}$$

$P_r(c_k)$ est la probabilité de se classifier dans la classe C_k. $P_r(f_i)$ est la probabilité de l'attribut f_i dans l'ensemble des attributs. $P_r(f_i, c_k)$ est la probabilité du document contenant la caractéristique f_i et a été classé dans la classe C_k. Cette mesure est nulle si C_i et f_i sont indépendantes et croît lorsque la dépendance augmente. Donc, si la valeur de *MI* est élevée alors la caractéristique f_i est importante et on la sélectionne. Dans le cas contraire, elle sera supprimée. Une autre méthode de sélection de caractéristiques, utilisée pour améliorer la catégorisation du texte est le Gain d'Information (*IG*). Il permet d'obtenir de l'information pour quantifier la corrélation d'un attribut f_i à une classe C_k donnée. Plus *IG* est élevé, plus il y a une forte corrélation de l'attribut pour cette tâche de classification [13].

Bien que ces méthodes offrent plusieurs techniques pour améliorer la classification et augmentent la précision des classificateurs, elles présentent quelques limites. Ces modifications dans la représentation des données (*TF-IDF*, fréquence) peuvent engendrer une perte de simplicité par rapport au modèle binaire original qui indique seulement la présence ou l'absence du mot. Selon Sholom M. Weiss et al., [6] la meilleure prédiction dépend du couplage de ces variations à une méthode spécifique d'apprentissage. L'effort additionnel

dans les transformations pour avoir une matrice *TF-IDF* ou de fréquences peut s'avérer inopérant et coûteux. Les autres méthodes les plus sophistiquées utilisent des mesures statistiques pour calculer le poids d'un attribut, évaluant l'importance d'un mot par rapport à un document dans un corpus du texte. Ces techniques demandent plus de ressources mais donnent des résultats de classification plus efficaces, selon plusieurs recherches [11][13]. Les méthodes de réduction d'espace de mots et de sélection de caractéristiques les plus importantes jouent un rôle important quant à la qualité de la prédiction.

1.2.4 La génération des vecteurs pour la prédiction

Après que le nettoyage, la standardisation et la sélection des caractéristiques soient faits, on peut établir la matrice mots-documents. Dans notre contexte formel de données, les lignes sont les exemples (documents) et les colonnes sont les attributs ou caractéristiques. Pour la classification du texte, on a une colonne additionnelle : une colonne pour l'étiquette identifiant à quelle classe appartient le document. Par exemple si on veut classifier des courriels selon qu'ils soient des pourriels ou non, cette colonne contiendra deux valeurs qui sont 1 (pour les spams) ou 0 (pour les légitimes). Chaque ligne de la matrice globale forme un vecteur de prédiction. Pour faire une prédiction fructueuse, on s'attend à trouver les modèles-attributs communs des mots dans les documents. On examine ainsi, les patrons relatifs à une étiquette quand cette dernière est la bonne réponse [6].

1.3 La prédiction

Le problème de prédiction classique du texte est appelé catégorisation du texte. Pour une bonne prédiction, les données doivent êtres bien structurées et préparées. Dans notre contexte formel de données, exprimé en lignes et colonnes, les lignes représentent les documents et les colonnes forment les attributs ou les caractéristiques qui vont êtres analysés. L'idée est de former des modèles-attributs communs en fouillant des similarités communes entre plusieurs documents. Les modèles de prédiction sont assez variés ; ils peuvent êtres partagés sur deux groupes : la première famille regroupe les méthodes d'apprentissage supervisées. Dans ce cas, le but est d'assigner une catégorie ou un thème à un nouveau document. L'ensemble des catégories est fixé à priori. La deuxième famille regroupe des techniques et des méthodes non supervisées. Dans ce cas, les catégories ne sont pas connues à

priori et l'objectif principal consiste à former des groupes «clusters», à partir d'un ensemble de documents, selon un critère de similarité, pour ensuite assigner une catégorie à chaque groupe.

1.3.1 Les méthodes d'apprentissage supervisées

Les méthodes d'apprentissage supervisées exigent que chaque ligne de la représentation matricielle des données possède une colonne additionnelle. Cette colonne identifie l'étiquette de chaque document ou vecteur. Par exemple, si nous voulons classifier des documents en deux catégories, nous aurons deux étiquettes, une pour la bonne réponse (1) et une pour la mauvaise (0). Le «Text Mining» dans le cas des problèmes supervisés n'est pas complètement automatique. L'assignement des étiquettes est une tâche à faire manuellement [6]. Comme nous l'avons évoqué, l'objectif est d'assigner une catégorie parmi les sujets fixés à priori à un nouveau document. Quand on aura un nouvel article, il sera transformé en vecteur (une ligne de la matrice). On analyse les attributs formant ce vecteur ensuite l'article sera assigné à un sujet en se basant sur les modèles-caractéristiques formés lors de l'étape de la formation [6]. La question qui se pose à cette étape est comment nous formons ces modèles-caractéristiques ? La réponse à cette question définit bien le concept d'apprentissage automatique où la prédiction est supervisée. En effet, la tâche de classification consiste à avoir une première matrice complète qui contient la colonne des étiquettes (label) et où les données sont déjà préparées. Cette colonne sera utilisée pour la formation de classificateurs à base de modèles ou patrons. La deuxième matrice de prédiction a le même format et les colonnes ont les mêmes significations. Dans la deuxième matrice, les exemples utilisés sont différents. On les appelle des documents de test. De plus, la deuxième matrice n'a pas les valeurs da la dernière colonne (label). L'objectif d'une méthode d'apprentissage est d'apprendre à partir de la première matrice quelques modèles qui puissent prédire la valeur de la dernière colonne de la deuxième matrice [6], en formant un classificateur à partir des données d'apprentissage (training). Le « Text Mining » prédictif supervisé a besoin d'un échantillon de données d'expériences a priori. À partir de cet échantillon, une méthode apprend comment faire des prédictions sur des nouveaux documents de test. Ainsi, pour une prédiction fructueuse, nous avons besoin d'un échantillon qui sera divisé en deux partie; une partie pour la formation d'un classificateur dont la colonne

d'étiquettes est indispensable (indique à quelle classe appartient le document ou l'exemple). Le reste de l'échantillon sera dédié au test du classificateur. Pour former un classificateur, nous devons nous baser sur les vecteurs qui ont un label positif, c'est-à-dire quand ce dernier a la bonne réponse. Prenons un exemple concret ou nous voulons former un classificateur qui catégorise des documents en deux sujets : sport ou non sport. La bonne réponse dans ce cas est sport qui prendra dans notre exemple la valeur 1. La matrice suivante est une matrice d'apprentissage du classificateur (matrice 1). Elle contient la colonne des étiquettes. Dans notre cas d'apprentissage, c'est une classification binaire.

	f1	f2	f3	f4	f5	...	f6	label
Document 1	1	1	0	1	0	1	0	1
Document 2	0	1	0	1	1	0	0	0
Document 3	1	1	0	1	0	1	0	1
Document 4	1	1	1	1	1	0	1	0
Document...	1	1	1	1	0	1	1	1
Document n	1	1	0	1	0	0	0	1

Tableau 1.1 Matrice binaire Documents-mots pour l'apprentissage (Training).

En se basant sur le label positif qui concerne le sujet sport dans notre cas (1), on peut voir que les trois mots *f1*,*f2* et *f4* se présentent toujours pour la «classe 1» (classe positive) et ne se présentent jamais ensemble pour la «classe 0» (classe négative). Notant que l'on ignore le «document 4» malgré qu'il contienne le même patron *"f1, f2 et f4"*, car il appartient à la classe négative. Nous essayons de trouver des modèles ou des patrons, en analysant la première matrice, et en trouvant une généralisation qui puisse être valide sur les nouveaux exemples de la deuxième matrice. On apprend donc de la première matrice et on applique les modèles inférés à la seconde matrice [6]. Les méthodes d'apprentissage automatiques supervisées, appliquées sur des données textuelles, peuvent êtres divisées en quatre catégories [6] : les méthodes logiques, les méthodes probabilistes, les méthodes statistiques et les méthodes du plus proche voisin. Dans cette partie, nous traitons les trois premières approches. L'approche du plus proche voisin sera abordée dans la partie de la prédiction non supervisée (1.3.2).

1.3.1.1 Approche logique

L'approche logique regroupe les méthodes qui se basent sur les règles de décision. Les règles de décision peuvent être des solutions satisfaisantes pour le « Text Mining ». L'enjeu de ces méthodes est principalement d'analyser les documents déjà préparés qui sont récupérés pour l'apprentissage, puis de générer la chaîne recherchée qui pourrait correspondre exactement à ces documents. L'analyse doit se baser sur les documents ayant une étiquette positive quand on est dans un problème binaire. Le problème est de trouver un ou plusieurs modèles qui produisent ces exemples positifs. Ces modèles sont donc des règles pour le groupe des exemples positifs. À l'étape de test, quand un nouveau document non étiqueté est présenté, l'attribution du label se fait selon que l'une des règles est satisfaite. Si tous les mots dans une règle sont trouvés, l'étiquette du document est positive, sinon, elle est négative. L'idée générale d'un algorithme de règles de décision, donnée par Sholom M. Weiss et al. [6], consiste globalement à trouver un ensemble de règles qui sépare complètement les deux classes. Ensuite, on divise l'ensemble des règles en plusieurs petits ensembles qui contiennent, évidement, moins de règles que l'ensemble de départ. Nous répétons cette étape d'élagage jusqu'à avoir une seule règle dans chaque ensemble. Enfin, on évalue ces ensembles de règles et on sélectionne le meilleur ensemble comme une solution finale. Ce concept d'élagage est partagé par de nombreux programmes d'apprentissage automatique comme C4.5, CART et l'algorithme ID3 [15]. Ce type d'élagage nous permet de créer des ensembles de règles de taille différente puis choisir un seul ensemble, comme solution au problème, par l'établissement d'une norme, le plus souvent l'erreur minimum comme dans le travail de C. Apte et al. [14], où les auteurs utilisent également l'élagage basé sur l'erreur minimum pour réduire le sur-apprentissage. Idéalement, l'ensemble de règles avec la plus faible erreur aura le plus haut pouvoir de prédiction. Une technique d'optimisation à la suite d'élagage a été évoquée dans le livre de Sholom M. Weiss et al. [6] ; elle est connue sous le nom de la mise en conformité (Backfitting). Cette technique se base sur l'échange de mots entre les règles crées et le dictionnaire de mots, afin d'éviter le chevauchement entre les règles et d'éviter que certains documents positifs n'aient pas d'occurrences dans les règles restantes. Pour n'importe quel mot dans nos phrases en cours, nous essayons de le remplacer par un autre mot dans le dictionnaire. Si un autre mot améliore les performances, nous les échangeons, déplaçant le nouveau mot dans la phrase et le vieux

mot dans le dictionnaire. Nous continuons le processus jusqu'à ce qu'aucun mot ne puisse être remplacé par un autre mot qui réduit les erreurs. L'avantage de la mise en conformité (backfitting) est qu'elle corrige des problèmes dans l'ensemble des règles, sans que l'on change sa taille [6].

Les algorithmes utilisant l'approche de règles de décision se basent des fois sur le modèle d'arbre de décision. L'arbre de décision est un modèle se présentant sous la forme d'un arbre dont chacun des nœuds internes représente un test sur un attribut, permettant de diviser l'ensemble d'apprentissage en deux ou plusieurs sous-ensembles, selon l'algorithme utilisé. Chaque nœud final, se présentant à la fin d'une branche, que l'on appelle une feuille, représente une classification ou un résultat du test. Le nœud situé au début de l'arbre est appelé la racine (voir Figure1.2). Un arbre de décision est formé de plusieurs DNF (disjonctive Normal Form) où chaque chemin complet (de la racine jusqu'à une feuille) ou une partie d'un chemin complet forme une disjonction [4]. La figure suivante donne un exemple d'un arbre de décision où on veut classifier une population entre une partie éligible à un emprunt et une autre non éligible.

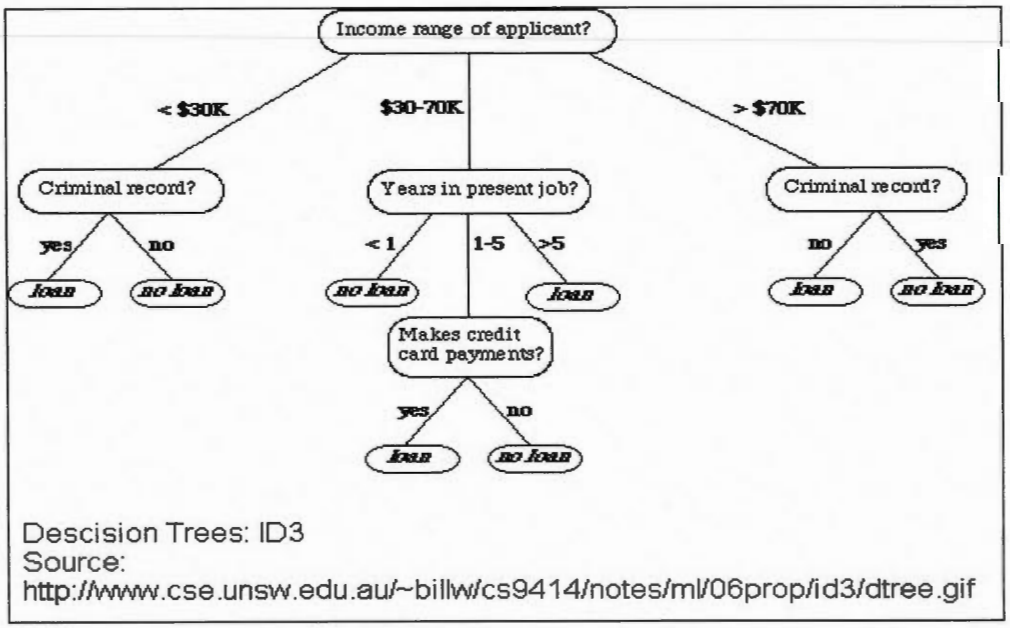


Figure 1.2 Un arbre de décision pour une classification binaire : éligibilité à un emprunt.

Nous prenons l'exemple de la figure 1.3 ci-dessous. La classe «emprunt» est donnée par la disjonction de conjonction suivante:

loan = [(Income range of applicant) < $30k **AND** (Criminal record)= false **] OR [** (Income range of applicant) IN $30k-$70K **AND** (years in present job) > 5 **] OR [**(Income range of applicant) IN $30k-$70K **AND** (years in present job IN 1-5 **AND** (makes credit card payments) = true**] OR [**(Income range of applicant) >$70K **AND** (criminal record) = false**]**.

Figure 1.3 Les règles de décision qui forment la classe « loan».

Les arbres de décision constituent une méthode bien connue en classification [28]. L'algorithme de base de génération des arbres de décision est donné par J. Han and M. Kamber, [15] et sera résumé dans la figure 1.4 suivante.

Algorithme : génération d'arbre de décision [15]

- Entrées : base d'apprentissage *D*, liste des attributs *L*,
- procédure de sélection d'attribut : *select (D, L)*.
- Sortie : arbre de décision.
- Méthode *generer_Arbre (D, L)*:

(1) Créer un nœud *N*.

(2) Si toutes les données appartiennent à la même classe *C*, alors retourner le nœud *N* qui devient un nœud feuille étiqueté par le label de la classe C.

(3) Si la liste des attributs est vide, alors retourner le nœud *N* qui devient une feuille étiquetée par le label de la classe majoritairement présente dans *D*.

(4) Appliquer la procédure *select (D, L)* pour identifier le critère de sélection permettant une meilleure partition de l'ensemble *D*.

(5) Nommer le nœud *N* avec la sortie (critère de sélection) de la procédure *select (D,L)*.

(6) Si l'attribut de sélection est composé de valeurs discrètes et la partition multiple est autorisée alors supprimer l'attribut de sélection de la liste des attributs *L*.

(7) Pour chaque partition *j* obtenue par le critère de sélection :

7.1 Soit *Dj* l'ensemble des éléments appartenant à *D* et satisfaisants le critère de sélection.

7.2 Si *Dj* est vide alors attacher au nœud *N* un nœud feuille étiqueté avec le label de la classe majoritairement présente dans *D*. Sinon, attacher au nœud *N* le nœud généré par *generer_Arbre*(D_j,*L*).

(8) Retourner *N*.

Figure 1.4 Algorithme de génération d'un arbre de décision.

À l'étape (5), une branche est créée pour chaque valeur de l'attribut de sélection choisi. Soit *A* cet attribut. Il existe trois scénarios possibles de partition des couples de la base d'apprentissage *D* : (i) si les valeurs de *A* sont discrètes et l'arbre admet les partitions multiples (ID3 et C4.5): une branche est créée à partir de *N* pour chaque valeur a_i de *A* et *A* sera retiré de la liste des attributs. (ii) Si *A* présente des valeurs continues (C4.5 et CART), le

test au nœud N présente deux sorties possibles ($A \leq$ valeur de partition et $A >$ valeur de partition) et deux branches seront créées portant respectivement le label «oui» et «non». (iii) Si les valeurs de A sont discrètes et l'arbre est binaire (CART) : le test au niveau du nœud doit être sous la forme «$A \in SA$? », SA est la partition obtenue des valeurs connues de A par la procédure *select(D, L)*. La réponse au test est binaire (oui, non). Deux branches seront créées portant le label «oui» ou «non». La procédure de sélection d'attribut *select(D,L)* est une heuristique. L'objectif est d'identifier le meilleur attribut à utiliser pour partitionner les éléments de la base d'apprentissage. Pour le trouver, nous utiliserons différentes techniques selon l'algorithme de génération de l'arbre. L'algorithme ID3 utilise la mesure de gain d'information, l'algorithme C4.5 utilise le ratio d'information et l'algorithme CART utilise l'indice de Gini [15]. Le processus de génération de l'arbre est récursif et s'arrête lorsque (i) toutes les données à partitionner appartiennent à la même classe ou, (ii) il ne reste pas d'attribut pour partitionner les données ou (iii) il n'existe pas de données pour une branche test.

1.3.1.2 Approche probabiliste

Plusieurs outils de classification utilisent les méthodes probabilistes. Soit C le label de la classe qui nous intéresse et X un vecteur de caractéristiques qui dénote la présence ou l'absence d'un mot dans un dictionnaire. Mathématiquement, l'objectif est d'estimer $P_r(C \mid x)$, la probabilité d'une classe, compte tenu de la présence ou de l'absence du mot d'un dictionnaire. Quand on est devant un problème de classification binaire, nous choisissons la classe dont $P_r(C \mid x)$ est la plus grande. Sinon, si nous abordons un problème de classification multiple, une façon de faire est de diviser la collection de documents en deux classes: une classe avec l'étiquette C et l'autre catégorie avec un Label qui n'est pas C. Par conséquent, il est facile de transformer un problème de classification multiple en un autre binaire si les étiquettes multiples sur les documents sont indépendantes les unes des autres. Il est possible d'afficher chaque label comme un problème de classification séparé avec deux classes. Le problème majeur réside dans le calcul de probabilité. Ce calcul est quasiment impossible même pour un dictionnaire de 150 mots qui aurait 2^{150} combinaisons possibles. Ainsi, une approche simplifiée pour l'estimation de probabilités, appelé Bayésienne avec indépendance ou Bayésienne naïve a souvent été essayée [6]. Les classificateurs simples supposent

l'indépendance des variables, ce qui réduit le temps de computation. Lorsque cette hypothèse s'avère vraie, elle permet aux classificateurs Bayésiens simples d'avoir un taux d'erreurs inférieur à celui des autres approches [15]. Le calcul devient efficace ce qui conduit à une large application de cette approche [11][13] [16][17], comme le client de messagerie Mozilla et le logiciel gratuit de projet SpamAssassin [16]. Les classificateurs simples, dit aussi naïfs, supposent que les variables permettant d'attribuer un élément à une classe donnée sont indépendantes l'une de l'autre. L'algorithme qui les décrit peut être résumé ainsi [15] : Soit une base d'apprentissage D composée des couples (X_i, C_j), avec X_i qui représente l'élément à classer et C_j la classe à laquelle il appartient. Chaque élément X_i est représenté par un vecteur $X_i = (x_1, x_2, ..., x_n)$ où x_1, x_2, ..., x_n représentent les valeurs de X_i correspondants aux attributs A_1, A_2, ..., A_n. Supposons que nous avons un ensemble de classe C_1, C_2, ..., C_m, le classificateur attribue X_i à la classe C_j ayant la probabilité $P(C_j|X_i)$ la plus élevée. En appliquant le théorème de Bayes, nous avons :

$$P(C_j \mid X_i) = \frac{P(X_i \mid C_j) P(C_j)}{P(X_i)}$$

Vu que la probabilité $P(X_i)$ est constante pour toutes les classes, le classificateur assigne X_i à la classe ayant le produit $P(X_i|C_j)*P(C_j)$ le plus élevé. Prenant l'hypothèse que toutes les variables sont indépendantes, alors :

$$P(X_i \mid C_j) = \prod_{k=1}^{n} P(x_k \mid C_j)$$

Le produit $P(X_i|C_j)*P(C_j)$ sera calculé pour toutes les classes C_j et l'élément X_i sera attribué à la classe ayant la valeur maximale. Cependant, dans la pratique, il y a des dépendances entre les attributs. Les algorithmes utilisant l'approche bayésienne en prenant en compte les dépendances entre les caractéristiques se basent sur le modèle de réseau bayésien. Un exemple de réseau bayésien est donné par la figure suivante, prise du livre de J. Han and M. Kamber [15], montrant aussi la table des valeurs des probabilité conditionnelles de la

variable LingCancer (LC) et toutes les combinaisons possibles avec les valeurs de ses nœuds parents, familyHistory (FH) et Smoker (S).

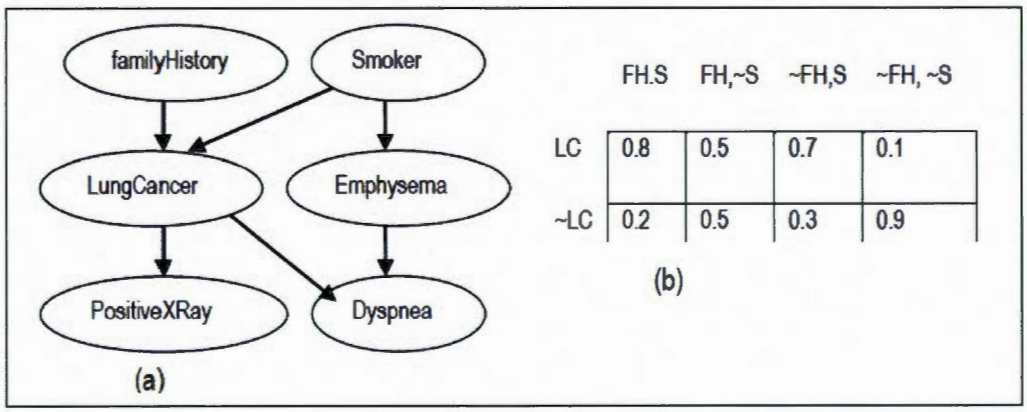


Figure 1.5 Exemple de réseau Bayésien avec la table des valeurs des probabilités conditionnelles (CPT) pour la variable LungCancer (LC) [15].

Les réseaux bayésiens permettent de définir et de traiter les dépendances entre les variables. Ils ont deux composants : (1) un modèle graphique représentant des liens de causalité entre les diverses variables (figure 1.5, a). Le modèle est défini par des nœuds représentants les variables (elles peuvent être discrètes ou continues, elles peuvent correspondre aux attributs de données d'apprentissage ou aux variables cachées qui représentent des relations entre les données [15]) et des arcs orientés représentant les liens de dépendance. Par exemple, si un arc est orienté d'un nœud Y vers un nœud Z, alors Y est un parent de Z, donc Z dépend de Y. Chaque variable est indépendante de ses non-descendants sachant ses prédécesseurs. (2) Une table des probabilités conditionnelles (*CPT*) pour chaque variable (figure 1.5, b). La *CPT* pour une variable Y spécifie les diverses probabilités conditionnelles $P_r(Y|Parent(Y))$. Au niveau de la figure 1.5, b, la *CPT* spécifie les diverses probabilités conditionnelles de la variable «LungCancer». Ainsi, $P_r(LungCancer=yes \mid FamilyHistyes=yes, Smoker = yes)=0.8$. Soit un élément $X=(x_1,x_2,...,x_n)$ tel que $x_1,x_2,...,x_n$ sont les valeurs de X correspondantes aux variables $Y_1,Y_2,..,Y_n$ respectivement, la probabilité jointe $P(x_1,x_2,...,x_n)$ est donnée par l'équation (1).

$$P(x_1,x_2,...,x_n) = \prod_n P_r(x_i|Parents(Y_i)) \text{ (1) où } P(x_1,x_2,...,x_n) \quad (1)$$

$P(x_1,x_2,...,x_n)$ correspond à une combinaison des valeurs de X et les valeurs de $P_r(x_i|Parents(Y_i))$ correspondent aux entrées dans la table *CPT* du Y_i. Le graphe permet d'identifier les probabilités conditionnelles au niveau de l'équation (1). L'apprentissage du réseau peut être réalisé selon divers scénarios. L'architecture du réseau peut être donnée à l'avance comme elle peut être inférée à partir des données. Les variables peuvent être observables ou cachées dans une partie ou dans l'ensemble des couples de la base d'apprentissage. Si l'architecture est déterminée et les variables sont observables alors l'apprentissage consiste à calculer les valeurs de la table *CPT*. Si l'architecture est donnée mais certaines ou toutes les variables ne le sont pas, l'apprentissage peut être effectué selon l'algorithme de descente du gradient (voir la section suivante).

1.3.1.3 Approche statistique

Tout d'abord, il faut mentionner que cette approche est souvent utilisée sous la forme de plusieurs variantes pour résoudre des problèmes différents (Cohen et al., 1999; Herbrich et al., 2000; Crammer and Singer, 2002; Freund et al., 2003) [18]. Généralement, cette approche est utilisée pour établir une relation d'ordre sur la population. On parlera dans ce cas d'un problème de scoring ou d'ordonnancement binaire (bipartite ranking problem) qui se réfère à Freund et al. (2003), comme dans le cas des moteurs de recherche ou des systèmes de recommandation. Cependant, cette approche peut être utilisée aussi pour la classification binaire où nous désirons classifier un ensemble de documents en deux catégories différentes, comme par exemple, spam et non spam. L'utilisation de cette approche est similaire dans les deux cas, c'est seulement la forme de la fonction que l'on doit apprendre de l'échantillon qui change. Dans le cas de l'ordonnancement, on cherche une des valeurs réelles de la fonction $f : X \longrightarrow \mathbb{R}$ qui induit un ordonnancement sur X. Dans la classification, on cherche une fonction binaire d'une valeur $h: X \rightarrow Y$ qui prédit la classe d'une nouvelle instance de X tel que $Y = \{-1, +1\}$ [18]. Dans notre recherche, nous considérons le problème de la distinction entre deux classes ; la classification basée sur cette approche, considère un critère mathématique, le taux d'erreur, pour évaluer une fonction de classification. Nous étudions le classificateur basé sur les réseaux de neurones qui apprend en ajustant les poids des connexions et en se basant sur la minimisation du taux d'erreur afin d'augmenter l'apprentissage du réseau. Cette approche peut être utilisée sur des méthodes, utilisant des techniques statistiques plus

sophistiquées, comme le classificateur de machines à vecteur du support ou séparateur à vaste marge (Support Vector Machine, SVM). Nous discuterons aussi du classificateur SVM qui apprend en formant un hyperplan qui sépare les données en deux classes et ayant les plus grandes marges en se basant sur une fonction qui minimise le taux d'erreur.

a) Réseau de neurones

Un réseau de neurones est un ensemble connecté d'unités d'entrés/sorties où un poids est associé à chaque connexion. Durant la phase d'apprentissage, le réseau apprend en ajustant les poids des connexions afin d'identifier la classe désirée. Parmi les réseaux les plus répandus pour l'implémentation des classificateurs supervisés, nous citons les perceptrons multicouches (PMC). Un PMC est un réseau constitué de plusieurs couches, les neurones de chaque couche sont connectés vers les neurones de la couche suivante et il n'y a pas de connections intra-couche. La figure ci-dessous, présente un exemple d'un réseau de neurones, prise du livre de J. Han and M. Kamber [15].

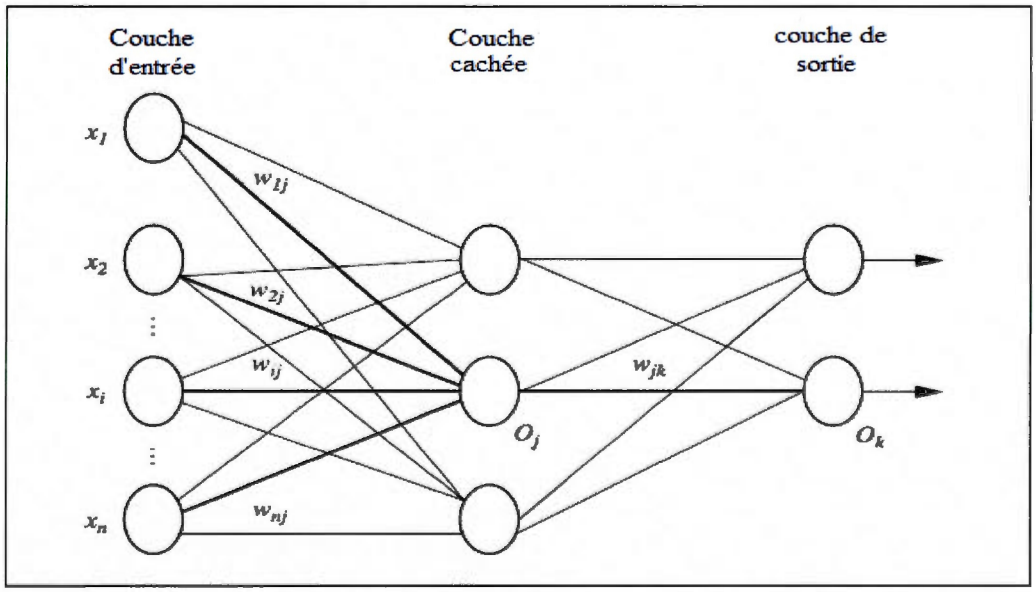


Figure 1.6 Exemple d'un perceptron multicouche (PMC) [15].

La détermination de la topologie du PMC exige la spécification de : (1) le nombre de couches cachées, (2) le nombre de neurones dans chaque couche, (3) les poids initiaux des connections, (4) les fonctions d'activation et (5) les biais des neurones. La spécification de l'architecture du réseau est un processus itératif d'essais et calcul d'erreurs. L'algorithme peut être résumé comme suit [15]:

(1) Initialiser les poids des connections et les biais des neurones.

(2) Tant que l'apprentissage n'est pas terminé :

(2.1) Propagation des inputs en avant (*forward*). Pour chaque unité nous calculons ses entrées et ses sorties. L'exemple d'apprentissage est transmis à la couche d'entrée dont la valeur de la sortie correspond à la valeur de l'entrée. L'entrée de chaque unité I_J des couches cachées et de la couche de sortie est calculée comme suit :

$$I_j = \sum_i W_{ij} O_i + \theta_j$$

où W_{ij} est le poids de la connexion entre l'unité i de la couche précédente et unité j et θ_j le biais de l'unité j.

(2.2) La sortie de chaque unité O_j des couches cachées et de la couche de sortie est calculée en appliquant la fonction d'activation[1] sur son entrée.

(2.3) Calculer l'erreur totale E pour savoir si le réseau a convergé tel que :

$$E = \frac{1}{2} \sum_{j=1}^{n} (d_j - O_j)^2$$

où d_j sortie désirée, O_j sortie obtenue au niveau du neurone J de la couche de sortie et n le nombre de neurones de la couche de sortie.

(2.4) Si l'erreur totale du réseau E est jugée suffisamment faible, l'apprentissage est terminé, sinon

2.4.1 pour chaque neurone J de la couche de sortie nous calculons le signal d'erreur :

[1] Exemple de fonction d'activation : fonction sigmoïde [15] $O_j = \frac{1}{1 + e^{-I_j}}$

$$Err_j = O_j(1-O_j)(d_j - O_j).$$

2.4.2 Pour chaque neurone j des couches cachées

$$Err_j = O_j(1-O_j)\sum_k Err_k W_{jk}$$

avec W_{jk} est le poids de la connexion entre l'unité J et l'unité suivante K, et Err_K est l'erreur au niveau de l'unité K.

2.4.3 Calculer le changement du poids de la connexion IJ selon la règle Δ :

$$\Delta W_{ij} = l.ErrO_i,$$

et l'adapter $W_i = W_i + \Delta W_i$ où l est le pas d'apprentissage.

2.4.4 Calculer et adapter les biais des neurones :

$$\Delta\theta_j = l.Err_j \text{ et } \theta_j = \theta_j + \Delta\theta_j.$$

Figure 1.7 Apprentissage d'un réseau de neurone pour la classification, en utilisant l'algorithme de rétro-propagation [15].

La fin de la phase d'apprentissage peut être obtenue grâce au point (2.4) ou si les changements des poids sont jugés très faibles par rapport à un certain seuil, ou encore si un nombre d'itérations seuil est atteint.

b) Les machines à vecteurs de support (SVM)

Les SVM sont des algorithmes qui utilisent une transformation non linéaire des données d'apprentissage. Ils projettent les données d'apprentissage dans un espace de plus grande dimension que leur espace d'origine. Dans ce nouvel espace, ils cherchent l'hyperplan qui permet une séparation linéaire optimale des données d'apprentissage en utilisant les vecteurs de support et les marges définies par ces vecteurs. Pour éclaircir nos propos, étudions d'abord le cas de données linéairement séparables. Ensuite nous aborderons le cas où les données ne sont pas linéairement séparables.

Soit l'ensemble d'apprentissage D tel que $X_1 \in Y_1$, $X_2 \in Y_{-1}$. Les éléments X sont des données de deux dimensions $X(x_1, x_2)$ avec x_1 et x_2 les valeurs de X pour les attributs A_1 et A_2 respectivement. Pour tout élément X de la base d'apprentissage D, le degré d'appartenance à Y est binaire $\{1,-1\}$.

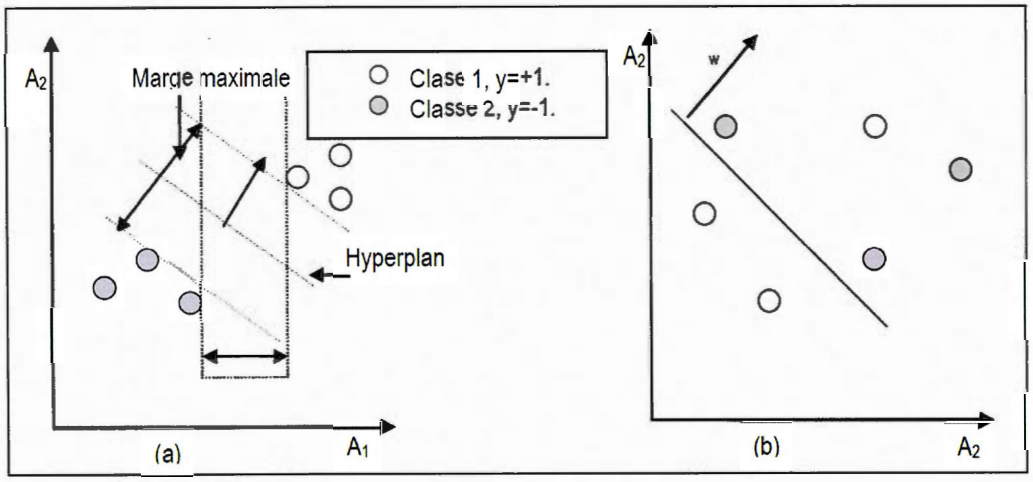


Figure 1.8 Classification binaire par SVM [15].

Comme l'illustre la figure ci-dessus (figure 1.8, a), il existe une infinité de lignes qui permettent de séparer les deux classes. L'objectif d'une méthode SVM est de trouver le meilleur séparateur (taux d'erreur minimal). Avec des données à deux dimensions, nous cherchons une ligne ; pour des données à trois dimensions nous cherchons un plan et pour des données à n dimensions, nous cherchons un hyperplan. Les machines à vecteur du support (SVM) cherchent l'hyperplan qui donne les résultats optimaux, c'est-à-dire, l'hyperplan ayant les plus grandes marges (MMH) [15]. La marge du MMH est la distance minimale entre lui et le point le plus proche de la classe Y_1 et Y_{-1}. Le MMH peut être définie par : $W.X+b=0$ **(1)** où W est le vecteur de poids et b est le biais. Dans notre exemple, l'équation **(1)** peut être représentée par: $w_0+w_1x_1+ w_2x_2=0$ **(2)**. Donc, si un point est situé au dessus du MMH, l'équation **(2)** aura un signe positif, sinon elle aura un signe négatif. Si nous ajustons les poids tel que les hyperplans définissent les cotés de la marges : $H_1 : w_0+w_1x_1+ w_2x_2 \geq 1$ **(3)** si X appartient à la classe Y, $H_2 : w_0+w_1x_1+ w_2x_2 \leq -1$ **(4)** si X n'appartient pas à la classe Y. Les équations **(3)** et **(4)** combinées nous donnent

$Y_i(: w_0+w_1x_1+ w_2x_2 \geq 1)$ **(5)** pour tout Y_i. Tous les couples situés sur les hyperplans H_1 ou H_2 constituent les vecteurs de support.

La question qui se pose est : comment définir le MMH? L'équation **(5)** peut être réécrite pour devenir ce que nous appelons un convexe (constrained quatratic optimization problem). À l'aide de techniques mathématiques avancées, nous arrivons à définir notre MMH comme une frontière de décision :

$$d(X^T) = \sum_{i=1}^{l} y_i \alpha_i X_i X^T + b_0 \textbf{ (6)},$$

avec Y_i est le label du vecteur de support, X^T le point à tester, α_i et b_0 sont des paramètres numériques déterminés par l'algorithme mathématique du SVM et l est le nombre de vecteurs de support. Lors de la classification, pour tout élément X, nous appliquons l'équation **(6)** : si le signe est positif, le SVM prédit que X appartient à la classe $+1$, sinon il prédit que X appartient à la classe -1.

Passons maintenant au cas où les données ne sont pas séparables. Si les données sont linéairement inséparables, aucune droite ne peut séparer les exemples en deux classes (figure 1.8, b). Dans ce cas, il faut généraliser le SVM pour qu'il puisse résoudre ce problème. Nous procédons en deux étapes. Premièrement, il est nécessaire de transformer les données d'entrées dans un espace de plus grandes dimensions. Deuxièmement, il faut chercher un hyperplan MMH dans ce nouvel espace. Le problème qui se pose ici est : comment pouvons nous choisir la méthode de passage d'un espace donné à un espace de dimension plus grande? Il y a aussi un problème lié au coût de calcul. Il est démontré que, lors de la résolution du problème de l'optimisation quadratique du SVM linéaire (dans le nouvel espace), les exemples d'apprentissage apparaissent uniquement sous la forme de produit vectoriel [15]. Le résultat de ce produit est équivalent à l'application de la fonction Kernel aux données dans l'espace d'origine. Donc, tous les produits vectoriels qui apparaissent dans l'algorithme d'apprentissage peuvent être remplacés par l'utilisation de la fonction Kernel. Dans ce cas, tous les calculs peuvent être effectués dans l'espace d'origine évitant ainsi la projection dans un espace de plus haute dimension. Une fois cette technique appliquée, nous pouvons procéder à la recherche du MMH. Plusieurs fonctions Kernel peuvent être utilisées dont la

fonction sigmoïde, la fonction polynomiale et la fonction radiale Gaussienne [15]. Il est intéressant de souligner que l'hyperplan séparateur trouvé par les SVM est semblable à celui obtenu par les réseaux de neurones dépendamment de la fonction Kernel utilisée. Par exemple, un SVM utilisant la fonction sigmoïde est semblable à un réseau de neurones multicouche (avec deux couches) [15]. Il n'existe pas une technique permettant de choisir la meilleure fonction Kernel, mais en pratique le choix de la fonction Kernel n'affecte pas grandement l'exactitude des résultats du classificateur. Les SVM réussissent toujours à trouver une solution globale au problème ce qui n'est pas le cas des réseaux de neurones (problème de minimas locaux). Il est possible de combiner plusieurs SVM pour résoudre des problèmes ayant un nombre de classes supérieur à deux. Les principaux avantages des SVM sont : (1) temps court d'apprentissage, (2) taux d'exactitude de résultats élevé et (3) faible sensibilité au problème de sur-apprentissage [15].

1.3.2 Les méthodes d'apprentissage non supervisées

Bien que le problème de prédiction puisse se présenter sous la forme de prédictions supervisées classiques où un ensemble de catégories est prédéfini, une autre forme de prédiction existe. Dans ce cas, les catégories ne sont pas connues à priori et l'objectif principal consiste à former des groupes «cluster» à partir d'un ensemble de documents et ensuite, assigner une catégorie à chaque groupe. Ce sont les techniques et les méthodes non supervisées qui interviennent dans ce cas. Ces méthodes évoquent souvent la notion de similarité entre les documents comme critère de regroupement. C'est pour cette raison que nous étudions l'approche basée sur le plus proche voisin dans cette partie du chapitre.

1.3.2.1 Approche basée sur la similarité

Trouver le voisin le plus proche d'un document est un processus très connu. Il est utilisé pour la recherche d'information comme par exemple dans les moteurs de recherche. On présente quelques mots clés, et le moteur de recherche retourne les documents appariés les plus proches. Ce cas d'utilisation est inclus dans un problème plus général qui consiste à prendre un nouveau document non étiqueté et prédire son étiquette ; c'est la prédiction non supervisée. Dans le contexte du «Text Mining», le problème est une variante de recherche d'information [6] ; nous prédisons à partir des documents recherchés et sélectionnés, un

document *X* contenant plusieurs mots est représenté sous la forme d'un vecteur. On essaye de trouver le correspondant le plus proche parmi les documents stockés qui sont représentés aussi sous forme des vecteurs. On choisi un groupe de documents qui ont les meilleures correspondances en se basant sur la distance entre deux vecteurs. On examine les étiquettes de ces documents sélectionnés et on choisit l'étiquette qui apparait le plus fréquemment, comme étiquette du document *X*. La figure suivante, prise du livre de Sholom M. Weiss et al. [6], compare la prédiction non supervisée et la recherche d'information, qui se basent toutes les deux sur l'approche de similarité.

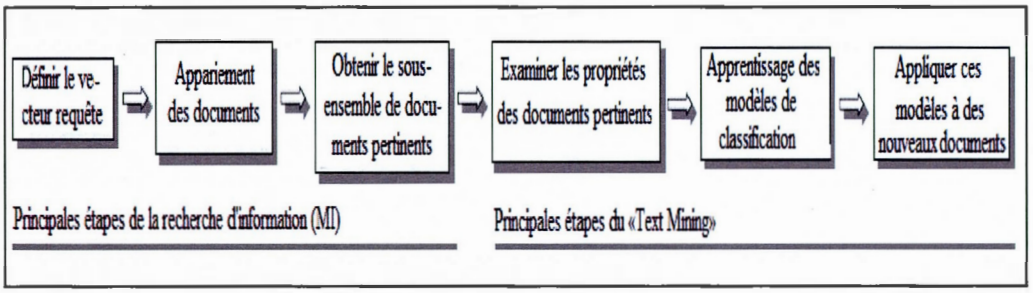


Figure 1.9 Comparaison entre la prédiction supervisée et la recherche d'information (RI) : la prédiction à partir des documents recherchés et sélectionnés.

L'équation la plus générale utilisée pour comparer la distance entre deux vecteurs est la distance euclidienne :

$$Distance(X, Y) \sqrt{(x1 - y1)^2 + \ldots + (xm - ym)^2}$$

Pour le texte, deux documents sont similaires s'ils ont le plus de mots communs en se basant sur les mots qui ont des valeurs positives (égale à 1 dans le cas de matrice binaire) dans les deux documents et non pas les valeurs égale à zéro 0. L'utilisation des mots positifs (égal à 1) pour le calcul de similarité est faite pour éliminer les valeurs bruits (égal à 0), pour que ces dernières n'affectent pas le résultat final. En pratique l'utilisation de la variante *TF-IDF* donnera souvent le meilleur résultat pour le calcul de similarité [6]. Le calcul effectif de la distance dans ce cas est appelé la « similarité cosinus » et a été largement utilisée pour la recherche d'information [6]. La similarité cosinus est fréquemment utilisée en tant que mesure de ressemblance entre deux documents. Un document vectorisé est constitué par les

mots de document et est comparé par mesure de cosinus de l'angle avec des vecteurs correspondants à tous les documents présents dans le corpus. Le calcul de la similarité-cosinus se fait de la manière suivante, sachant que le poids du mot dans le document W_j est calculé par la formule *TF-IDF* (term frequency–inverse document frequency), où j est le $j^{ème}$ mot dans le dictionnaire, TF_j est sa fréquence dans le document, N est le nombre de documents dans la collection d'apprentissage et enfin, DF_j est le nombre des documents dans lesquels le mot apparait :

$$\begin{aligned} w(j) &= \mathrm{tf}(j) * \log_2(N/\mathrm{df}(j)), \\ \mathrm{norm}(D) &= \sqrt{\sum w(j)^2}, \\ \mathrm{cosine}(d1, d2) &= \sum (w_{d1}(j) * w_{d2}(j))/(\mathrm{norm}(d1) * \mathrm{norm}(d2)) \end{aligned}$$

Figure 1.10 Calcul de la similarité-cosinus.

Au lieu de comparer un document X à chaque document du corpus du texte, une façon pratique évoquée dans [6], consiste à construire un vecteur composite de documents et ensuite le comparer à X. Ce vecteur s'appelle un vecteur résumant étant donné qu'il résume un ensemble de documents appartenant au même groupe. Le vecteur moyen est calculé pour le groupe une seule fois, et le nouveau vecteur résultant sera utilisé pour une comparaison directe avec d'autres vecteurs. L'utilisation de ce vecteur moyen est considérée comme une comparaison plus rapide que de comparer des paires individuelles de différents groupes [6].

1.3.2.2 Les méthodes de prédiction les plus connues

a) Regroupement k-means

k-means est une méthode de classification classique qui a été adaptée aux documents textes. Elle est largement utilisée pour les documents texte et est relativement efficace. Le concept est qu'on commence avec une seule pile contenant tous les documents, ensuite cette pile est distribuée en plusieurs petites piles. À des fins de prédiction, chaque pile de documents peut être considérée comme une catégorie qui possède un label unique. L'algorithme procède comme suit [6] :

1. Distribuer tous les documents entre les *k* piles
2. Calculer le vecteur moyen pour chaque pile
3. Comparer le vecteur de chaque document au vecteur moyen de la pile et noter le vecteur moyen qui se rapproche le plus
4. Déplacer tous les documents vers leurs piles les plus semblables
5. Si aucun document a été déplacé vers une nouvelle pile (stabilité), arrêter, sinon, passez à l'étape 2.

Figure 1.11 L'algorithme k-means [6].

Le nom k-means implique qu'on aura k groupes utilisés. La moyenne de chaque groupe aura un important impact. Les documents seront déplacés entre un nombre fixe de piles jusqu'à ce qu'on ne détecte aucun changement entre les groupes. Il existe diverses approches pour déplacer initialement les documents. L'approche classique consiste à assigner d'une façon aléatoire les documents entre les k piles. Étant donné que l'algorithme s'arrête à un minimum local, plusieurs exécutions avec une assignation aléatoire peuvent être nécessaires pour obtenir la meilleure réponse. Une approche alternative consiste à calculer le vecteur moyen de tous les documents, calculer la similarité de tous les documents par rapport à ce vecteur, trier les mesures de similarité et les documents correspondants et attribuer un nombre égal de documents pour chaque pile dans l'ordre de tri. L'algorithme du k-means est ensuite appliqué. Dans ce cas, seulement les déplacements entre les piles adjacentes sont considérés pour déplacer les documents. Notons que si les documents originaux sont présentés sous la forme de vecteurs normalisés, le vecteur moyen peut-être aussi normalisé. La figure suivante est un exemple de l'application de l'algorithme k-means sur 5 documents, pour deux groupes ($k = 2$) et où les vecteurs ne contiennent qu'un seul mot mesuré par la fréquence.

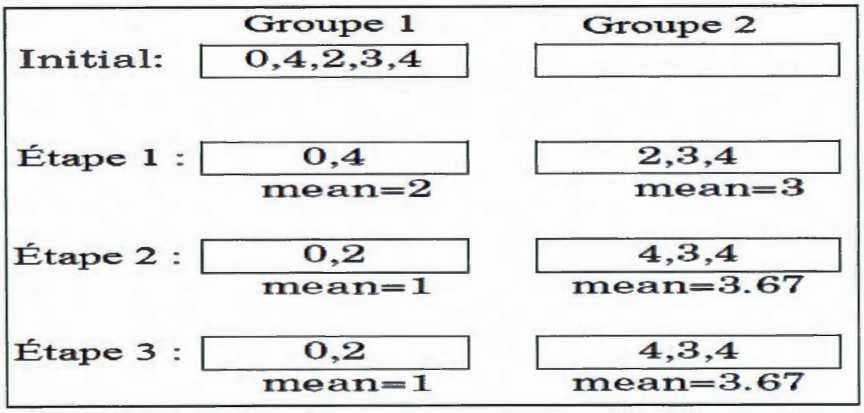


Figure 1.12 Exemple d'exécution de K-means pour $k=2$ [6].

Quand le nombre de piles augmente, l'efficacité diminue. Le grand problème pour cet algorithme est de déterminer k, le nombre de groupes. La méthode n'a aucune connaissance pour déterminer k.

b) Regroupement hiérarchique

Le regroupement hiérarchique est une alternative populaire au k-means. Comme prévu, la méthode produit des groupes homogènes, mais ils sont organisés en une hiérarchie. La similarité est décidée par les moyennes et aussi par la distance maximale et minimale entre les documents au sein d'un cluster. Définissons tout d'abord la distance maximale et la distance minimale. La distance minimale ou le lien simple est la plus petite distance entre un élément d'un groupe et un élément d'un autre groupe, c'est-à-dire, $Dis(K_i, K_j) = Min(t_{ip}, t_{jq})$. La distance maximale ou le lien complet concerne les deux documents les moins similaires entre deux groupes. C'est la plus grande distance entre un élément d'un groupe et un élément d'un autre groupe, c'est-à-dire, $Dis(K_i, K_j) = Max(t_{ip}, t_{jq})$. La faiblesse majeure du regroupement hiérarchique est le temps d'exécution et la complexité du calcul [6]. La méthode est destinée à regrouper un nombre relativement petit de documents. L'algorithme compare toutes les paires de documents et non pas la moyenne de tous les documents. Pour cela, la complexité est de l'ordre du nombre de documents au carré $O(n^2)$. L'algorithme est le suivant [15]:

1. Commencer par l'ensemble des documents comme étant un seul groupe.
2. Trouver la meilleure paire de clusters n'ayant pas de parents: B et C.
3. Combiner les documents de B et C dans un cluster parent A.
4. Si plus d'un cluster reste sans des parents, aller à l'étape 2.

Figure 1.13 L'algorithme de regroupement hiérarchique [6].

Le résultat serait un arbre binaire. Le nœud parent contient tous les documents, et il est récursivement partitionné en groupes plus petits. Contrairement à l'induction d'arbres de décision pour la prédiction, les arbres construits à partir d'un algorithme de regroupement hiérarchique agglomératif (HAC) sont compilés à partir du bas vers le haut. Les paires de clusters sont récursivement combinées jusqu'à ce qu'il ne reste qu'un seul cluster. Les nœuds terminaux de l'arbre sont les clusters réels. Un arbre peut être coupé pour trouver un petit arbre qui est préférable à l'arbre trouvé. La figure suivante donne un petit aperçu du résultat d'exécution de la méthode hiérarchique sur un ensemble de données [15].

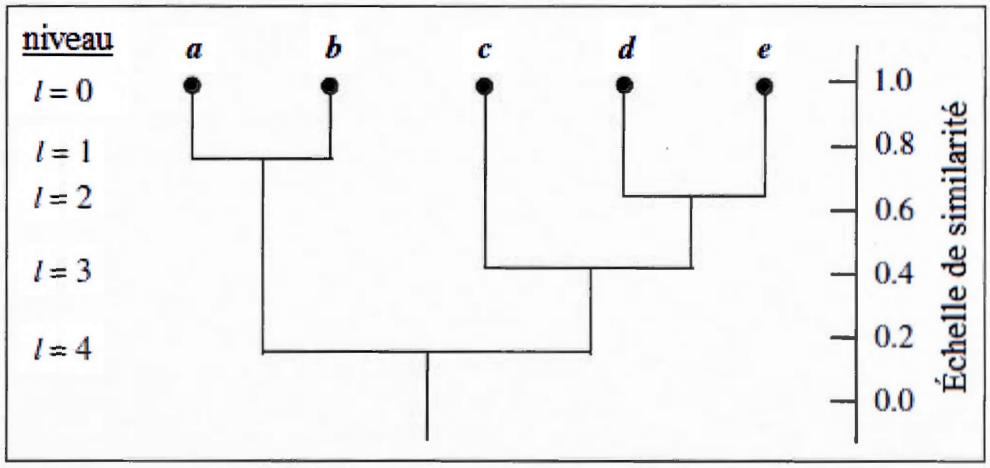


Figure 1.14 Un dendrogramme représentatif du regroupement hiérarchique des données a, b, c et d.

On peut avoir le nombre de groupes voulu en coupant l'arbre de base à l'endroit approprié. Par exemple, pour avoir deux groupes, on coupe l'arbre de décision juste au niveau du noeud racine.

1.3.3 Comparaison entre les différentes méthodes d'apprentissage

La prédiction non supervisée basée sur l'approche de similarité peut être appliquée lorsqu'on n'a pas beaucoup d'information sur la structure des documents. En termes d'application de prédiction, le regroupement des documents fournit une solution directe à un problème d'acquisition de données. Il nous évite de faire un effort additionnel pour collecter des données étiquetées. Nous n'avons pas d'étiquettes. Concernant l'algorithme k-means, vu précédemment, on attend que le nombre de groupes soit petit par rapport au nombre de documents. Chaque comparaison est avec la moyenne de groupe, ainsi le calcul et la comparaison augmentent d'une façon linéaire avec le nombre de documents [6]. Pour le regroupement hiérarchique, la similarité entre toutes les paires de documents est calculée avant le regroupement, en comparant toutes les paires de documents et non pas la moyenne avec tous les documents. Le problème majeur des méthodes basées sur la similarité est la détermination du paramètre k, le nombre de groupes. Généralement, le nombre de groupes k doit être plus petit que le nombre de documents, sinon, la prédiction et la généralisation sur des nouveaux documents seraient affaiblies [6]. La performance du classificateur KNN est principalement déterminée par (i) le choix approprié de k qui peut être très difficile si les données ne sont pas uniformément réparties ou s'il ya des données bruitées, et (ii) la distance métrique appliquée. L'avantage de l'algorithme KNN est qu'il offre un apprentissage rapide [4]. Mais, la précision de l'algorithme se dégrade avec l'augmentation du bruit dans les données d'apprentissage. En ce qui concerne les méthodes de classification supervisées, selon J. Han and M. Kamber [15], on peut les comparer en se basant sur (i) l'exactitude qui réfère à l'habilité du modèle à identifier avec exactitude la classe des données, (ii) le temps de traitement, (iii) la robustesse qui réfère à l'habilité du modèle à réaliser des prédictions exactes en présence de données erronées ou en cas de données insuffisantes, (iv) l'extensibilité qui définit la capacité de générer le modèle lors de l'utilisation d'une grande base d'apprentissage et (v) l'interprétabilité, c'est-à-dire, la capacité d'interpréter les résultats

fournis par le modèle. En se basant sur ces cinq propriétés, nous essaierons d'identifier les principales caractéristiques distinctives des principaux modèles de classification.

Les principaux points forts des arbres de décision sont : (1) l'apprentissage et la classification sont généralement rapides, (2) le taux d'exactitude est comparable aux autres méthodes de classification (réseaux de neurones, k-NN, etc.) [15] (3) ils sont très utiles dans des domaines basés sur des règles, (4) leurs résultats sont interprétables, en effet, ils permettent la production de règles explicites, et (5) ils sont caractérisés par la facilité du calcul lors de la classification. Enfin, des erreurs sur les données sont admissibles et on peut avoir des attributs manquants. Leurs points faibles sont : (1) il n'y a pas de contrôle sur le nombre de branches élaborées ; certaines branches peuvent représenter des anomalies dans les données (absences ou données erronées), plusieurs techniques (prepruning, postpruning) ont été utilisées pour élaguer l'arbre (C4.5 et CART), (2) temps de computation coûteux et (3) ils peuvent exiger de grandes ressources en mémoire [19].

Les points forts des classificateurs bayésiens sont : (1) la simplicité, (2) la performance en présence d'une grande base d'apprentissage, et (3) théoriquement, ils ont le taux d'erreurs minimal comparé aux autres classificateurs [15]. Leurs points faibles viennent du fait que dans la pratique, leur performance peut diminuer dans certains domaines [15] ; Ceci est dû aux faits que l'hypothèse d'indépendance entre les variables n'est pas toujours vérifiable dans le cas des classificateurs naïfs, et à la difficulté de déterminer les probabilités conditionnelles de certaines variables dans le cas de réseaux bayésiens. Toutefois, les classificateur bayésiens naïfs sont prédisposés à « l'empoisonnement bayésien », une situation où un document classifié dans une catégorie *X*, mêle une grande quantité de données qui caractérisent une autre catégorie *Y*, pour contourner le mécanisme de classification probabiliste [4].

En ce qui concerne les réseaux de neurones, (1) ils sont utiles dans le cas où l'accent est mis sur le taux de précision et non pas sur l'interprétation et lorsqu'il a suffisamment d'exemples d'apprentissage. (2) Ils sont tolérants aux bruits. Leur point faible est qu'ils (1) demandent une durée d'apprentissage assez longue, (2) ils nécessitent la spécification de certains paramètres qui sont généralement déterminés de manière empirique (architecture du réseau, initialisation des poids, etc.) et (3) il est difficile d'interpréter la signification

symbolique des divers changements de poids et d'extraire des règles d'induction (interprétabilité). Ceci dit, il existe des algorithmes d'extraction de règles à partir des réseaux de neurones [1]. Enfin, les méthodes d'apprentissage utilisées par le réseau de neurones peuvent rencontrer le problème de sur-apprentissage à cause de l'application de ces méthodes plusieurs fois et du fait qu'on n'a pas une condition d'arrêt logique. Au contraire, l'arrêt de l'algorithme d'apprentissage se fait d'une manière empirique [15].

Dans le même contexte d'approche, les SVM réussissent toujours à trouver une solution globale au problème ce qui n'est pas le cas des réseaux de neurones. Les principaux avantages des SVM sont : (1) le temps d'apprentissage court, (2) le taux d'exactitude élevé et (3) la faible sensibilité au problème du sur-apprentissage [15]. Cependant, elles ne produisent pas de connaissances explicites et interprétables.

1.4 Conclusion

La tâche d'apprentissage automatique sur des données textuelles doit obligatoirement passer par une étape primordiale qui consiste au prétraitement de données textuelles. Cette étape est longue et suit des procédures soigneusement organisées pour former des données structurées sous forme de matrice. Cette étape commence par une collecte de données qui peuvent être publiques (LingSpam, Reuters) ou obtenues suite à un processus de collection de données. Le processus de préparation de données passe par plusieurs étapes. Chaque étape permet de modifier les données passées en entrée. Toutes ces étapes de préparation de données doivent être appliquées d'une façon organisée, dans un processus bien déterminé et bien ordonné afin de passer d'un ensemble de données textuelles brutes vers un vecteur numérique. L'ensemble de tous les vecteurs numériques forme la matrice des données documents-mots.

La prédiction se fait en utilisant la matrice documents-mots finale. Les modèles de prédiction sont assez variés. Ils peuvent être partagés en deux groupes selon le type de prévision. Ainsi, la première famille regroupe les méthodes d'apprentissage supervisées et la deuxième famille regroupe les méthodes non supervisées. Selon un autre angle de vue qui prend en compte un critère technique de prédiction, on peut diviser les méthodes

d'apprentissage automatique en quatre catégories : les méthodes logiques, les méthodes probabilistes, les méthodes statistiques et les méthodes de plus proche voisin.

Nous présentons dans le prochain chapitre l'application des méthodes d'apprentissage automatiques sur un type de données textuelles particulier ; les courriels. Un état de l'art sur le filtrage des courriels au moyen de diverses méthodes d'apprentissage automatique sera détaillé.

CHAPITRE II

FILTRAGE DE COURRIELS AU MOYEN DE DIVERSES MÉTHODES D'APPRENTISSAGE AUTOMATIQUE : UN ÉTAT DE L'ART

2.1 Introduction

Le filtrage des courriels dans le cadre de l'apprentissage automatique consiste à classifier les messages reçus dans la boite de réception client en messages légitimes et en messages spams. C'est une classification textuelle binaire qui se base sur les méthodes d'apprentissage automatique. Les recherches dans ce domaine sont très actives [29][30][31][32][33][34] ; il y a plusieurs façons d'appliquer le filtrage ou la détection des spams au moyen de méthodes d'apprentissage automatique. Plusieurs recherches se sont concentrées sur l'utilisation de tout le contenu de courriel pour la classification. D'autres se sont contentées de la sélection de quelques caractéristiques comme attributs d'apprentissage. Toutefois, la plupart des recherches appliquent les méthodes d'apprentissage automatique supervisées afin de catégoriser les courriels. Les méthodes de prédiction non supervisées sont généralement utilisées dans quelques travaux, exclusivement pour soutenir les méthodes d'apprentissage supervisées [7]. La performance de ces méthodes diffère d'un travail à un autre. En effet, plusieurs outils de mesure des performances ont été utilisés. Dans ce chapitre, nous commençons par établir les techniques de mesure de performances adoptées dans les recherches les plus récentes. Nous présentons ensuite les approches les plus utilisées dans la classification des courriels au moyen de méthodes d'apprentissage automatique.

2.2 Évaluation de performance

Pour évaluer la performance d'une solution, on divise l'échantillon de données en deux catégories. Typiquement, la première partie sera dédiée à l'apprentissage (training), et nous testons sur le reste de l'échantillon. Selon Sholom M. Weiss et al. [6], la plupart des

évaluations des solutions pour le « Text Mining » ordonnent l'échantillon dans le temps et utilisent les documents antérieurs à l'apprentissage et les nouveaux documents pour le test comme le montre la figure suivante, prise du livre des auteurs cité ci-dessus [6].

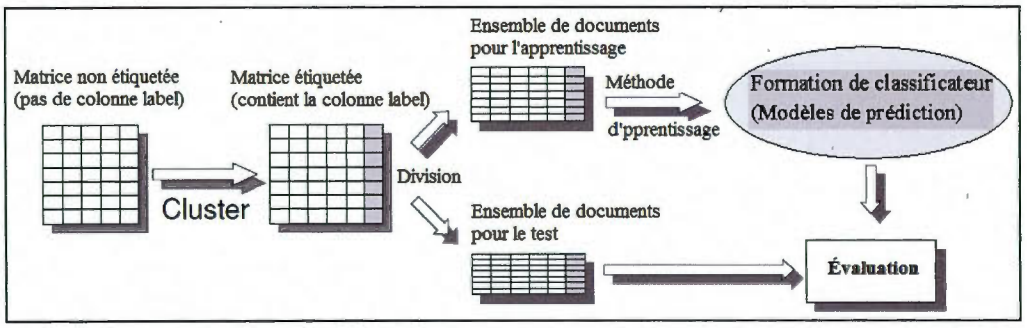


Figure 2.1 La partition de l'ensemble de données pour la prédiction et l'évaluation [6].

La performance peut être estimée en fonction de plusieurs mesures. Ces mesures s'appuient sur des ratios dont les plus importants sont le ratio de faux positifs et le ratio des faux négatifs. Le ratio de faux positifs correspond aux courriels désirables qui ont été classifiés comme des spams par le classificateur, contrairement au ratio de faux négatifs, qui représente les spams qui ont été classifiés comme des courriels légitimes. Toutes les méthodes de prédiction doivent pouvoir être en mesure de maîtriser ces risques et viser à rendre le ratio de faux positifs nul et de baisser au maximum le ratio de faux négatifs. Le ratio de faux positifs est le plus critique car il a un impact plus dangereux sur l'utilisateur que le ratio de faux négatifs, en rejetant un message légitime que le classificateur considère comme un spam. En effet, La mesure la plus générale et la plus standard est le "taux d'erreur" (Error Rate), son erreur-type (Standard Error, SE) et la précision (Accuracy). Le taux d'erreur n'est que le nombre d'erreur de classification (nombre de documents mal classifiés) divisé par le nombre total de documents. L'erreur-type est donnée par l'équation suivante [6]:

$$SE = \frac{\sqrt{\text{erate} - (1 - \text{erate})}}{\sqrt{\text{nombre de documents}}}$$

La précision qui est complémentaire du taux d'erreur ($Err = 1 - Acc$) attribue des pondérations égales aux deux types d'erreurs (faux positifs et faux négatifs). C'est le rapport entre le nombre de bonnes prévisions positives et le nombre de prédictions positives. Elle mesure la sécurité, c'est-à-dire le degré avec lequel les messages bloqués sont vraiment des spams. Le rappel est le rapport entre le nombre de bonnes prédictions positives et le nombre total d'exemples. Il mesure le pourcentage de messages non légitimes que le filtre a bloqué, à savoir son efficacité. Le seuil de rentabilité est quand la précision et le rappel sont égaux [22]. Bien que le taux d'erreurs et sa précision associée soient en général utiles pour estimer la performance de la prédiction pour la plupart des applications du texte, d'autres mesures plus sophistiquées sont utilisées (F-mesure, TCR et le point d'équilibre entre le rappel et la précision (Precision/Recall-Breakeven point)). La plupart de ces mesures combinent la précision et le rappel pour avoir une seule mesure. Dans [17], les auteurs utilisent une variante de la précision et du rappel (recall). La variante est la même pour les deux mesures et elle met l'accent sur le fait que les faux positifs sont plus critiques que les faux négatifs. On attribut donc aux faux positifs un poids λ pour souligner que les faux positifs sont λ plus coûteux que les faux négatifs. Les auteurs utilisent aussi une autre mesure qui est le TCR, le ratio de coût total (total cost ratio). Le TCR mesure le temps nécessaire pour supprimer manuellement tous les messages spams quand aucun filtre n'est utilisé (N_{spam}), comparé au temps nécessaire pour supprimer manuellement tous les messages spams qui ont passé le filtre (faux négatif), plus le temps nécessaire pour récupérer les messages légitimes bloqués (faux positif) [17].

$$TCR = \frac{WErr^{b}}{WErr} = \frac{N_{spam}}{\lambda \cdot n_{legit \rightarrow spam} + n_{spam \rightarrow legit}}$$

L'importance de la mesure TCR réside dans sa capacité de combiner entre la précision et le rappel ce qui facilite énormément la comparaison entres les classificateurs en se basant sur une seule mesure unifiante. S. M. Namburu et al. [22], utilisent le point d'équilibre précision/rappel qui indique l'égalité entre la précision et le rappel. Cette mesure permet aussi d'unifier les deux mesures. Sholom M. Weiss et al. [6], utilisent la F-mesure qui est définie comme la moyenne harmonique de la précision et au rappel.

$$F-mesure = \frac{2}{\frac{1}{presicion} + \frac{1}{recall}}$$

Toutes ces mesures sont utilisées essentiellement pour comparer les différentes approches et les différentes méthodes utilisées dans la classification des courriels.

2.3 Aperçu sur la classification des courriels au moyen des méthodes d'apprentissage automatique

La classification des courriels qui consiste à l'application binaire de la tâche de catégorisation du texte en classant un ensemble de courriels en deux catégories, spam et non spam, met en jeu plusieurs techniques d'apprentissage. Un travail d'investigation des travaux qui se concentre sur la façon de préparer et de traiter les données nous mène à les répartir en trois grandes famille. On peut citer l'approche statistique de classification du texte qui met en jeu les méthodes classiques d'apprentissage automatique en se basant sur tout le contenu des courriels et sur l'occurrence des mots dans les documents ; elle suppose que les attributs sont indépendants les uns des autres [14][16][17]. Contrairement à cette dernière, il existe une autre approche qui utilise quelques caractéristiques comme attributs d'apprentissage en se basant sur un critère de sélection de caractéristiques ou en se basant principalement sur des caractéristiques particulières des courriels comme « Subject » ou « To » [8][10][20][32][34]. Une autre approche est basée sur le contexte des mots dans le corpus du texte en prenant en compte les dépendances linguistiques et syntaxiques complexes, entre les attributs et les caractéristiques, comme indicateurs de contexte [23][21][24]. Les travaux reposant sur l'approche à base de contexte ne sont pas très évolués [4]. Les méthodes existantes analysent la pertinence contextuelle des mots dans le corpus du texte. Elles utilisent des algorithmes linguistiques sophistiqués [6] pour identifier les significations des mots et leurs différentes relations sémantiques. Cette approche met l'accent sur la dépendance et la corrélation entre les mots dans un document. Par exemple, il y a quelques mots qui ne sont significatifs que lorsqu'ils se présentent avec d'autres mots. Les méthodes de classification à base de contexte identifient les associations entre les mots afin d'améliorer la classification [23]. Pour ce faire, on doit passer par des procédures linguistiques et sémantiques. Ces procédures viennent après la tokenisation et avant la génération des vecteurs numériques. Prenons quelques exemples de

ces techniques linguistiques. Tout d'abord, (1) la détermination de la limite des phrases est essentielle pour pouvoir décider si une ponctuation donnée est une limite d'une phrase ou elle est belle et bien un caractère. (2) l'étiquetage grammatical (POS, Part-Of-Speech Tagging) est une sorte de classification du texte sur des classes grammaticales (nom, verbe,...). Pour cette raison, on utilise souvent des dictionnaires de correspondance mot/POS (parti de discours), comme le «Penn Tree Bank», qui contient le plus grand ensemble de catégories dans la langue anglaise. L'étiquetage grammatical doit être en mesure de déterminer la fonction grammaticale d'un mot dans un contexte ambigu, comme par exemple "voie" qui peut être un nom ou un verbe conjugué. C'est la tâche la plus difficile dans le marquage de la partie de discours. Dans une perspective sémantique, (3) résoudre le problème de l'ambigüité sémantique du sens de mot (Disambiguation) joue aussi un rôle important dans la détermination de la sémantique des mots. Il existe des outils capables de déterminer le sens d'un mot comme WordNet [6]. WordNet est une base de données lexicale développée à l'université de Princeton [42]. Il se base sur un dictionnaire qui stocke toutes les significations de chaque mot. (4) La reconnaissance des entités nommées est aussi une procédure importante pour identifier des types particuliers comme le temps, les dates, les endroits, les noms propres,... En effet, il existe plusieurs outils dont le rôle est de déterminer les liaisons et les dépendances entre les mots comme RIPPER [23]. RIPPER est une méthode qui implémente un algorithme de règles de décision, avec une répétition incrémentale pour produire une réduction d'erreur [41]. C. Donghui and L. Zhijing [21] utilisent par exemple les modèles de Markov cachés (HMM, Hidden Markov Models) afin d'éliminer les informations inutiles dans les documents et améliorer la classification. L'avantage de l'approche à base de contexte est qu'elle permet d'augmenter l'espace de description et de réduire le taux de rappel (recall) [4]. Dans [24], les auteurs proposent une approche non-paramétrique en combinant la méthode KNN avec la technique d'indexation LSA (Latent Semantic Analysis). Cette technique analyse les données et attribue un ensemble de liens de dépendance entre les attributs et le contexte dans lequel l'attribut est utilisé. Les résultats sur la base de données standards LingSpam montrent une précision très élevée, plus de 99.56 %. Cependant, la classification à base de contexte entraine un effort et un coût supplémentaire avant la classification [6].

L'approche statistique classique qui utilise tout le contenu des courriels se caractérise principalement par l'exploitation de l'occurrence des mots existants dans les documents. C'est l'approche la plus utilisée et la plus standard [4]. Cette approche nous permet d'épargner l'effort de l'analyse de la pertinence contextuelle des mots dans un document. Elle évite l'étude linguistique des attributs et ne se préoccupe que de la fréquence des mots dans le corpus du texte. Plusieurs travaux de recherche se penchent sur ce type d'approche. Dans [16], les auteurs utilisent une variante de classificateur bayésien naïf qui utilise des vecteurs d'attributs basés sur la position de mot. En plus de l'utilisation des probabilités à posteriori qui supposent que les attributs sont conditionnellement indépendants, le travail introduit une hypothèse additionnelle : la probabilité d'un mot W_i qui se présente à la postions i est identique à la probabilité du même mot se présentant à une position k. Le classificateur résultant est équivalent à un classificateur bayésien naïf qui se base sur un modèle d'événement multinomial. Il utilise aussi la notion de N-gram. Le résultat de la classification a donné de bonnes performances. Dans [17], I. Androutsopoulos et al. ont crée un nouveau corpus de courriels contenant des messages légitimes et des spams. Ils démontrent que la précision du classificateur Bayésien naïf dépasse 99%. C. Apte et al [14] traitent tout le contenu des messages dans des dictionnaires locaux. Ensuite un dictionnaire global est généré pour les attributs de fréquences globales supérieures à 4. Les auteurs ont expérimenté une approche qui utilise un arbre de décision comme méthode d'apprentissage, couplée avec le ré-échantillonnage adaptif (boosting), pour former plusieurs arbres de décision à partir du même ensemble de données, et en remplaçant à chaque étape de construction d'arbre, l'échantillon d'apprentissage. Cette méthode réduit le phénomène de sur-apprentissage du à la formation d'un arbre de décision unique et très spécifique. Les résultats de cette approche sur la base des données Reuters-21578 ont été extrêmement bons [14].

L'approche qui sélectionne les caractéristiques les plus importantes pour la classification est semblable à l'approche statistique classique qui utilise tout le contenu des courriels. Elle introduit en plus, des mesures d'information et des mesures mathématiques comme l'information mutuelle (MI) et le gain d'information (IG) pour sélectionner les attributs les plus pertinents. Elle peut utiliser dès le début, quelques caractéristiques particulières aux courriels, comme attributs, au lieu d'utiliser tout le contenu. H. Chen et al

[32], proposent un modèle de classification des courriels dans la langue chinoise basé sur un arbre de décision. La classification se base principalement sur quelques caractéristiques extraites de la collection de documents. Ces caractéristiques permettent de classifier les courriels en spams ou en messages légitimes ; ce sont la taille du courriel, la quantité d'images dans le courriel, le sujet, le nom de l'émetteur,... Ensuite, ils appliquent un algorithme d'arbre de décision pour analyser les règles d'association potentielles entre ces attributs des courriels. Il en résulte une base de règles par laquelle les courriels seront classés en spams ou non spams.

L'application de ce travail sur 1000 courriels, moitié spams et moitié non spams, montre que son efficacité n'est pas inferieure aux méthodes qui se basent sur tout le contenu des courriels. Elle diminue entre autre la consommation des ressources de système. Dans [10], H. Dalianis investigue sur les parties de courriels les plus prédictives pendant un fil de discussion. Il expérimente cette recherche sur quatre types de données : question seulement, seulement la réponse, la question et la réponse ou tout le fil de discussion. Les résultats ont montré que seulement la partie contenant la première question et la première réponse d'un courriel contenant un fil de discussion est importante. A. Kolcz et J. Alspector [20] évoquent la nécessité de procéder à la classification avec un grand espace d'attributs en se basant sur tout le contenu des messages, l'entête et le corps du message et en éliminant la partie HTML. Il supprime les mots de fréquence inferieure à 3. Il adapte enfin la sélection de caractéristiques en s'appuyant sur le critère de l'information mutuelle. Pour former un classificateur, les auteurs ont choisi l'algorithme SVM qui est considéré comme le classificateur le plus performant capable de gérer des données qui se caractérisent par une grande dimensionnalité des attributs telles que les données textuelles. Z. Q. Wang et al. [34] proposent une méthode de classification des courriels en utilisant la sélection des caractéristiques et l'algorithme SVM. En plus de l'application de la lemmatisation des mots et de l'utilisation de «Stop List», les auteurs appliquent la sélection des caractéristiques pour réduire la haute dimension des données textuelles. Le critère de sélection utilisé est le gain d'information (IG). L'application expérimentale montre que cette méthode donne des bons résultats avec une précision égale à 94,75 %. Un autre travail où on a recours à la sélection

des caractéristiques pour faire face à la grande dimensionnalité des données textuelles est celui de Y. Li et al. [8]. Il combine plusieurs techniques de sélection de caractéristiques.

Nous avons réparti les travaux de recherche en trois approches, en nous basant sur la façon avec laquelle les données sont préparées. L'approche la plus sophistiquée est celle à base de contexte. L'approche la plus simple est d'utiliser tout le contenu. Une alternative à l'approche qui se base sur tout le contenu est d'appliquer une sélection des caractéristiques sur le contenu des courriels. Cette alternative simplifie la tâche de classification mais entraine un effort et un coût supplémentaire lors du traitement de données textuelles. Certes, il n'existe pas un travail qui compare les trois approches sur les mêmes données textuelles et avec les mêmes mesures d'évaluation, mais les trois approches ont donné de bons résultats [24][14][34].

Nous abordons dans ce qui suit, un aperçu sur les classificateurs de filtrage anti-spams à l'aide de l'apprentissage automatique, sous un autre angle qui repose sur la méthode utilisée pour la classification des courriels. Les premières recherches, principalement axées sur le filtrage d'courriels au moyen de l'apprentissage automatique, sont celles de W. Cohen [29] en 1996 qui classe les courriels, en s'appuyant sur les règles de décision. Les règles de classification utilisées sont des combinaisons booléennes de chaines et d'expression régulières. M. Sahami et al [30] ont appliqué l'algorithme Bayésien naïf (NB) pour construire un classificateur anti-spam. L'application de la méthode SVM a également été proposée par H. Drucker et al. [31] en 1999. En 2004, P. I. Nakov et al [24] ont proposé aussi une approche de filtrage des courriels qui introduit un classificateur KNN. Une approche plus récente qui s'appelle « Boosting » et qui consiste à former plusieurs classificateurs, comme les arbres de décision, sur le même ensemble de données et en utilisant une technique d'échantillonnage, a été introduite en 2007 par J. He et B. Thiesson [33]. Cette approche permet de réduire la variance dans les données et évite le sur-apprentissage. La plupart des méthodes ont donné de bons résultats [16][32][24], ce qui prouve l'efficacité de la classification des courriels au moyen des méthodes d'apprentissage automatique. Dans [16], J. Hovold montre que la méthode bayesienne naïve peut donner de bons résultats avec une précision égale à 98,8 % et un rappel égal à 91,67 %. Toutefois, les classificateurs bayésiens naïfs doivent faire face au problème d'empoisonnement bayésien [4]. Les expériences de H.

Chen et al. [32] sur un ensemble de courriels chinois se basant sur l'arbre de décision ont donné aussi de bons résultats de classification. Bien que les arbres de décision fournissent des explications sur la classification, la performance des ces derniers peut diminuer à cause de la grande quantité de données et la grande variation dans les données textuelles [14]. L'utilisation d'un classificateur KNN [24] a donné également des résultats adéquats. Bien que toutes les méthodes statistiques ont montré leurs efficacité dans la classification des courriels, plusieurs travaux de recherche affirment que la méthode SVM est la plus efficace pour la classification binaire des données de grande dimensionnalité, telles que les données textuelles [6][21][34][22]. Les machines à vecteur de support ont surpassé la plupart des algorithmes de classification [21]. Cette méthode ne s'affaibli pas à cause de la dimensionnalité de l'espace de caractéristiques. Elle se concentre plutôt sur la maximisation de la marge entre les exemples positifs et les exemples négatifs du corpus de données. Z. Q. Wang et al. comparent dans [34], plusieurs méthodes de classification sur le même ensemble de données textuelles. Les résultats ont donné une précision de l'algorithme SVM supérieur à celui de NB et KNN. Un autre travail de recherche qui compare les méthodes de classification est fait par C. Apte et al. [14]. Leur expérimentation est faite sur la base de données REUTERS-21578 et a montré que la performance de l'algorithme SVM a devancé les méthodes bayésiennes naïves, Rocchio, arbres de décision (C4.5), kNN et les règles d'induction optimisées. La figure suivante montre les résultats obtenus [14].

Prediction Method	*Performance Breakeven (%)*
Naive Bayes (linear)	73.4
Rocchio (linear)	78.7
Decision Tree C4.5	78.9
k-nearest neighbor	82.0
Optimized Rule Induction	82.0
Support Vector (rbf)	86.3

Figure 2.2 Comparaison des performances de quelques méthodes d'apprentissage sur l'ensemble de données textuelles publique REUTERS-21578.

2.4 Conclusion

Nous avons établi dans ce chapitre un aperçu sur les travaux qui ont été faits pour classifier les courriels au moyen de méthodes d'apprentissage automatique. Notre état de l'art à été établi selon deux points de vues ; un se concentrant sur les méthodes de classification utilisées (SVM, KNN, NB,...) et l'autre sur la façon dont les travaux ont procédé avec les données, avant la classification. La plupart des travaux ont donné de bons résultats. La méthode SVM a donné les meilleurs résultats dans plusieurs travaux. Nous nous concentrons dans le prochain chapitre sur le prétraitement des données textuelles. Nous présentons ainsi notre outil de préparation de données basé sur la programmation modulaire que nous appelons PDTPM : plateforme de Préparation de Données Textuelles basée sur la Programmation Modulaire. Cet outil sera utilisé pour manipuler l'ensemble de données que nous traiterons pour expérimenter les méthodes d'apprentissage automatique dans le chapitre 4. Non seulement il permet de nettoyer les données textuelles brutes, PDTPM permet surtout la sélection des caractéristiques jugées les plus pertinentes.

CHAPITRE III

PDTPM : UNE PLATEFORME DE PRÉPARATION DE DONNÉES TEXTUELLES BASÉE SUR LA PROGRAMMATION MODULAIRE

3.1 Introduction

Pour pouvoir utiliser un ensemble de données textuelles dans l'apprentissage automatique, nous avons besoin de le prétraiter et de l'organiser sous une forme bien structuré. Nous avons essayé d'utiliser des outils déjà existants mais nous avons contrarié plusieurs limites. Plusieurs travaux ne différencient pas l'étape de préparation de données et l'étape d'analyse et de classifications. Pour profiter des modules de préparation de données textuelles intégrés, ces outils nous obligent d'utiliser leurs méthodes d'analyse; ce qui est limitant et nous ne permettent pas d'utiliser des nouvelles méthodes d'apprentissage automatiques comme SVM. Prenons par exemple les outils Alceste, Lexico ou QDA Miner qui offrent des services de préparations de données très limités et intégré dans un processus global d'analyse et de classification. D'autres systèmes n'offrent pas toutes les fonctionnalités de préparation de données et n'intègrent que quelques modules de préparation de données. En fait, la plupart des outils de préparation de données n'offrent que quelques modules de préparations de données et ne sont ni extensible ni flexibles. Toutes ces limites nous a obligé à concevoir un système capable de contourner toutes ces limites.

Le prototype que nous avons conçu est nommé : plateforme de Préparation de Données Textuelles basée sur la Programmation Modulaire (PDTPM). Cet outil nous a été très utile dans l'étape de préparation de données. Il ne s'occupe que des étapes de préparation de données, ce qui lui permet d'être indépendant des méthodes d'apprentissage automatiques et capable d'être utilisé par n'importe quelle méthode. Il permet de créer plusieurs chaînes de préparation de données d'une façon très flexible en choisissant des modules et en supprimant d'autres selon les besoins et la nature des méthodes d'apprentissage qui seront utilisées après.

Sa programmation modulaire nous permet aussi de l'étendre et d'ajouter facilement d'autres modules de préparation de données. Il nous a permis aussi de créer des matrices finales spécifiques à chaque outil d'apprentissage automatique utilisé dans nos expérimentations.

La plateforme prend en entrée un ensemble de données textuelles brutes. Ces données subissent plusieurs modifications pour obtenir des données bien structurées sous la forme d'une matrice de données documents-mots (lignes/colonnes). Le schéma suivant clarifie les domaines qui interviennent durant le processus.

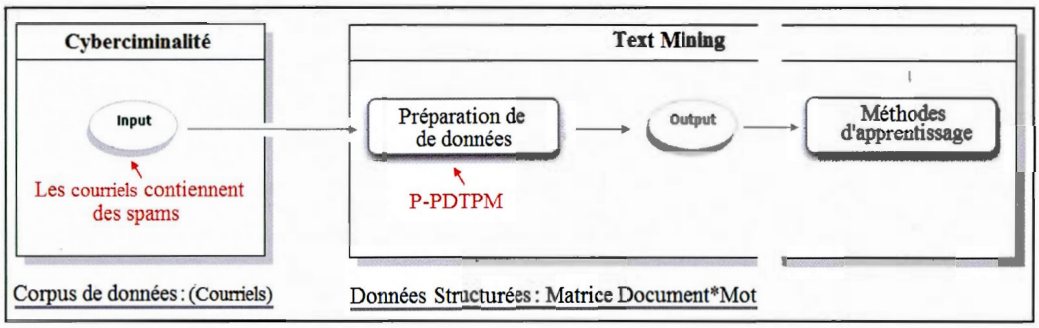


Figure 3.1 Position de PDTPM vis à vis de la cybercriminalité et de l'apprentissage automatique.

Dans notre cas, nous exploitons comme corpus de données un ensemble de courriels afin de les catégoriser en courriels spams et courriels normaux. L'apprentissage automatique prend part au processus lors de l'utilisation des techniques et des méthodes de « Text Mining » sur l'ensemble des courriels. Notre système prépare, en premier lieu, les données entrées en input pour former un contexte formel, c'est-à-dire, la matrice documents-mots, sur laquelle nous pouvons appliquer les techniques d'apprentissage automatique. Notre plateforme est modulaire : elle sépare les composantes de transformation des données textuelles et les présente sous forme d'exécutables ; il suffit donc d'ajouter des composantes exécutables de « Text Mining » pour qu'elle soit capable de gérer tout le processus d'apprentissage automatique, de la préparation jusqu'à la catégorisation. Notre système permet la formation des chaînes de traitement de données textuelles avec une grande flexibilité. Il offre à l'utilisateur une marge de liberté dans la formation des chaînes de

traitement de données textuelles. La figure ci-dessous montre la flexibilité que PDTPM offre pour la formation des chaînes exécutables.

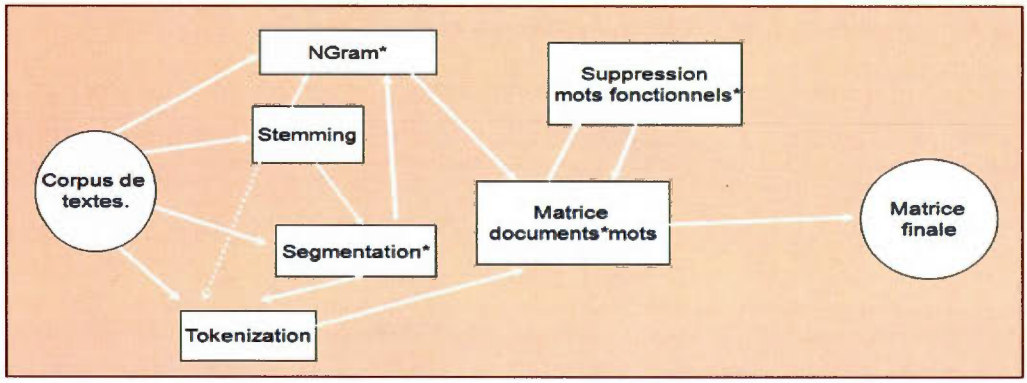


Figure 3.2 Exemple de chaînes exécutables que l'utilisateur peut former au moyen de PDTPM.

Une chaîne de préparation de données exécutable est donc une chaîne qui contient plusieurs modules dans un ordre précis et que l'on applique sur un ensemble de données textuelles non structurées et bruitées. Elle permet d'obtenir, en sortie, des données textuelles structurées sous la forme de matrices documents-mots. Une composante exécutable est un module qui représente une étape du processus de préparation de données. En d'autre terme, une composante exécutable est un composant d'une chaîne de préparation de données. Nous proposons plusieurs modules exécutables comme *ComposerMatrice*, *MatriceBinaire*, *MatriceTfIdf*, *N-gram*, *Tokenization*, *MatriceSansMotsFonctionnels*,... Nous détaillons chaque composante dans le chapitre 4.

Dans ce chapitre, nous commençons par une présentation générale du système en exposant ses objectifs et en dévoilant les utilisateurs et les acteurs de la plateforme de préparation de données. Ensuite, nous présentons une vue fonctionnelle du système à l'aide de diagrammes de cas d'utilisation. Puis, nous mettons de l'avant le schéma relationnel et les exigences logiques de la base de données que nous concevons et qui interagit avec notre système. Ensuite, nous illustrons les scénarios possibles du PDTPM avec des captures

d'écrans de quelques interfaces graphiques. Enfin, une conclusion résume notre chapitre et met l'accent sur les évolutions possibles de notre système.

3.2 Description des utilisateurs et des acteurs

Dans notre système de préparation des données textuelles, nous allons concevoir une interface graphique IHM (Interface Homme Machine) permettant à l'utilisateur principal d'interagir facilement avec le système. Notre système interagit aussi avec une base de données. En effet, il existe un seul utilisateur qui exploite la plateforme. Cet utilisateur principal interagit directement avec l'interface graphique (GUI) de notre plateforme. Il doit avoir des connaissances sur le domaine de l'apprentissage automatique pour qu'il puisse former des chaînes pertinentes. C'est à l'utilisateur de former les modules de préparation des données en glissant les modules voulus et en les mettant dans l'ordre d'exécution approprié. Il est donc très important que l'utilisateur connaisse au moins les étapes de préparation de données, pour que les chaînes ne soient pas erronées. Un exemple d'une chaîne erronée est la suivante :

Stemming(Racinisation) ⟶ ***Segmentation*** ⟶ ***matriceTf*Idf*** ⟶ ***Tokenization*** ⟶ ***Suppression Mots Fonctionnels (SeuilMinimal).***

Dans cette chaîne, nous constatons bien que le module de tokenisation est situé après le module de matrice-TfIdf, ce qui est faux! De plus, la suppression des mots fonctionnels doit être placée avant le module matrice-TfIdf. Pour éliminer cet obstacle, nous avons programmé un module qui fournit des chaînes prêtes à priori. Ces chaînes ont pour but l'assistance de l'utilisateur.

Il existe trois acteurs secondaires qui sont : (1) une base de données qui interagit avec la plateforme pendant l'ajout, la suppression et la consultation d'une composante exécutable (module) et pendant l'enregistrement, la suppression ou la consultation d'une chaîne de préparation de données (CPD), (2) un dossier contenant le corpus de données (input). Ce dossier système qui interagit avec la plateforme pendant l'exécution d'une CPD contient un ou plusieurs fichiers texte. Il doit contenir aussi le fichier Stop-List. (3) Enfin, un dossier contenant le résultat final (Matrice documents-Mots, output). Ce dossier système contient la

matrice finale que nous obtiendrons lorsque le système exécute une chaîne de préparation de données. La figure suivante expose le type de matrice que nous aurons comme résultat.

Attribut / Documents	*Mot1*	*Mot2*	*Mot3*	...	*Motn*
Document 1	*2.0*	*0*	*0*	*0*	*0*
Document ...	*0*	*0*	*3.0*	*0*	*5.0*
Document n	*0*	*1.0*	*0*	*1.0*	*0*

Figure 3.3 Matrice Documents-Mots obtenue en exécutant une chaîne de préparation de données sur un corpus de données.

Nous avons regroupé le dossier contenant le corpus de données (input) et le dossier contenant le résultat final (Matrice documents-Mots) (Output) dans un dossier que nous avons nommé WorkSpace.

3.3 Exigence logique de la base de données

Nous avons utilisé le système de gestion de base de données Mysql. Les classes *Module*, *Chaîne*, *ParametreOption* et *OptionModules* définissent respectivement les tables *Module*, *Chaîne*, *ParametreOption* et *OptionModules* dans notre base de données *BDplateforme*. La base de données permet la sauvegarde des modules et des chaînes formées. Le schéma relationnel suivant représente les tables qui forment la base de données :

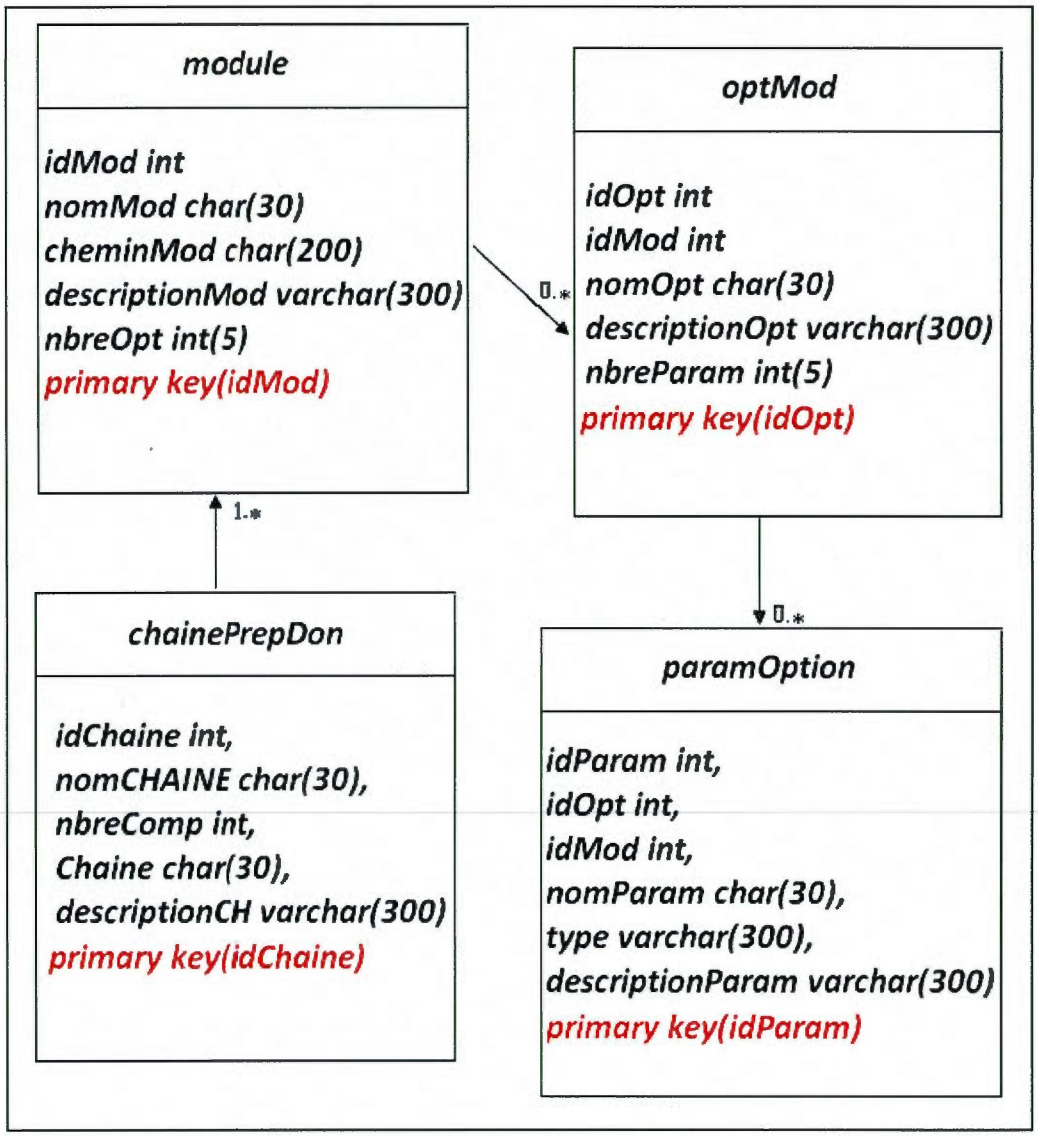


Figure 3.4 Schéma relationnel de la base de données BDplateforme de PDTPM.

3.4 Spécification fonctionnelle de PDTPM

L'outil de préparation de données PDTPM permet principalement de fournir un ensemble de données textuelles très bien structuré, en exécutant une chaîne de modules indépendants. Les principaux cas d'utilisation proposés par PDTPM sont : (1) ajouter de nouveaux modules à la plateforme, (2) former des chaînes de traitement de données textuelles

et les enregistrer, (3) exécuter une chaîne de préparation de données, ou bien, (4) exécuter un seul module de préparation de données et (5) convertir un format de fichier textuel (.txt) en format Weka (.arff). PDTPM permet entre autre de faire des mises à jour sur les chaînes formées, comme la modification, la suppression et la consultation. La suppression et la consultation peuvent aussi se faire sur les modules exécutables. Dans ce qui suit, nous présentons chaque cas à part.

3.4.1 Consultation ou suppression d'un module exécutable ou d'une chaîne

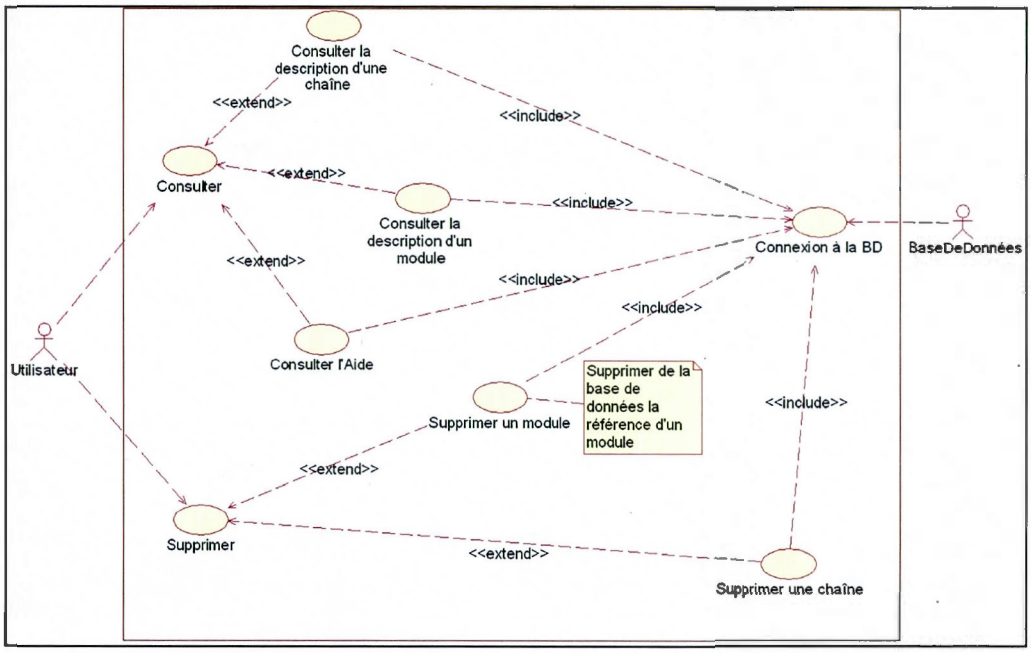


Figure 3.5 C-U «Mise à jour à un module exécutable ou à une chaîne».

Comme tout système, PDTPM permet la consultation de la description d'un module exécutable ainsi que les chaînes. Le système affiche toutes les informations concernant un module. Cela permet d'avoir un aperçu sur la fonctionnalité d'un tel module et sur la façon dont l'utilisateur applique ce module. L'affichage de la description d'un module ou d'une chaîne se fait dans l'espace d'affichage des messages, intitulé «panneau sud» dans la figure suivante.

Figure 3.6 GUI-Fenêtre principale de PDTPM.

Le système permet aussi de consulter l'aide, dont le rôle est d'expliquer aux utilisateurs comment former une chaîne juste et comment utiliser des chaînes créées au préalable. D'autre part, l'utilisateur a la possibilité de supprimer une chaîne déjà enregistrée. Il sélectionne la chaîne voulue et clique sur le bouton *«supprimer»*. Le système affiche un message demandant la confirmation de l'utilisateur. L'utilisateur confirme et le système supprime la chaîne de la base de données, puis affiche un message de suppression. Toutes ces fonctionnalités entrainent la connexion à la base de données pour la mettre à jour.

3.4.2 Ajouter un nouveau module

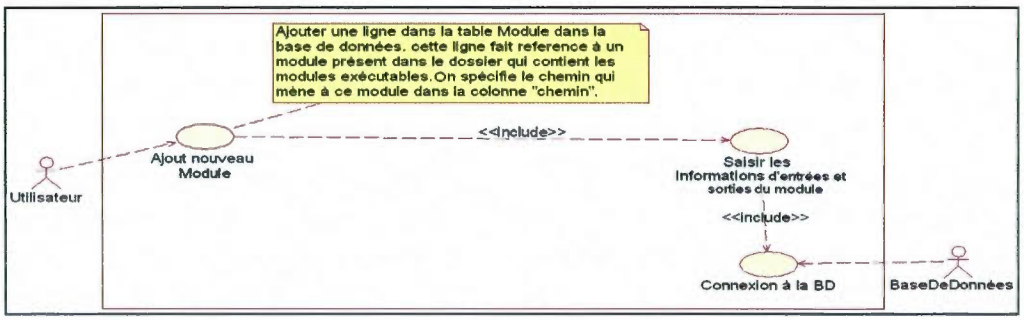


Figure 3.7 C-U «Ajouter un nouveau module».

Le système permet à l'utilisateur d'ajouter un nouveau module au système ; précisément, il permet d'ajouter une référence à un module donné. Il entre les informations nécessaires sur ce module et surtout le chemin vers ce dernier à l'aide de plusieurs fenêtres successives. Ces informations seront enregistrées dans la base de données. Ce cas d'utilisation implique une connexion à la base de données où une référence à ce module est ajoutée avec toutes les informations qui le concerne (nom, nombre d'options, nombre de paramètres, chemin,...).

3.4.3 Créer une chaîne

L'utilisateur peut créer une chaîne complète de préparation de données d'une façon très flexible, en glissant les modules désirés qui se trouvent dans la liste des modules (Panneau ouest dans la figure 3.6), dans l'espace de traitement des chaînes (Panneau central dans la figure 3.6), en supprimant ceux considérés comme inutiles, ou bien, en changeant l'ordre des modules dans la chaîne. Enfin, il peut l'enregistrer dans la base de données. Nous présentons dans la figure suivante, le cas d'utilisation qui permet de créer une chaîne de préparation de données.

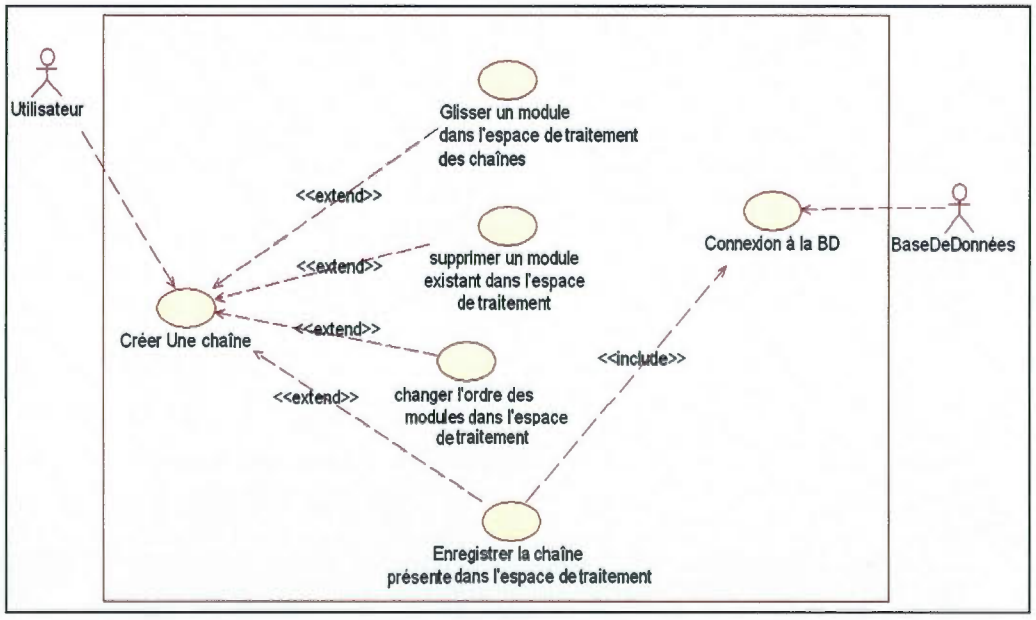


Figure 3.8 C-U «Créer une chaîne de préparation de données».

3.4.4 Exécuter une chaîne/un module

Précisons tout d'abord que le dossier Input est le dossier où se trouvent les fichiers formant le corpus de données et où se trouve aussi le fichier *StopList*. Le dossier Output contiendra le fichier *matrice.txt* qui sera le résultat final. L'utilisateur a la possibilité de choisir entre l'exécution d'une chaîne de préparation de données enregistrée dans la base de données ou bien l'exécution d'un seul module présenté dans la liste des modules (Panneau ouest dans la figure 3.6). Il a aussi la possibilité de choisir une chaîne parmi celles crées au préalable. Ensuite, le processus d'exécution est le même pour une chaîne ou un module, puisqu'un module représente une chaîne qui contient un seul module. La figure suivante représente ce cas d'utilisation.

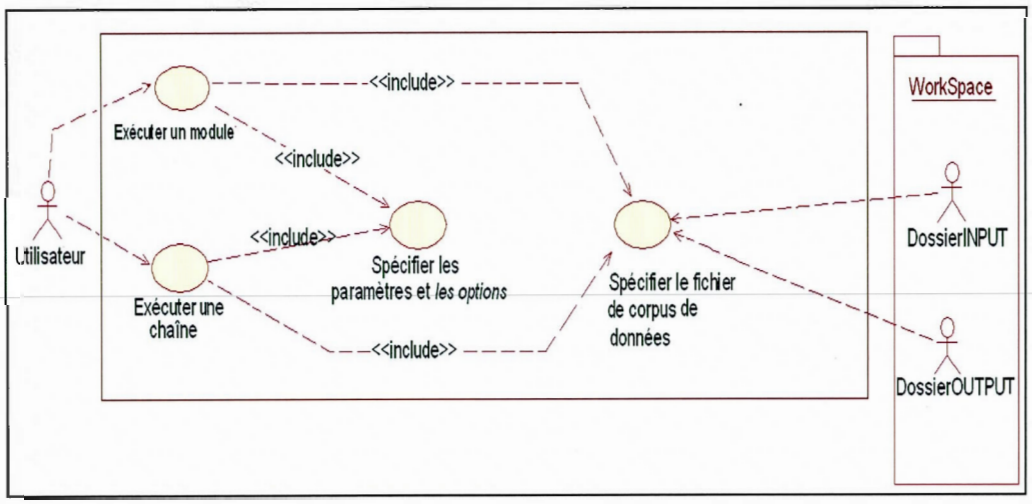


Figure 3.9 C-U «Exécuter une chaîne/un module».

L'utilisateur sélectionne la chaîne voulue. Il clique sur *exécuter*. Il doit sélectionner une option pour chaque module appartenant à la chaîne. Ensuite, il parcourt et sélectionne le corpus de données pour chaque module. À chaque exécution de chaque module, un ou plusieurs fichiers seront crées. Les fichiers donnés en entrée dans le dossier Input (*C:\RepFichier\FichierEntrees*) ne seront pas modifiés. Le résultat final, qui est une matrice documents-mots, sera aussi enregistré dans un fichier, dans le dossier Output (*C:\RepFichier\FichierSorties*).

3.4.5 Convertir un fichier textuel (*.txt*) en format Weka (*.arff*)

Nous avons ajouté cette fonctionnalité étant donné que l'utilisation de l'outil *Weka* nécessite que le fichier, contenant le corpus textuel, soit organisé selon une forme spécifique à *Weka* avec une extension *.arff*. Pour ce faire, nous avons ajouté ce cas d'utilisation qui permet de convertir un fichier textuel qui contient une matrice complète et bien organisée, en format *Weka*. La figure suivante correspond au cas d'utilisation *«Convertir un fichier textuel (.txt) en format Weka (.arff)»*.

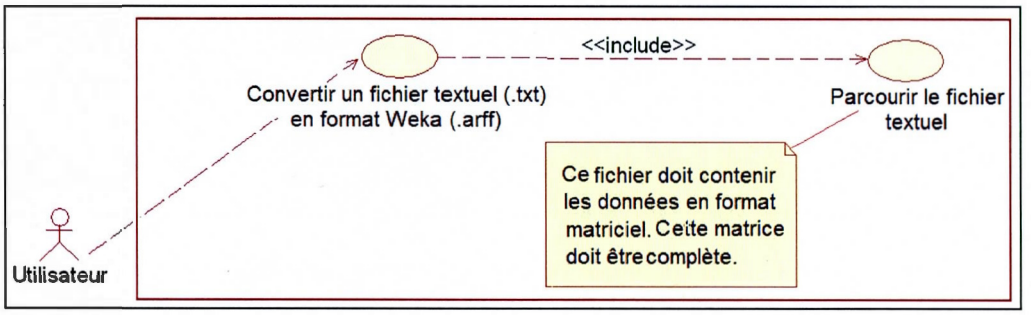


Figure 3.10 C-U «Convertir un fichier matriciel textuel (.txt) en format Weka (.arff)».

L'utilisateur doit donc parcourir le fichier textuel qui contient la matrice de données organisée et complète. Cette matrice doit contenir les attributs en colonnes (les mots). Le résultat final sera enregistré dans un fichier d'extension *«.arff»* dans le dossier output (*C:\RepFichier\FichierSorties*).

3.5 Principaux scénarios des cas d'utilisation

Pour implémenter notre système de préparation de données, nous utilisons le langage de programmation Java et en particulier Swing qui est un ensemble de packages qui reprend les principes développés dans les classes du package java.awt, mais en étendant leur comportement. De nouvelles classes ont aussi été ajoutées. Dans notre application, nous utilisons le patron de conception (design pattern) Modèle-Vue qui est une variante du patron de conception Modèle-Vue-Contrôleur. Ce patron de conception divise l'IHM en un modèle

(modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation).

Le système de gestion de base de données que nous utilisons est MySQL 5.1. Notre modèle de données contiendra les méthodes nécessaires pour accéder à la base de données afin de sélectionner, ajouter, modifier ou supprimer les informations. Parmi les méthodes appartenant au modèle de données nous citons :

afficherMatrice(), ajouterOption(), supprimerOption(), ajouterParametre(), supprimerParametre(), ajouterModule(), supprimerModule(), consulterModule(), importerChaîne(), consulterChaîne(), executerchaîne(),...

Ces méthodes, appartenant au modèle de données, gèrent les données et les passent soit aux vues ou bien à la base de données, selon le besoin. Tout au long de la description des principaux scénarios du système, nous prenons comme exemple le module exécutable *«matriceSansMotsFonctionnels»*. Ce module permet de nettoyer la matrice (générée par tokenisation, n-gram,...) afin de réduire sa taille et de supprimer les unités qu'ils jugent non nécessaires. Il permet la suppression des mots fonctionnels selon plusieurs choix ; StopList, fréquences ou manuelle. Il retourne la matrice sans les mots supprimés. Ce module contient précisément 5 options :

- 0: suppression par *«StopList»*. Cette option ne contient pas de paramètres.
- 1: suppression par une valeur seuil minimale. Cette option contient un seul paramètre, *SeuilMin*.
- 2: suppression par une valeur seuil maximal. Cette option contient un seul paramètre, *SeuilMax*.
- 3: suppression par intervalle (conserve les éléments qui sont inclus inclusivement dans l'intervalle). Cette option contient deux paramètres, *SeuilMin* et *SeuilMax*.
- 4: suppression manuelle. Cette option contient un seul paramètre listMots. Les mots doivent êtres séparés par un espace.

Nous étalons tous les scénarios possibles des principaux cas d'utilisation du système en s'appuyant sur ce module comme exemple d'exécution.

3.5.1 Ajouter module

Pour ajouter un module, l'utilisateur clique sur le bouton *«ajouter»* dans la barre de menu. Ensuite, il clique sur *«Ajouter nouveau module»*. Une nouvelle fenêtre *«ajout d'un nouveau module»* s'ouvre. L'utilisateur doit entrer le chemin du module en parcourant le fichier .class souhaité. Il saisie le nom et la description du module ainsi que le nombre d'options. Enfin, il clique sur *«Continuer»*. La figure suivante représente un exemple d'ajout du module exécutable *«matriceSansMotsFonctionnels»*.

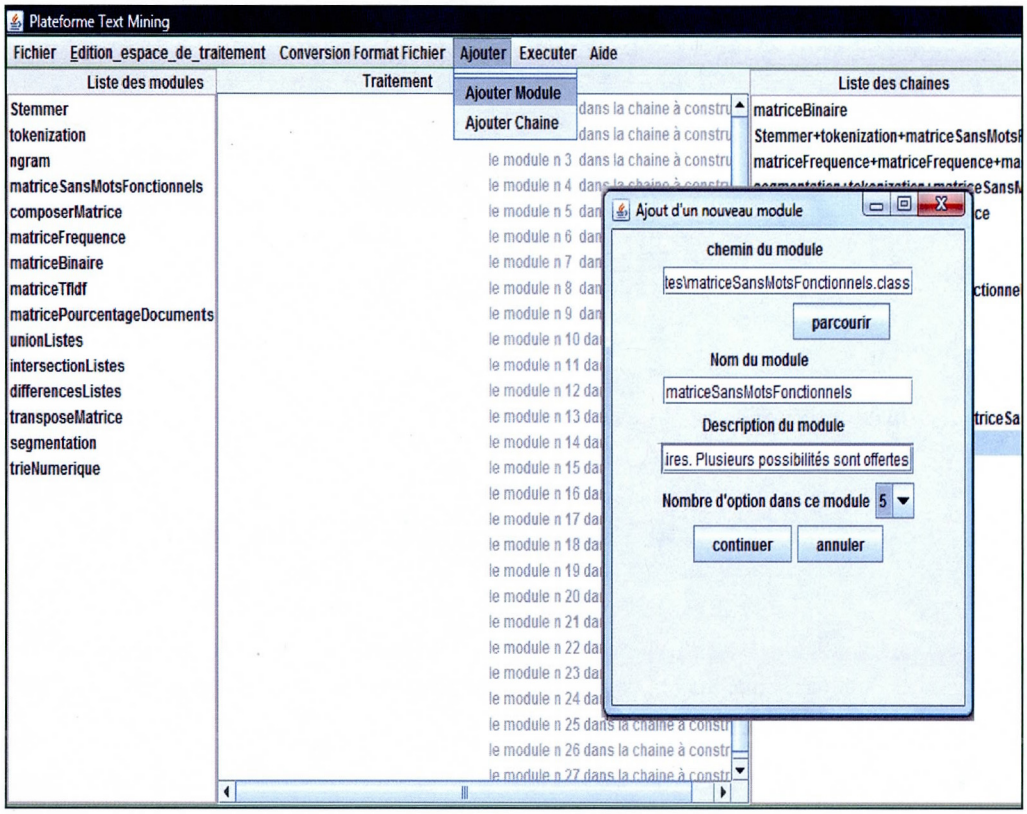


Figure 3.11 GUI-Ajouter un module exécutable.

Une autre fenêtre s'ouvre et contient autant d'onglets que le nombre d'options entré par l'utilisateur. Par exemple, si l'utilisateur a entré cinq options, nous aurons une fenêtre *«Ajout des divers options pour le module en cours»* qui contient cinq onglets pour les cinq

options. L'utilisateur doit entrer le nom, la description et le nombre de paramètres pour chaque option et cliquer sur *«Continuer»*.

La figure suivante montre la fenêtre «Ajout des divers options pour le module en cours» pour le module «matriceSansMotsFonctionnels».

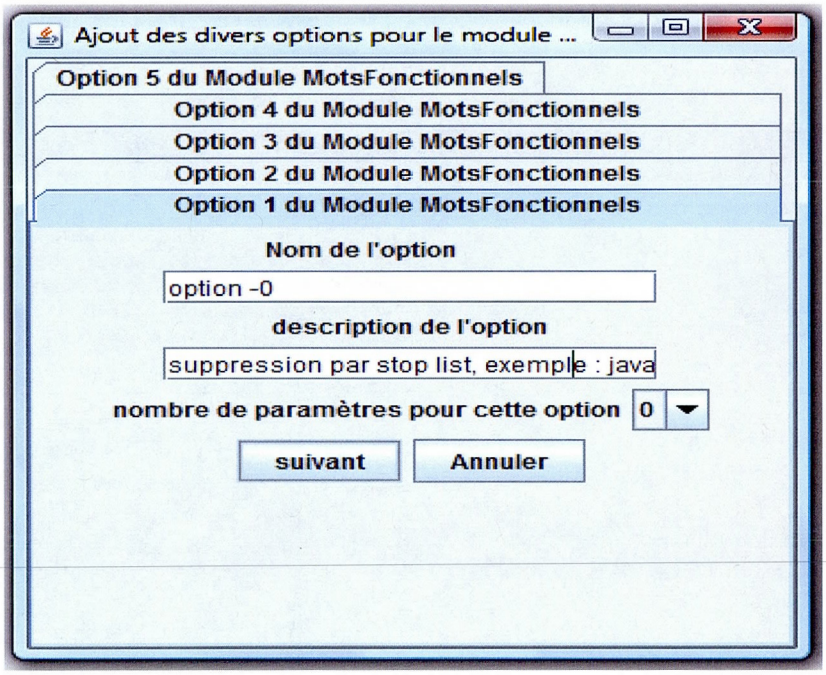


Figure 3.12 GUI-Ajouter les options d'un module exécutable.

Après avoir cliqué sur *«Continuer»*, une nouvelle et dernière fenêtre s'ouvre. C'est la fenêtre *«Ajout des divers paramètres pour chaque option»*. Le nombre d'onglets dans cette fenêtre est aussi dynamique : il correspond à la somme des paramètres de toutes les options. L'utilisateur doit donc saisir le nom, le type et la description de chaque paramètre. En cliquant sur le bouton *«Continuer»* et si les critères de saisie sont respectés, le module s'ajoute dans la base de données, il s'affiche dans la liste des modules (panneau ouest dans la figure 3.6) et la fenêtre *«Ajout des divers paramètres pour chaque option»* se ferme. En prenant le même exemple, la figure suivante est une capture d'écran de la fenêtre *«Ajout des divers paramètres pour chaque option»*.

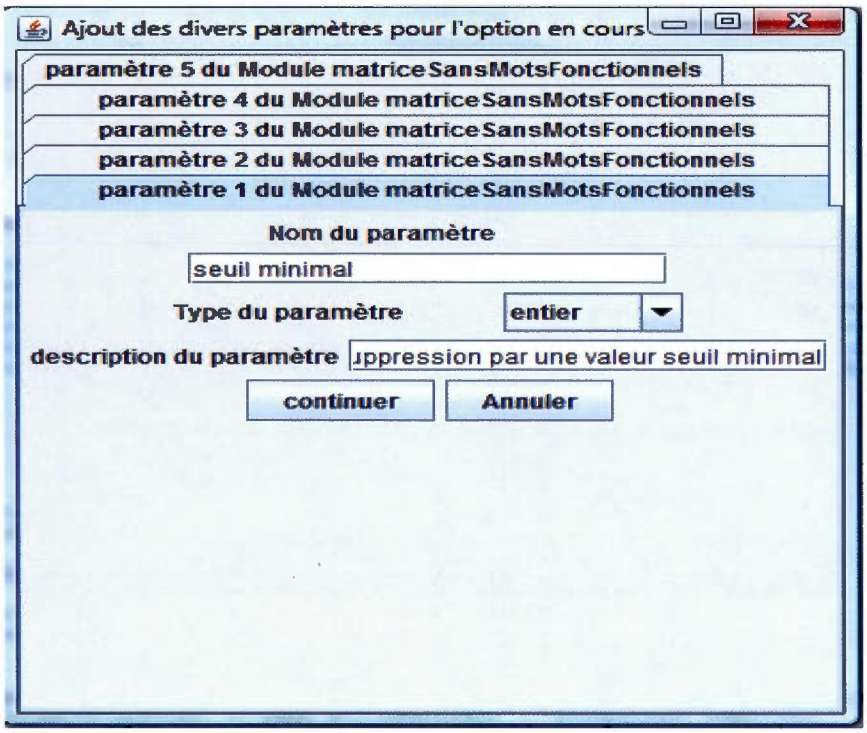


Figure 3.13 GUI-Ajouter les paramètres des options saisies pour un module.

3.5.2 Créer une chaîne de préparation de données

Pour créer une chaîne de préparation de données, on doit avoir plusieurs modules déjà enregistrés. En effet, nous avons programmé cette fonctionnalité en nous basant sur la technique Glisser-Déposer (Drag and Drop). L'utilisateur clique sur le module qu'il veut utiliser et le déplace de la liste de modules (panneau ouest Figure 3.6) vers l'espace de traitement des chaînes, en utilisant la technique de Glisser-Déposer. L'utilisateur peut aussi supprimer un module déjà ajouté dans cet espace. Enfin, en formant la chaîne voulue, l'utilisateur clique sur le bouton *«Ajouter»* de la barre de menu, puis sur *«Ajouter chaîne»*. En effectuant la sauvegarde, la chaîne de préparation de données s'affiche dans la liste des chaînes (panneau Est figure 3.6) et sera enregistrée dans la base de données. La figure suivante illustre la fenêtre principale qui contient l'espace de traitement des chaînes.

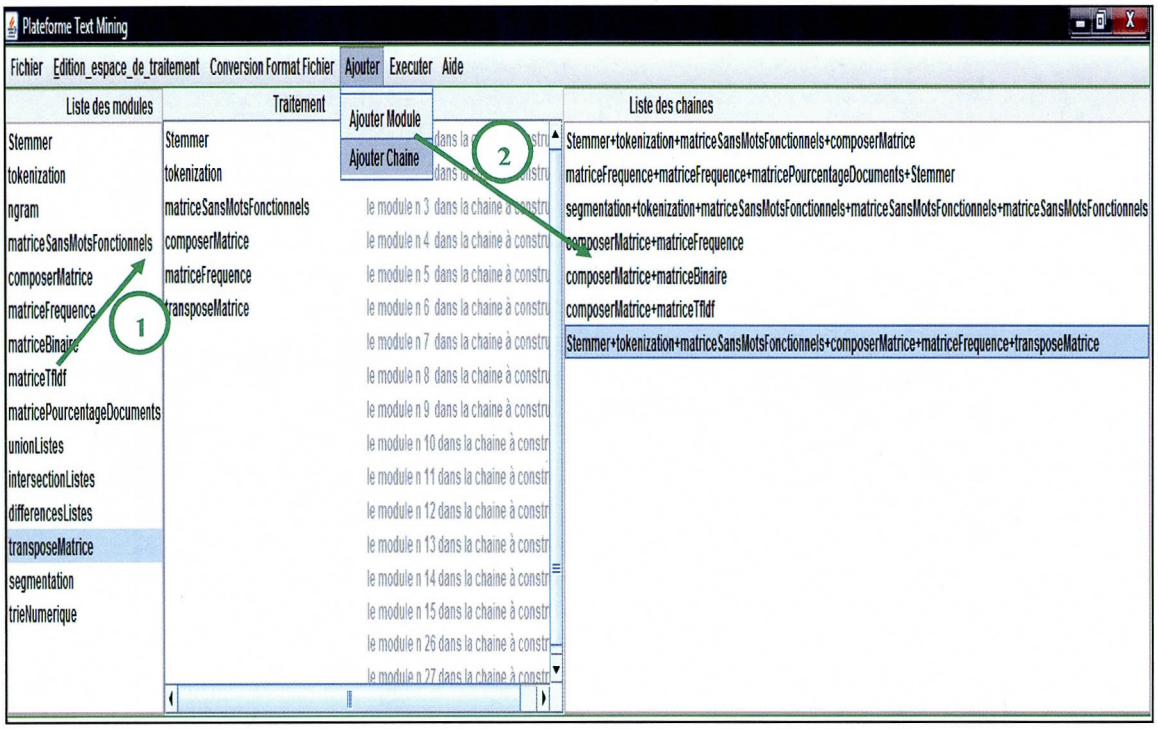


Figure 3.14 GUI-Créer une chaîne de préparation de données.

3.5.3 Exécuter une chaîne de préparation de données

Comme nous l'avons déjà évoqué, on peut exécuter une des chaînes présente dans la liste des chaînes ou bien une chaîne préparée au préalable et offerte par le système. Pour le premier cas, l'utilisateur clique droit sur la chaîne dans la liste des chaînes (panneau Est figure 3.6). Un menu s'ouvre et l'utilisateur choisi l'option *«exécuter»*. Pour le deuxième cas, l'utilisateur clique sur le bouton affichage dans la barre de menu et sélectionne la chaîne souhaitée. Ensuite, l'exécution est la même dans les deux cas : une nouvelle fenêtre s'ouvre qui contient autant d'onglets que le nombre de modules dans la chaîne. Dans chaque onglet, l'utilisateur doit choisir l'option qu'il veut utiliser ainsi que les paramètres s'il y en a. En appuyant sur *«Suivant»*, on passe d'un onglet à un autre jusqu'au dernier onglet, comme le montre la figure suivante.

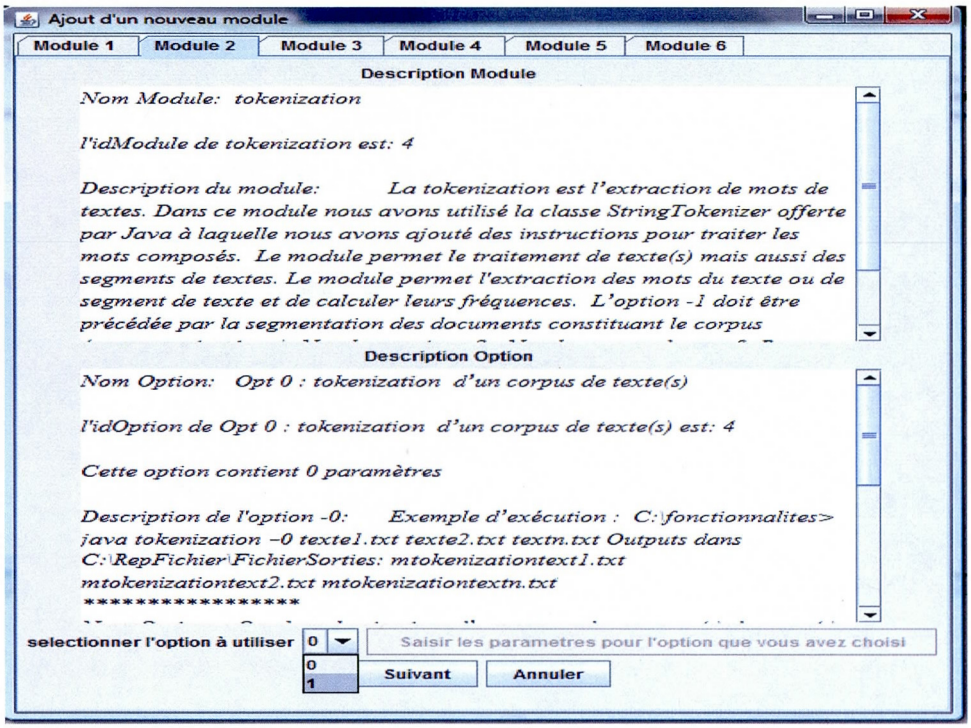


Figure 3.15 GUI-Exécuter une chaîne de préparation de données sur un corpus de texte.

À cette étape, nous avons configuré notre chaîne en choisissant l'option et le paramètre que nous voulons utiliser pour chaque module. Ensuite, une nouvelle fenêtre intitulée *«Sélection des fichiers qui forment le corpus du texte»* s'affiche. À chaque onglet, on doit parcourir les fichiers textes à utiliser. Ensuite, on ajoute le fichier au corpus du texte en cliquant sur *«Ajouter le fichier parcouru au corpus de données»*. En appuyant sur *«Exécuter»*, le système exécute le module sur le corpus choisi par l'utilisateur et affiche le résultat dans l'espace d'affichage des messages (panneau sud dans la figure 3.6). L'utilisateur doit répéter ce processus pour chaque module. Nous avons choisi cette façon de faire parce que l'exécution diffère d'un module à un autre, et pour respecter la flexibilité qu'a notre système d'introduire n'importe quel nouveau module. Par exemple, il y a des modules qui nécessitent la sélection d'un corpus de données alors qu'il en y a d'autres qui utilisent des fichiers déjà créés par d'autres modules. Dans ce cas, l'utilisateur ne doit pas entrer un corpus de données, au contraire, il doit cliquer directement sur le bouton *«Exécuter»*. En effet, le

système affiche les propriétés et les spécifications d'exécution de chaque module dans la chaîne en cours d'exécution dans l'espace de description des modules. La figure suivante montre une capture d'écran dans la fenêtre *«Description du module qui correspond à ce corposDuTexte»*. Ainsi, en s'appuyant sur la description du module, il pourrait savoir si on doit ajouter des fichiers textuels ou non.

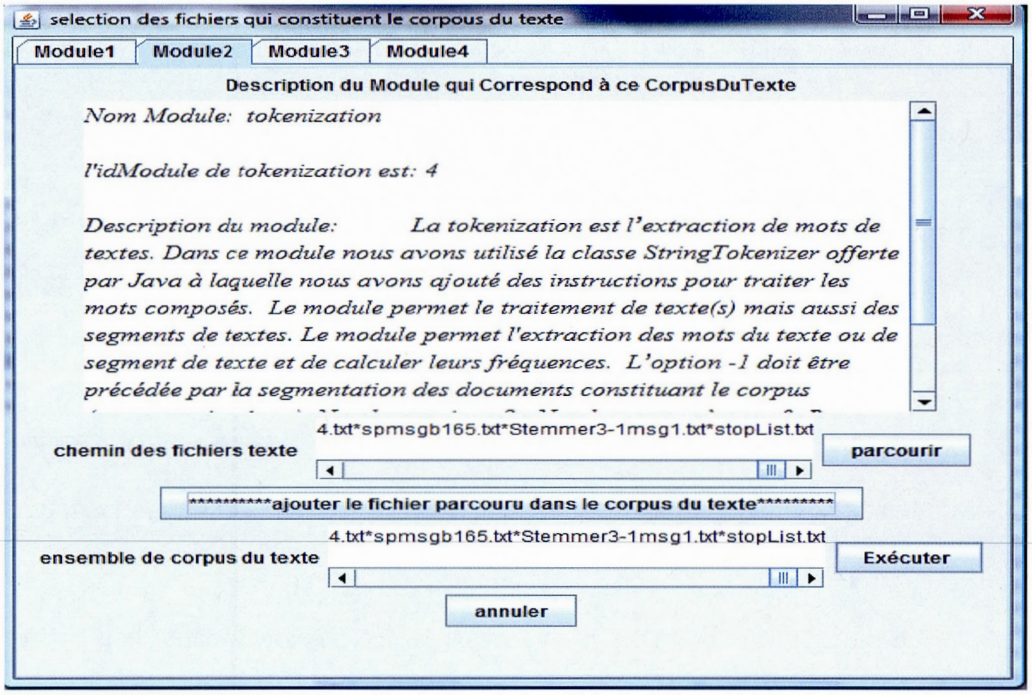


Figure 3.16 GUI-Sélection du corpus de texte qui sera exécuté par la chaîne de préparation de données.

3.5.4 Convertir une matrice textuelle *(.txt)* en format Weka *(.arff)*

Le programme d'apprentissage automatique *WEKA* qui sera utilisé pour mener des expérimentations, a besoin d'une forme spécifique de données. Nous avons donc besoin de transformer une matrice textuelle en format *Weka*, d'extension *.arff*. L'utilisateur clique sur le bouton *«Conversion format fichier»*, ensuite, il choisit le sous menu *«Convert txtToArff»* comme le montre la figure suivante :

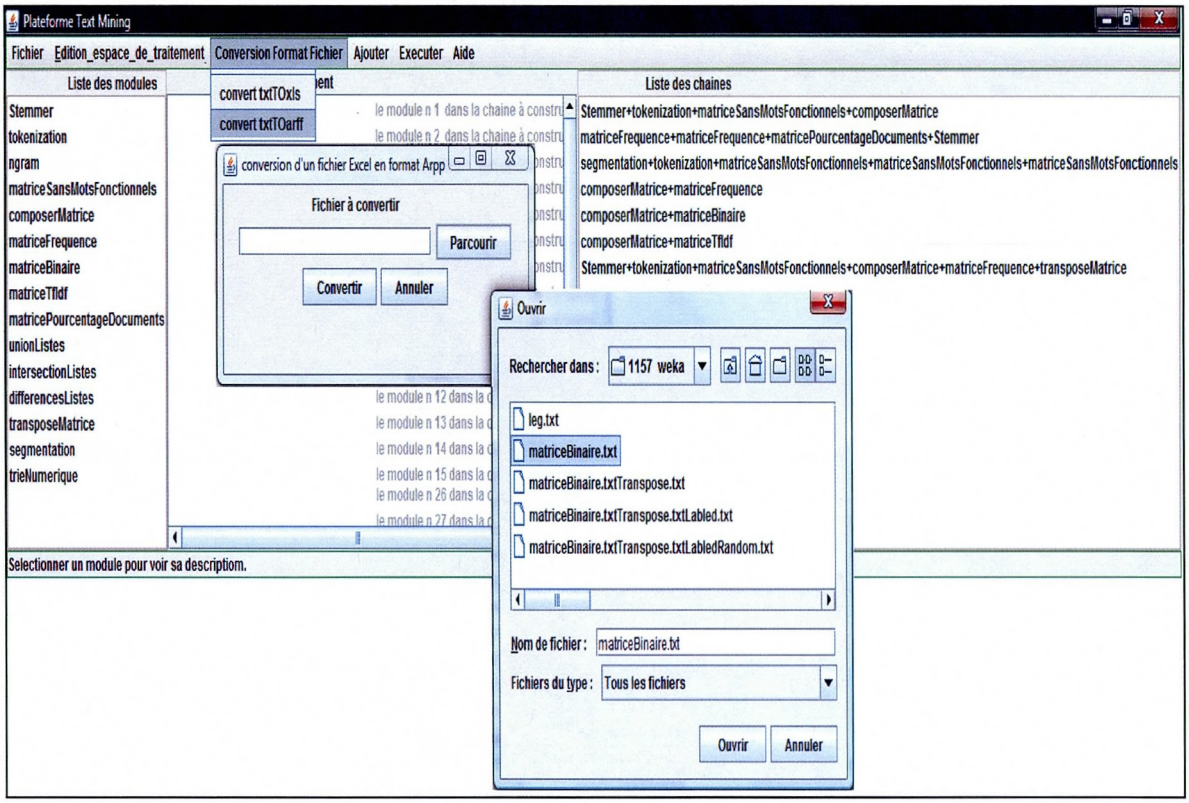


Figure 3.17 GUI-Convertion une matrice textuelle *(.txt)* en format *Weka(.arff)*.

Ensuite, l'utilisateur doit sélectionner le fichier textuel qui contient le corpus de données et il le convertit en cliquant sur le bouton *«Convertir»*. Il obtient un fichier d'extension *.arff*, de même nom et dans le même répertoire que le fichier source.

3.6 Conclusion

Nous avons proposé un système qui permet de nettoyer et de structurer des données textuelles brutes en utilisant la programmation modulaire. Nous utilisons des composantes exécutables pour former des chaînes. En exécutant ces chaînes, elles permettent de transformer un corpus de données brutes en contexte formel : matrice documents-mots. Nous détaillons dans le chapitre 4 la description des composantes exécutables déjà introduites dans la plateforme de préparation de données textuelles. En se basant sur une méthode modulaire, notre système peut évoluer facilement : nous pouvons ainsi ajouter des composantes

exécutables qui concernent les techniques d'apprentissage automatique comme *Cluster*, *K-means*,... Ainsi notre système permet non seulement la préparation des données textuelles brutes mais aussi l'apprentissage automatique.

CHAPITRE IV

APPLICATION SUR UNE BASE DE DONNÉES PUBLIQUE : EXPÉRIMENTATION DES MÉTHODES DE CATÉGORISATION DU TEXTE SUR L'ENSEMBLE DE DONNÉES TEXTUELLES LINGSPAM

4.1 Introduction

Après avoir exposé la plateforme de préparation de données, nous présentons dans ce chapitre les expérimentations faites en exploitant PDTPM. Dans une première partie, nous exposons la base de données textuelles sur la quelle nous expérimentons plusieurs méthodes d'apprentissage automatique. Dans la deuxième partie, à l'aide de notre plateforme de préparation de données PDTPM, nous prétraitons les données textuelles en suivant trois chaînes de préparation de données. Chaque chaîne sera discutée en fonction des composantes de préparation des données utilisées. Ensuite, nous exécutons diverses méthodes d'apprentissage automatique sur les trois matrices obtenues à partir de l'application des trois chaînes de préparation de données, sur le corpus de données LingSpam. Enfin, dans la dernière partie, nous discutons les résultats trouvés en les comparant aux résultats d'autres travaux. Nous commentons les résultats obtenus en nous basant sur deux axes : un axe qualitatif et un axe quantitatif.

4.2 Base de données publique : LingSpam

Nous avons choisi la base de données textuelles LingSpam, formée d'un ensemble de courriels. Cette base de données publique a été citée et utilisée dans plusieurs travaux de recherche [17], [20], [24] et [34]. LingSpam comprend 2893 messages, 2412 messages linguistiques légitimes et 481 messages spams. Les messages indésirables représentent 16,6 % de tout le corpus [17]. Les messages indésirables ont été rassemblés à partir de plusieurs sources (collections publiques, des participants et des pièges spams (SpamTraps)). Les

messages légitimes sont aussi collectés par plusieurs sources (des volontaires, des courriels résultant de l'abonnement à une liste de diffusion,...).

D'après A. Kolcz et J. Alspector [20], l'avantage dans l'utilisation de LingSpam est qu'il contient assez de données par rapport à d'autres corpus de texte dans le domaine de filtrage des courriels au moyen de méthodes d'apprentissage automatique. Il contient 2412 messages linguistiques légitimes, ce qui est difficile à rassembler en raison de la protection de la vie personnelle [34]. De plus, LingSpam est largement utilisé dans les travaux de recherche et permet donc la comparaison entre plusieurs travaux et plusieurs approches. Cependant, LingSpam est une base de données publique statique et ne suit pas l'évolution du contenu de spams qui change rapidement [24]. LingSpam contient quatre catégories dans quatre dossiers différents :

i. *Bare*: lemmatisation désactivée, *StopList* désactivé.
ii. *Lemm*: lemmatisation activée, *StopList* désactivé.
iii. *Stop*: lemmatisation désactivée, *StopList* activé.
iv. *Lemm-Stop* : lemmatisation activée, *StopList* activé.

Nous considérons la première catégorie afin d'appliquer nos propres modules exécutables de lemmatisation et de *StopList*. Nous sélectionnons 1157 fichiers dont 192 messages spams et 965 messages légitimes. Nous avons réduit le contenu de la base de données LingSpam à cause de la contrainte d'espace mémoire ; le serveur de notre laboratoire de recherche a une capacité de mémoire système qui ne permet d'utiliser que la moitié de la base de données LingSpam. Toutefois, 1157 courriels nous ont donné suffisamment d'attributs. En fait, après la tokenisation et la formation de la matrice intermédiaire documents-mots, et avant la sélection des attributs les plus pertinents, nous avons obtenu 9468 attributs.

4.3 Préparation de données

Nous avons utilisé exclusivement notre plateforme de préparation de données PDTPM pour effectuer l'étape de préparation de données. PDTPM offre non seulement une souplesse et un choix flexible dans la création des chaînes de préparation de données mais, elle permet

aussi de former des matrices spécifiques à chaque outil d'apprentissage automatique comme Weka ou Matlab, que nous utiliserons dans une étape ultérieure. Les trois chaînes de préparation de données crées contiennent des composantes exécutables spécifiques et d'autres composantes communes. Chaque composante ou module correspond à une étape de préparation de données. Nous présentons dans ce qui suit, les trois chaînes de préparation de données que nous avons formées et qui seront utilisées pour construire les matrices de données, exploitables par des méthodes d'apprentissage automatique.

4.3.1 Chaîne 1 : Chaîne pour former la matrice binaire

Cette chaîne de préparation de données textuelles est formée de 10 composantes exécutables. Chaque composante correspond à une étape de processus de préparation de données. L'application de chaque module entraine la création d'au moins un nouveau fichier textuel. Après l'application de module TransposerMatrice, nous obtenons le fichier texte contenant la matrice Documents-Mots. Le contenu de cette matrice est binaire ; la valeur 0 indique que le mot n'existe pas dans le document alors que la valeur 1 indique la présence du mot dans le document, et c'est pour cela que nous l'appelons une matrice binaire. Ce fichier, contenant la matrice binaire, sera ensuite manipulé de deux façons différentes afin d'avoir à la fin deux fichiers de texte : un fichier qui est spécifique à l'outil Weka et l'autre fichier spécifique à l'outil Matlab. La figure suivante montre cela.

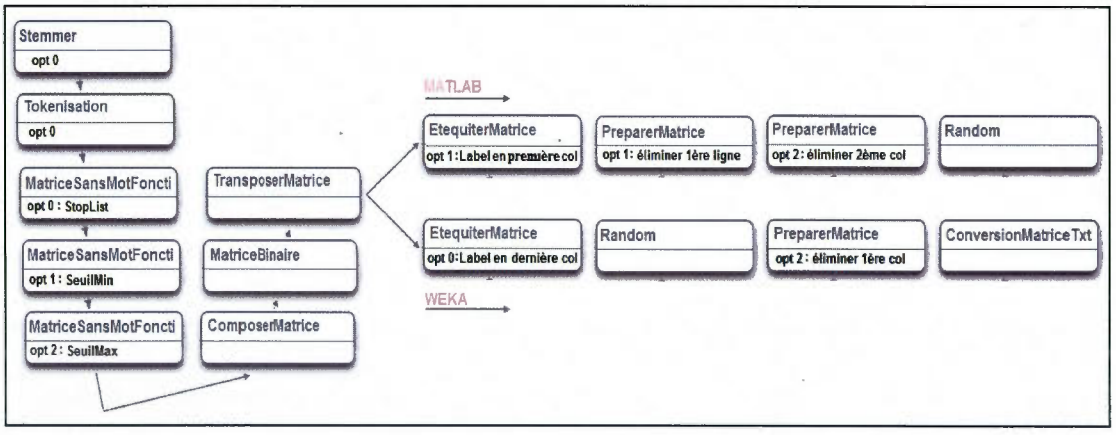


Figure 4.1 Chaîne de préparation de données textuelles pour former des matrices binaires par l'intermédiaire de l'outil PDTPM.

a) Stemmer

La racinisation (stemming) consiste à supprimer les suffixes et les préfixes des mots afin de ne conserver que la racine. L'algorithme suivi est celui présenté par M. Porter [35]. Nous avons réalisé des modifications sur le code source (direction des inputs et des outputs) pour l'adapter aux traitements que nous désirons réaliser. L'exécution de cette composante donne, au moins, en output un fichier texte qui prend comme nom *Stemmer[NomFichInput].txt*, localisé dans le dossier input crée par PDTPM : *C:\RepFichier\FichierEntrees*. Dans notre cas, nous donnons en input 1157 courriels et nous obtenons en outputs 1157 fichiers.

b) Tokenisation

La tokenization est l'extraction de mots du texte. Dans ce module, nous avons utilisé la classe *StringTokenizer* offerte par Java, à laquelle nous avons ajouté des instructions pour traiter les mots composés (exemple, c'est-à-dire). Le module permet le traitement de textes mais aussi de segments de textes. Il contient deux options : l'option 0 qui permet la tokenisation d'un corpus de texte, et l'option 1 qui permet la tokenisation d'un corpus de segment de texte. La deuxième option nécessite l'utilisation du module *segmentation.exe* avant l'utilisation de la composante *Tokenisation.exe*. Nous utilisons dans cette chaîne binaire l'option 0 qui permet la tokenisation d'un corpus de texte formé de 1157 courriels. L'exécution de cette composante produit au moins, en output un fichier texte qui prend comme nom *mTokenization[NomFichInput].txt*, localisé dans le dossier output crée par PDTPM : *C:\RepFichier\FichierSorties*. Étant donné que nous avons appliqué la composante Stemming avant, le nom de fichier sera dans notre cas *mTokenizationStemmer[NomFichInput].txt*. La figure suivante représente une capture d'écran d'une partie d'un fichier qui a subit une tokenisation.

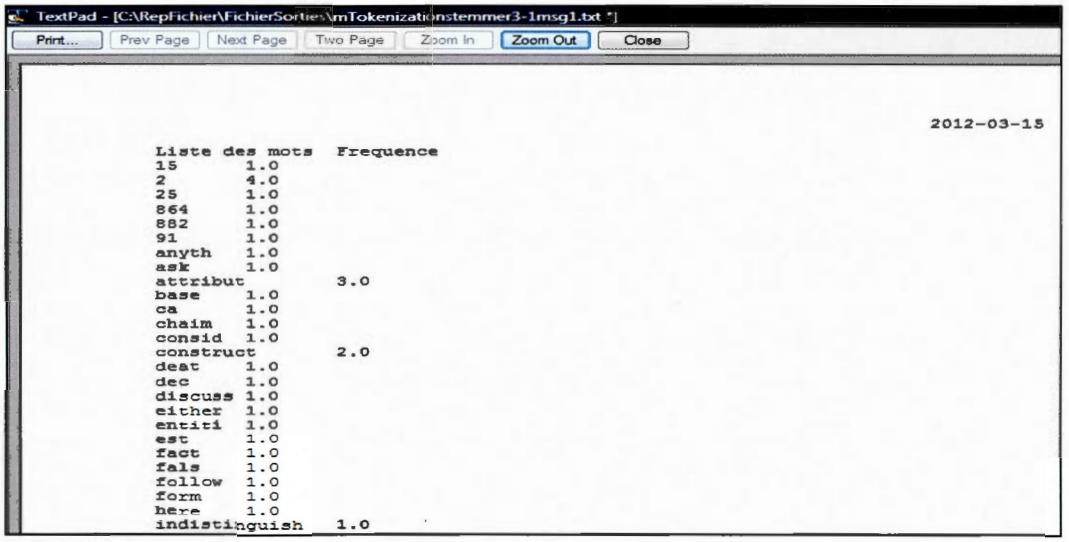


Figure 4.2 Organisation des données dans un fichier textuel qui a subit une tokenisation par le biais du module Tokenization.exe intégré dans PDTPM.

c) Matrice Sans Mots Fonctionnels

Après avoir effectué la tokenisation, les utilisateurs ont souvent besoin de nettoyer la matrice générée afin de réduire sa taille et de supprimer les unités qu'ils jugent non nécessaires. Plusieurs possibilités sont offertes, et la première possibilité (extraction par stop liste) ne concerne que la tokenisation. Cependant les trois autres peuvent êtres utilisées aussi pour nettoyer la matrice générée suite à l'extraction des N-grams. La première option consiste en la suppression des mots fonctionnels par Stop Liste. Pour que ce module avec cette option (option 0) soit fonctionnel, il est nécessaire de placer au niveau du répertoire *"C:\RepFichier\FichierEntrees"* un fichier *StopList* est de le nommer *"stopList.txt"*. La deuxième option (option 1) consiste en la suppression par fréquence minimale. On conserve uniquement les unités ayant une fréquence supérieure ou égale à un nombre donné en paramètre (param1, SeuilMin). La troisième option consiste en la suppression par fréquence maximale (option 2). Dans ce cas, on conserve uniquement les unités ayant une fréquence inférieure ou égale à un autre nombre entré en paramètre (param1, SeuilMax). Une quatrième option est offerte par cette composante : c'est la suppression par un intervalle de fréquences (option 3), où on conserve uniquement les unités ayant une fréquence située inclusivement

entre deux valeurs entrées en paramètres (param1, SeuilMin et param2, SeuilMax). La dernière option consiste en une surpression manuelle (option 4) : dans ce cas, on supprime les mots donnés en paramètre (param1, ChaineDeMotsÀSupprimer) dont les mots sont séparés par un espace. Nous avons utilisé dans cette chaîne les trois options suivantes : la suppression des mots fonctionnels par *StopList*, la suppression par fréquence minimale et la suppression par fréquence maximale. En appliquant cette composante, nous n'obtenons pas de nouveaux fichiers. Les fichiers obtenus à l'aide de la composante tokenisation seront modifiés en éliminant les noms fonctionnels.

d) Composer Matrice

Ce module permet de produire une matrice intermédiaire nécessaire à l'élaboration des matrices mots*documents ou N-Gram*documents. Le passage par ce module est obligatoire pour la suite des traitements d'un corpus composé de documents. Nous avons décidé de passer par une matrice intermédiaire afin d'offrir plus de flexibilité à l'utilisateur tout en réduisant le temps de traitement. Pour le traitement de segments de textes, il n'est pas nécessaire de passer par ce module vu que les modules *tokenization.exe* et *Ngram.exe* génèrent directement la matrice intermédiaire en output. Cette composante permet donc de rassembler tous les fichiers formés par les composantes précédentes en un seul fichier appelé *matriceIntermédiaire.txt* et rangé dans le dossier output : *C:\RepFichier\FichierSorties*. La figure suivante permet de visionner une partie d'un fichier *matriceIntermidiaire.txt* formé par cette chaîne binaire.

	A	B	C	D	E	F	G
1	liste_mot	mtokenization3-1msg1.txt	mtokenization3-1msg2.txt	mtokenization3-1msg3.txt	mtokenization3-375msg1.txt	SommeFrequence	PourcentageDocument
2	institutional	0	0	0	0	0	0
3	dinar	4.0	0	2.0	0	6	1.5
4	event	0	0	0	0	0	0
5	$	0	1.0	0	0	1.0	0.25
6	european	1.0	0	0	0	1	0,25
7	quality	0	0	7.0	2.0	9.0	2.25
8	business	0	0	0	4.0	4.0	1
9	royal	0	0	0	0	0	0
10	telephone	0	3.0	0	0	3.0	0.75

Figure 4.3 Organisation des données sous la forme d'une matrice textuelle après l'exécution du module *ComposerMatrice.exe* intégré dans PDTPM.

e) Matrice Binaire

Après avoir généré, la matrice intermédiaire, il est possible de générer la matrice *mots*documents*, *Ngrams*documents*, *mots*Segments* ou *Ngrams*Segments*. La représentation de cette matrice peut prendre plusieurs formes, comme *matriceBinaire*, *matrice-Frequence* ou *matriceTfIdf*. Nous avons choisi dans notre chaîne la composante *matrice-Binaire.exe*. Ce module permet de générer la matrice en tenant compte de la présence ou de l'absence de l'unité dans le document. Le résultat de l'application de cette composante est un fichier textuel nommé *matriceBinaire.txt* dans le dossier output *FichierSorties*.

f) Transposer Matrice

Ce module permet la transposition des matrices de telle sorte que les lignes deviennent des colonnes et les colonnes deviennent les lignes. Par exemple, la matrice *mots-documents* devient *documents-mots*. Nous appliquons cette composante afin de respecter la forme exigée par l'outil Weka et Matlab qui considèrent des vecteurs de documents. Le fichier obtenu est nommé *[NomFichierInitial]Transposer.txt*, situé dans le dossier output *C:\RepFichier\FichierSorties*.

g) Etiqueter Matrice

Ce programme permet d'étiqueter automatiquement une matrice non étiquetée. *EtiqueterMatrice.exe* contient deux options : 1) L'option 0 qui permet d'ajouter la colonne de label à la fin de la matrice, c'est à dire à la dernière colonne. Cette forme de matrice est spécifique à l'outil Weka. 2) L'option 1 qui permet d'ajouter la colonne de label au début de la matrice, c'est à dire à la première colonne. Cette forme est spécifique à l'outil Matlab. En effet, cette composante exécutable n'est faisable que si on a une régularité dans les noms des fichiers (les documents). Il faut que les noms des documents d'une classe donnée aient une particularité qui la différencie des autres classes. Prenons l'exemple de notre cas binaire ou il ya deux classes, les fichiers de la classe spam commencent par «spm». Les noms de la classe message légitime ont une régularité qui consiste à avoir la sous chaîne «msg». En exécutant cette composante, le programme demande d'entrer le nombre de classes. Ensuite, une nouvelle fenêtre s'affiche permettant de saisir la valeur de chaque classe et la sous-chaîne

correspondante qui permet de détecter les documents appartenant à cette classe. À chaque utilisation de cette composante, nous obtiendrons un fichier dans le dossier output nommé *[NomFichierIniatial]Labled.txt.* Nous appliquons dans notre cas cette composante avec les deux options pour avoir deux fichiers : un fichier au format Weka et un autre fichier au format Matlab.

h) Preparer Matrice

Le module *PreparerMatrice.exe* a plusieurs fonctionnalités. L'objectif de cette composante est de présenter les matrices finales *mots-documents* sous une forme qu'un logiciel de catégorisation ou de classification peut traiter. Cette fonctionnalité présente plusieurs options. L'option 0 permet de supprimer la première ligne et la première colonne. L'option 1 supprime la ligne dont le numéro est entré en paramètre (param1, *numLigne*). L'option 2 supprime la colonne dont le numéro est entré en paramètre (param1, *numColon*). L'option 3 permet de modifier le séparateur entre les valeurs de la matrice (param1, tab). L'option 4 divise une matrice en deux sous-matrices selon le pourcentage saisi par l'utilisateur (param1, *TauxDiv*). L'option 5 identifie dans l'ordre suivant, les méta-données d'une matrice numérique : dimensions de la matrice (nombre de lignes, nombre de colonnes), *MAX*, *MIN*, *SUM* et *AVG* de chaque colonne. Comme cela est illustré dans la figure 4.1, la figure 4.4 et la figure 4.5, nous avons utilisé la composante *PreparerMatrice.exe* dans plusieurs scénarios pour préparer le fichier sous forme Weka et Matlab. Pour former le fichier contenant la matrice en format sous Weka, nous avons utilisé l'option 2 pour éliminer la première colonne qui contient les noms des documents. Pour former le fichier contenant la matrice en format Matlab, nous avons utilisé l'option 1 pour éliminer la première ligne qui contient les attributs et nous avons utilisé l'option 2 pour éliminer la deuxième colonne qui contient les noms des documents, étant donné que la première colonne contient le label de chaque document. À chaque utilisation de la composante *PreparerMatrice.exe*, nous obtenons dans le dossier output *FichierSorties*, un fichier nommé *MP[NomFichier-Iniatial].txt.*

i) Random

Le module *Random.exe* permet d'entrecroiser les enregistrements pour que nous obtenions une matrice homogène qui contient des enregistrements de toutes les classes bien repartis sur toute la matrice. Il suffit de donner le nombre d'enregistrements de chaque classe et le programme procède à la répartition automatique et au hasard des lignes de la matrice. À chaque utilisation de la composante *Random.exe*, nous obtenons un fichier nommé *[Nom-FichierIniatial]Random.txt,* dans *FichierSorties*.

j) Conversion d'une matrice en extension *.txt* en format Weka (*.arff*)

Le module *ConversioMatriceTxtEnFormatWeka.exe* permet de convertir une matrice structurée dans un fichier texte en format *arff* pour être exploitée par le programme Weka. Ce programme ne peut pas être utilisé que si la matrice dans le fichier source respecte les conditions suivantes:

- (1) la matrice doit contenir les noms des attributs (mots) dans la première ligne;
- (2) la matrice doit contenir la colonne label qui désigne à quelle classe appartient l'enregistrement. Cette colonne doit être la dernière colonne dans la matrice;
- (3) enfin, la matrice ne doit pas contenir les noms des fichiers (documents) dans la première colonne.

Nous avons respecté la condition (2) en utilisant la composante exécutable *EtequiterMatrice.exe* avec l'option 0. La condition (3) a été respectée en utilisant la composante exécutable *PreparerMatrice.exe* avec l'option 2 dans une étape précédente, comme le montre la figure 4.1. À chaque utilisation de la composante *ConversionMatriceTxtEnFormatWeka.exe*, nous obtiendrons un fichier dans le dossier output *FichierSorties*, nommé *[NomfichierSource]WEkaFormat.arff*.

4.3.2 Chaîne 2 : Chaîne pour former la matrice fréquence

Cette chaîne permet d'avoir une matrice *Documents-Mots* dont le contenu correspond à la fréquence de chaque mot dans chaque document. Elle permet aussi d'avoir un fichier *arff* qui contient la fréquence de chaque attribut dans chaque document pour être utilisé par l'outil

Weka. Nous présentons dans ce qui suit les composantes qui n'ont pas été définies dans la partie 4.3.1.

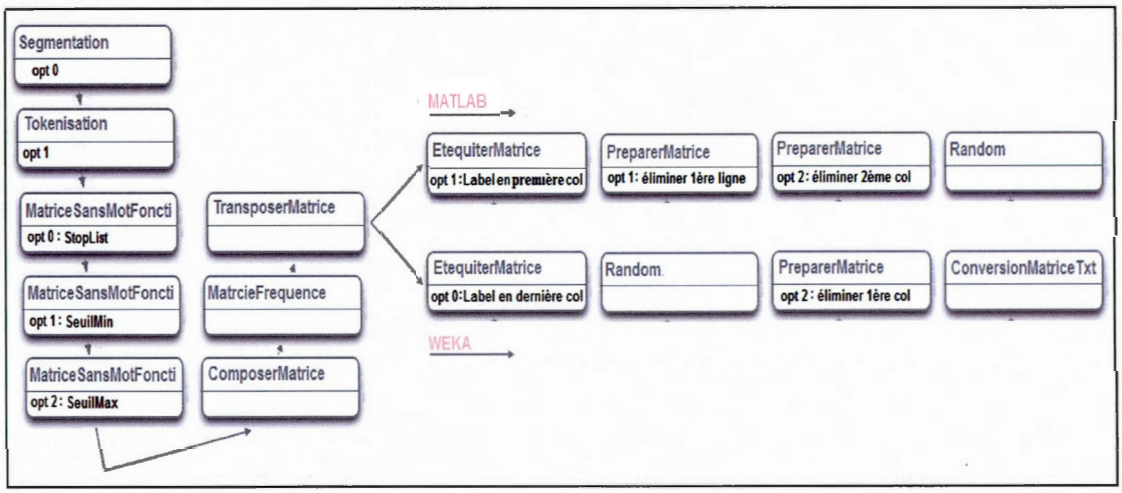


Figure 4.4 Chaîne de préparation de données textuelles pour former des matrices fréquences par l'intermédiaire de l'outil PDTPM.

a) Segmentation

La segmentation est la division d'un texte en plusieurs fragments. La segmentation peut se faire selon plusieurs possibilités que l'utilisateur identifie. Le module que nous présentons, Segmentation.exe, offre les options suivantes: l'option 0 permet une segmentation par paragraphe. Dans cette option le module permet la division de texte en plusieurs paragraphes. Chaque paragraphe est identifié par le retour de chariot "\r". L'option 1 permet la segmentation par nombre de mots qui est donné en paramètre (param1 *NbrMots*). Dans cette option, le module permet la division du texte en plusieurs segments, identifiés par le nombre de mots qu'ils présentent. L'option 2 permet la division par balises. Chaque segment est identifié par une balise ouvrante *<Balise>* et une balise fermante *</Balise>*. Nous avons appliqué l'option 0 qui permet une segmentation par paragraphes. En utilisant *Segmentation.exe*, nous obtenons, un fichier nommé *fichierRegroupantSegments[Nomfichier-Source].txt* dans le dossier *FichierSorties*.

b) Tokenisation

La tokenisation a été définie dans la partie 4.3.1.b. Nous appliquons cette composante dans cette chaîne avec l'option 1 qui permet la tokenisation d'un ensemble de segments, étant donné que nous avons appliqué le module de segmentation.

c) Matrice fréquence

Nous avons choisi dans notre chaîne la composante matriceFrequence.exe. Ce module permet de générer la matrice en tenant compte du nombre de fois où l'unité est présente dans le document. Le résultat de l'application de cette composante est un fichier textuel nommé matriceFrequence.txt dans le dossier output *FichierSorties*.

4.3.3 Chaîne 3 : Chaîne pour former la matrice TfIdf

Cette chaîne de préparation de données textuelles permet d'avoir une matrice *Documents-Mots* dont le contenu de chaque cellule correspond à la valeur TfIdf de chaque mot dans chaque document. Cette chaîne de préparation de données permet aussi de créer un fichier de type *arff* spécifique à l'outil Weka. Il contient la valeur TfIdf de chaque mot pour chaque document.

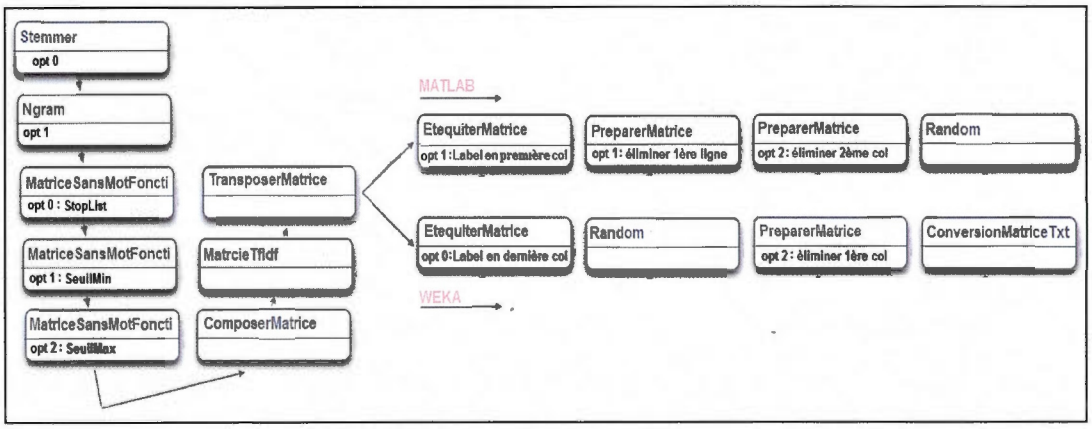


Figure 4.5 Chaîne de préparation de données textuelles pour former des matrices TfIdf par l'intermédiaire de l'outil PDTPM.

a) Ngram

L'extraction par N-Gram permet la division de textes ou de segments de textes en unités comptées par leurs longueurs (nombre de caractères). Par exemple, l'extraction bi-gram du mot "TEXT" est : _T, TE, EX, XT, T_. L'algorithme utilisé est celui présenté par W. B. Cavnar et J. M. Trenkle [36]. Cette composante exécutable contient deux options. L'option 0 permet l'extraction N-Gram d'un corpus de texte formé de plusieurs fichiers texte. L'option 1 permet l'extraction N-Gram d'un corpus de texte formé de plusieurs segments de texte. Avant d'utiliser ce module, il faut d'abord procéder à la segmentation des documents constituant le corpus. Nous avons utilisé l'option 0 étant donné que nous n'avons pas segmenté le texte dans cette chaîne de préparation de données. Le résultat de l'application de cette composante est un fichier textuel nommé *matriceIntermédiaire.txt* dans le dossier output *FichierSorties*.

b) Matrcie TfIdf

Ce module permet de générer la matrice *documents-mots* en tenant compte de la valeur *tf*Idf* de l'unité dans le document. *TF* correspond à la fréquence du terme dans un document. Plus le *TF* est élevé, plus le terme est important. *DF* correspond à la fréquence de documents, c.à.d. le nombre de documents contenant le terme *t*. *IDF* définit l'inverse de *DF*. Il représente l'inégalité de la répartition du terme dans le corpus, tel que, plus le terme est réparti uniformément, moins il est spécifique à un document particulier [37]. Ainsi la valeur *tf*Idf* est plus importante lorsque l'unité est fréquemment présente dans un document et moins présente dans le reste des documents du corpus. La formule *tf*idf* peut se calculer de différentes manières. Nous avons choisi d'adopter les formules les plus couramment utilisées et qui sont inspirées du cadre de travail proposé par J. Y. Nie [37]. Ainsi, la formule de pondération *tf*idf* peut s'écrire comme suit :

$$Weight(t,D) = tf(t,D) * idf(t) \text{ tel que,}$$

$$tf(t,D) = log(freq(t,D) + 1; \qquad Idf = Log(N/n(t).$$

Tel que;

freq(t, D) est la fréquence du terme t dans le document D;

N nombre de documents dans le corpus;

et enfin *n(t)* nombre de ceux qui contiennent le terme.

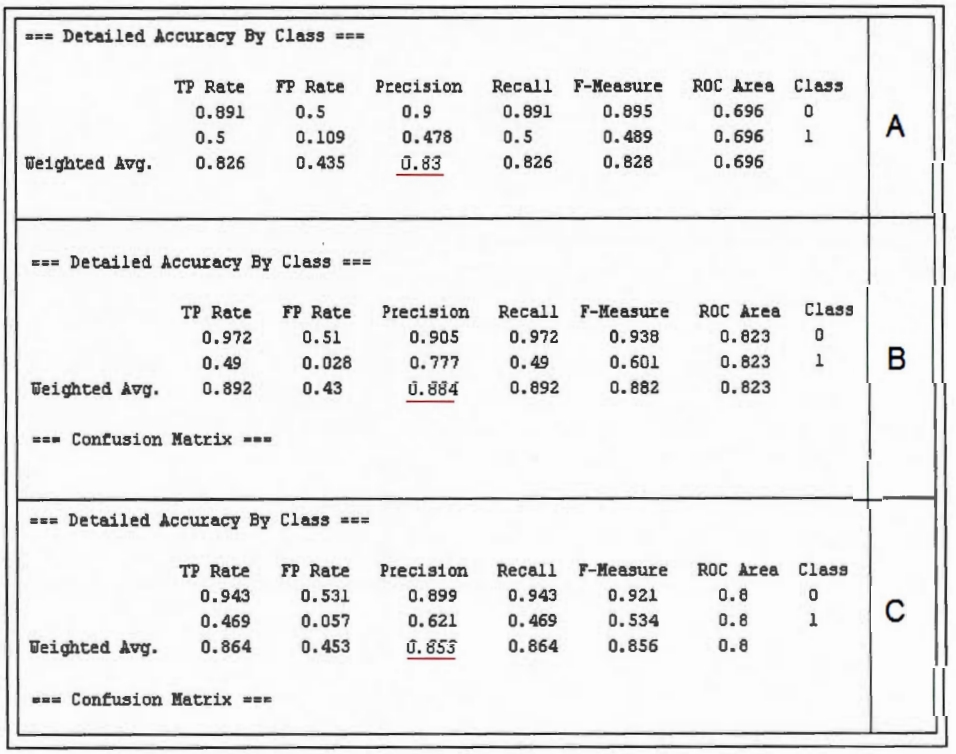
Dans cette chaîne, nous élaborons la matrice *mots-documents* selon la formule *tf*idf* présentée ci-dessus. Le résultat de l'application de cette composante est un fichier textuel nommé *matriceTfIdf.txt* dans le dossier output *FichierSorties*.

4.4 Expérimentation de plusieurs méthodes d'apprentissage sur les données prétraitées extraites de la base de données LINGSPAM

En exécutant les trois chaînes de préparation de données définies dans la partie précédente, nous avons obtenu 6 fichiers textuels finaux : deux fichiers textuels de type binaire, deux fichiers textuels de type fréquence et deux autres fichiers de types TfIdf. Chaque paire comporte un fichier textuel avec la colonne label au début de fichier pour être appliqué avec l'outil Matlab et un autre fichier textuel de type *arff* pour être utilisé avec Weka. Chaque méthode a été appliquée en utilisant une validation croisée (cross validation) pour améliorer les résultats. Nous avons adopté la précision comme mesure de performance pour toutes les méthodes d'apprentissage automatique. Nous présentons dans ce qui suit les méthodes d'apprentissage automatique que nous avons utilisées et les résultats obtenus pour chaque expérimentation.

4.4.1 KNN

Dans le chapitre 1, nous avons reparti les méthodes d'apprentissage automatique selon 4 approches, l'approche probabiliste, l'approche statistique, l'approche logique et l'approche basée sur la similarité (chap1, section 3.2.1) à laquelle appartient l'algorithme KNN. Nous avons appliqué la méthode IBK (Instance-based learning algorithm) intégrée dans l'outil Weka. Cette méthode a été conçue par D. W. Aha et al. [38]. En se basant sur la précision, le résultat obtenu avec la matrice fréquence a donné une précision plus élevée par rapport à la matrice binaire et la matrice TfIdf. La figure suivante représente une capture d'écran des résultats obtenus, les parties A, B et C correspondant respectivement aux résultats des matrices binaires, fréquence et TfIdf.



A

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.891     0.5        0.9      0.891     0.895      0.696     0
                 0.5       0.109      0.478    0.5       0.489      0.696     1
Weighted Avg.    0.826     0.435      0.83     0.826     0.828      0.696
```

B

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.972     0.51       0.905    0.972     0.938      0.823     0
                 0.49      0.028      0.777    0.49      0.601      0.823     1
Weighted Avg.    0.892     0.43       0.884    0.892     0.882      0.823

=== Confusion Matrix ===
```

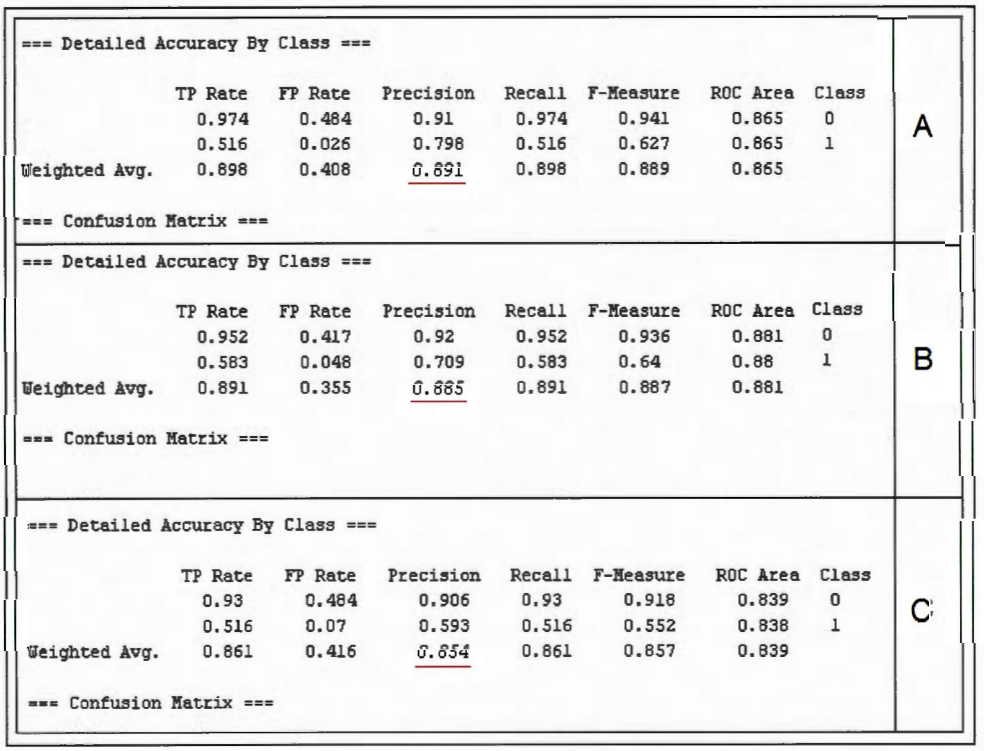
C

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.943     0.531      0.899    0.943     0.921      0.8       0
                 0.469     0.057      0.621    0.469     0.534      0.8       1
Weighted Avg.    0.864     0.453      0.853    0.864     0.856      0.8

=== Confusion Matrix ===
```

Figure 4.6 Taux de précision de la méthode IBK sur les trois matrices textuelles (binaire, fréquence et Tfldf).

4.4.2 Naive Bayes

La méthode bayesienne naïve appartient à l'approche probabiliste évoquée dans le chapitre 1 (section 3.1.2). Nous avons appliqué la méthode NaiveBayes intégrée dans l'outil Weka. Cette méthode a été conçue par G. H. John and P. Langley [39]. En se basant sur la précision, le meilleur résultat est obtenu par la matrice binaire avec 89.1 % de précision. La figure suivante représente une capture d'écran des résultats obtenus, les parties A, B et C correspondant respectivement aux résultats des matrices binaire, fréquence et *Tf Idf*.



A

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.974     0.484       0.91     0.974      0.941      0.865    0
                 0.516     0.026       0.798    0.516      0.627      0.865    1
Weighted Avg.    0.898     0.408       0.891    0.898      0.889      0.865

=== Confusion Matrix ===
```

B

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.952     0.417       0.92     0.952      0.936      0.881    0
                 0.583     0.048       0.709    0.583      0.64       0.88     1
Weighted Avg.    0.891     0.355       0.885    0.891      0.887      0.881

=== Confusion Matrix ===
```

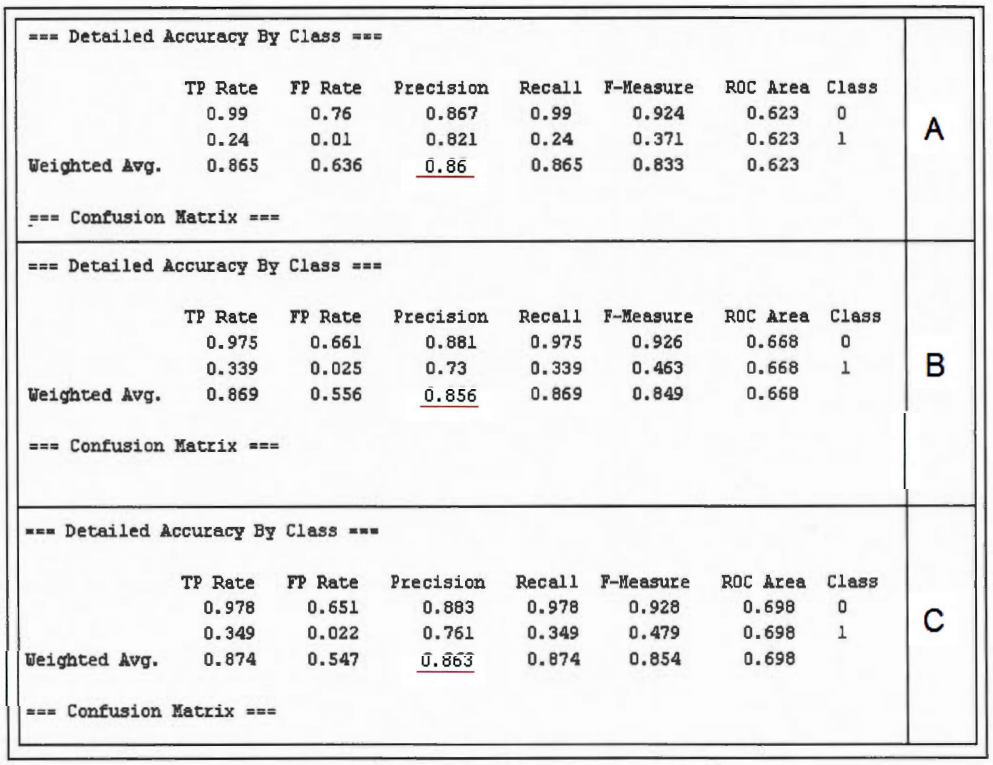
C

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.93      0.484       0.906    0.93       0.918      0.839    0
                 0.516     0.07        0.593    0.516      0.552      0.838    1
Weighted Avg.    0.861     0.416       0.854    0.861      0.857      0.839

=== Confusion Matrix ===
```

Figure 4.7 Taux de précision de la méthode Naive Bayes sur les trois matrices textuelles (binaire, fréquence et Tf Idf).

4.4.3 J48

La méthode J48 fait partie de l'approche logique vue dans le chapitre 1 (section 3.1.1). Elle correspond à une variante de C4.5 étant donné qu'elle génère un arbre de décision élagué. Nous avons appliqué la méthode J48 intégrée dans l'outil Weka. Cette méthode a été conçue par J. R. Quinlan [40]. En se basant sur la précision, le meilleur résultat est donné par la matrice TfIdf avec une précision égale à 86.3 %. La figure suivante représente une capture d'écran des résultats obtenus et où les parties A, B et C correspondent respectivement aux résultats des matrices binaire, fréquence et TfIdf.



```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.99      0.76       0.867     0.99      0.924      0.623    0
                 0.24      0.01       0.821     0.24      0.371      0.623    1
Weighted Avg.    0.865     0.636      0.86      0.865     0.833      0.623

=== Confusion Matrix ===
```

A

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.975     0.661      0.881     0.975     0.926      0.668    0
                 0.339     0.025      0.73      0.339     0.463      0.668    1
Weighted Avg.    0.869     0.556      0.856     0.869     0.849      0.668

=== Confusion Matrix ===
```

B

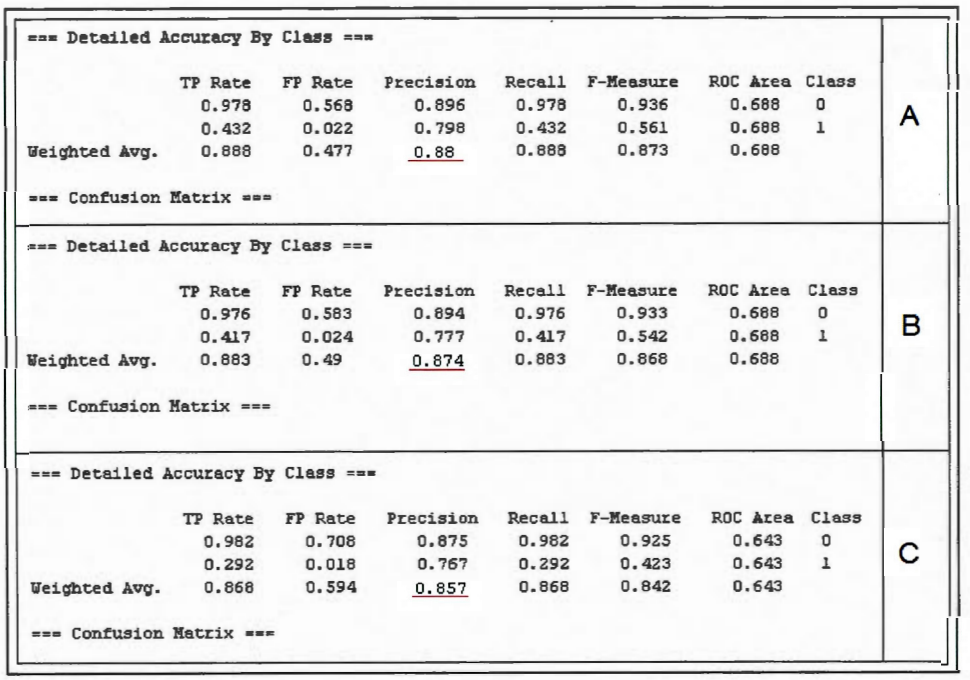
```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.978     0.651      0.883     0.978     0.928      0.698    0
                 0.349     0.022      0.761     0.349     0.479      0.698    1
Weighted Avg.    0.874     0.547      0.863     0.874     0.854      0.698

=== Confusion Matrix ===
```

C

Figure 4.8 Taux de précision de la méthode J48 sur les trois matrices textuelles (binaire, fréquence et Tf Idf).

4.4.4 JRip

JRip ou JRipper (Repeated Incremental Pruning to Produce Error Reduction) est une méthode qui implémente un algorithme de règles de décision, avec une répétition incrémentale pour produire une réduction d'erreur. Cette méthode appartient à l'approche logique vue dans le chapitre 1 (section 3.1.1). Nous avons appliqué la méthode JRip intégrée dans l'outil Weka qui est une variante de l'algorithme principal. Cette méthode a été conçue par William W. Cohen [41]. La matrice binaire a donné le meilleur résultat avec une précision égale à 88 %. La figure suivante représente une capture d'écran des résultats obtenus et où les parties A, B et C correspondent respectivement aux résultats des matrices binaire, fréquence et Tf Idf.



A

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.978     0.568       0.896     0.978     0.936      0.688    0
                 0.432     0.022       0.798     0.432     0.561      0.688    1
Weighted Avg.    0.888     0.477       0.88      0.888     0.873      0.688

=== Confusion Matrix ===
```

B

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.976     0.583       0.894     0.976     0.933      0.688    0
                 0.417     0.024       0.777     0.417     0.542      0.688    1
Weighted Avg.    0.883     0.49        0.874     0.883     0.868      0.688

=== Confusion Matrix ===
```

C

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.982     0.708       0.875     0.982     0.925      0.643    0
                 0.292     0.018       0.767     0.292     0.423      0.643    1
Weighted Avg.    0.868     0.594       0.857     0.868     0.842      0.643

=== Confusion Matrix ===
```

Figure 4.9 Taux de précision de la méthode JRip sur les trois matrices textuelles (binaire, fréquence et Tf Idf).

4.4.5 SVM

Les machines à vecteurs de support (SVM, support vector machine) sont un ensemble de techniques statistiques d'apprentissage supervisé destinées à résoudre des problèmes de classification. SVM appartient à l'approche statistique évoquée dans le chapitre 1 (section 3.1.3). Nous avons appliqué une variante de SVM nommé *svmtrain*, intégrée dans l'outil Matlab. Cette variante utilise une fonction noyau (Kernel fonction) polynomiale. En effet, une méthode SVM contient trois paramètres ajustables qui sont la pente *alpha*, le *biais* et le degré polynomial. On a gardé l'ordre polynomial donné par défaut par Matlab qui est égal à trois. Les trois matrices ont donné des résultats égaux à 88.14 %. La figure suivante représente une capture d'écran des résultats obtenus avec la matrice binaire.

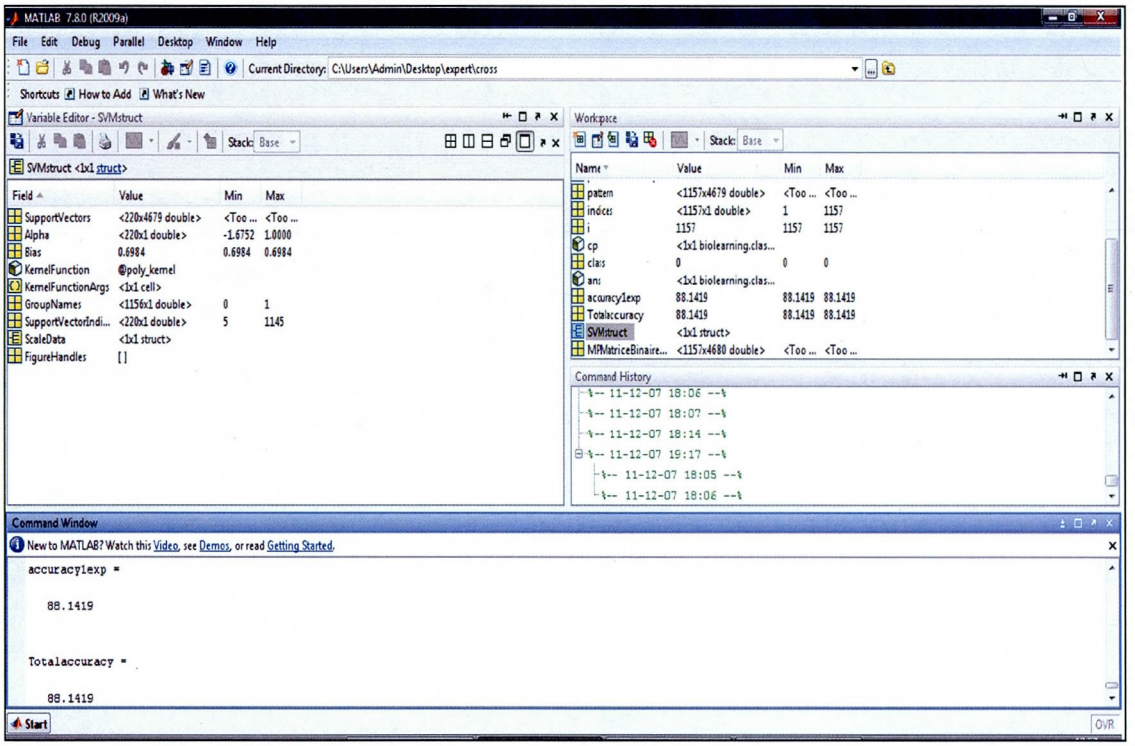


Figure 4.10 Taux de précision de la méthode SVM sur la matrice textuelle binaire.

4.4.6 Réseaux de neurones (NN)

Les réseaux de neurones (NN) sont placés au sein de la famille des applications statistiques. Nous avons utilisé un perceptron multicouches avec apprentissage par rétro-propagation. Pour cela, nous avons eu recours à la fonction *newff* (new feed-forward backpropagation network) qui permet de créer un réseau de neurones de type Perceptron Multi Couches. Nous avons crée un réseau de neurones à deux couches cachées. La première couche cachée contient deux neurones et la deuxième couche cachée contient trois neurones. En effet, nous avons testé diverses architectures, concernant les couches cachées, et nous avons obtenu le meilleur résultat en choisissant l'architecture citée ci-dessus. Nous avons choisi *trainrp* au lieu de la fonction *trainlm* donnée par défaut car cette dernière a besoin de beaucoup de mémoire pour son exécution alors que *trainrp* est certes plus lente mais ne demande pas trop de mémoire. La fonction de transfert utilisée pour les deux couches cachées est *transig* qui est une fonction d'activation sigmoïde. La fonction de transfert pour la couche

de sortie est *purelin* qui est une fonction d'activation ou de transfert linéaire. Nous présentons dans la figure suivante les caractéristiques du réseau de neurones que nous avons conçu.

• Construction du réseau (Code Matlab)

```
for i=1:1157
test=(indices==i);train1=~test;
net =newff(pattern(train1,:)',target(train1,:)',[3 2],{'tansig','tansig','purelin'},'trainrp');
net.trainParam.lr=0.1;
net=train(net,pattern(train1,:)',target(train1,:)');
Y=sim(net,pattern(test,:)');
newtarget=target(test,:)';
e = newtarget - Y
meansqe = mse(e)
performance1exp = performance1exp + mse(e)
end
```

• Couches cachées
Nombre de couches cachées : 2
Nombre de neurones dans la couche1 : 3
Nombre de neurones dans la couche2 : 2
Fonctions de transfert pour la première couche cachée : tansig
Fonction de transfert pour la deuxième couche caché : tansig

• Couche de sortie
Fonction de transfert pour la couche de sortie : Purelin.
Purelin une fonction d'activation ou de transfert linéaire.

• Apprentissage
Fonction d'apprentissage du réseau de neurones : Trainrp.
Trainrp fait l'apprentissage de réseau qui met à jour les valeurs des poids suivant l'optimisation de l'algorithme de rétro-propagation résilient (Rprop).

Figure 4.11 Construction du réseau de neurones : architecture proposée.

L'apprentissage commence en appliquant la fonction Matlab *train* et en simulant le réseau à l'aide de la fonction *SIM* de Matlab. Nous avons obtenu le meilleur résultat comparé aux autres méthodes d'apprentissage avec une précision égale à 91.20%. La figure suivante représente une capture d'écran des résultats obtenus avec la matrice binaire.

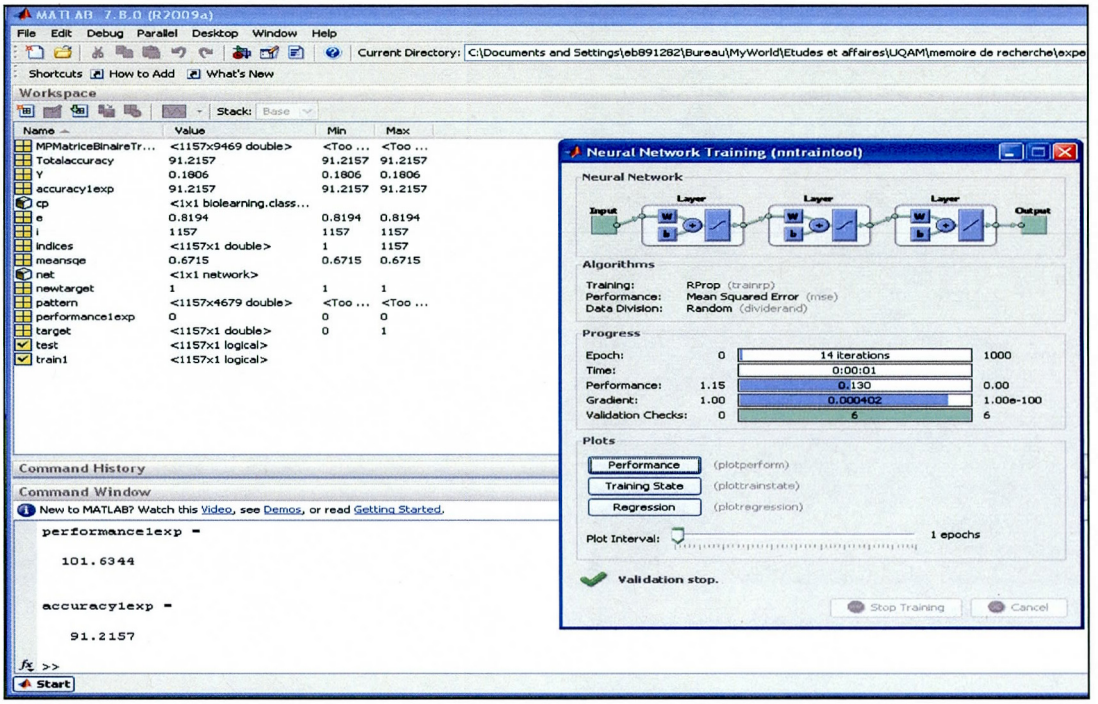


Figure 4.12 Taux de précision de la méthode réseau de neurones sur la matrice textuelle binaire.

Le tableau suivant récapitule les résultats de toutes les méthodes d'apprentissage automatique que nous avons utilisées sur toutes les matrices de données que nous avons préparées.

	KNN	NAIVE BAYES	J48	JRIP	SVM	*NN*	MOYENNE PAR CHAÎNE
Fréquence	88	88.5	85	87	88.14	91.20	87.97
Binaire	83	89.1	86	88	88.14	91.20	87.57
Tfldf	85.3	85.48	86.3	85.7	88.14	91.20	87.02
MOYENNE / MÉTHODE	85.5	87.7	85.8	86.9	88.14	91.20	87.54

Tableau 4.1 Tableau récapitulatif de tous les résultats obtenus.

4.5 Discussion des résultats obtenus

4.5.1 Discussion quantitative

Selon les résultats des expérimentations obtenus sur la matrice binaire, la matrice fréquence et la matrice TfIdf, le meilleur taux de précision a été produit par la méthode des réseaux de neurones (NN) avec une valeur égale 91.20 % suivie de la méthode de la machine à vecteurs de support (SVM) avec une valeur de précision égale à 88.14 %. Les méthodes appartenant à l'approche statistique ont réalisé des taux de précision plus élevés que les méthodes qui se basent sur la similarité, la logique ou les méthodes probabilistes.

Dans l'article [34], Z. Q. Wang et al. ont utilisé le même corpus de données LingSpam pour comparer plusieurs méthodes d'apprentissage automatique. Nous avons obtenu des résultats semblables pour les méthodes KNN et Naive Bayes. Cependant, les auteurs ont obtenu une précision supérieure à ce que nous avons obtenu en appliquant SVM sur LingSpam. Cette différence peut être expliquée par la méthode SVM utilisée. Nous avons utilisé la méthode *SVMtrain*, intégrée dans Matlab qui utilise une fonction noyau polynomiale. Nous avons maintenu les paramètres par défaut donnés par Matlab comme l'ordre de la fonction polynomiale qui est égal à 3 dans notre cas.

Les résultats que nous avons obtenus prouvent ce que nous avons évoqué dans le deuxième chapitre : SVM et NN sont les méthodes qui répondent le mieux à la haute dimension des donnés textuelles. Dans notre cas, nous avons autour de 4679 attributs dans les matrices que nous avons formées. Vue ce grand nombre de caractéristiques, la méthode NN a donné le meilleur résultat. Nous avons reconfiguré cette méthode dans notre cas afin d'avoir de bons résultats. Pour SVM, nous avons presque utilisé la configuration par défaut de Matlab qui semble être la plus utilisée. Malgré que les méthodes KNN, Naive Bayes, J48 et Jrip ont été moins efficaces que SVM et NN, elles ont donné quand même des résultats satisfaisants. Le meilleur résultat donné par KNN est égale à 88% lors de son application sur la matrice fréquence. Naives Bayes, appliquée sur la matrice binaire, a donné une précision égale à 89,1. Ce résultat dépasse la performance de SVM appliquée sur la même chaîne.

Nous pouvons constater qu'à part NN et SVM, toutes les autres méthodes réagissent quant au contenu de la matrice et quant à la façon que les données ont été traitées. Par

exemple KNN, exécutée sur la chaîne binaire, a donné la plus mauvaise précision qui est égale à 83%. Mais en l'appliquant sur la chaîne fréquence, il a donné un bon résultat de prédiction (88%). Ce résultat peut être expliqué principalement en raison du contenu de la matrice. Étant donné que le modèle de classification établi par IBK se base sur le calcul de la similarité entre les documents, IBK a besoin donc des valeurs comme la fréquence ou Tfldf pour calculer la distance de similarité entre les documents plus que la présence ou l'absence du mot (matrice binaire). Une autre raison pour expliquer ces résultats est le fait que dans la matrice fréquence nous avons utilisé la composante Segmentation.exe qui augmente le nombre d'observations. Cette composante permet de diviser un seul document en plusieurs enregistrements. En ce qui concerne la méthode Naives Bayes, le résultat de prédiction est en moyenne de 87,7% pour les trois chaînes obtenues. Le meilleur résultat est obtenu suite à son application sur la matrice binaire. La précision dans ce cas (89,1%) a dépassé la méthode SVM. JRip et J48 ont donné des résultats semblables et s'adaptent mieux avec les données prétraitées selon la chaîne binaire. En effet, cette chaîne utilise le module exécutable *Stemming.exe*, ce qui permet de convertir plusieurs mots en leur racine ; cela permet donc de diminuer le nombre d'attributs en rassemblant les différentes variantes flexionnelles d'un mot autour d'un même mot racine (stemme). Ainsi, en diminuant la dimension de la matrice de données, nous augmentons la performance des méthodes de règles/arbres de décision qui fonctionnent mieux avec un ensemble d'attributs restreint. Enfin, d'un point de vue quantitatif, la méthode des réseaux de neurones et la méthode SVM ont donné les meilleurs résultats avec une précision qui dépasse 90%. Ce résultat coïncide avec les caractéristiques déjà évoquées de NN et SVM qui sont tolérants à la grande dimension des données textuelles et sont capables de donner une solution globale à un problème de classification. Nous discutons dans la suite, de la performance de ces méthodes d'apprentissage du point de vue qualitatif.

4.5.2 Discussion qualitative

La méthode de la machine à vecteurs du support et celle des réseaux de neurones ont dépassé les autres méthodes de classification, d'un point de vue quantitatif, mais elles demandent une durée d'apprentissage assez longue. De plus, il est difficile d'interpréter le résultat obtenu puisqu'elles ne fournissent pas de règles compréhensibles et exploitables. Le

réseau de neurones correspond à une boite noire qui n'identifie pas des règles et des modèles de classification. Cependant, Jrip et J48 produisent des résultats interprétables et explicites. Ainsi, en appliquant J48, nous avons obtenu un arbre de décision que nous présentons dans la figure suivante.

```
J48 pruned tree
------------------
money <= 0
|  free <= 0
|  |  check <= 0
|  |  |  $ <= 6: 0 (1033.0/101.0)
|  |  |  $ > 6
|  |  |  |  study <= 0
|  |  |  |  |  double <= 0
|  |  |  |  |  |  case <= 0
|  |  |  |  |  |  |  addition <= 0: 1 (18.0/3.0)
|  |  |  |  |  |  |  addition > 0: 0 (2.0)
|  |  |  |  |  |  case > 0: 0 (2.0)
|  |  |  |  |  double > 0: 0 (3.0)
|  |  |  |  study > 0: 0 (7.0)
|  |  check > 0
|  |  |  card <= 0
|  |  |  |  work <= 0
|  |  |  |  |  security <= 0
|  |  |  |  |  |  adult <= 0: 0 (10.0/2.0)
|  |  |  |  |  |  adult > 0: 1 (2.0)
|  |  |  |  |  security > 0: 1 (2.0)
|  |  |  |  work > 0: 1 (3.0)
|  |  |  card > 0: 1 (13.0)
|  free > 0: 1 (34.0/6.0)
money > 0: 1 (28.0/2.0)

Number of Leaves  : 13
Size of the tree :  25
```

Figure 4.13 Arbre de décision formé lors de l'application de J48 sur la matrice fréquence.

Cet arbre est obtenu en appliquant J48 sur la matrice fréquence. Il contient 13 feuilles, donc 13 règles de décision. Par exemple, si on a un document qui a : *money <= 0 AND free <= 0 AND Check <=0 AND $<= 6* alors c'est un message légitime. Si on a un document qui a: *Check > 0 AND card > 0 AND Work > 0 AND Adult > 0 AND Security > 0* alors c'est un message spam.

La méthode Jrip produit aussi des règles de décision sur lesquelles elle se base pour classifier les documents. Ces règles sont explicites et interprétables. La figure suivante présente les trois règles que nous avons obtenues lors de l'application de Jrip sur la matrice TfIdf.

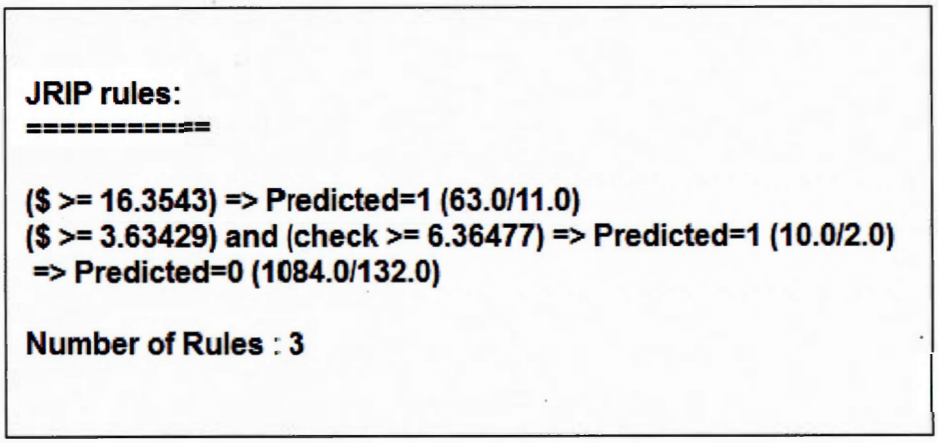


Figure 4.14 Les règles de décision obtenues lors de l'application de Jrip sur la matrice fréquence.

Le résultat est donc compréhensible et il peut être compris et utilisé par un humain. Par exemple, si on a un document qui a : *$ >=3,6 AND Check >= 6,36* alors il appartient à la classe 1 et par conséquence c'est un spam.

En plus de produire des règles interprétables, Jrip et J48 associent une information sur la qualité de chaque règle. Ils donnent un poids pour chaque règle ; à la fin de chaque règle, Weka affiche deux valeurs entre parenthèses (voir figure 4.13 et figure 4.14). Le premier nombre, à gauche, représente le nombre d'instances correctement classifiées, c'est à dire les instances parmi tous les exemples d'apprentissage où la règle vérifie toutes les conditions de chaque instance et le résultat de la classification est vrai. Le deuxième nombre, à droite, représente les instances incorrectement classifiées, c'est à dire les instances parmi tous les exemples d'apprentissage où la règle vérifie toutes les conditions de chaque instance mais le résultat de classification est incorrect. Par exemple, la règle *money <= 0 AND free <= 0 AND Check <=0 AND $<= 6* est pondérée par les valeurs (1033.0/101.0). Ces deux algorithmes ne produisent pas seulement des règles explicites, mais ils donnent aussi une information sur la qualité de ces règles.

Enfin, les règles obtenues peuvent êtres exploitées pour enrichir une base de connaissances. Les règles explicites formées par Jrip et J48 peuvent être intégrées dans la

base de connaissances d'un système à base de connaissances. Ces connaissances sont généralement exploitées par un expert afin de prendre des décisions mais aussi, par un système automatique d'aide à la décision. Avec la pondération des règles, nous pouvons améliorer la qualité da la base de connaissances en n'intégrant que les règles ayant un taux de pondération supérieur à un seuil minimum donné par l'expert.

4.6 Conclusion

Nous avons consacré ce chapitre à expérimenter notre plateforme de préparation de données textuelles (PDTPM) et les méthodes d'apprentissage automatique les plus utilisées. Dans une première partie, nous avons organisé les données textuelles qui proviennent du corpus de données public LingSpam à l'aide de PDTPM. Cet outil est très utile dans l'étape de préparation de données ; il a permis de créer plusieurs chaînes de préparation de données d'une façon très adaptable. Il nous a permis aussi de créer des matrices finales spécifiques à chaque outil d'apprentissage automatique, utilisé dans nos expérimentations. PDTPM offre plusieurs modules exécutables de préparation de données (13 composantes). La programmation modulaire nous permet d'intégrer facilement d'autres composantes. Dans une deuxième partie, nous avons exécuté plusieurs méthodes d'apprentissage automatiques sur les matrices formées par PDTPM. Les méthodes appartenant à l'approche statistique telle que NN et SVM ont donné les meilleurs résultats. Cependant, elles ne produisent pas de connaissances explicites et interprétables. Les méthodes qui se basent sur les règles/arbres de décision ont donné des résultats de prédiction satisfaisants mais surtout interprétables. Naives Bayes a donné une précision égale à 89.1% dans le cas où elle était appliquée sur la matrice binaire, et KNN a donné un résultat de la précision égale à 83 %, quand nous l'avons appliqué sur la matrice binaire. Mais cette dernière méthode, appliquée sur la matrice fréquence, a donné une meilleure précision (88%). Nous avons montré que les modules impliqués dans les chaînes de préparation des données affectent le résultat de la précision, d'où l'importance d'avoir un outil capable de générer facilement des chaînes de préparation de données flexibles comme cela est le cas avec PDTPM.

CONCLUSION

5.1 Introduction

Dans ce chapitre final, nous récapitulons les éléments importants de cette recherche sur l'application des méthodes de «Text Mining» pour classifier des courriels et l'élaboration d'un outil de préparation des données. Nous faisons un survol de notre recherche et nous présentons notre contribution dans ce travail. Ensuite, nous étalons les limites rencontrées lors de cette recherche et projetons des perspectives futures.

5.2 Survol de la recherche

Dans la première partie, nous avons commencé notre travail par une étude détaillée du domaine de la fouille de données textuelles (Text Mining). La tâche d'apprentissage automatique sur des données textuelles doit obligatoirement passer par une étape primordiale qui consiste au prétraitement de ces données. Cette étape est longue et suit des procédures soigneusement organisées pour former des données structurées sous forme de matrices. Cette étape commence par une collecte de données que peuvent être publiques (LingSpam, Reuters) ou obtenue suite à un processus de collection de données. Les bases de données publiques nous offrent une grande quantité de données et nous permettent de nous comparer aux travaux antérieurs. Mais ces données sont statiques et ne sont pas mises à jour. Les données obtenues suite à un processus de collecte de données nous obligent à accomplir un effort supplémentaire de nettoyage et de standardisation. L'avantage est que les données sont mises à jour et reflètent la réalité.

Le processus de préparation de données passe par plusieurs étapes. Chaque étape permet de modifier les données passées en entrée et se distingue par ses objectifs. Nous trouvons des étapes qui visent à réduire l'espace de données comme StopList, seuil-Minimal, seuil-Maximal ou N-Gram. Ces composantes de préparation de données permettent de réduire la haute dimensionnalité des données textuelles qui pose un problème quant à la performance des algorithmes d'apprentissage automatique. D'autres étapes de préparation de

données ont pour objectif de représenter les données textuelles sous une forme bien structurée comme la segmentation, la composition de la matrice documents-mots, l'étiquetage de la matrice ou la transposition de la matrice de données textuelles. Enfin, d'autres composantes mettent l'accent sur le contenu des données comme la Tokenisation ou la transformation de la matrice documents-mots en matrice binaire, matrice Tfldf ou matrice fréquence. Toutes ces étapes de préparation de données doivent être appliquées d'une façon organisée, dans un processus bien déterminé et bien ordonné afin de passer d'un ensemble de données textuelles brutes vers un ensemble de vecteurs numériques. L'ensemble de tous les vecteurs numériques forme la matrice des données documents-mots.

La prédiction se fait en utilisant la matrice documents-mots finale. Les modèles de prédiction sont assez variés et peuvent être divisés en deux groupes selon le type de prévision. Ainsi, la première famille regroupe les méthodes d'apprentissage supervisées. Dans ce cas, le but est d'assigner une catégorie ou un thème à un nouveau document. L'ensemble des catégories est fixé à priori. Deuxièmement, nous trouvons le groupe des méthodes non supervisées. Dans ce cas, les catégories ne sont pas connues à priori et l'objectif principal consiste à former des groupes (clusters) à partir d'un ensemble de documents, selon un critère de similarité, pour ensuite assigner une catégorie à chaque groupe.

Selon un autre angle de vue, et selon un critère technique de prédiction, on peut diviser les méthodes d'apprentissage automatique selon quatre catégories : les méthodes logiques, les méthodes probabilistes, les méthodes statistiques et les méthodes du plus proche voisin. L'apprentissage logique regroupe les méthodes qui se basent sur des règles de décision ; l'enjeu de ces méthodes est principalement d'analyser les documents déjà préparés puis de générer des modèles qui pourraient correspondre à ces documents. Ces modèles sont donc les règles identifiant le groupe des exemples positifs. Les algorithmes utilisant l'approche de règles de décision se basent souvent sur le modèle d'arbre de décision comme l'algorithme CART et l'algorithme C4.5. Les arbres de décision constituent une méthode de classification bien connue. Une autre approche regroupe les méthodes probabilistes. Mathématiquement, l'objectif est d'estimer la probabilité d'une classe C compte tenu de la présence ou de l'absence du mot x d'un dictionnaire. Le problème majeur réside dans le calcul de

probabilités complètes ; ce calcul est quasiment impossible même pour un dictionnaire de 150 mots qui aurait 2^{150} combinaisons possibles. Ainsi, une approche simplifiée pour l'estimation des probabilités, appelée bayesienne naïve, a souvent été essayée. Les classificateurs simples supposent l'indépendance des variables ; cette hypothèse réduit le temps de computation et quand elle s'avère vraie, elle permet aux classificateurs bayesiens simples d'avoir un taux d'erreur inferieur, ce qui conduit à une large application de cette approche [11][13][16][17]. Cependant, dans la pratique, il y a des dépendances entre les attributs. Les algorithmes utilisant l'approche bayesienne en prenant en compte les dépendances entre les caractéristiques se basent sur le modèle de réseau bayesien. La troisième approche est l'approche statistique ; un critère mathématique qui est le taux d'erreur est généralement utilisé pour évaluer une fonction de classification. Deux méthodes sont très connues et utilisées pour la prédiction : les réseaux de neurones et les machines à vecteur de support (SVM). Enfin, l'approche basée sur la similarité qui consiste à trouver le plus proche voisin d'un document est un processus non supervisé très connu et utilisé dans le «Text Mining». Un document X contenant plusieurs mots est représenté sous la forme de vecteur ; on essaye de trouver le correspond dont le plus proche des documents stockés qui sont représentés aussi sous forme des vecteurs. Deux documents sont similaires s'ils ont le plus possible de mots en commun. K-means est la méthode de classification la plus utilisée, basée sur la similarité.

Dans une deuxième partie, nous avons établi un état de l'art sur le filtrage des courriels au moyen de diverses méthodes d'apprentissage automatique. Nous avons donné dans cette partie un aperçu sur les travaux qui ont été produits pour classifier les courriels au moyen de méthodes d'apprentissage automatique. Notre état de l'art a été établi suivant deux points de vue. Premièrement, nous nous sommes concentrés sur la façon dont les travaux ont procédé avec les données textuelles avant la classification, une vue qui se concentre sur la façon de préparer et de traiter les données, et nous mène à répartir les travaux en trois grandes lignes. Il y a une approche statistique de préparation de texte qui se base sur tout le contenu des courriels et sur l'occurrence des mots dans le document. Cette approche suppose que les attributs sont indépendants les uns des autres [14][16][17]. Contrairement à cette dernière, il existe une approche qui utilise quelques caractéristiques comme attributs d'apprentissage en

se basant sur un critère de sélection de caractéristiques ou en se basant principalement sur des caractéristiques particulières des courriels comme «Subject» ou «To» [8][10][20]. Enfin, une autre approche est basée sur le contexte des mots dans le corpus de texte en prenant en compte les dépendances linguistiques et syntaxiques entre les attributs. Dans ce cas, les méthodes existantes analysent la pertinence contextuelle des mots dans le corpus de texte. Elles utilisent des algorithmes linguistiques sophistiqués [6] pour identifier les significations des mots et leurs différentes relations sémantiques.

Deuxièmement, nous nous sommes concentrés sur les méthodes de classification utilisées. Les premières recherches principalement axées sur le filtrage des courriels au moyen des méthodes d'apprentissage automatique sont celles de W. Cohen [29] en 1996, qui classe les courriels, en s'appuyant sur des règles de décision. M. Sahami et al [30] ont appliqué l'algorithme bayesien naïf pour construire un classificateur anti-spams. L'application de la méthode SVM a également été proposée par H. Drucker et al. [31] en 1991. En 2004, P. I. Nakov et al [24] ont proposé aussi une approche de filtrage des courriels qui introduit un classificateur KNN. Une approche plus récente qui s'appelle «Boosting» et qui consiste à former plusieurs classificateurs, comme les arbres de décision, sur le même ensemble de données et en utilisant une technique d'échantillonnage, a été introduite en 2007 par J. He et al [33]. La plupart des méthodes ont donné de bons résultats [16][24][32], ce qui prouve l'efficacité de la classification des courriels au moyen des méthodes d'apprentissage automatique. Enfin, nous avons présenté comment les travaux évaluent la performance des méthodes d'apprentissage automatique. La performance peut être estimée en termes de plusieurs mesures. Ces mesures de performance s'appuient sur des ratios dont les plus importants sont le ratio de faux positifs et le ratio de faux négatifs. Toutes les méthodes de prédiction doivent être en mesure de maîtriser ces risques en visant à rendre le ratio de faux positifs nul et en baissant au maximum le ratio de faux négatifs. Les mesures les plus générales et les plus standards sont le taux d'erreur (Error rate), son erreur type et la précision.

Dans la troisième partie, nous avons introduit notre outil de préparation de données textuelles que nous avons appelé PDTPM : plateforme de Préparation de Données Textuelles basée sur la Programmation Modulaire. Nous avons conçu une plateforme qui permet de

prendre en entrée un ensemble de données textuelles brutes et de générer en sortie ces données sous une forme bien structurée qui consiste en une matrice de données documents-mots. Nous utilisons des composantes exécutables pour former des chaînes de préparation de données. En exécutant ces chaînes, elles permettent de transformer un corpus de données brutes en un format bien structuré. En se basant sur une programmation modulaire, notre système peut évoluer facilement : nous pouvons ajouter des composantes exécutables qui concernent l'apprentissage automatique comme K-means. Ainsi, notre système permettra non seulement la préparation des données textuelles brutes mais aussi l'apprentissage automatique.

Enfin, dans la quatrième partie de ce travail, nous avons expérimenté plusieurs méthodes d'apprentissage automatique sur un ensemble de données publiques. Tout d'abord, nous avons organisé les données textuelles qui proviennent du corpus des données publiques LingSpam à l'aide de PDTPM. Cet outil nous a été très utile dans l'étape de préparation de données. Il a permis de créer plusieurs chaînes de préparation de données d'une façon très adaptable. Il nous a permis aussi de créer des matrices finales spécifiques à chaque outil d'apprentissage automatique utilisé dans nos expérimentations. Ensuite, nous avons exécuté plusieurs méthodes d'apprentissage automatique sur les matrices formées dans la première partie. Les méthodes appartenant à l'approche statistique telle que NN et SVM ont donné les meilleurs résultats, sans produire de règles interprétables. Les méthodes qui se basent sur les règles de décision ont donné des résultats de prédiction satisfaisants mais surtout interprétables. Naïves Bayes a donné une précision égale à 89.1% dans le cas où elle était appliquée sur la matrice binaire. KNN a donné le plus bas résultat de précision (83 %) quand nous l'avons appliqué sur la matrice binaire. Mais cette dernière méthode, appliquée sur la matrice fréquence, a donnée une bonne précision (88%). Nous avons montré que les modules appliqués dans les chaînes de préparation des données affectent le résultat de précision, d'où l'importance d'avoir un outil capable de générer facilement des chaînes de préparation de données flexibles comme PDTPM.

5.3 Contribution de la recherche

Dans le cadre du travail décrit dans ce mémoire, nous avons tout d'abord développé et intégré une plateforme de préparation de données textuelles basée sur la programmation

modulaire. Cet outil était très utile dans notre travail car nous l'avons utilisé pour préparer les courriels de la base de données LingSpam, afin d'appliquer ensuite les méthodes d'apprentissage automatique. Nous avons aussi fait des expérimentations à partir de plusieurs méthodes d'apprentissage automatique en utilisant MATLAB et WEKA. Nous avons obtenu des résultats très satisfaisants. Nous avons discuté les résultats obtenus en les comparant aux résultats d'autres travaux. Nous avons aussi discuté et comparé les résultats obtenus dans ce travail en nous basant sur un critère qualitatif et sur un critère quantitatif.

5.4 Limites de la recherche

Bien que cette recherche ait apporté une contribution théorique et pratique, les résultats et les contributions formulés au terme de cette étude sont toutefois limités par certaines contraintes. Nous avons montré dans le deuxième chapitre qu'il existe plusieurs façons de préparer les données avant l'apprentissage. Cependant, nous nous sommes concentrés seulement sur l'approche statique qui se base sur tout le contenu des courriels et nous avons supposé que les attributs sont indépendants les uns des autres. Nous n'avons pas testé l'approche qui sélectionne quelques caractéristiques comme attributs d'apprentissage en se basant sur un critère syntaxique ou sémantique de sélection des caractéristiques [23][21][24]. Cette limite est due à l'absence des modules exécutables qui permettent de sélectionner les attributs. Une deuxième contrainte rencontrée consiste à l'incapacité d'expérimenter tous les courriels offerts par la base de données publique LingSpam ; nous avons utilisé 1157 courriels, ce qui équivaut à 50% de la base de données. Cette limite est dûe au serveur de notre laboratoire de recherche qui a une capacité de mémoire système limitée. Toutefois, 1157 courriels nous ont donné suffisamment d'attributs. En fait, après la tokenisation et la formation de la matrice intermédiaire documents-mots, et avant la sélection des attributs les plus pertinents, nous avons obtenu 9468 attributs.

5.5 Recherches futures

Au niveau des perspectives de recherche, il serait intéressant d'enrichir notre plateforme de préparation de données en concevant des modules de préparation de données qui permettent d'établir les dépendances syntaxiques entre les attributs pour que nous sélectionnions les attributs les plus prédictifs et améliorer ainsi la prédiction. Nous comptons

aussi intégrer des modules exécutables qui implémentent des méthodes d'apprentissage automatique. Enfin, après avoir obtenu les meilleurs résultats en appliquant la méthode des réseaux de neurones, et en tenant en compte de la limite de cette méthode au niveau de l'interprétabilité des résultats obtenus, nous envisageons nous concentrer sur cette méthode et essayer d'explorer les techniques capables d'extraire des interprétations et des explications.

BIBLIOGRAPHIE

[1] P. Hayati and V. Potdar. Evaluation of Spam Detection and Prevention Frameworks for Email and Image Spam - A State of Art, Proceedings of iiWAS2008, AIIDE, Acm, 2008.

[2] D. Cavit and al. Rapport Microsoft sur les données de sécurité, Microsoft, Volume 10, Juillet à décembre, Washington, États-Unis, 2010.

[3] MessageLabs Intelligence Report, Symantec, États-Unis, www.symantec.com/about/news/release/article.jsp?prid=20100526_01, 2010.

[4] Upasana and S. Chakravarty. A Survey on Text Classification Techniques for E-mail Filtering, Second International Conference on Machine Learning and Computing, pp. 32-36, 2010.

[5] S. Marsland. Machine learning An Algorithm Perspective, Chapman & Hall. CRC PRESS, 2009.

[6] S.M. Weiss, N. Indukhya, T. Zhang and F.J. Damerau. TEXT MINING: Predictive Methods for Analyzing Unstructured Information, Springer, 2005.

[7] M. F. Saeedian and H. Beigy. Spam Detection using Dynamic Weighted Voting based on Clustering, Second International Symposium on Intelligent Information Technology Application, IEEE, 2008.

[8] Y. Li and D. F. Hsu and S. M. Chung . Combining Multiple Feature Selection Methods for Text Categorization by Using Rank-Score Characteristics, 21st IEEE International Conference on Tools with Artificial Intelligence, 2009.

[9] C. C. Aggarwal and P. S. Yu. Finding Generalized Projected Clusters in High Dimensional Spaces, Proceedings of ACM Sigmod International Conference on Management of Data, ACM, Ney York, USA, pp. 70–81, 2000.

[10] H. Dalianis, M. Rosell and E. Sneiders. Clustering E-Mails for the Swedish Social Insurance Agency – What Part of the E-Mail Thread Gives the Best Quality? In Proceedings of the 7th international conference on Advances in natural language processing, Springer-Verlag, Berlin, pp. 115-120, 2010.

[11] GuoQiang, Research and Improvement for Feature Selection on Naive Bayes Text Classifier. 2nd International Conference on Future Computer and Communication, pp.V2-156-V2-159, 2010.

[12] G. Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification, The Journal of Machine Learning Research, pp. V3-1289–V3-1305, 2003.

[13] D. Wei and L.X. Yang. Weighted naive Bayesian classifier model based on information gain, International Conference on Intelligent System Design and Engineering Application, IEEE, pp.V2819-V2822, 2010.

[14] C. Apte, F. Damerau and S.M. Weiss. Text Mining with Decision Trees and Decision Rules, Conference on Automated Learning and Discovery Carnegie-Mellon University, 1998.

[15] J. Han and M. Kamber. Data Mining Concepts and Techniques. Morgan Kaufmann, 2006.

[16] J. Hovold. Naive Bayes Spam Filtering Using Word-Position-Based Attributes, In Conference on Email and Anti-Spam, 2004.

[17] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos and P. Stamatopoulos. Learning to Filter Spam E-Mail: A Comparison of a Naïve Bayesian and a Memory-Based Approach, Proceedings of the workshop "Machine Learning and

Textual Information Access", 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, France. pp. 1-13, 2000.

[18] S. Agarwal, th. Graepel, R. Herbrich, S. Har-Peled and D. Roth. Generalization Bounds for the Area Under the ROC Curve, Journal of Machine Learning Research 6. pp. 393–425, 2005.

[19] M. Kamber, L. Winstone, W. Gong, S. Cheng and J. Han. Generalization and Decision Tree Induction: Efficient Classification in Data Mining, Proceedings of the 7th International Workshop on Research Issues in Data Engineering, IEEE, 1997.

[20] A. Kolcz and J. Alspector. SVM-based Filtering of E-mail Spam with Content-specific Misclassification Costs, In Proceedings of the Workshop on Text Mining, 2001.

[21] C. Donghui and L. Zhijing. A new text categorization method based on HMM and SVM, 2nd International Conference on Computer Engineering and Technology (ICCET), IEEE, pp.V7-383-V7-386, 2010.

[22] S. M. Namburu, H. Tu, J. Luo and K. R. Pattipati. Experiments on Supervised Learning Algorithms for Text Categorization, Aerospace Conference, IEEE , pp.1-8, 2005.

[23] A. K.S. Wong, J. W.T. Lee and D. S. Yeung. Use of Linguistic Features in Context-Sensitive Text Classification, In Proceedings of the 4th international conference on Advances in Machine Learning and Cybernetics, Springer, pp. 701–710, Berlin, 2006.

[24] P. I. Nakov and P. M. Dobrikov. Non-Parametric Spam Filtering based on kNN and LSA, Scientific Literature Digital Library and Search Engine, USA, 2008.

[25] Z. Pei, Y. Zhou, L. Liu. L. Wang, Y. Lu and Y. Kong. A Mutual Information and Information Entropy Pair based Feature Selection Method in text classification, International Conference on Computer Application and System Modeling, IEEE, pp.V6-258-V6-261, 2010.

[26] L.T. Vinh, N.D. Thang and Y.K. Lee. An Improved Maximum Relevance and Minimum Redundancy Feature Selection Algorithm Based on Normalized Mutual Information, 10th Annual International Symposium on Applications and the Internet, IEEE, pp.395-398, 2010.

[27] S.M. Weiss, C. Apte, F.J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz and T. Hampp. Maximizing Text-Mining Performance, IEEE Intelligent Systems, IBM T.J. Watson Research Center, pp. v14-63–v14-69, 1999.

[28] C. Apte and S. Weiss. Data Mining with Decision Trees and Decision Rules, Future Generation Computer Systems, pp. v13-197-v13-210, 1997.

[29] W.W. Cohen. Learning Rules that Classify E-Mail, Proceedings of the AAAI Spring Symposium on Machine Learning in Information Research, Stanford, California, 1996.

[30] M. Sahami, S. Dumais, D. Heckerman and E. Horvitz. A bayesian approach to filtering junk e-mail, In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, 1998.

[31] H. Drucker, D. Wu and V. N. Vapnik. Support vector machines for spam categorization, IEEE Transactions on Neural Networks, pp. V10-1048- V10-1054, 1999.

[32] H. Chen, Y. Zhan and Y. Li. The application of decision tree in Chinese email classification, Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, pp. v1-305-v1-308, 2010.

[33] J. He, and B. Thiesson. Asymmetric gradient boosting with application to spam filtering, Proceedings of Fourth Conference on Email and Anti-Spam CEAS, California, USA 2007.

[34] Z.Q. Wang, X. Sun, X. Li and D. X. Zhang. An efficient svm-based spam filtering algorithm, Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, IEEE, pp. 3682-3686, 2006.

[35] M. Porter. An algorithm for suffix stripping, Program, pp. V14-130- V14-137, http://tartarus.org/~martin/PorterStemmer/index-old.html, 1980.

[36] W.B. Cavnar and J.M. Trenkle. N-gram based text categorization, In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, LasVegas, UAS, pp. 161–175, 1994.

[37] J.Y. Nie. Cross Language Information Retrieval. Computational Intelligence Bulletin, IEEE, pp. V2-19-v2-24, 2003.

[38] D.W. Aha, D. Kibler and M.K. Albert. Instance based learning algorithms, Machine Learning, pp. v6-37- v6-66, 1991.

[39] G.H. John and P. Langley. Estimating Continuous Distributions in Bayesian Classifiers, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338-345, Morgan Kaufmann, San Mateo, 1995.

[40] J. R. Quinlan. C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, pp. V16-235-v16-240, 1993.

[41] W. W. Cohen. Fast effective rule induction, In Proceedings of the Twelfth International Conference on Machine Learning, pp. 115-123, 1995.

[42] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, K. Miller. WordNet: An online lexical database. Int. J. Lexicograph. 3, 4, pp. 235–244, 1990.