UNIVERSITÉ KASDI MERBAH OUARGLA
Faculté des Nouvelles Technologies de
L'Information et de la Communication
Département d'Informatique et des Technologies de l'Information

MEMOIRE

En vue de l'obtention du diplôme de :
MASTER ACADEMIQUE

Domaine : Informatique
Filière : Informatique
Spécialité : Informatique Fondamentale

Présentées par :

- **BOURANE Zohra**
- **BERREKBIA Ahlam**

Thème :

UN SYSTÈME DE CLASSIFICATION ET DE RECHERCHE DE DOCUMENTS TEXTUELS DE LA LANGUE ANGLAISE

Soutenu publiquement
le : .02/07/2017.

Devant le jury :

Mme KHALDI Amine	Pr. Président	UKM Ouargla
Mr ZITOUNI Farouq	Encadreur/rapporteur	UKM Ouargla
Melle TOUMI Chahrazad	C.Encadreur	UKM Ouargla
M. BOUKHAMALA Akram	Examinateur	UKM Ouargla

ANNEE UNIVERSITAIRE : 2016/2017

Dédicace

Je dédie ce modeste ouvrage a mon plus cher être au monde parents dont la patience n'a égal que leur grandeur d'âme.

A mes sœurs, mes frères.

A mon marie qui a m'aidé beaucoup avec ses encouragements.

A mon bébé Mohammed Abdelkafi

A tous mes amis (es) et particulièrement à ma chère binôme Ahlam et sa famille

A tous ceux qui m'ont aidé.

Zohra

Dédicace

Je dédie ce modeste travail :

A mes parents pour tous les efforts et sacrifices consentis ;

A mon frère, à mes sœurs, à mes belles sœurs, à mon beau frère et à toute ma famille;

A mes enseignants ; A Tous ceux qui m'ont connue, soutenue et aimée ;

A tous mes amis (es) et camarades;

A toux ceux qui m'ont aidée;

Et tout particulièrement à ma chère binôme Zohra et à toute sa famille pur que notre amitié dure et se consolide.

Ahlam

Remerciments

Tout d'abord nous nous inclinons humblement et avec le plus grand respect devant « Dieu » : le tout puissant qui nous avons donné la volonté et la force d'aller d'avant.

Pour cela, nous faisons part de notre reconnaissance a notre Encadreurs Mr ZITOUNI Farouq et Melle TOUMI Chahrazad

qui ont nous guidés et ont su nous orienter chaque fois que c'était nécessaire, Que messieurs les membres de jury présents a notre soutenance, trouvent ici le témoignage de notre profond respect.

Enfin nous remercions tous ceux qui nous aidés.

Table des matières

Introduction générale.....

Chapitre 1: Concepts généraux en Recherche d'Information

Introduction ..... *Erreur ! Signet non défini.**16***

1 La recherche d'information ..... *Erreur ! Signet non défini.**16***

1.1 Définitions .....Erreur ! Signet non défini.**16**

1.2 Concepts de base de la recherche d'information .....Erreur ! Signet non défini.**17**

1.3 Les modèles de recherche d'information.....Erreur ! Signet non défini.**17**

2 Les systèmes de recherche d'informations ..... *Erreur ! Signet non défini.**19***

2.1 Définition.....Erreur ! Signet non défini.**19**

2.2 Notions de document et de requête .....Erreur ! Signet non défini.**19**

2.2.1 Document ..... **Erreur ! Signet non défini.**19

2.2.2 Requête..... **Erreur ! Signet non défini.**20

2.3 Principales phases du processus de recherche d'information...Erreur ! Signet non défini.**20**

2.3.1 L'indexation..... **Erreur ! Signet non défini.**20

2.3.2 Fonction de pondération ..... **Erreur ! Signet non défini.**20

3 Recherche d'information sur le Web ..... *Erreur ! Signet non défini.**21***

3.1 Outils de recherche d'information .....Erreur ! Signet non défini.**21**

4 Prétraitement linguistiques ..... *Erreur ! Signet non défini.**21***

4.1 Segmentation .....Erreur ! Signet non défini.**22**

4.2 Normalisation .....Erreur ! Signet non défini.**23**

4.2.1 Normalisation textuelle ..... **Erreur ! Signet non défini.**23

4.2.2 Normalisation linguistique..... **Erreur ! Signet non défini.**24

4.2.3 Filtrage par un anti dictionnaire..... **Erreur ! Signet non défini.**25

5 Les deux lois de base en recherche d'information..... *Erreur ! Signet non défini.**26***

5.1 Loi de Heaps .....Erreur ! Signet non défini.**26**

5.2 Loi de Zipf.....Erreur ! Signet non défini.**27**

5.3 Représentation documentaire .....Erreur ! Signet non défini.**28**

5.4 Pondération des termes .....Erreur ! Signet non défini.**29**

5.5 Index inversé .....Erreur ! Signet non défini.**32**

5.6 Indexation dans des collections statiques .....Erreur ! Signet non défini.**33**

5.7 Indexation par bloc à base de tri.....Erreur ! Signet non défini.**33**

5.8 Indexation distribuée .....Erreur ! Signet non défini.**34**

5.9 Indexation dans des collections dynamiques.....Erreur ! Signet non défini.**34**

Conclusion.....*Erreur ! Signet non défini.**35***

Chapitre 2 : Classification et catégorisation

Introduction.....*Erreur ! Signet non défini.**37***

1 Classification et catégorisation de documents.....*Erreur ! Signet non défini.**37***

2 Méthodologie.....*Erreur ! Signet non défini.**37***

3 Méthodes algorithmiques.....*Erreur ! Signet non défini.**37***

4 Sélection de variables.....*Erreur ! Signet non défini.**39***

5 Le seuillage sur la mesure Document Frequency (df).....*Erreur ! Signet non défini.**40***

5.1 L'information mutuelle ponctuelle (IMP).....Erreur ! Signet non défini.**40**

5.2 L'information mutuelle (IM).....Erreur ! Signet non défini.**41**

5.3 Les Modèles d'apprentissage.....Erreur ! Signet non défini.**42**

5.3.1 Modèles génératifs ..... **Erreur ! Signet non défini.**42

5.3.2 Modèle multinomial..... **Erreur ! Signet non défini.**43

5.4 Modèles discriminants.....Erreur ! Signet non défini.**45**

5.5 Modèle logistique.....Erreur ! Signet non défini.**48**

6 Mesures d'évaluation.....*Erreur ! Signet non défini.**49***

Conclusion.....*Erreur ! Signet non défini.**51***

Chapitre 3: Conception du système

Introduction.....*Erreur ! Signet non défini.**53***

1 Architecture générale du système.....*Erreur ! Signet non défini.**53***

1. Prétraitements.....*Erreur ! Signet non défini.**54***

2. Calcul de similarité et création des classes.....*Erreur ! Signet non défini.**54***

3. Apprentissage.....*Erreur ! Signet non défini.**55***

4. Conception du système.....*Erreur ! Signet non défini.**55***

4.1 Les points forts d'UML.....Erreur ! Signet non défini.**55**

4.2 Diagramme de cas d'utilisation.....Erreur ! Signet non défini.**56**

4.3 Diagramme de séquences.....Erreur ! Signet non défini.**56**

4.4 Diagramme de classe.....Erreur ! Signet non défini.**58**

4.5 Diagramme d'activités.....Erreur ! Signet non défini.**59**

Conclusion.....*Erreur ! Signet non défini.**60***

Chapitre 4: Réalisation du système

Introduction ***..... Erreur ! Signet non défini.61***

1 Présentation des outils de développement..... Erreur ! Signet non défini.61

1.1 Plateforme matérielle ..... Erreur ! Signet non défini.**61**

1.2 Plateforme logicielle ..... Erreur ! Signet non défini.**61**

1.2.1 Langage JAVA ..... **Erreur ! Signet non défini.**61

1.2.2 Environnement de développement Eclipse ..... **Erreur ! Signet non défini.**62

1.2.3 Weka ..... **Erreur ! Signet non défini.**62

1.2.4 Langage XML ..... **Erreur ! Signet non défini.**63

2 Descriptions du corpus utilisé ..... Erreur ! Signet non défini.64

3 Description du système..... Erreur ! Signet non défini.64

5. Expérimentation et Résultats..... Erreur ! Signet non défini.75

Conclusion..... Erreur ! Signet non défini.75

Conclusion Générale et Perspectives ..... Erreur ! Signet non défini..

Bibliographie .....

Listes des figures

Figure 1 : Architecture du système de la recherche d'informations......5
Figure 2: Constitution du vocabulaire, index inversé des termes et représentation des documents dans l'espace des termes pour une collection de documents donnée. 8
Figure 3: Illustration de la loi de Heaps sur la collection du Wikipédia français. 13
Figure 4: Illustration de la loi de Zipf sur la collection du Wikipédia français (gauche).Pour plus de visibilité nous avons utilise une double échelle logarithmique, ou In est le logarithme népérien. La droite qui interpole le mieux les points, au sens des moindres carrés......14
Figure 5 : Création de l'index inversé pour une collection statique constitué de 3documents. 19
Figure 6 : illustration de l'indexation distribuée. Nous supposons ici que l'indexation de la collection a lieu lorsque cette dernière ne subit pas de modifications dans le temps. 21
Figure 7 : Catégorisation de documents en deux phases :a) entrainement d'un classifieur à partir d'une collection de documents étiquetés et b) prédiction des étiquettes de classe des documents d'une base test avec le classifieur appris......24
Figure 8 : Trois fonctions de cout pour un problème de catégorisation à deux classes en fonction du produit $c \times h$, ou h est la fonction de cout sont l'erreur de classification. 33
Figure 9 : L'architecture générale du système. 39
Figure 10 : Diagramme de cas d'utilisation......42
Figure 11 : Diagramme de séquence du cas d'utilisation « Ajouter un texte »......43
Figure 12 : Diagramme de classes. 44
Figure 13 : Diagramme d'activités représentant la création des classes du corpus......45
Figure 14 : Diagramme d'activités représentant les traitements subit par un nouveau document......45
Figure 15 : Interface d'accueil de système. 51
Figure16 : Paramètre d'ajustement du classifieur. 51
Figure 17 :** Chargement du corpus avec succès**......52
Figure 18 : Menu de système comment chargé le calcul tf-idf. 52
Figure 19 : Message de chargement de vecteurs de pondération......53
Figure 20 : Interface d'apprentissage avant le nettoyage. 53
Figure 21 : Interface d'apprentissage après le nettoyage. 54
Figure 22 : Vecteurs de termes pondérés du corpus......54
Figure 23 : Interface de similarités entre les documents......55
Figure 24 : Le corpus avant le nettoyage. 55
Figure 25 : le corpus après le nettoyage. 56
Figure 26 : Pondération des vecteurs de termes......56
Figure 27 : Les similarités entre les documents. 57
Figure 28 : Taux de classification avec la formule tf-idf......57
Figure29 : Interface affiche la classification du document. 58
Figure 30 : L'affichage de classe du document. 59
Figure 31 : Interface de prétraitement d'un nouveau document. 59
Figure 32 : les résultats de calculs...... **Erreur ! Signet non défini.**60

.

Liste des tables

Tab 1 : Les mots les plus fréquents (et peu informatifs) présent dans la collection du Wikipédia français. ----------12
Tab 2: Quelques statistiques sur la collection de Reuters-RCV2----------25
Tab 3: Tableau de contingence comptabilisant les nombres de documents appartenant, ou non, à la classe c et contenant, ou non, le terme t d'une base d'entrainement de taille m.----------26
Tab 4: Tableau de contingence indiquant les nombres de bonnes et de mauvaises prédictions. --------35
Tab 5: Description des classes.----------44
Tab 6 : Répartition des documents du corpus utilisé. ----------50

RESUME

La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information, elle est aujourd'hui un champ pluridisciplinaire, intéressant même les sciences cognitives, elle a plusieurs domaines parmi ces derniers la catégorisation qui connait des améliorations et des innovations importantes dont le but est de faire apprendre à une machine comment attribuer un texte à la bonne catégorie. Face à l'accroissement du volume d'information disponible en ligne, la catégorisation automatique de texte s'impose de plus en plus comme une étape essentielle qui permet d'accélérer, cibler et d'améliorer la recherche d'information.

Dans ce mémoire nous présentons un système de prédiction de thématiques d'un document donné en entrée. Nous disposons Pour cela d'un ensemble de textes pour lesquels la catégorie est connue (corpus d'apprentissage et qui nous servent à entrainer un classifieur qu'il sera ensuite testé sur d'autres documents auxquels la catégorie est connue également (corpus de test). L'objectif de notre approche est d'améliorer les performances et l'efficacité des systèmes de recherche et de classification de textes pour mieux cibler l'information pertinente basant sur le pouvoir discriminatif des termes. Faire. A la fin nous obtiendrons un système qui est capable de prédire les thèmes enfouis dans un document donné.

Dans notre travail nous avons proposé d'adapter une formule de pondération à ce qui a permis de réduire le temps de calcul de vecteur de pondération et d'améliorer significativement le taux de classification.

Mots Clés: RI, SRI ,TF-IDF,SVM, classification automatique de texte, représentation vectorielle, mesures de similarité, Apprentissage.

Abstract

Information retrieval is a branch of informatics that is concerned with the acquisition, organization, storage, research and selection of information. Today, it is a multidisciplinary field, interesting even in the sciences Cognitive, it has several areas among these categorization which is experiencing major improvements and innovations whose aim is to teach a machine how to assign a text to the right category. Given the increasing volume of information available online, automatic text categorization is increasingly becoming an essential step in accelerating, targeting and improving information retrieval.

In this paper we present a system of predicting themes of a given document as input. For this purpose, we have a set of texts for which the category is known (learning corpus and which are used to train a classifier that it will then be tested on other documents to which the category is also known (corpus test) The objective of our approach is to improve the performance and efficiency of text search and classification systems in order to better target relevant information based on the discriminative power of terms. System that is capable of predicting embedded themes in a given document.

In our work we proposed to adapt a weighting formula to what has allowed to reduce the time of computation of vector of weighting and to improve significantly the rate of classification.

Keywords: RI, SRI, TF-IDF, SVM, automatic text classification, vector representation, similarity measurements, Learning.

ملخـــــص

البحث عن المعلومات هو فرع من فروع علم الحاسوب الذي يركز على اكتساب وتنظيم وتخزين والبحث واختيار المعلومات، هو الآن حقل متعدد التخصصات، للاهتمام حتى العلم المعرفي، لديها العديد من المناطق بين التصنيف الأخير الذي يعرف التحسينات والابتكارات الهامة التي تهدف لتعليم آلة كيفية تعيين النص إلى الفئة الصحيحة. وفي مواجهة تزايد حجم المعلومات المتاحة على الإنترنت، التصنيف التلقائي للنص أصبحت أكثر وأكثر خطوة أساسية أن يسرع، والتركيز، وتحسين استرجاع المعلومات.

في هذه الورقة نقدم نظام للتنبؤ مواضيع المدخلات وثيقة معينة. لدينا ذلك لمجموعة من النصوص التي يعرف الفئة (الإحضار التدريب والتي نستخدمها لتدريب المصنف من شأنها أن يكون من الممكن اختبارها على الوثائق الأخرى التي يعرف الفئة أيضا (اختبار الإحضار). والهدف من نهجنا هو تحسين أداء وكفاءة نظم البحوث وتصنيف النص إلى أفضل المعلومات المستهدفة ذات الصلة على أساس القوة التمييزية للمصطلحات. ل. في النهاية نحن سوف تحصل على نظام قادر على التنبؤ المواضيع دفن في وثيقة معينة.

في عملنا اقترحنا على التكيف مع صيغة الترجيح لما تقليل الوقت الحوسبة الوزن النواقل وتحسن ملحوظ في معدل التصنيف.

كلمات البحث: RI، SRI، TF-IDF-ا، SVM، تصنيف التلقائي النص والتمثيل ناقلات والتدابير التشابه، التعلم.

INTRODUCTION GÉNÉRALE

Aujourd'hui l'information joue un rôle primordial dans le quotidien des individus et dans l'essor des entreprises. Cependant, le développement de l'Internet et la généralisation de l'informatique dans tous les domaines ont conduit à la production d'un volume d'information sans précédent. En effet, la quantité d'information disponible particulièrement à travers le web, se mesure en milliards de pages.il est par conséquent, de plus difficile de localiser précisément ce que l'on recherche dans cette masse d'information. La recherche d'information (RI) est le domaine par excellence qui s'intéresse à répondre à ce type d'attente. En effet, l'objectif principal de la RI est de fournir des modèles, des techniques, et des outils pour stocker et organiser des masses d'informations et localiser celles qui seraient pertinentes relativement à un besoin en information d'un utilisateur, souvent exprime à travers une requête. A cet ère, il est devenu plus astreignant aux utilisateurs de trouver des informations pertinentes satisfont leurs besoins, de conserver et de classer ces informations. Afin d'aider les utilisateurs à accéder au contenu désiré la classification de texte est de ressembler les textes similaires selon un certain critère, au sein d'une même classe. Deux types d'approches de classification automatique peuvent être distingués :

La classification supervisée et la classification non supervisée. Ces deux méthodes diffèrent sur la façon dont les classes sont déterminées automatiquement par la machine, par contre, dans l'approche supervisée, la classification de texte consiste à affecter à un texte à une catégorie prédéfinie par un expert. Ces catégories peuvent être par exemple le sujet du texte, son thème, l'opinion qu'il l'exprime, etc. Nous disposons pour cela d'un ensemble de textes pour lesquels la catégorie est connue (corpus d'apprentissage) et qui nous servent à entrainer notre classifieur qu'il sera ensuite, testé sur d'autres documents auxquels la catégorie est connue également (corpus de test).

Dans ce mémoire intitulée : ≪ **Un système de Classification et de recherche de documents textuels de la langue Anglaise »,** nous essayons de réaliser une classification automatique de documents. Pour cela, nous avons utilisé un corpus d'apprentissage téléchargé depuis le site de partage de publications scientifiques Citeulike.

Organisation du mémoire

Ce mémoire est organisé de la façon suivante :

Une introduction générale présente le conteste général du mémoire nos objectifs et nos motivations, le premier chapitre, expose un état de l'art sur les concepts généraux en recherche d'information (RI) que ce soit sa définition ou son rôle, le deuxième chapitre consiste à présenter la classification, le troisième chapitre est consacré à l'étude conceptuelle, le quatrième chapitre décrit les étapes d'implémentation, les outils utilisés ainsi que les résultats et les expérimentions réalisées.

En fin nous terminons par une conclusion et des perspectives.

CHAPITRE I
CONCEPTS GÉNÉRAUX EN RECHERCHE D'INFORMATION

Introduction

Plusieurs tache se regroupent sous le vocable de la RI, la plus ancienne est la recherche documentaire, on y trouve également d'autres taches plus au moins récentes comme : le filtrage d'information, l'extraction d'information, la recherche d'information multilingue les questions réponses, la recherche d'information sur le web, etc.
Dans ce chapitre nous allons définir La recherche d'information les aspects de cette dernières. Après, nous allons passer en revue l'ensemble des prétraitements qui aboutit à la création du vocabulaire. Nous présenterons ensuite les deux lois de base :(loi de Heaps) et (loi de Zipf), puis nous présenterons successivement le model .Nous allons aussi y traiter en particulier deux algorithmes de compression du vocabulaire et de l'index.

1 La recherche d'information

La recherche d'information est un domaine historiquement lié aux sciences de l'information et à la bibliothéconomie qui ont toujours eu le souci d'établir des représentations des documents dans le but d'en récupérer des informations à travers la construction d'index. L'informatique a permis le développement d'outils pour traiter l'information et établir la représentation des documents au moment de leur indexation, ainsi que pour rechercher l'information [3].

1.1 Définitions

Plusieurs définitions de la recherche d'information ont vu le jour dans ces dernières années, nous citons dans ce contexte les trois définitions suivantes:

- ***Définition 1*** *:* La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations [2].

- ***Définition 2*** : La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information [3].

- ***Définition 3*** *:* La recherche d'information est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information[3].

1.2 Concepts de base de la recherche d'information

La recherche d'information est considérée comme l'ensemble des techniques permettant de sélectionner à partir d'une collection de documents, ceux qui sont susceptibles de répondre aux besoins de l'utilisateur. A cet effet une synthèse des travaux nous a permis de dégager les concepts suivants [5]:

Collection de documents, Document, Besoin d'information.

1.3 Les modèles de recherche d'information

Un modèle de RI a pour rôle de fournir une formalisation du processus de RI et un cadre théorique pour la modélisation de la mesure de pertinence. Ces modèles ont en commun le vocabulaire d'indexation basé sur le formalisme mots clés et diffèrent principalement par le modèle d'appariement requête-document. Le vocabulaire d'indexation V = {ti}, i ∈ {1, ..., n} est constitué de n mots ou racines de mots qui apparaissent dans les documents. Un modèle de RI est défini par un quadruplet (D, Q, F, R (q, d)) : où
– D est l'ensemble de documents

– Q est l'ensemble de requêtes

– F est le schéma du modèle théorique de représentation des documents et des requêtes

– R (q, d) est la fonction de pertinence du document d à la requête q Nous présentons dans la suite les principaux modèles de RI : le modèle booléen, le modèle vectoriel et le modèle probabiliste [1].

a- Modèle booléen

Le modèle booléen est basé sur la théorie des ensembles. Dans ce modèle, les documents et les requêtes sont représentés par des ensembles de mots clés. Chaque document est représenté par une conjonction logique des termes non pondérés qui constitue l'index du document. Un exemple de représentation d'un document est comme suit : d = $t_1 \wedge t_2 \wedge t_3 \ldots \wedge t_n$tn.

Une requête est une expression booléenne dont les termes sont reliés par des opérateurs logiques (OR, AND, NOT) permettant d'effectuer des opérations d'union, d'intersection et de différence entre les ensembles de résultats associés à chaque terme. Un exemple de représentation d'une requête est comme suit : " q = $(t_1 \wedge t_2) \vee (t_3 \wedge t_4)$ " La fonction de correspondance est basée sur l'hypothèse de présence/absence des termes de la requête

dans le document et vérifie si l'index de chaque document dj implique l'expression logique de la requête q. Le résultat de cette fonction est donc binaire est décrit comme suit : RSV (q, d) = {1,0} [3].

b- Modèle vectoriel

Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est définie par des mesures de distance dans un espace vectoriel. Le modèle vectoriel représente les documents et les requêtes par des vecteurs d'un espace à n dimensions, les dimensions étant constituées par Les termes du vocabulaire d'indexation.

L'index d'un document dj est le vecteur $= (W_1, j, W_2, j, W_3, j, \dots, W_n, j))$, où $W_{k,j} \in [0\ 1]$. dénote le poids du terme t_k dans le document dj. Une requête est également représentée par unvecteur $= (W_1, q, W_2, q, W_3, q, \dots, W_n, q)$, où w_k, q est le poids du terme t_k dans la requête q.

La fonction de correspondance mesure la similarité entre le vecteur requête et les vecteurs documents. Une mesure classique utilisée dans le modèle vectoriel est le cosinus de l'angle formé par les deux vecteurs : RSV (q, d) = cos q d [2].

c- Modèle probabiliste

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête. Le principe de base consiste à retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents. Etant donné une requête utilisateur Q et un document D, il s'agit de calculer la probabilité de pertinence du document pour cette requête.

Deux possibilités se présentent : R, D est pertinent pour q et D, n'est pas pertinent pour q. Les documents et les requêtes sont représentés par des vecteurs booléens dans un espace à n dimensions. Un exemple de représentation d'un document dj et une requête q est le suivant : $\mathbf{d}_j = (W_1, j, W_2, j, W_3, j, \dots, W_n, j)$,q), $\mathbf{q} = (W_1, q, W_2, q, W_3, q, \dots, W_n, q)$ Avec $W_{k,j} \in [0\ 1]$ et $W_{k,q} \in [0\ 1]$ La valeur de $W_{k,j}$ (resp. W_k q) indique si le terme "# apparaît ou non dans le document $\mathbf{d}_j$ (resp. q) [1].

Le modèle probabiliste évalue la pertinence du document $\mathbf{d}_j$ pour la requête q. Un document est sélectionné si la probabilité que le document d soit pertinent, notée p(R/D), est supérieure

à la probabilité que d soit non pertinent pour q, notée p(R/D) où R est l'événement de pertinence et est l'événement de non pertinence.

Le score d'appariement entre le document d et la requête Q, noté RSV (Q, D) est donné par : $(,) = \frac{(/)}{(/)}$

Ces probabilités sont estimées par de probabilités conditionnelles selon qu'un terme de la requête est présent, dans un document pertinent ou dans un document non pertinent [3].

2 Les systèmes de recherche d'informations

2.1 Définition

Un Système de Recherche d'Informations (SRI) est un système informatique qui permet de retourner à partir d'un ensemble de documents, ceux dont le contenu correspond le mieux à un besoin en informations d'un utilisateur, exprimé à l'aide d'une requête [2].

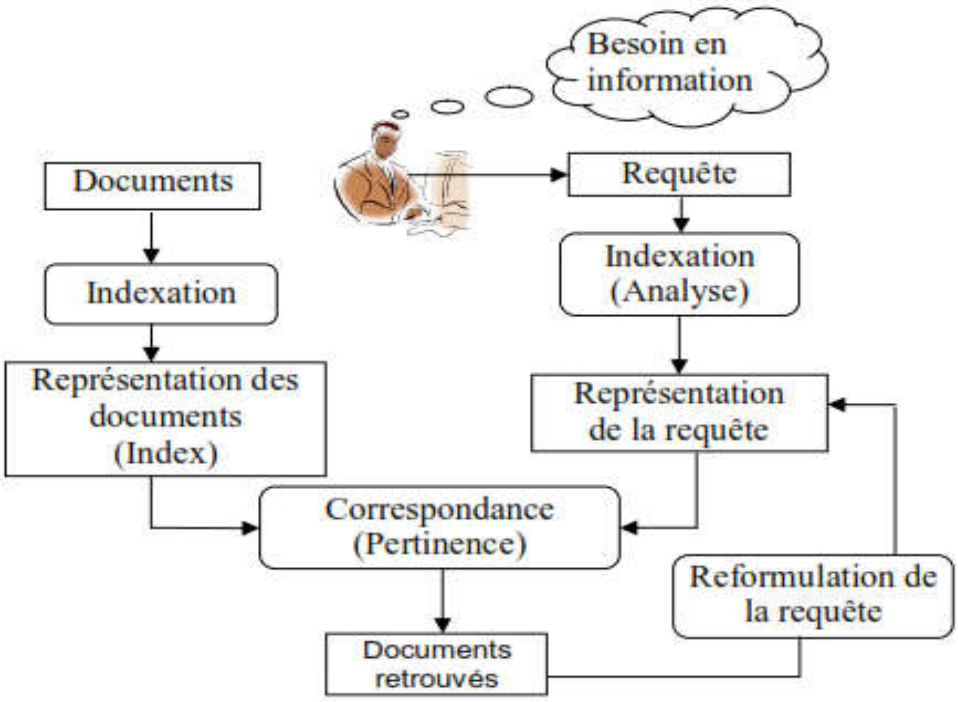


Figure 1 : *Architecture du système de la recherche d'informations.*

2.2 Notions de document et de requête

2.2.1 Document

Le document représente le conteneur élémentaire d'information, exploitable et accessible par le SRI. Un document peut être un texte, une page WEB, une image, une bande

vidéo, etc. Dans notre contexte, nous appelons document toute unité qui peut constituer une réponse à un besoin en information exprimé par un utilisateur [2].

2.2.2 Requête

Une requête constitue l'expression du besoin en informations de l'utilisateur. Plusieurs systèmes utilisent des langages différents pour décrire la requête [3].

2.3 Principales phases du processus de recherche d'information

L'objectif fondamental d'un processus de RI est de sélectionner les documents ''les plus proches'' du besoin en information de l'utilisateur décrit par une requête. Ceci induit deux principales phases dans le déroulement du processus : indexation et appariement requête/documents.

2.3.1 L'indexation

L'indexation est une étape très importante dans le processus de RI. Elle consiste à déterminer et extraire les termes représentatifs du contenu d'un document ou d'une requête, qui couvrent au mieux leur contenu sémantique [3].

Les groupes de mots forme ce que l'on appelle un thesaurus. Ce dernier inclut des relations de type linguistiques (équivalence, association, hiérarchisation) et statistiques (pondération) .L'indexation peut être caractérisée par son mode et fonction de pondération [3].

- **Le Mode d'indexation :**

L'indexation peut être manuelle, automatique ou semi-automatique.

2.3.2 Fonction de pondération

La pondération permet d'affecter à chaque terme d'indexation une valeur qui mesure son importance dans le document où il apparaît. Pour trouver les termes du document qui représentent le mieux son contenu sémantique, a défini la fonction de pondération d'un terme dans un document connue sous la forme de Tf.Idf, qui est reprise dans différentes versions par la majorité des SRI On y distingue :

- *Tf* (*tem fréquence*) : cette mesure est proportionnelle à la fréquence du terme dans le document. L'idée sous-jacente est que plus un terme est fréquent dans un document, plus il est important dans la description de ce document. Le Tf est souvent exprimé selon l'une des

déclinaisons suivantes [3] : a. *Tf* : utilisation brute b. $\mathbf{0.5 + 0.5}\frac{\boldsymbol{Tf}}{\boldsymbol{Max(Tf)}}$. .

- *Idf* (*Inverse of Document Frequency*) : mesure l'importance d'un terme dans toute la collection. L'idée sous-jacente est que les termes qui apparaissent dans peu de documents de la collection sont plus représentatifs du contenu de ces documents que ceux qui apparaissent dans tous les documents de la collection. Cette mesure est exprimée selon l'une des déclinaisons suivantes [1] : a. $\boldsymbol{Idf = \log\left(\frac{N}{df}\right)}$ b. $\mathbf{Idf = \log\left(\frac{N-df}{df}\right)}$.

3 Recherche d'information sur le Web

Est le domaine qui étudie la manière de retrouver des informations dans un corpus. Celui-ci est composé de documents d'une ou plusieurs bases de données, qui sont décrits par un contenu ou les métadonnées associées. Les bases de données peuvent être relationnelles ou non structurées, telles celles mises en réseau par des liens hypertexte comme dans le World Wide Web, l'internet et les intranets. Le contenu des documents peut être du texte, des sons, des images ou des données [2].

3.1 Outils de recherche d'information

Il existe de nombreux outils de recherche d'information sur le Web, ces outils qui se spécialisent en fonction des services utilisés et du type d'information qu'ils recensent. Il convient en effet de distinguer différents types d'outils de recherche sur l'Internet [1].

4 Prétraitement linguistiques

Dans cette partie nous allons présenter l'ensemble des prétraitements linguistiques conduisant à la constitution de l'index inversé et de la représentation documentaire à partir d'une collection de documents donnée. Ces prétraitements comprennent la segmentation des séquences de caractères présents dans les documents en mots distincts, et le filtrage qui vise à supprimer les mots les plus fréquents.ces processus sont schématisés par le cube grisé dans la figure 1.2 [1]

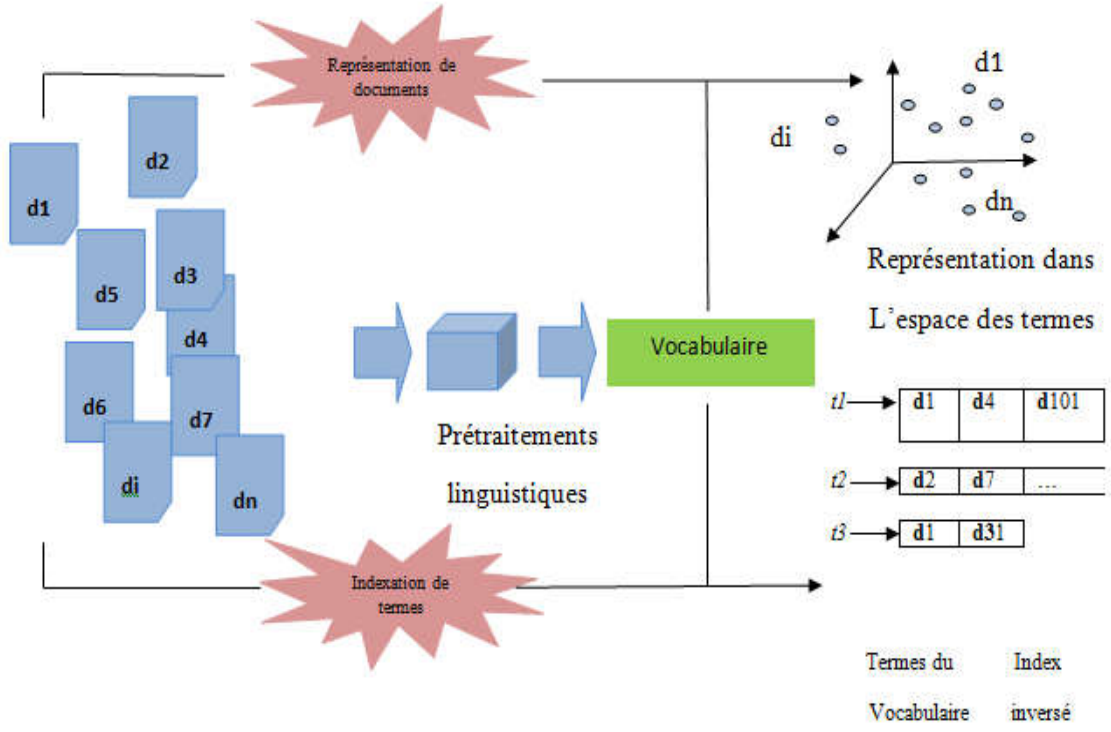


Figure 2: *Constitution du vocabulaire, index inversé des termes et représentation des documents dans l'espace des termes pour une collection de documents donnée.*

4.1 Segmentation

La segmentation (*en anglais tokenisation*) consiste à séparer une suite de caractères en éléments sémantiques, ou mots. Un type de mot est la classe de tous les mots ayant la même séquence de caractères et un terme est un type de mots que l'on garde pour former le vocabulaire. Dans l'exemple suivant :

Au sens qui, l'information est ce donne une forme à l'esprit

Nous avons 14 mots :

Au, sens, étymologique, l', information, est, ce, qui, donne, une, forme, à, l', esprit

Mais seulement 13 types, puisqu'il y a deux instances de {l'}. Apres filtrage par un anti dictionnaire, il ne restera plus que 6 termes {*sens, étymologique, information, donne, forme, esprit*} dans le vocabulaire. Cette étape de segmentation est une étape difficile et cruciale puisqu'une mauvaise segmentation a un impact négatif direct sur le résultat de la recherche. En effet, si dans une collection certains termes ne sont pas indexés, les requêtes composées de ces termes ne pourront jamais être appariés avec les documents de la collection contenant ces termes. Une bonne segmentation dépend de la prise en compte des spécificités de la langue des textes traités .Par exemple, pour le français, nous avons :

- Les composés lexicaux à trait d'union comme *chassé-croisé, peut-être, rendez-vous*, etc.
- Les expressions idiomatiques comme *au fait, poser un lapin, tomber dans les pommes,* Les formes contractées comme *Gad'zart(les gars des Arts et Métiers), M'esieur, j', etc.*
- Les sigles et les acronymes comme *K7, A.R, CV, P-V, etc.*

4.2 Normalisation

La normalisation de mots est le processus qui transforme tous les mots d'une même famille sous une forme normale ou canonique de façon a ce que l'appariement entre les termes d'une requête et ceux de l'index puissent avoir lieu, et ce malgré les différences qui peuvent exister entre les séquences de caractères de leurs mots assoies[1].

4.2.1 Normalisation textuelle

La normalisation textuelle rend les mots d'une même famille sous leur forme canonique en effectuant quelques transformations superficielles sur les séquences de caractères de ces mots. L'exemple précédent avec P.V. constitue un cas de normalisation textuelle.

- ✓ ***Les ponctuations***

La règle de base appliquée à l'exemple ci-dessus, et qui concerne tous les acronymes, est d'enlever les points et les traits d'union apparaissant dans les mots.

- ✓ ***La case***

Une stratégie classique est de réduire certaines lettres en minuscules. Elle ne peut pas être généralisée à cause de certains noms propres (Blanc, Noir) et des acronymes (ACE-Association des cinémathèques européennes, CAS-Centre d'analyse stratégique) dont la réduction des majuscules peut conduire à les confondre avec des noms communs qui s'écrivent pareillement (CAS⟶ cas).

- ✓ ***Les accents***

L'impact des accents et des diacritiques sur le résultat de la recherche est souvent marginal. La règle appliquée consiste généralement à enlever les diacritiques sur tous les mots (par exemple ambigüe devient ambigue ou forêt qui devient foret).

- ✓ ***Les dates et les valeurs monétaires***

Les règles pour normaliser ces types de mots sont assez spéciales et dépendent de la langue et des usages particuliers de chaque pays. Pour certaines langues comme l'anglais d'Amérique du Nord ,le problème des dates est même plus compliqué puisque dans les documents d'une collection nous pouvons rencontrer des occurrences de année/mois/jours ou mois/jour/année pour une même date [1][15].

4.2.2 Normalisation linguistique

La normalisation linguistique consiste à ramener un mot fléchi sous sa forme canonique. Il existe deux types de normalisation linguistiques: la racinisation et lemmatisation.

- ➢ ***La racinisation***

La racinisation se rapporte au procédé qui cherche à regrouper les différentes variantes morphologiques d'un mot auteur d'une même racine(en anglais stem).Ce procédé repose sur un ensemble de règles pour supprimer les flexions et les suffixes des mots, et il est fortement dépendant de la langue utilisée. Par exemple, les mots, français suivant ont la même racine morphologique (en gras) [1] :

Chanter, chante, chantes, chantent, chanté, chantée, chants ⇨ ***chant***

La racinisation peut aussi ramener des mots à des formes qui ne sont pas de vrais mots :

Cheval, chevaux, chevalier, chevalerie, chevaucher ⇨ **cheva**

Les premiers algorithmes de racinisation, appelés stemmer, ont été étudié à partir de la fin des années 1960. Ces stemmers sont conçus pour l'anglais et ils procèdent en traitant les mots par des étapes prédéfinies. A une étape donnée, on détermine si la terminaison d'un mots se trouve dans une liste prédéfinie le cas échéant, le stemmer la supprime ou la transforme suivant la règle à appliquer .D'autres algorithmes de racinisation pour d'autres langues ont été conçus suivant le principe pour le français, on peut par exemple citer [7] :

- Le stemmer Snowball, disponible sur Internet et dont les règles de racinisation sont expliquées sur la page de téléchargement[3] En RI, le but de la racinisation est de trouver les documents contenant toutes les formes fléchies des mots d'une requête .En contrepartie, le risque est qu'on peut aussi ramener des documents qui contiennent les mots ayant les mêmes racines que les mots de la requête mais qui sont sémantiquement totalement différents de ces derniers [stemmer] [6].

Par exemple, si la requête porte sur les chevaux, en considérant les documents contenant le mot normalisé ***cheva***, on sait qu'on aura une bonne couverture du sujet, puisqu'on obtiendra les documents contenant toutes les occurrences des mots *cheval* et *chevaux*.

Par contre, dans la liste des documents retournés, figureront également ceux qui contiennent les mots *chevaucher, chevalerie* et *chevaliers* qui pourront être un peu éloignés du sujet recherché.

- ***La lemmatisation***

La lemmatisation fait une analyse linguistique poussée destinée à enlever les variantes flexionnelles des mots afin de les ramener sous leur forme lemmatisée ou encyclopédique. Pour les verbes, cette transformation fournit la forme à l'infinitif, pour les adjectifs la forme masculin singulier et pour les noms la forme singulier5. A l'opposé de la racinisation, le résultat de la lemmatisation n'est autre que des mots de l'encyclopédie, elle ne conduit ainsi pas à l'agrégation de mots très différents [2].

4.2.3 Filtrage par un anti dictionnaire

Un anti dictionnaire (Stopword list en anglais), ou liste de mots vides, est une liste de mots qui tendent à être présents avec une fréquence élevée dans tous les documents d'une collection et qui n'apportent que peu d'information sur le contenu d'un document. Le filtrage par un anti dictionnaire est très pratiqué dans les taches de catégorisation (chapitre2) .En effet, la distribution des mots vides est identique dans tous les documents d'un corpus, ceci quelles que soient les thématiques auxquelles ils appartiennent, et les mots vides n'ont ainsi pas un pouvoir discriminant élevé pour ces taches. L'idée de ce filtrage vient de (Luhn 1958), un des pionniers en RI [1] [8].

Il existe des anti-dictionnaires indépendants du domaine de la collection que l'on considère et d'autres qui en sont dépendants.

De	La	Le	Et	En	L'	Du
Des	D'	Les	Est	Un	une	Il
Au	Dans	Par	Pour	Sur	date	A
Qui	Que	Avec	Son	Plus	Se	Sans
Quel	Quelle	S'	Pas	N'	Je	Y
Ou	Se	Sont	Aux	Qu'	Sa	Elle

***Tab 1** : Les mots les plus fréquents (et peu informatifs) présent dans la collection du Wikipédia français.*

Une fois cette liste des mots constituée, on filtre les documents du corpus en enlevant les mots présents dans cet anti dictionnaire. Le tableau 1.1 montre un exemple d'anti dictionnaire relatif à la collection du Wikipédia français. Après le filtrage des documents par anti dictionnaire le filtrage des documents par anti dictionnaire, l'ensemble des termes restants constituent le vocabulaire de la collection, qui sera ensuite utilisé pour représenter les documents dans un espace vectoriel [1].

5 Les deux lois de base en recherche d'information

5.1 Loi de Heaps

La loi de (Heaps 1978) stipule que la taille du vocabulaire (V) croit exponentiellement en fonction du nombre de mots présents dans une collection donnée (M).Autrement dit :

$$V = K \times M^{\beta}$$

Ou K et B sont des paramètres dépendant de la collection considérée. La figure 2.2 montre l'évolution exponentielle de la taille du vocabulaire en fonction du nombre de mots de la collection de Wikipédia .En utilisant une régression des moindres carrés, nous avons trouvé que les paramètres de la loi sont, dans ce cas, approximativement égaux à $K \approx 95$ et $\beta \approx 0.43$

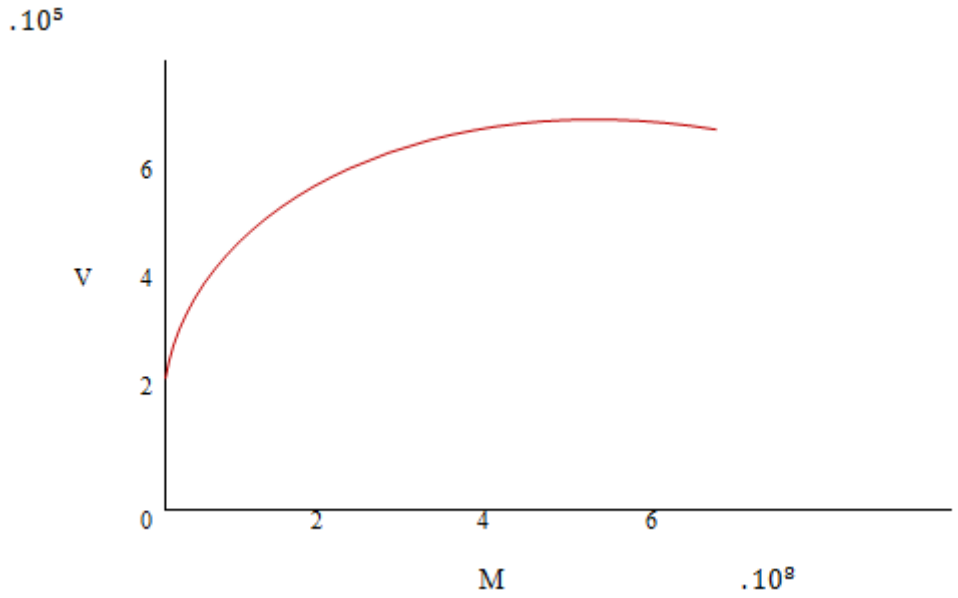


Figure 3: *Illustration de la loi de Heaps sur la collection du Wikipédia français.*

Le paramètre b dépend de la langue de la collection mais le paramètre K dépend beaucoup des prétraitements appliqués à une collection pour extraire les mots et les termes .En effet, une mauvaise segmentation a une incidence directe sur les mots que L'on extrait (et donc sur leur nombre) et des stratégies de normalisation vont bien réduire le nombre de types de mots (d'à peu prés 21% dans notre cas) [1].

Quelques que soient les valeurs de ces paramètres, la loi de Heaps suggère que la taille du vocabulaire croit au fur et à mesure que la collection croit. Dans le cas de la collection de Wikipédia, les nouvelles pages amènent inexorablement leurs lots de nouveaux noms propres, de noms de villes, de termes scientifique, de noms d'entreprises, de produits etc qui seront ajoutés au vocabulaire. La conséquence directe de cette loi est donc que les grandes collections ont des vocabulaires de grande taille.

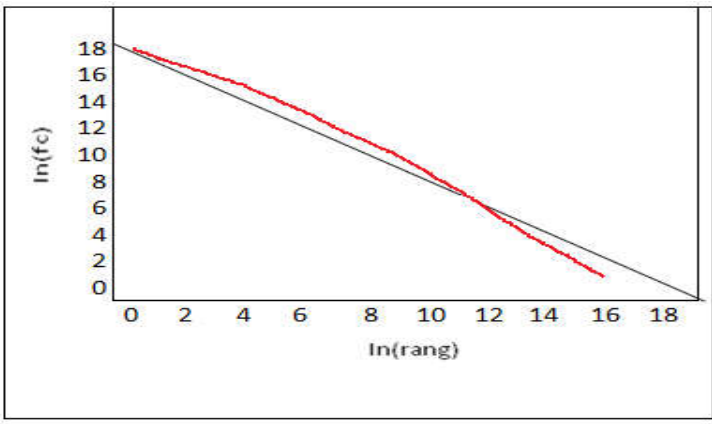
5.2 Loi de Zipf

La loi de (Zipf 1935) spécifie que la fréquence d'occurrence fc(m) d'un mot m dans une collection de documents est inversement proportionnelle à son rang, le rang étant ici celui obtenu lorsque l'on trie par ordre décroissant des fréquences les mots de la collection :

$$\forall\, m, \mathrm{fc}(m) = \frac{\lambda}{rang(m)}$$

Ou λ est ici aussi un paramètre dépendant de la collection considérée.

La figure 1.3 (ci contre) montre le lien entre les rangs et les fréquences d'occurrence des mots ainsi que les fréquences des mots de la collection de Wikipédia dont le rang vaut au plus dix. Sur la double échelle logarithmique nous voyons effectivement que la droite In(fc) = 17,42 – In(rang) représenté une bonne approximation de la relation entre logarithmes des fréquences et des rangs des mots de cette collection. Une conséquence de la loi de Zipf est que la fréquence du second mot le plus fréquent est la moitié de la fréquence du premier, etc. Une autre conséquence est que la fréquence des mots décroit rapidement avec leur rang Ceci explique pourquoi le filtrage par les 200 mots les plus fréquents diminue si drastiquement la taille des documents de la collection de Wikipédia [1]



rang	mot	Fréquence
1	de	36 875 868
2	la	16 565 726
3	Le	12 639 034
4	Et	11 587 487
5	En	10 885 221
6	l'	8 937 203
7	Du	8 541 846
8	Des	8 302 026
9	d'	7 766 576
10	Les	7 722 951

Figure 4: *Illustration de la loi de Zipf sur la collection du Wikipédia français (gauche).*

5.3 Représentation documentaire

En recherche documentaire, le but est de trouver les documents qui contiennent les termes d'une requête donnée. Pour d'autres taches comme la classification ou le partitionnement, on cherche à discriminer ou à regrouper les documents entre eux. La représentation du contenu d'une collection et donc une phase primordiale dans tous les processus de l'accès à l'information et a motivé de nombreux travaux. Dans la suite nous allons présenter les techniques et algorithmes classiques pour la représentation documentaire. La section suivante sera dévolue aux algorithmes classiques d'indexation [3].

a. Modèle vectoriel

Le model vectoriel (ou Vector Space Model), proposé par **Gerard Salton** (Salton 1975) Définit la représentation documentaire communément utilisée dans différentes taches de l'accès à l'information. Avec cette représentation on associe à chaque document *d* d'une collection *C,* un vecteur **d** dont la dimension correspond à la taille du vocabulaire. L'espace vectoriel considéré est donc un espace vectoriel de termes dans lequel chaque dimension est associée à un terme de collection.

$$\forall d \in C, \boldsymbol{d} = (w_{id}) \text{ i}\epsilon\{1, \dots, \text{V}\}$$

Dans ce cas, w_{id} est le poids que le terme d'indice i du vocabulaire a dans le document *d*. Nous verrons plus loin qu'il existe plusieurs façons de calculer ce poids. Il est toutefois important de noter ici que toutes ces méthodes assignent un poids nul à un terme non présent dans le document .Avec cette représentation, l'ordre d'apparition des termes dans le document n'est pas pris en compte (figure 2.4).Pour cette raison, cette représentation est aussi connue sous le nom de la représentation par sac de mots [8].

Au niveau terminologique, la matrice,X_{vXn} composée des vecteurs des documents représentés dans l'espace des termes du vocabulaire est notée la matrice termes-documents.

La représentation vectorielle (et les poids qu'elle manipule en général) implique que :

Les documents contenant des termes sémantiquement proches mais n'utilisant pas les mêmes termes se trouvent sur des dimensions différentes de l'espace vectoriel.

Les vecteurs de documents contiennent peu valeurs non nulles (0,05% de caractéristiques non nulles en moyenne pour les documents de la collection de Wikipédia –tableau 2.2).

Pour ne pas stocker inutilement les valeurs nulles présentes dans les vecteurs de représentation des documents, ces derniers sont souvent codés en ne gardant que les indices correspondant aux valeurs non nulles [2].

5.4 Pondération des termes

Malgré ces limitations, cette représentation est largement utilisée en catégorisation et partionnement de documents, ainsi que dans des modèles vectoriels de recherche .Elle est de plus très similaire à celle utilisée dans les modèles probabilistes de recherche d'information et présente l'avantage d'être facilement manipulable et exploitable par des techniques statistiques génériques. Une proche simple et classique pour estimer les poids $(w_{id})_{i=1}^{V}$du

vecteur associé au document d est de calculer le nombre d'occurrences des termes du vocabulaire dans le document. Cette approche connue en anglais sous le nom de *term frequency (tf)* définit ainsi chaque poids associé à un terme par le nombre de fois ou ce terme apparait dans un document :

$$\forall i \in \{1, \dots, V\}; w_{id} = tf_{ti,d}$$

Cela dit cette représentation fréquentielle introduit un biais problématique, les documents longs tendant à comporter davantage de répétitions de termes que les courts. Ainsi, un mot qui apparait 10 fois dans un document court (de100 mots par exemple) Est bien plus important pour ce document que le même mot apparaissant 100 fois dans un document très long (de 100 000 mots par exemple), différence dont ne rend pas compte le poids précédent (qui conduit en fait à l'importance inverse) .Comme nous le verrons plus loin, différents types de normalisation permettent de pallier ce problème. Elevé. En d'autres mots, ces termes ont un pouvoir de discrimination plus grand que celui des termes qui apparaissent, dans beaucoup de documents. Par exemple dans la collection du Wikipédia français. Le terme *algèbre* est spécifique à un domaine particulier, et sa présence dans un document donne une meilleure indication quant au sujet qui y est traité que le terme *calcul*, qui est un terme commun à plusieurs domaines présents dans la collection. Dans le calcul des poids, on prend en compte le nombre de documents dans lesquels un terme *t* du vocabulaire apparait, appelé *document frequency* en anglais et usuellement noté par df, en multipliant le nombre d'apparitions du

terme t dans un document d donné par son idft[7], le logarithme de l'inverse normalisé de dft :

$$idf_t = In\frac{N}{df_t}$$

Ou N est le nombre de document de la collection. On peut monter que l'idf d'un terme est, à un constant pré, l'entropie conditionnelle entre la variable aléatoire caractérisant les documents d'une collection .En recherche d'information, le codage le plus courant des documents, connu sous le nom du codage *tf-idf* est défini comme :

$$\forall i \in \{1, \dots, V\}; w_{id} = tf_{ti,d} \text{ X } idf_{ti}$$

De très nombreuses variantes de ce codage ont été proposées et elles ont fait l'objet d'un grand nombre de comparaisons expérimentales. Elles utilisent souvent un terme de

normalisation, *n*, dépendant de la taille du document considéré, en plus des deux termes à base de tf et de idf. La forme du codage des termes est ainsi :

$$\forall i \in \{1, \dots, V\}; w_{id} = n_d \text{ x } ptf_{ti,d} \text{ x } pdf_{ti}$$

Ou $ptf_{ti,d}$ et pdf_{ti} sont des poids fondés sur la fréquence du terme t_i dans d, ($ptf_{ti,d}$) ainsi que sur le nombre de documents le contenant(df_{ti}). Le tableau 2.3 récapitule les variantes les plus communes, proposées dans SMART le système de RI développé parle groupe de Gerard Salton (Salton 1975), et nous allons présenter dans la suite les trois pondérations à base de fréquence des termes les plus utilisées dans la littérature [1].

a) Pondération fréquentielle normalisée

Comme nous l'avons noté plus haut, la pondération fréquentielle est très sensible aux variations de la taille des documents. Par exemple, si on triple la taille d'un document, les poids des termes seront triplés et ce document augmenté sera considéré comme plus pertinent pour une requête composée de certains de ses termes que le document de départ. Une façon générale de remédier à ce problème est de normaliser [1]

$$\forall d \in C, \forall i \in \{1, \dots, V\}; w_{id} = n_d \text{ x } ptf_{ti,d} \text{ x } pdf_{ti}$$

$ptf_{ti,d}$	pdf_{ti}	n_d
$tf_{ti,d}$	1	1
$\frac{1 + \ln((tf_{ti,d})}{1 + In(moy\_tf(d))}$	idf_{ti}	$\frac{1}{\|d\|} = \frac{1}{\sqrt{\sum_{i=1}^{v} w_{id}}}$
$\begin{cases} 1 + \ln(tf_{ti,d}) \text{ si } tf_{ti,d} > 0 \\ 0 \quad sinon \end{cases}$		
0.5+0.5 x $\frac{tf_{ti,d}}{tf_{maxd}}$	Max{0,$In\frac{N-df_{ti}}{df_{ti}}$}	$\frac{1}{(Char_d)^{\alpha,}} 0 < \alpha < 1$
$\begin{cases} 1 \quad si \quad tf_{ti,d} > 0 \\ 0 \quad sinon \end{cases}$		

Tab.1.2 : *Différentes variantes du codage tf-idf, proposes dans SMART (Salton).* ($Char_d$ *Correspond à la taille du document d, en nombre de caractères le constituent.*

Les fréquences des termes apparaissant dans un document d par la fréquence maximale dans le document $tf_{maxd} = max_{t \in d}\, tf_{t,d}$

$$\forall i \in \{1, \dots, V\}; ptf_{ti,d} = \lambda + (1 - \lambda) \text{ x } \frac{tf_{ti,d}}{tf_{maxd}}$$

Ou λ ϵ[0,1] est un terme de lissage, généralement fixé à 0,5.

b) Pondération logarithmique des termes

Dans le cas de documents de grande taille, la pondération *tf-idf* linéaire privilégie le terme *tf* lorsque ce dernier est grand, alors même que la répétions d'un terme n'est pas le seul facteur déterminant son importance dans le calcul du poids, comme nous l'avons vu plus haut. Pour diminuer cet effet, et ainsi la disparité entre les caractéristiques du vecteur de représentation associé à un document *d,* on utilise souvent le logarithme du nombre d'occurrences des termes (Buckley et al.1992) [1]:

$$\forall i \in \{1, \dots, V\}; ptf_{ti,d} = \begin{cases} 1 + \ln(tf_{ti,d}) \text{ si } tf_{ti,d} > 0 \\ 0 \qquad\qquad\qquad\qquad sinon \end{cases}$$

c) **Pondération logarithmique normalisée**

Pour tenir compte des documents de petite taille, plusieurs études ont proposé de normaliser la caractéristique logarithmique précédente par une quantité qui dépend de la moyenne des caractéristiques fréquentielles d'un document d donné,

moy_tf(d) $=\sum_{i=1}^{v} \frac{tf_{ti,d}}{|d|}$, ou |*d*| représente le nombre de termes du vocabulaire différents dans le document *d* (si d ne contient qu'un seul terme, qui apparait 3 fois, |*d*|=1) :

$$\forall i \in \{1, \dots, V\}; ptf_{ti,d} = \frac{1 + \ln((tf_{ti,d})}{1 + In(moy\_tf(d))}$$

Après avoir exposé la représentation classique des documents sous forme vectorielle, nous allons maintenant nous intéresser à la construction de l'index inversé, qui fournit, pour chaque terme, la liste des documents qui le contiennent [3]**.**

5.5 Index inversé

L'utilisation d'une structure qui fait correspondre chaque terme du vocabulaire à la liste des documents qui le contiennent est la façon la plus rapide pour trouver un terme d'une requête donnée dans une collection de documents. Cette structure s'appelle communément un index inversé et elle peut prendre plusieurs formes. La forme la plus simple est celle qui est illustrée au bas de bas de la figure 1.1 dans laquelle, pour chaque terme du vocabulaire, *l'indexeur* ne fournit que la liste des identifiants de documents ou ce apparait. D'autres index indiquent aussi le nombre de documents contenant les termes, df, leur nombre d'occurrences dans les documents, tf (figure 1.5), ou la position des termes dans les documents [2].

5.6 Indexation dans des collections statiques

La figure 1.5 montre les différentes étapes de création d'index inversé des termes pour une collection statique de document, ces étapes sont :

- **L'extraction** de paires d'identifiants (terme, document), en opérant une passe complète sur la collection. Ces identifiants sont généralement des clés numériques uniques associées à chaque terme et chaque document de la collection. Dans la figure 2.5, on a assimilé les identifiants des documents à leur indices et aussi chaque terme à son identifiant pour plus de clarté [3].
- **Le tri** des paires suivants les clés d'identifiants de termes puis les clés identifiants de documents.

- **Le regroupement** de paires en construisant pour chaque identifiant de terme, la liste des identifiants de documents dans lesquels le terme apparait. A ce stade, nous pouvons ajouter dans cette liste d'identifiants de documents d'autres informations comme le *df* ou le *tf* des termes [1].

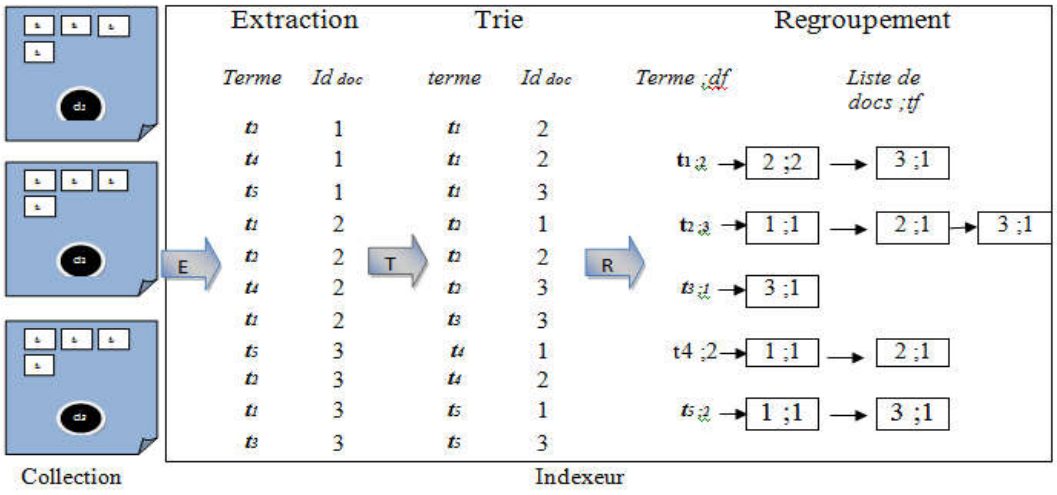


Figure 5 : *Création de l'index inversé pour une collection statique constitué de 3documents.*

5.7 Indexation par bloc à base de tri

L'étape la plus complexe suivant ce schéma est la phase de tri qui pour des grandes collections de documents ne peut pas être réalisée en mémoire. Dans ce cas, pour éviter de stocker la liste complète des paires à trier sur le disque et d'avoir recours à des accès disque gourmands en temps, on stocke les paires d'identifiants (terme, document) par bloc en mémoire et on effectue le tri avant de passer au bloc suivant[1].

Algorithme 1. **Algorithme d'indexation par bloc à base de tri**

Entrée : une collection de documents prétraitée C
Initialisation : *b*←0 ; pos← début de C
Répéter
b←*b*+1 ;
buffer ← Θ ;
Répéter
buffer ← *buffer* U{paires d'dentifiant(IDterm*pos* , IDdoc*pos*)} ;
pos← *nextpos* ;
jusqu'à (*buffer*=*plein*) V(*pos*=*fin de* C) ;
L*b*←Tri(*buffer*) ; // Les paires sont triées d'abord suivant les clés
// d'identifiants de termes puis suivant les clés
// d'identifiants de documents
R*b*←Regrouper (L*b*) ;
Ecrire R*b* sur le disque ;
jusqu'à pos= fin de C ;
Sortie : index inversé de C en fusionnant R1 ..,.R*b*

5.8 Indexation distribuée

Pour des collections de documents très volumineuses comme la Toile (ou Web en anglais), l'indexation ne peut pas être effectuée sur une seule machine. Dans ces cas, on utilise un cluster de calcul et de traitement de données en mode distribué pour construire l'index inversé des termes. Dans la phase *Map,* la collection de données est divisée en différentes parties dont chacune peut être traitée rapidement par un ordinateur (appelé analyseur dans la figure 1.5).C'est un serveur qui, suivant la disponibilité des machines sue cluster, affecte une partie des données à un analyseur. Chaque analyseur exécute ensuite l'étape d'extraction, présentée un peu plus haut, sur la partie qui lui est assignée. Les paires extraites d'identifiants de termes et de documents sont ensuite écrites par intervalle égal sur le disque de la machine (trois intervalles schématiquement représentés par **A-I ,J-R** et **S-Z** dans la figure 1.5)[1].

5.9 Indexation dans des collections dynamiques

L'indexation dynamique consiste à actualiser l'index après ajouts, modifications ou suppressions des documents d'une collection dynamique. Une solution triviale à ce problème est de construire périodiquement un nouvel index et de transférer la recherche sur ce dernier une fois sa construction terminée [3].

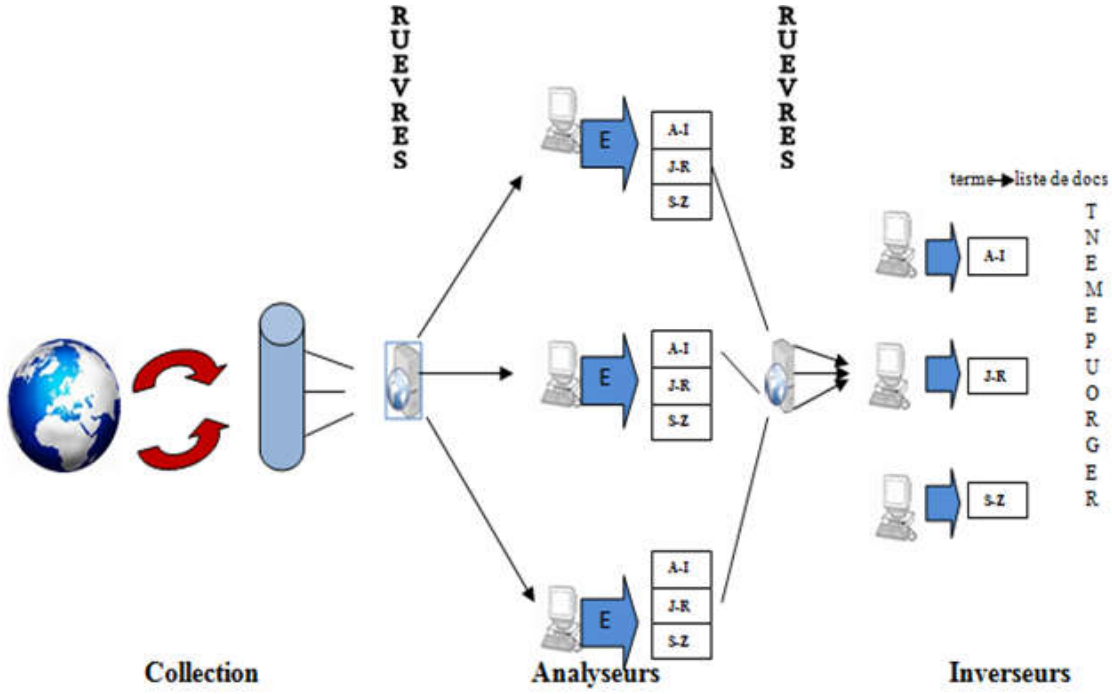


Figure 6 : *illustration de l'indexation distribuée. Nous supposons ici que l'indexation de la collection a lieu lorsque cette dernière ne subit pas de modifications dans le temps.*

Conclusion

Dans ce chapitre nous avons la recherche d'information comme une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information. Nous allons présenter par la suite des techniques de classification de documents relative aux besoins d'utilisateurs.

CHAPITRE II

CLASSIFICATION

ET

CATEGORISATION

Introduction

Dans ce chapitre, nous allons présenter les techniques automatiques de catégorisation de documents issues du domaine de l'apprentissage statistique, techniques qui sont maintenant largement utilisées dans les systèmes de RI. Nous allons tout d'abord exposer le formalisme de la tache de catégorisation, avant de passer en revue, les techniques de sélection de variables permettant de trouver le sous-ensemble des caractéristiques pertinentes parmi celles utilisées pour représenter les documents d'une collection.

1 Classification et catégorisation de documents

La classification et catégorisation de documents est l'activité du Traitement automatique des langues naturelles qui consiste à classer de façon automatique des ressources documentaires, généralement en provenance d'un corpus. Cette classification peut prendre une infinité de formes. On citera ainsi la classification par genre, par thème, ou encore par opinion. La tâche de classification est réalisée avec des algorithmes spécifiques, mis en œuvre par des systèmes de traitement de l'information. C'est une tâche d'automatisation d'un processus de classement, qui fait le plus souvent appel à des méthodes numériques (c'est-à-dire des algorithmes de recherche d'information ou de classification de type mathématique) [10].

2 Méthodologie

Le déploiement d'un système de classification repose sur plusieurs étapes. On peut les schématiser ainsi :

- ✓ Définition des classes (exemple : catégories "Sport", "Politique", "Diplomatie", ou encore Opinion "bonne/mauvaise")
- ✓ Apprentissage des classes avec un système de classification en utilisant un corpus d'apprentissage
- ✓ Évaluation des performances du système avec un corpus de test [11].

3 Méthodes algorithmiques

Comme toute tâche de classification, la catégorisation de documents peut être réalisée en mode supervisé, ou non-supervisé. En mode supervisé, des éléments pré-définis seront utilisés pour classer un document : ce peut être un index, ou encore un dictionnaire de mots correspondant à une classe particulière et servant à pré-étiqueter les documents. En mode non-

supervisé, c'est sur la phase d'apprentissage que reposera l'entraînement du classifieur, et ses performances ultérieures. On utilise dans les systèmes de classification de document des algorithmes numériques. Les plus performants sont ceux à base de SVM ou encore de Boosting (qui reposent sur AdaBoost). D'autres méthodes de mesure de similarité (tel que la similarité cosine), ou encore probabiliste (les classifieurs bayésiens naïfs) peuvent également être mises en œuvre. On utilise dans les systèmes les plus performants une combinaison de plusieurs systèmes de classification départagés par une méthode de vote [1]

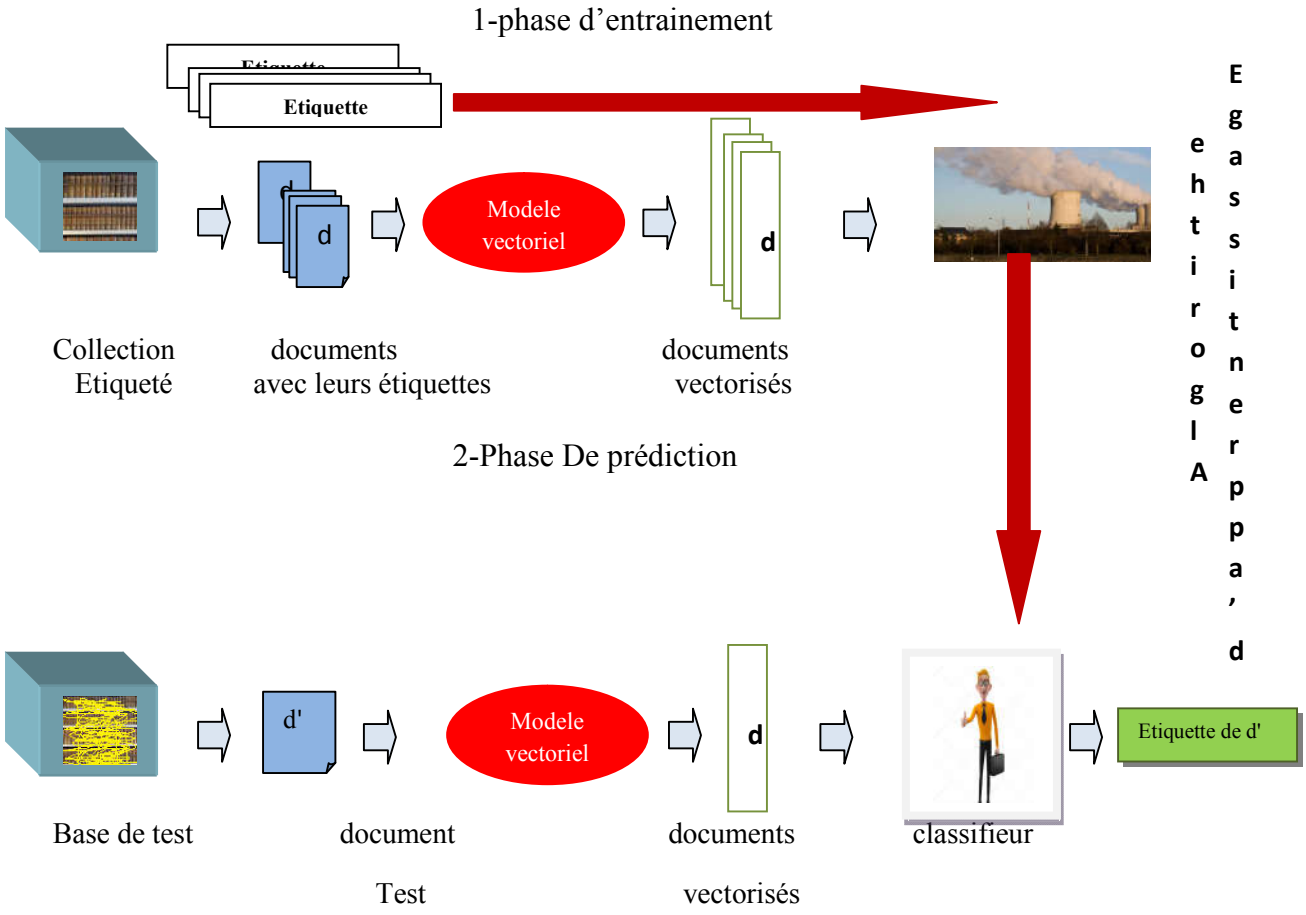


Figure 7 : *Catégorisation de documents en deux phases .*

Nous utiliserons la collection de Reuters-RCV2 français [3] pour illustrer les caractéristiques et les performances des différentes techniques et algorithmes que nous présentons. Cette collection contient 85 385 articles de presse écrits par des journalistes français de l'agence Reuters entre 20 aout 1996 et le 19 aout 1997. Ces articles ont été assignés manuellement à 3 586 catégories différentes, un document pouvant appartenir à plusieurs classes, ces classes étant structurées hiérarchiquement. Enfin, les classes dans cette collection ne sont pas équilibrées, certaines ne contenant qu'un seul document. De plus , si un document appartient à plus d'une classe , nous l'affectons à la classe minoritaire de l'ensemble de ses classes .

Pour les prétraitements, nous avons utilisé l'espacement pour segmenter les termes, et l'anti dictionnaire *French stopword list*4 pour filtrer les mots vides. Le tableau 2.1 donne les caractéristiques de cette collection [3].

Variables	Symboles	Valeurs
# de documents de la collection		85 167
# de classes initiales		3 586
# de documents considérés	N	70 703
# de termes du vocabulaire	V	141 146
# moyen de termes par document		136,7
# de documents considérés	K	29
Tailles de la base d'entrainement	M	35 000
Taille de base de test		35 703

***Tab 2**: Quelques statistiques sur la collection de Reuters-RCV2.*

4 Sélection de variables

Une des difficultés principales que soulève l'application des algorithmes de catégorisation développés en apprentissage aux données textuelles est la grande dimensionnalité de l'espace vectoriel utilisé pour représenter les documents. Alors que ces algorithmes ont été majoritairement conçus pour des problèmes de catégorisation ou la dimension de l'espace d'entrée est assez réduite, la taille du vocabulaire d'une collection de documents, qui est égale à la dimension des vecteurs représentatifs de ces derniers, dépasse généralement les dizaines voire certaines de milliers de termes.

Des méthodes de sélection de variables ont été proposées pour réduire la dimensionnalité de l'espace de représentation en supprimant du vocabulaire les termes non informatifs. L'objectif est ici d'obtenir un gain, en temps de calcul, sur l'apprentissage d'un classifieur (ce dernier étant appris à partir d'un ensemble réduit de termes sélectionnés) sans sacrifier à la performance de ce dernier sur la base de test [2].

5 Le seuillage sur la mesure Document Frequency (df)

Ce seuillage consiste à enlever les termes du vocabulaire présent dans la base d'entrainement et qui ont une mesure *df* , en-dessous d'un certain seuil prédéfini L'hypothèse sous-jacente est que les termes qui apparaissent dans très peu de documents ne sont pas informatifs pour une classe donnée et n'influent pas ainsi sur la performance globale d'un classifieur .Cette technique est certainement la plus simple pour réduire la taille du vocabulaire d'une collection . Dans le cas ou les termes apparaissant dans très peu de

documents sont peu informatifs, cette approche permet des gains de performance par rapport au cas ou le classifieur est appris à partir de tout le vocabulaire. Toutefois, un tel seuillage ne peut être très agressif car il transgresserait alors une des hypothèses de la RI qui stipule que moins la mesure df d'un terme est élevée, plus ce dernier est informatif. La technique précédente est une méthode non supervisée puisqu'elle n'utilise pas l'association entre les documents et les classes de la base d'entrainement pour sélectionner les termes [6].

5.1 L'information mutuelle ponctuelle (IMP)

Cette mesure estime l'information que la présence d'un terme *t* ϵ V dans les documents d'une classe *c* apporte sur la catégorisation dans cette classe .En notant *T* (resp.C) la variable aléatoire discrète qui est égale à 1 si un document contient le terme *t* (resp. appartient à la classe *c*) et à 0 sinon, on compare avec ce critère la probabilité jointe d'observer les deux événements correspondant aux cas T = 1 et C = 1,aux probabilités de les observer indépendamment :

$$IMP(t,c) = \log_2 \frac{P(T = 1, C = 1)}{P(T = 1)xP(C = 1)}$$

	T = 1	T = 0	Total
C = 1	n_{ct}	$\|C\| - n_{ct}$	$\|C\|$
C = 0	$df_t - n_{ct}$	$m - df_t - (\|C\| - n_{ct})$	$m - \|C\|$
Total	df_t	$m - df_t$	*M*

***Tab 3**: Tableau de contingence comptabilisant les nombres de documents appartenant, ou non, à la classe c et contenant, ou non, le terme t d'une base d'entrainement de taille m.*

Ainsi, lorsqu'un terme *t* n'apparait dans aucun document de la classe *c* (le cas ou les variables aléatoires *T* et *C* sont indépendantes).IMP(*t,c*) = 0 . Pour le cas opposé ou le terme t n'apparait que dans les documents de la classe *c*, ce critère est maximal. De plus, en développant la probabilité jointe dans l'équation 2.1, l'IMP entre un terme *t* et une *c* s'écrit :

$$IMP(t,c) = \log_2 P(T = 1|C = 1) - \log_2 P(T = 1)$$

Nous pouvons alors constater d'après cette expression que parmi les termes ayant des probabilités marginales, $(T = 1|C = 1)$, identiques, ceux qui sont plus courants(qui ont une probabilité $P(T = 1)$ plus élevés) seront considérés comme moins informatifs.

Les estimations des probabilités intervenant dans le calcul de l'IMP sont généralement déduites, au sens du maximum de vraisemblance, à partir d'une base d'entrainement *S* .En considérant le tableau de contingence 5.2 comptabilisant différentes statistiques sur la cooccurrence des documents appartenant à une classe *c* et contenant ou non un terme t, on peut établir[1]: $P(T = 1, C = 1) = \frac{n_{ct}}{|C|}$

$$P(T = 1) = \frac{df_t}{m}$$

$$IMP(t, c) = \log_2 \frac{m\ x\ n_{ct}}{df_t\ x\ |C|}$$

Pour obtenir un critère de sélection final, on prend généralement la moyenne pondérée des différentes mesures IMP entre chaque terme du vocabulaire et les classes existantes :

$$\forall\ t\ \epsilon\ \mathrm{V}\ ,\ \mathrm{I(t)}=\sum_{\mathrm{c}\ \epsilon \mathrm{Y}} P(C = 1)\ x\ IMP(t, c)$$

5.2 L'information mutuelle (IM)

Cette mesure estime l'information que la présence et l'absence d'un terme $t\ \epsilon\ \mathrm{V}$ dans les documents d'une classe *c* et les documents d'autres classes apportent pour prendre la bonne décision de catégorisation sur *c* Formellement cette mesure se calcule comme l'espérance mathématique jointe de e sur toutes les réalisations des variables aléatoires discrètes

T et *C* :

$$IM(t, c) = \mathrm{E}_{\mathrm{T,C}} \left[\log_2 \frac{\mathrm{P(T,C)}}{P(T)xP(C)}\right]$$

$$IM(t, c) = \sum_{Vc\ \epsilon\{0,1\}}\ \sum_{Vt\epsilon\{0,1\}} P(T = Vt, C = Vc)\log_2 \frac{\mathrm{P(T}=Vt,\mathrm{C}=Vc)}{P(T=Vt)P(C=)Vc}$$

Si la distribution d'un terme *t* dans une classe *c* est la même que sa distribution dans la base d'entrainement c'est-à-dire sur toutes les classes la mesure IM(*t,c*) est alors nulle .Comme avec le critère IMP la mesure IM(*t,c*) atteint son maximum dans le cas ou le terme *t* n'apparait que dans les documents de la classe *c.* L'algorithme 7 montre un procédé simple de sélection de *L* termes d'un vocabulaire décrivant au mieux une classe *c* d'après la mesure IM [6].

Algorithme 7 : Sélection de variables avec la mesure IM

Entrée

- Une base d'entrainement S=(($\boldsymbol{d}_1, c_1$),…,($\boldsymbol{d}_m$,c_m)) ;
- *L*, le nombre de termes à sélectionner pour la classe *c*.

Initialisation :

- Extraire le vocabulaire V des documents de S(chapitre 1) ;
- Initialiser la table de hachage % H⟵Θ

Pour *t* ϵ V faire

| % H{t}⟵IM(*t,c*)

Fin

Sortie : les *L* termes associés aux *L* mesures IM les plus grandes d'après % H{.}

5.3 Les Modèles d'apprentissage

5.3.1 Modèles génératifs

Les modèles génératifs sont les premiers modèles d'apprentissage automatique développés en RI pour la tache de catégorisation. L'hypothèse fondamentale des modèles génératifs est que chaque vecteur représentatif d'un document ***d*** est la réalisation d'une variable aléatoire multidimensionnelle et est généré par le mélange de k densités de probabilité avec des propositions (ou probabilités *a priori*) π_1,.. π_k Vérifiant :

$$\sum_{k=1} \pi_1 = 1 \text{ et } \pi_k >= 0 \text{ (k=1,…,k)}$$

Chaque fonction de densité de ce mélange est une fonction paramétrique modélisant la distribution de probabilité conditionnelle de classe associé à une classe k ϵ Y donné :

$$\forall K \in Y , P(\boldsymbol{d}|C_K =1) = f_K(d, \Theta_K)$$

Ou C_K est la variable aléatoire binaire exprimant l'apparence ($C_K = 1$) ou non $C_K = 0$) d'un document à la K^e classe, la densité de mélange modélise ainsi la génération de d par ces K densité de probabilité et elle s'exprime comme linéaire convexe de ces dernières :

$$P(d,\Theta) = \sum_{k=1}^{k} \pi_k f_K(d, \Theta_K)$$

Ou Θ est l'ensemble des proportions π_k et de tous les paramètres définissant les fonctions de densité f_K :

$$\Theta = \{\Theta_K, \pi_k : k \in \{1,…,K\}\}$$

Le but de l'apprentissage est alors d'estimer l'ensemble des paramètres Θ pour que le modèle de mélange explique au mieux, au sens du maximum de vraisemblance , les exemples de la

base d'entrainement . Une fois l'estimation de ces paramètres terminée, un nouveau document d'est affecté à une classe de l'ensemble Y d'après la règle de décision Bayésienne :

d' appartient à la classe k ssi k= $\text{argmax}_{h\epsilon Y}$ P(C_h =1|**d**')

Ou de façon équivalente et après la règle de Bayes :

d' appartient à la classe k ssi k= $\text{argmax}_{h\epsilon Y}$ π_h f_K(|**d**', Θ_h)

dans les sections suivantes nous représentons les deux modèles génératifs les plus utilisés et connus en catégorisation automatique de documents [5].

5.3.2 Modèle multinomial

Dans le modèle multinomial on considère un document d = (t_1,…, t_{L_d}) comme une séquence ordonnée de termes générée à partir d'un vocabulaire V donné. Avec l'hypothèse de Naive Bayes qui rappelons le, suppose que la probabilité de génération d'un terme une composante du mélange (ou une classe) est indépendante du contexte et de la position de ce terme dans le document les densités de probabilité s'écrivent :

$$\forall\, k \,\epsilon\, Y,\ P(d = (t_{1,\dots,} t_{l_d}) |\ C_k = 1) = \prod_{t_h \epsilon d} \Theta_{t_{h|K}}$$

Ou, $\forall\, h \,\epsilon\, \{1,\dots V\}$, $\Theta_{t_{h|K}} = P(t_{h|}\ C_K = 1)$ représente la probabilité de génération du terme du vocabulaire par la k^e composante du mélange. Ces probabilités vérifient :

$$\Theta = \{\Theta_{t_{i|K}} : i \,\epsilon\, \{1, \dots, V\}, k \,\epsilon\{1, \dots, k\}; \pi_k : k \,\epsilon\{1, \dots, k\}\}$$

Le modèle multinomial capture l'information de fréquence des termes dans les documents, et non plus seulement l'information de présence/absence. Avec les notions de la section précédente , les estimations au sens du maximum de vraisemblance des paramètres de ce modèle avec le même lissage de Laplace s'obtiennent d'après le principe développé à :

$$\forall\, i \,\epsilon\, \{1,\dots V\}, \forall\, k \,\epsilon\, Y,\ \Theta_{t_{i|K}} = \frac{\sum_{d\epsilon} tf_{t_{i,d}} + 1}{\sum_{i=1}^{V} \sum_{d\epsilon S_K} tf_{t_{i,d}} + V}$$

$$\forall\, k \,\epsilon\, Y, \pi_k = \frac{N_k(S)}{m}$$

Pour l'estimation des probabilités conditionnelles des termes d'une classe k donnée (équation 5.12) on compte le nombre d'occurrences des termes du vocabulaire dans les documents de cette classe. En pratique il est plus simple de concaténer les' documents d'une classe donnée

avant de réaliser ce comptage. L'algorithme 10 montre le procédé d'estimation des paramètres du modèle multinomial pour la phase d'apprentissage et l'algorithme 11 présente la phase de test avec ce modèle [6].

ALGORITHME 10 **Modèle multinomial, phase d'apprentissage**

Entrée :
- Une base d'entraiment S = $((d_1, c_1), \ldots, (d_m, c_m))$.

Initialisation :
- Extraire le vocabulaire V = { $t_1, \ldots, t_v$ } de S ;
- Initialiser N [.] [.] ← 0, D [.] ← 0.

Pour chaque k ϵ {1, …, K} faire
  N[k] ← $\sum_{d \in S_k} 1$; // S_k = {d ∈ S | d appartient à la classe k }
  P_i [K] ← $\frac{N_k}{m}$; // équation 5.13
  Pour chaque i ∈ {1, …, V} faire
    tf [k] [i] ← $\sum_{d \in S_k} tf_{id}$;
    D[k] ← D [k] + tf [k] [I];
  Fin
Fin
Pour chaque k ϵ {1, …, k} faire
  Pour chaque k ∈ {1, …, V} faire
  PC [k][i] ← $\frac{tf[k][i]+1}{D[k]+V}$; // équation
  Fin
Fin
Sortie : les tableaux P_i[.] et PC [.][.]

ALGORITHME 11. Modèle multinomial, phase de test

Entrée :
- Un document d', un vocabulaire V = {t_1, …, t_v } ;
- Les tableaux P_I[.] et PC [.][.] obtenus avec l'algorithme 10.

Initialisation :
- Obtenus la représentation vectorielle d' = $(w'_{id})_{i \epsilon \{1,\ldots,V\}}$ avec V ;
- Initialisation Pif [.] ← 0.

Pour chaque i ϵ {1,…, K} faire
  Pif[k] ← ln (Pi[k]) ;
  Pour chaque k ϵ {1,…, V} faire
    Pif [K] ← PiF[k] + w'_{id} × ln (PC[k][i]) ;
  Fin
Fin
Sortie : classe de d' = $argmax_{h \epsilon y}$ PiF[h]

D'après ces résultats, nous constatons que le modèle multivarié de Bernoulli est plus performant que le modèle multinomial sur des vocabulaires de très petites tailles et que cette tendance s'inverse en faveur du modèle multinomial lorsque la taille du vocabulaire s'accroit. Ce phénomène peut facilement se comprendre dans la mesure où la méthode IM sélectionne les termes informatifs par rapport aux classes et que, plus un terme représentatif d'une classe donnée apparait dans un document, plus ce document n'est susceptible d'appartenir à cette classe. Le seconde constat est que les performances des deux modèles génératifs obtenues avec un vocabulaire contenant les 50 000 meilleurs termes (au sens de l'information mutuelle, IM) sont à peu près égales à leurs performances lorsque tout le vocabulaire, de taille 141 146, est utilisé pour représenter les documents [3].

5.4 Modèles discriminants

Les modèles discriminations trouvent directement une fonction de classification $f : \mathbb{R}^v Y$ qui résout le problème de catégorisation sans émettre d'hypothèses sur la génération des exemples comme on vient de le voir avec les modèles génératif dans la section précédente. Le classifieur recherché est cependant supposé appartenir à une classe de fonctions donnée F et sa forme analytique est trouvée en minimisant une certaine *fonction d'erreur* (appelée aussi *risque* fondée sur une *fonction de perte* (ou fonction de cout) de la L : $y \times y \rightarrow \mathbb{R}^+$

Par exemple, la fonction de perte généralement considérée en catégorisation est l'erreur de classification définie comme :

$$\forall(\boldsymbol{d}, c), L(f(\boldsymbol{d}), c) = \mathbb{1}_{f(\boldsymbol{d}) \neq c}$$

Ou 1_π est la fonction indicatrice qui vaut 1 si le prédicat π est vrai et 0 sinon. Comme mentionné dans la section 5.1, le classifieur qu'on apprend doit être capable de produire de bonnes prédictions sur de nouveaux exemples, ou commettre une faible erreur de généralisation, ce qui, avec l'hypothèse de génération i.i.d. des exemples, s'écrit :

$$R(f) = \mathbb{E}_{(\boldsymbol{d},c)\sim D} L(\boldsymbol{d}, c) = \int_{x \times y} L(h(\boldsymbol{d}), c) D(\boldsymbol{d}, y)$$

Ou $\mathbb{E}_{(\boldsymbol{d},c)\sim D} L(\boldsymbol{d}, c)$ représente l'espérance mathématique de la variable aléatoire L lorsque la paire d'exemples (d, c) suit la loi D. Comme la distribution de probabilité D est inconnue, la

fonction de risque *R* ne peut pas être minimisée directement et les algorithmes d'apprentissage tentent de trouver la fonction de classification f en minimisant un estimateur non biaisé de *R* sur une base d'entrainement $S = ((d_j, c_j))_{i=1}^{m}$ L'estimateur habituellement utilisé correspond à *l'erreur empirique*, définie par :

$$\hat{R}_m(f, S) = \frac{1}{m}\sum_{j=1}^{m} L(f(\boldsymbol{d}_j), c_j)$$

(Vapnik 1999), par exemple, montre le lien qui existe entre erreur de généralisation d'un classifieur et erreur empirique, d'une part, et entre ces erreurs et la classe de fonctions considérée f , d'autre part. Cette étude apporte une justification formelle au principe de *minimisation du risque empirique* (MRE), qui consiste à sélectionner la fonction dont l'erreur empirique sur la base d'entrainement est minimale [1].

La théorie de Vapnik a aussi donné naissance au développement d'un modèle très intéressant, appelé séparateurs à vaste marge (SVM), et qui est devenu le modèle par excellence pour la catégorisation de documents en RI. Ce modèle a été développé pour le cas de la classification en deux classes et différentes stratégies ont été proposées pour l'adapter au cas multi classe. Pour simplifier la présentation et afin de nous mettre dans le cadre d'étude des SVM, nous allons considérer dans la suite le problème de classification en deux classes (problème bi-classe). Dans ce cas l'espace de sortie considérée est généralement $y = \{-1, +1\}$ et la fonction de classification est aussi une fonction $h : \mathbb{R}^V \rightarrow \mathbb{R}$, qui a une forme analytique propre à la classe de fonctions considérée. Avec le principe de MRE, on cherche la fonction *h* qui minimise le risque empirique sur la base d'entraînement. Géométriquement, l'ensemble des exemples de l'espace d'entrée $\{\boldsymbol{d}|h(\boldsymbol{d}) = 0\}$ définit une frontière de décision séparant l'espace représentation en deux sous-espaces :

$\{\boldsymbol{d}|h(\boldsymbol{d}) > 0\}$ et $\{d|h(\boldsymbol{d}) \leq 0\}$. La classe associée à chaque point du premier sous-espace est +1 et, dans le second sous-espace, la classe associée est _1. Une fois la fonction h fixée, le classifieur associé est alors défini comme $\left(h(\boldsymbol{d})\right)$, ou sgn est la fonction signe :

$$\forall x \in \mathbb{R}, sgn(x) = \begin{cases} +1 : si\ x > 0 \\ 0 \quad : si\ x = 0 \\ -1 : si\ x < 0 \end{cases}$$

Dans ce cas particulier, l'erreur de classification 2.15 s'écrit aussi :

$$\forall(d,c); L(h(\boldsymbol{d}),c) = \mathbb{1}_{c\times h(\boldsymbol{d})\leq 0}$$

Cette fonction de risque n'est pas dérivable et une optimisation directe de l'erreur empirique de classification estimée sur une base d'entrainement est impossible. De ce fait, les algorithmes d'apprentissage, qui cherchent un classifieur ayant la plus petite erreur de classification, considère généralement une borne supérieure convexe et dérivable de l'erreur de classification existantes, les deux fonctions de couts suivantes ont été plus particulièrement étudiées dans l'état de l'art à cause de leurs propriétés mathématiques:

$$L_l(h(\boldsymbol{d}),c) = \ln\left(1 + exp\left(-c\times h(\boldsymbol{d})\right)\right) \textbf{ (Cout logistique)}$$

$$L_l(h(\boldsymbol{d}),c) = e^{-c\times h(d)} \textbf{ (Cout exponentiel)}$$

La figure 2.3 montre les graphes de ces deux fonctions de cout ainsi que l'erreur de classification en fonction du produit $\times\, h$.

Le reste de cette section est consacré à la description du modèle logistique et des séparateurs à vaste marge. Trois autres algorithmes : le Perceptron, les k plus proches voisins et le Boosting, toujours utilisés en catégorisation de documents mais moins populaires [1].

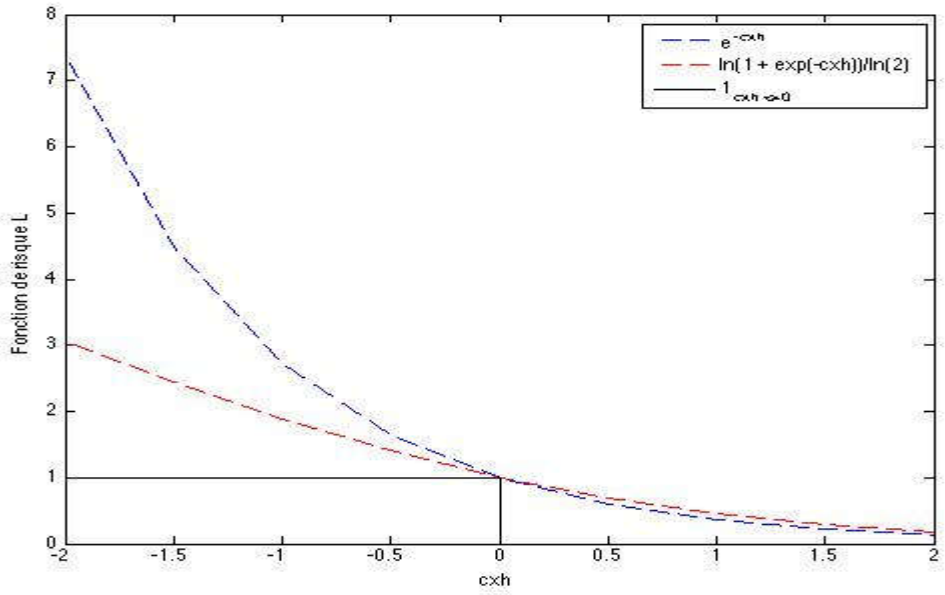


Figure 8 : *Trois fonctions de cout pour un problème de catégorisation à deux classes en fonction du produit $c\times h$, ou h est la fonction de cout sont l'erreur de classification.*

5.5 Modèle logistique

Le modèle logistique a été d'abord étudié en statistique et puis largement employé dans les autres domaines pour des problèmes de catégorisation. Comme on vient de le mentionner, ce modèle peut être obtenu en minimisant l'erreur empirique fondée soit sur le cout logistique. Dans cette section, nous allons présenter un algorithme permettant de construire itérativement le modèle logistique tout en minimisant l'erreur empirique exponentielle grâce à une fonction auxiliaire [21].

Une condition nécessaire à l'utilisation de cet algorithme est que les vecteurs représentatifs des exemples soient normalisés de façon à ce que :

$$\forall \boldsymbol{d} = (w'_{id})_{i\epsilon\{1,\dots,V\}}, \forall i \in \{1, \dots, V\}, w'_{id} \geq 0 \text{ Et } \sum_{i=1}^{v} w'_{id}=1$$

Sur une base d'apprentissage donnée $S = \left(\left(\boldsymbol{d}_j, c_j\right)\right)_{j=1}^{m}$ on cherche alors une fonction à

Valeurs réellesh_λ, de paramètres $= (\lambda_1, \dots, \lambda_V)$, dans la classe des fonctions linéaires ($\langle ., . \rangle$ dénote le produit scalaire) :

$$\mathcal{F} = \{f_\lambda : \mathbf{d} \mapsto \langle \mathbf{d}, \lambda \rangle | \lambda \in \mathbb{R}^V\}$$

qui minimise l'erreur empirique : $\hat{\mathrm{R}}_m^e(\lambda, S) = \frac{1}{m}\sum_{j=1}^{m} e^{-cf_\lambda(\mathbf{d}_j)}$

ALGORITHME 10 **Modèle logistique, phase d'apprentissage**

Entrée :

- Une base d'entraiment S = ((d_1, c_1), …, (d_m, c_m)).
  /* S_+ (Respectivement S_-) est les sons ensemble des documents de la base D'entrainement appartenant à la classe +1 (respectivement -1). */

Initialisation :

- Normaliser les vecteurs des documents avec leur norme$-L^1$;
- Initialiser aléatoirement les poids $\lambda^{(0)}$;
- $l = 0$.

Répéter

Pour chaque $\boldsymbol{i} \in \{1, \dots, V\}$ **faire**

$$\lambda_i^{(l+1)} \leftarrow \lambda_i^{(l)} + \frac{1}{2}\ln\frac{\sum_{d\epsilon S_+} w_{id}\ e^{-\langle d..\lambda^l\rangle}}{\sum_{d\epsilon S_-} w_{id}\ e^{\langle d..\lambda^l\rangle}}$$

$\boldsymbol{l \leftarrow l + 1}$ **;**

Jusqu'à *convergence de*

Fin

Sortie : les paramètres du modèle linéaire de paramètres λ^l

L'algorithme 12^4 montre ce procédé itératif avec la formule de mise à jour des poids établie à l'exercice 5.1. À chaque itération de cet algorithme, et pour un ensemble de poids $\lambda + \delta = (\lambda_1 + \delta_1, \dots, \lambda_V + \delta_V)$ qui conduisent à un nouveau modèle de plus faible erreur empirique. Ce procédé est poursuivi jusqu'à ce que le minimum de la fonction de risque, convexe, $\hat{R}_m^e(\lambda, S)$ (équation 5.18) soit atteint sur la base d'entrainement *S*. Une fois terminée l'estimation des poids du modèle logistique, la sortie de la fonction linéaire associée est utilisée pour prédire la classe d'un nouveau document *d′* suivant la règle :

$$\text{Classe de } d\text{'e} \begin{cases} +1 : \text{si } f_\lambda(\mathbf{d}') \geq 0 \\ -1 : \text{sinon.} \end{cases}$$

Autrement dit, les positions des nouveaux documents par rapport aux sous-espaces définis par l'hyperplan séparateur, de vecteur normal λ trouvé par l'algorithme, indiqueront les classes de ces documents [1].

6 Mesures d'évaluation

L'application des algorithmes de catégorisation en deux classes aux problèmes multi-classes s'effectue généralement suivant l'approche un contre tous qui consiste à apprendre autant de classes existantes. Dans ce cas, chaque classifieur associé à une classe donnée est appris en considérant les documents appartenant à cette classe comme des documents de la classe positive et tous les autres documents comme ceux appartenant à la classe négative. En prédiction, un nouveau document est assigné à la classe dont le score du classifieur associé est le plus grand. Pour chacun de ces classifieurs, il est alors possible d'estimer différentes mesures d'évaluation dont les quatre les plus populaires sont la *précision*, le *rappel*, la *mesure* F_1 et le *taux de bonne classification* [20].

Classe *K*		Jugement de l'expert	
		Dans la classe	Pas dans la classe
Prédiction du classifieur	Dans la classe	VP_k	FP_k
	Pas dans la classe	FN_k	VN_k

Tab 4: *Tableau de contingence indiquant les nombres de bonnes et de mauvaises prédictions.*

du classifieur associé à une classe k d'après le jugement d'un expert.Pour une classe k donnée, on considère le tableau de contingence indiquant, pour un classifieur, les nombres de bonnes et de mauvaises prédictions. *VP* désigne les vrais positifs, c'est-à-dire les documents que le classifieur a affecté à la classe et qui sont réellement dans la classe, *FP* les faux positifs, *VN* les vrais négatifs et *FN* les faux négatifs. Les performances de ce classifieur par rapport aux mesures d'évaluation mentionnées plus haut sont alors [20] :

Précision $P_k = \frac{VP_k}{VP_k+FP_k}$

Rappel $R_k = \frac{VP_k}{VP_k+FP_k}$

Mesure F_1 $F_1 = \frac{2P_kR_k}{P_k+R_k}$

Taux de bonne classification $TBC_k = \frac{VP_k+VN_k}{VP_k+VN_k+FP_k+FN_k}$

Pour obtenir des mesures globales sur l'ensemble des classes, deux types de moyenne sont envisagées sur les performances des classifieurs calculées pour chacune des classes. Le premier est la macromoyenne (*macro-averaging* en anglais) qui est la moyenne arithmétique simple des performances calculées par classe ; le second est la micromoyenne (ou *micro-averaging* en anglais) qui se calcule après avoir fusionné les tableaux de contingence en sommant chacune des cellules correspondantes des différents tableaux [10].

Par exemple, pour la précision, ces deux moyennes sont :

$$P_{macro} = \frac{1}{|y|}\sum_{k\in|y|} P_k, \quad P_{micro} = \frac{\sum_{k\in y} VP_k}{\sum_{k\in y} VP_k+FP_k}$$

Conclusion

Dans ce chapitre nous avons présentés la catégorisation de texte qu'est un domaine de recherche qui connait des améliorations et des innovations importantes dont le but est de faire apprendre à une machine comment attribuer un texte à la bonne catégorie. Face à l'accroissement du volume d'information, la catégorisation automatique de texte s'impose de plus en plus comme une étape essentielle qui permet d'accélérer, cibler et d'améliorer la recherche d'information.

CHAPITRE III

CONCEPTION

DU

SYSTÈME

Introduction

Notre application consiste à concevoir un système de recherche d'information et de classification de documents textuels, cette application permet de présenter un système de prédiction de thématique d'un document donnée en entrée. Pour ce faire, nous avons utilisé un corpus qui est un ensemble de documents textuels téléchargé depuis le site de partage de publications scientifiques Citeulike[1] qui contient une grande masse de documents répartis sur plusieurs thèmes. D'abord, ces documents on été convertis en représentations numérique sous forme de vecteurs TF-IDF, qui seront utilisés comme base d'apprentissage et de test pour notre classification.

Dans ce la suite de ce chapitre nous allons décrire, l'architecture, la conception ainsi que modélisation du notre système.

1 Architecture générale du système

Le système est réalisé en deux phases essentielles : une phase d'apprentissage et une phase de classification. La figure (Figure 3.1.) montre l'architecture générale du système :

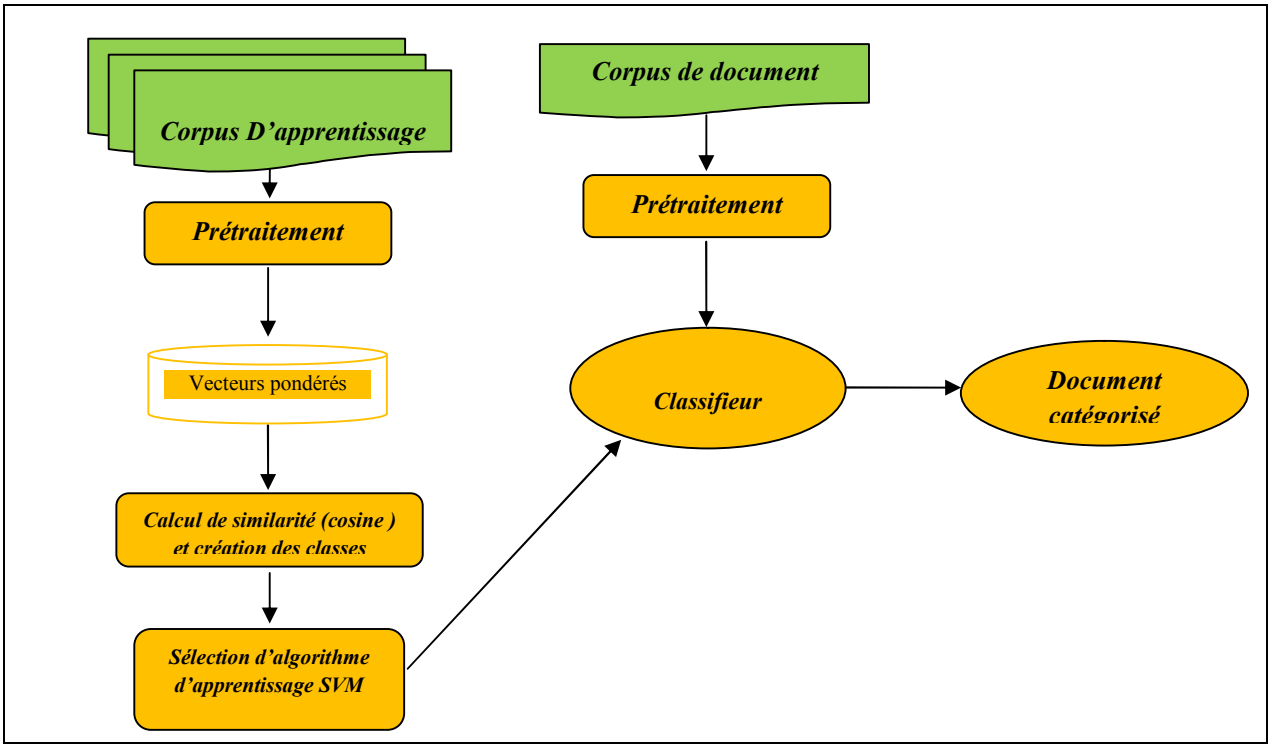


Figure 9 : *L'architecture générale du système.*

[1] http://www.citeulike.org/

1. Prétraitements

Cette étape comporte les sous étapes suivantes :

- ✓ **L'analyse lexicale :** est un processus qui convertit le texte d'un document en un ensemble de termes ou un terme est considéré comme une unité lexicale.Cette analyse permet de reconnaitre les espaces de séparation des chiffres, des mots, des ponctuations, … etc.
- ✓ **Suppression des mots vides :** Ce traitement permet de garder seulement les termes significatifs qui représentent le contenu de document. Afin d'éliminer les mots vides de sens, nous avons utilisé une liste, appelée stop Word list (ou parfois anti-dictionnaire) qui contient tous les pronoms personnels, les adverbes, les articles, les conjonctions de coordination, les verbes auxiliaires de la langue anglaise qui se trouve sur ce site.
- ✓ **Calcul des poids :** Dans le but d'attribuer à chaque terme un poids qui mesure son importance au sein du corpus, on a opté pour l'utilisation de formule de pondération suivante :
  - ➢ Formule TF_IDF.

Nous avons décrit cette formule dans le chapitre 1.

2. Calcul de similarité et création des classes

Dans le but de créer des classes contenant des documents similaires on a calculé la similarité entre vecteurs de documents.

Après avoir testé plusieurs mesures de similarité sur un échantillon de « 50 » documents, le choix est tombé sur la mesure de cosinus qui a donné de meilleurs de calcul de similarité. Les classes trouvées correspondent à un regroupement de documents similaire guidé en quelque sorte par un seuil établi = 0.5.

La similarité entre deux documents d'une même classe doit être inférieure ou égale au seuil, et la similarité entre deux documents de classes différentes est strictement supérieure au seuil.

3. Apprentissage

A cette étape le corpus de documents constitué de 500 documents a été subdivisé en 2 ensembles l'ensemble d'apprentissage qui contient 330 documents et l'ensemble de test qui contient 170 documents.

Les SVM[2] ont montré leur puissance et efficacité dans la résolution des problèmes de classification de texte. Alors et en utilisant cette technique de classification on a essayé d'entrainer le classifieur SVM sur le corpus d'apprentissage qui permet de prédire la classe de chaque nouvel article introduit au système par le biais de son titre et son abstract.

- Test : ceci permet de tester le classifieur en utilisant les documents de l'ensemble de test, en calculant les probabilités à posteriori d'appartenance des documents testés aux différentes classes.

La génération du modèle de classification se fait en appelant la fonction « SMO » existante sous weka[3].

Classification de nouveaux documents (articles) : le document est assigné à la catégorie ayant la probabilité à postériori la plus grande.

4. Conception du système

Dans cette section du mémoire, nous présentons la conception de notre application en utilisant le langage UML (Unified Modeling Langage de pour langage de modélisation Unifiée en français), qui est un langage de modélisation graphique à base de pictogrammes. UML est le fruit de la fusion Booch, OOM et OOSE qui sont des langages de modélisation. A présent UML est un standard défini par l'Objet Management groupe (OMG). Depuis sa standardisation par l'OMG en 1997, UML est devenu un standard incontournable [13].

4.1 Les points forts d'UML

- UML es**t** un langage semi-formel et normalisé, donc il garantit, d'obtenir une modélisation de haut niveau indépendante des langages et des environnements.
- Il permet de générer automatiquement la partie logicielle d'un système.
- La modélisation UML apporte une compréhension rapide du programme à d'autres développeurs externes en cas de reprise du logiciel et facilite sa maintenance [13].

Dans ce qui suit, nous allons présenter la conception du notre système. Pour cela, nous avons utilisé les diagrammes d'UML suivants :

1. Diagramme de cas d'utilisation.
2. Diagrammes de séquences.
3. Diagramme de classes.
4. Diagramme d'activités.

[2] *Support Vector Machines*
[3] http://www.cs.waikato.ac.nz/ml/weka/

4.2 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation (use case) capture le comportement d'un système, tel qu'un utilisateur extérieur le voit. Le rôle de ce diagramme est l'analyse, l'organisation des besoins et le recensement des grandes fonctionnalités d'un système. Il s'agit donc de l'étape UML la plus importante dans l'analyse d'un système.

Un diagramme de cas d'utilisation est un graphe d'acteurs, un ensemble de cas d'utilisation englobés par la limite du système, des relations (ou associations) des communications (participations) entre les acteurs et les cas d'utilisation, et des généralisations de ces cas d'utilisation.

- **L'utilisateur** est le seul acteur dans notre système qui peut interagir avec le système et il est capable de l'évaluer en commentant les résultats obtenu.

Les cas d'utilisation de notre système sont :

1. **Ajouter un texte :** c'est pour l'insertion de nouveau texte, coller le titre et l'abstract.

D'où la présentation de notre diagramme de cas d'utilisation :

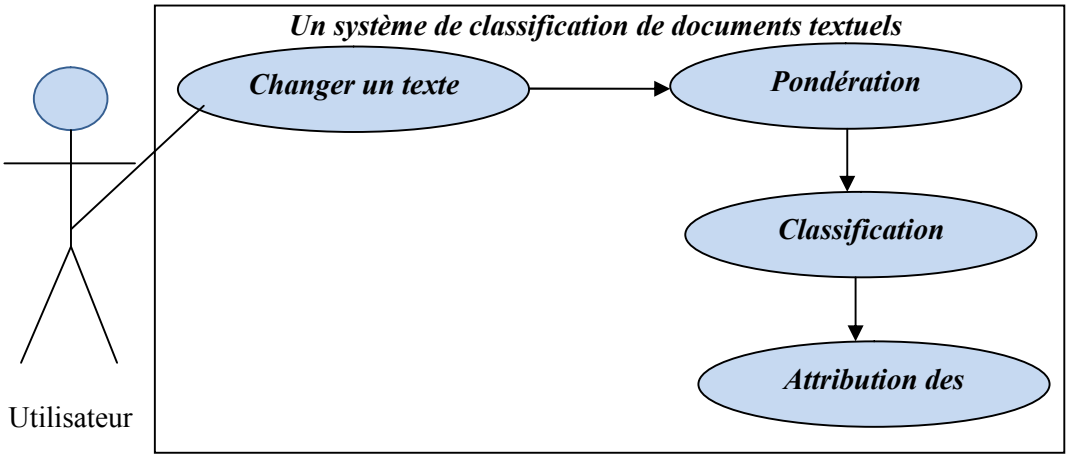


Figure 10 : *Diagramme de cas d'utilisation.*

4.3 Diagramme de séquences

C'est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs ou objets. Plusieurs types de messages (actions) peuvent transiter entre les acteurs et les objets. (Message simple, avec durée de vie, message synchrone, message asynchrone, message dérobant). Voici quelques notions de base du diagramme :

- **Une interaction :** un comportement qui comprend un ensemble de messages échangés par un ensemble d'objets dans un certain contexte pour accomplir une certaine tache.

- **Un message :** représente une communication unidirectionnelle entre objets qui transporte de l'information avec l'intention de déclencher une réaction chez le récepteur.

Les figures suivantes représentent les diagrammes de séquence des cas d'utilisation cités plus haut.

Diagramme de séquence du cas d'utilisation «Ajouter un texte» :

L'utilisateur peut ajouter dans cette étape un nouveau texte.

Les échanges des messages entre l'utilisateur, le système et les données (fichier data) sont :

- ✓ L'utilisateur colle titre et l'abstract dans les champs titre et l'abstract puis sur soumettre.
- ✓ Le système stock le texte et effectue les traitements nécessaires (étapes de prétraitement) et calcule le vecteur pondéré par la formule TF-IDF est présentées dans le chapitre 1.
- ✓ Le système compare le vecteur du nouveau texte avec les autres vecteurs de corpus stockés dans le fichier data.
- ✓ Le système affiche le texte avant et après le traitement et le vecteur de pondération TF-IDF ainsi que la classe appropriée.

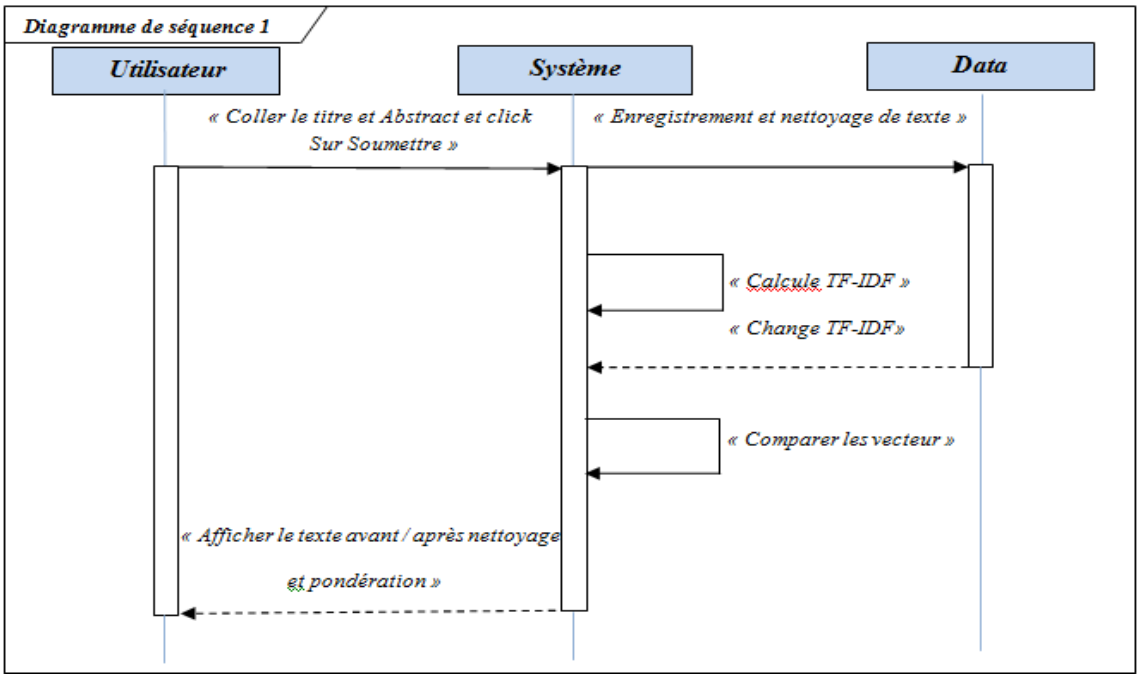


Figure 11 : *Diagramme de séquence du cas d'utilisation « Ajouter un texte ».*

4.4 Diagramme de classe

Il s'agit d'un ensemble d'éléments statique qui montre qui montre la structure d'un modèle (les classes, leur type, leur contenu et leur relation)

Description des classes

La classe	Les attributs	Les méthodes
Utilisateur	//	Ajouter (un document)
Texte	Titre Abstract	//
Classe	Nom	//
Système	//	Classifier

***Tab 5**: Description des classes.*

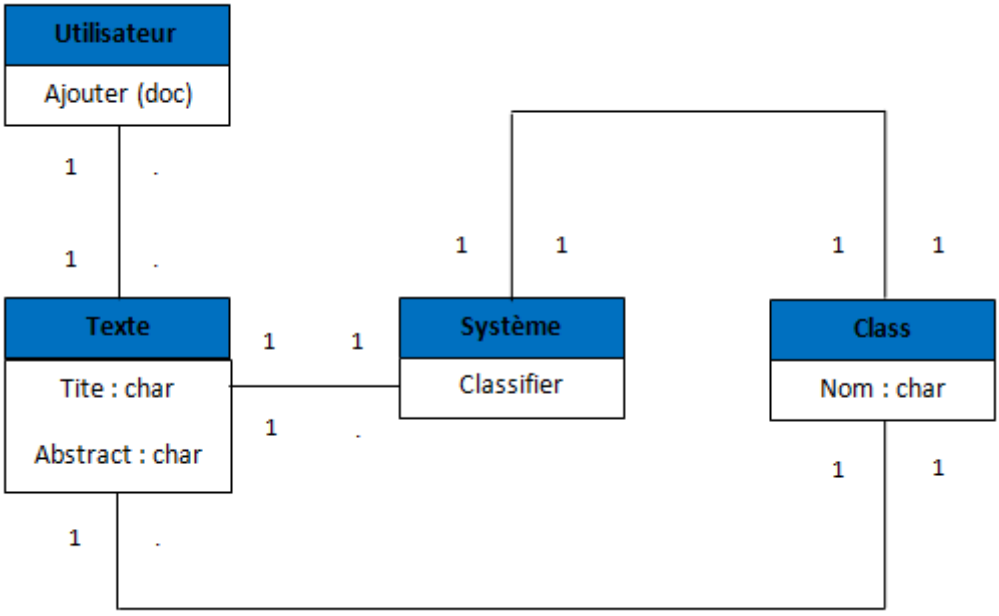


***Figure 12 :** Diagramme de classes.*

4.5 Diagramme d'activités

Le diagramme d'activités permet de mettre l'accent sur les traitements ainsi la représentation graphique du comportement d'une méthode ou le déroulement d'un cas d'utilisation.

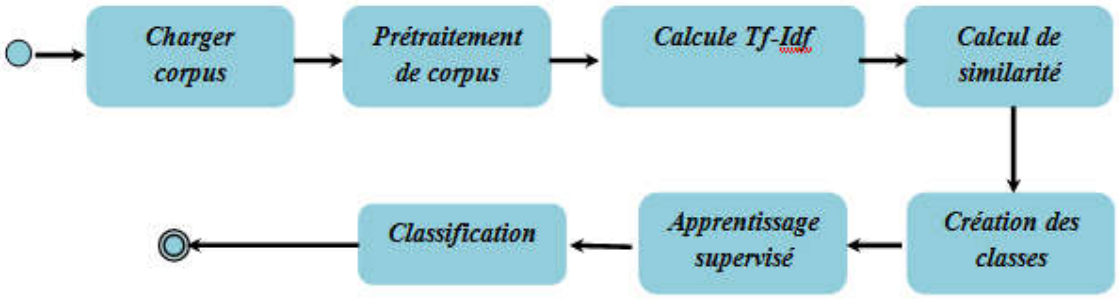


Figure 13 : *Diagramme d'activités représentant la création des classes du corpus.*

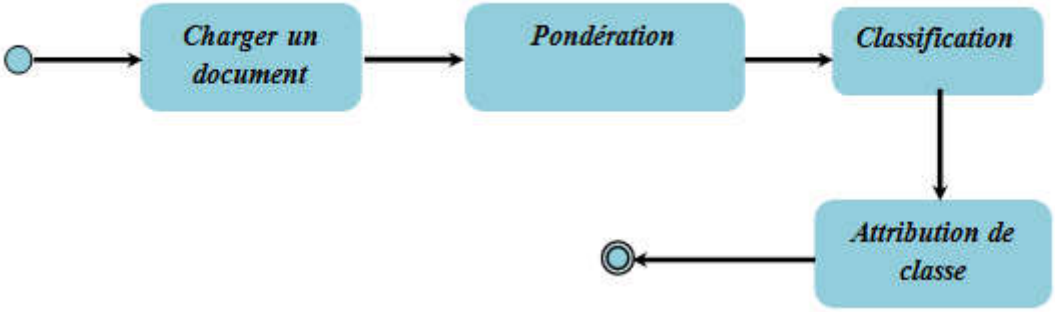


Figure 14 : *Diagramme d'activités représentant les traitements subit par un nouveau document.*

Conclusion

Dans ce chapitre, nous avons effectués la modélisation de notre système de classification des documents textuels. La conception a été faite en utilisant le langage de modélisation UML. Les différents diagrammes de cas d'utilisation, de séquences, de classes et d'activités ont été présentés pour bien expliquer le déroulement de l'application.

Une fois la conception faite, nous arrivons à la phase de développement et de réalisation de l'application qui doit s'appuyer sur la conception.

CHAPITRE IV
RÉALISATION DU SYSTÈME

Introduction

Dans les chapitres précédents nous avons présenté en détail le fonctionnement du système de classification des documents textuels. Le présent chapitre est consacré à la réalisation pratique proprement dite. Nous citons l'environnement de développement mise en place, ensuite nous donnons un aperçu des outils utilisés tel que Weka et le langage de programmation choisi. Enfin, nous terminons par des illustrations des principales interfaces du système.

1 Présentation des outils de développement

1.1 Plateforme matérielle

L'implémentation de l'application est réalisée par un ordinateur portable ayant les caractéristiques suivantes :

- Machine : SONY
- Processeur : Intel (R) Core (TM) i3-3120M CPU
- Fréquence : 2.50 Ghz
- Ram : 4.00 GO
- Carte graphique : Intel (R)
- Système exploitation Microsoft Windows 8 unilingue.

1.2 Plateforme logicielle

Le choix d'un langage de programmation est donc une décision importante. Dans ce qui suit, nous présentons brièvement le langage utilisé pour l'implémentation « JAVA » et l'environnement de développement intégré IDE « Eclipce ».

1.2.1 Langage JAVA

Notre choix du langage de programmation s'est porté sur le langage JAVA, et cela parce qu'il est un langage orienté objet simple ce qui réduit les risques d'incohérence et il possède une riche bibliothèque de classes comprenant des fonctions diverses telles que les fonctions standards, le système de gestion de fichiers ainsi que beaucoup de fonctionnalités qui peuvent être utilisé pour développer des applications diverses. Aussi, il offre un nombre important de fonctions de traitement de texte.

- ✓ **Java Api**

Une interface de programmation (Application Programming Interface AIP) et un ensemble de fonctions procédures ou classes mises à dispositions des programmes informatiques par une bibliothèque logicielle, un système d'exploitation ou un service. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

1.2.2 Environnement de développement Eclipse

Pour le choix de l'environnement de développement, on a opté pour Eclipse car il possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme plugins.
- Supports de plusieurs plateformes d'exécution : Windows, Lunix, Mac OS.
- Malgré son écriture en Java, Eclipse est très rapide à l'exécution grâce à l'utilisation de la bibliothèque SWT.

La version que nous avons utilisée MARS.2.

1.2.3 Weka

Pour obtenir une bonne classification nous avons choisi d'intégrer l'outil WEKA dans notre IDE ; en fait,(Waikato Enveronment Knowledge Analysis) offrent un ensemble d'algorithmes permettant de manipuler et d'analyser des fichiers de donnés. Il permet à l'utilisateur d'implémenter la plupart des algorithmes d'apprentissage entre autres : des SVM, les arbres de décision et les réseaux de neurones [12].

Les algorithmes peuvent soit être appliqués directement à un ensemble de données ou appelés à partir d'un code Java.

Il se compose principalement :

- De classe Java permettant de charger et manipuler des donnés.
- De classe Java pour implémenter les principaux algorithmes de classification supervisée et non supervisée.
- D'outils de sélection d'attributs, des statistiques sur ces attributs.
- De classes permettant de visualiser les résultats.

On peut l'utiliser à trois niveaux :

- Via l'interface graphique, pour charger un fichier de données, lui appliquer un algorithme, et vérifier son efficacité.
- Invoquer un algorithme sur la ligne de commande.
- Utiliser des classes définies dans ses propres programmes pour créer d'autres méthodes, implémenter d'autres algorithmes, comparer ou combiner plusieurs méthodes :

- La version que nous avons utilisée est Weka 3.6.3.
- Nous avons choisi les SVM comme un algorithme d'apprentissage supervisé pour tous les points forts que nous avons présentés.

1.2.4 Langage XML

XML est un format fondé sur des balises comme HTML mais il d'écrit le contenu plutôt que la présentation du contenu. Les balises peuvent avoir aussi des attributs, et cela fait aux langages de programmation à objets.

Nous avons choisi ce format pour la manipulation de nos documents car les données décrites par XML sont sous une forme structurée (les balises) arborescente et extensible ce qui permet de décrire les données brutes. Il est ainsi facile à parser le contenu des documents par des logiciels tout en étant lisible par les humains [20].

L'API JDOM

JDOM est une API open source java, vue comme un modèle de documents objets dont le but de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML.JDOM propose une intégration de SAX, DOM XSLT et XPath.

Un document XML est encapsulé dans un objet de type Documents. Les éléments d'un document sont encapsulés dans des classes dédiées : Elément, Attribut, Text, Processing Instruction, Namspace, Comment, DocType, EntityRef, CDATA, JDOM permet aussi de vérifier que les données contenues dans les éléments respectent les normes XML.

JDOM propose des réponses à certaines faiblesses de SAX et DOM .La simplicité d'utilisation de JDOM lui permet d'être une API dont l'utilisation est assez répondue.

2 Descriptions du corpus utilisé

Le corpus est une collection qui comprend environ 500 documents à la langue anglaise. Ces documents sont téléchargés depuis le site de partage de publications scientifiques « citeulike ».

Il comporte 23 classes avec 330 document documents pour l'enemble d'apprentissage et 170 pour l'ensemble de test.

Le tableau suivant montre une description détaillée de notre corpus :

N°	Classe	Apprentissage	Test	N°	Classe	Apprentissage	Test
01	Chat	15	09	13	Méthodologie	02	00
02	Collaborative Web	04	00	14	Protein Kinase	03	10
03	Biochemistry	18	05	15	Geophysics	03	00
04	Fossils	02	00	16	Cholesterol	03	00
05	Modeling	11	00	17	Mathematics	02	00
06	Echology	02	00	18	Spectroscopy	225	106
07	Dopamine	03	00	19	Collisions	02	00
08	Genetics	01	02	20	Literatures	00	04
09	Cortex	02	00	21	Communication	00	22
10	Particle physics	01	02	22	Hydrolysis	00	02
11	Immunology	04	00	23	Biodiversity	00	02
12	Divers	27	06		Doc 27		

Tab 6 : *Répartition des documents du corpus utilisé.*

3 Description du système

Dans cette partie nous nous intéressons à la présentation de notre système en montrant quelques exemples de captures d'écran des différentes interfaces réalisées.

Nous avons essayé de créer une interface graphique qui montre le plus possible les détails d'exécution de notre application.

Au lancement de l'application la fenêtre suivante s'affiche :

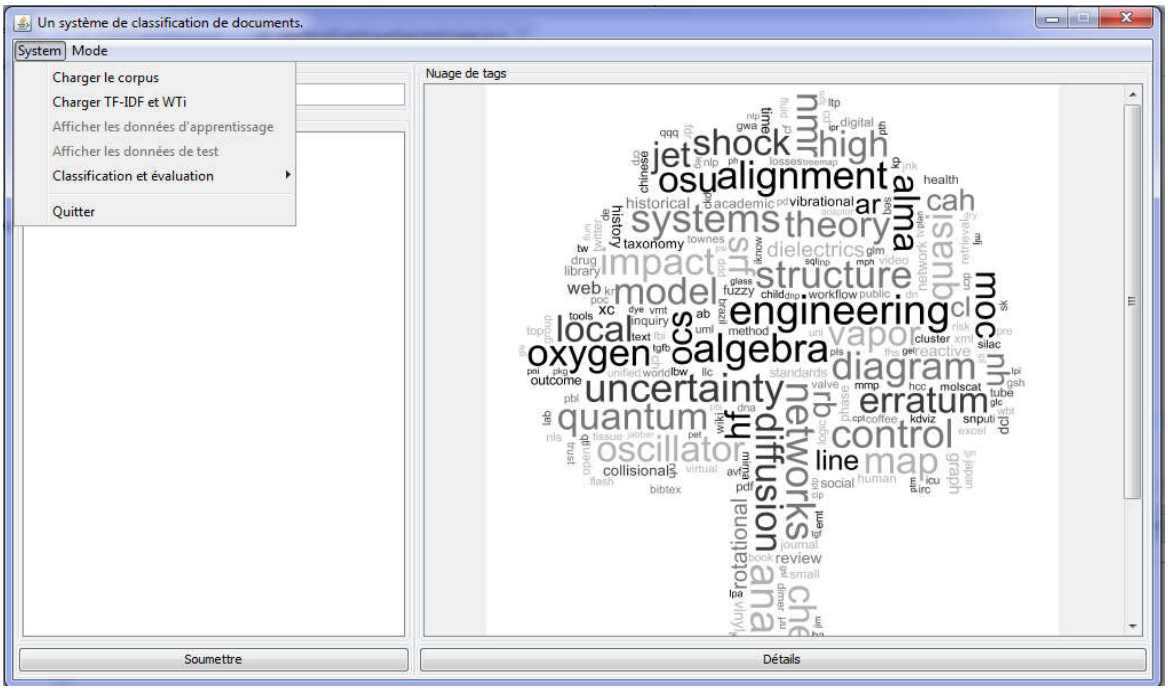


Figure 15 : *Interface d'accueil de système.*

La fenêtre principale précédente montre l'interface principale de notre application, elle permet d'accéder aux différents traitements décrits précédemment.

Le premier item du menu « system » permet le chargement du corpus de texte et le transformer d'un format « .txt » au format « XML » lorsqu'on clique sur cette commande la fenêtre suivante s'affiche pour choisir le nombre de document d'apprentissage et de test.

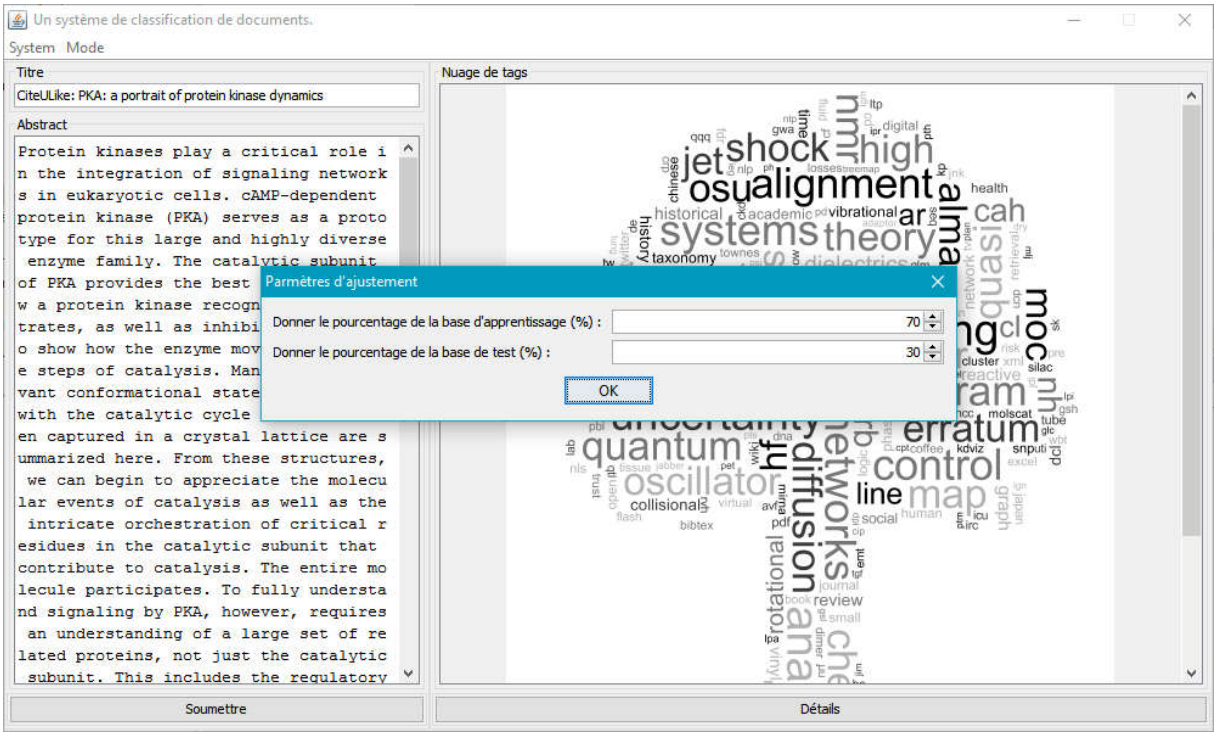


Figure 16 : *Paramètre d'ajustement du classifieur.*

Une fois les paramètres sont choisis pour le bouton « OK », le message suivant s'affiche :

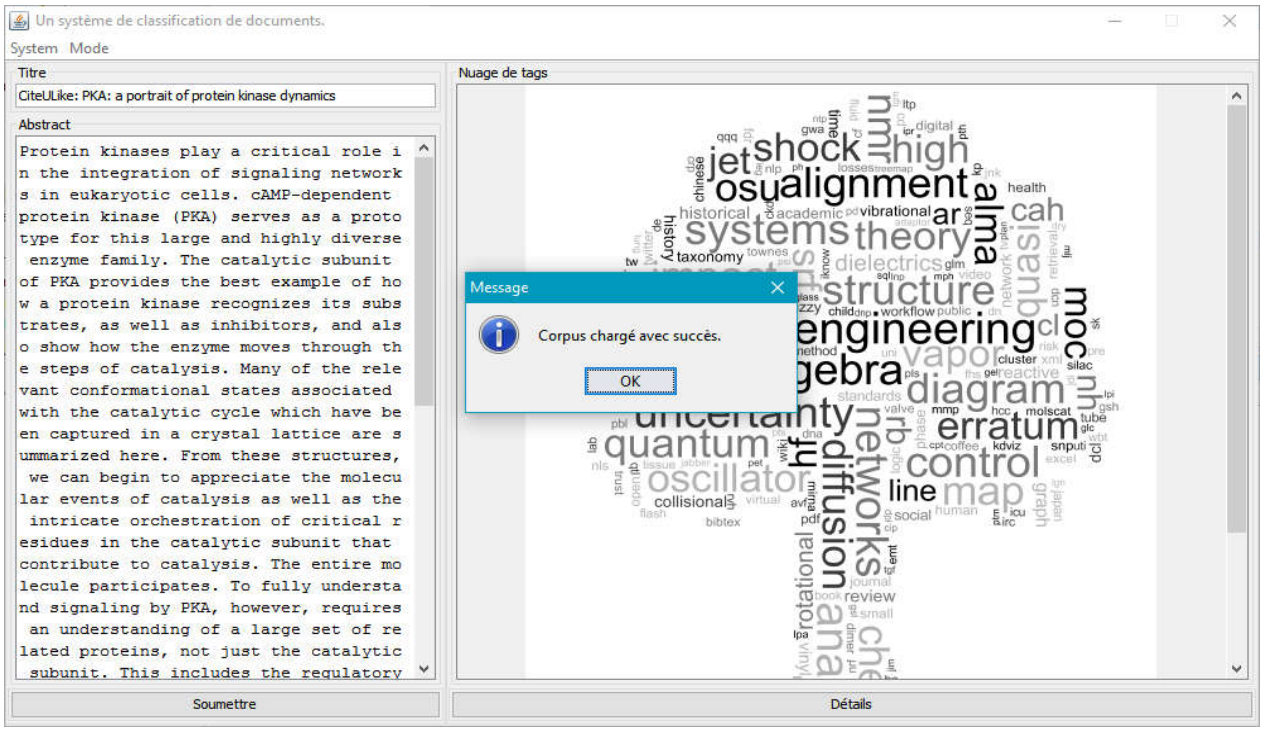


Figure 17 : *Chargement du corpus avec succès.*

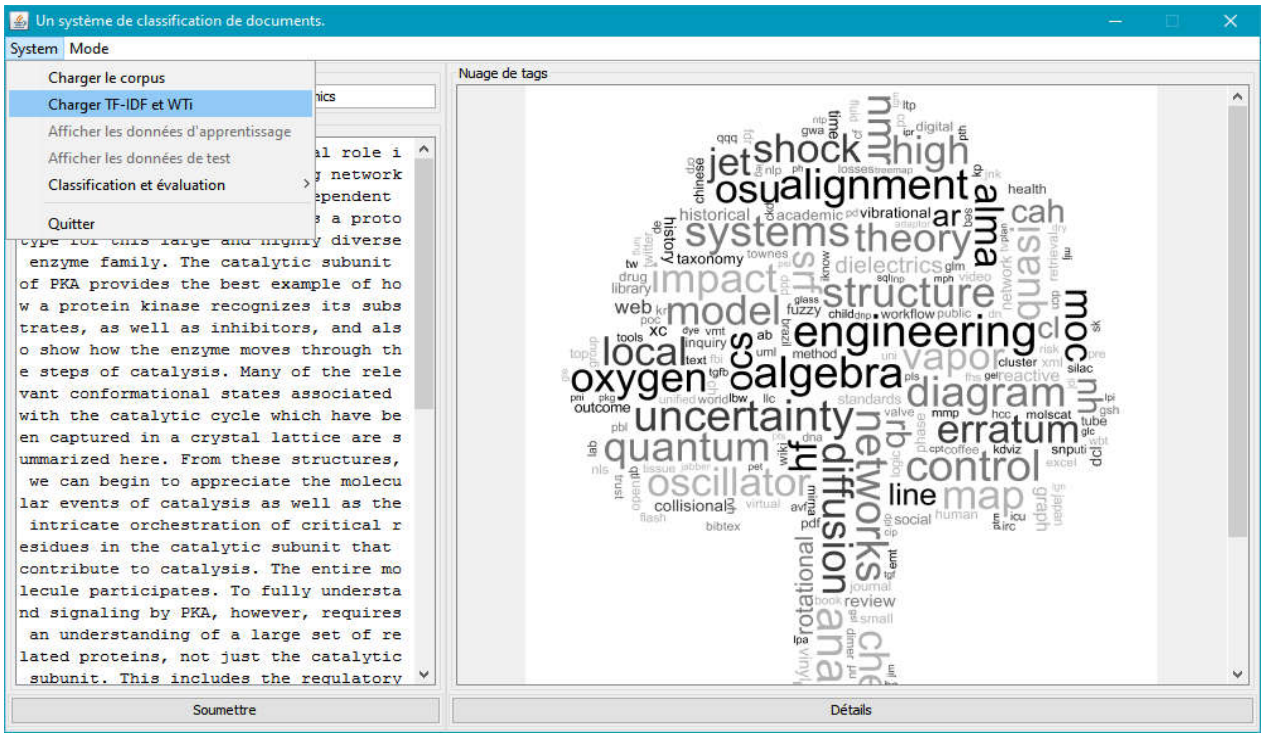


Figure 18 : *Menu de système comment chargé le calcul tf-idf.*

- Chargé tf-idf : avec un clic gauche sur ce bouton le vecteur de terme pondéré est calculé et chargé pour qu'on puisse entamer la classification d'un de 500 vecteurs.

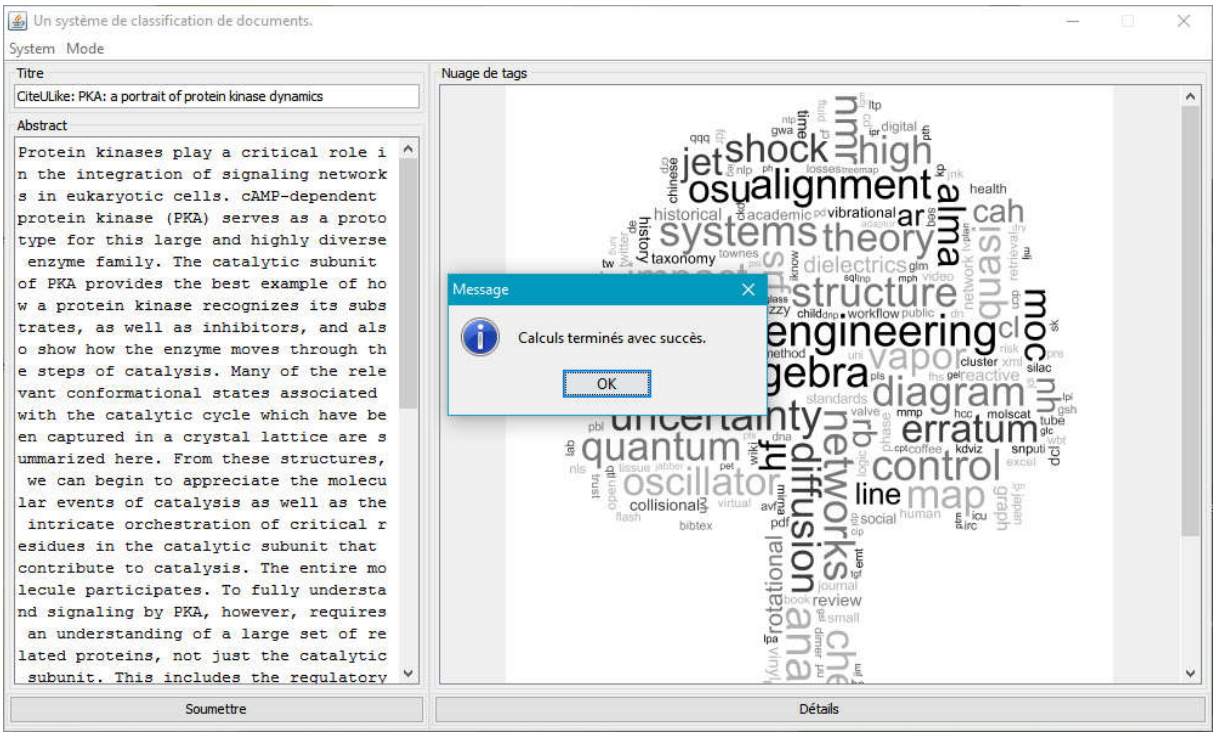


Figure 19 : *Message de chargement de vecteurs de pondération.*

- Apprentissage : cette interface contient la liste des documents du corpus d'apprentissage et il permet d'afficher tous les documents avant et après le nettoyage et l'affichage de vecteur de tf-idf de tout le corpus à l'aide de la fonction de pondération comme il la montre la figure suivante :

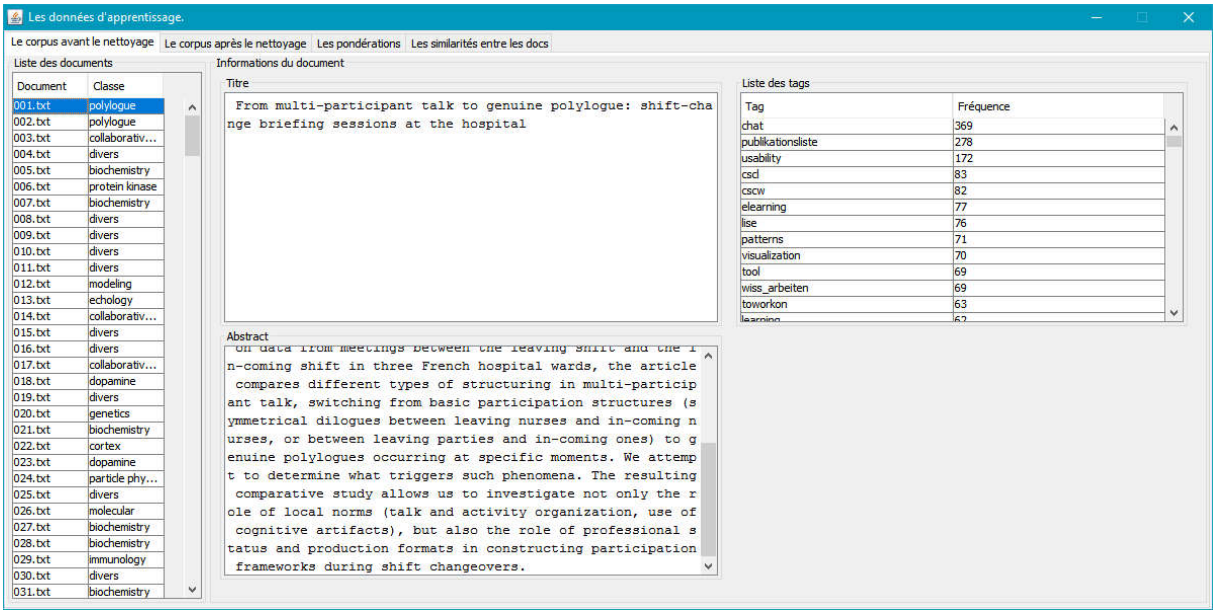


Figure 20 : *Interface d'apprentissage avant le nettoyage.*

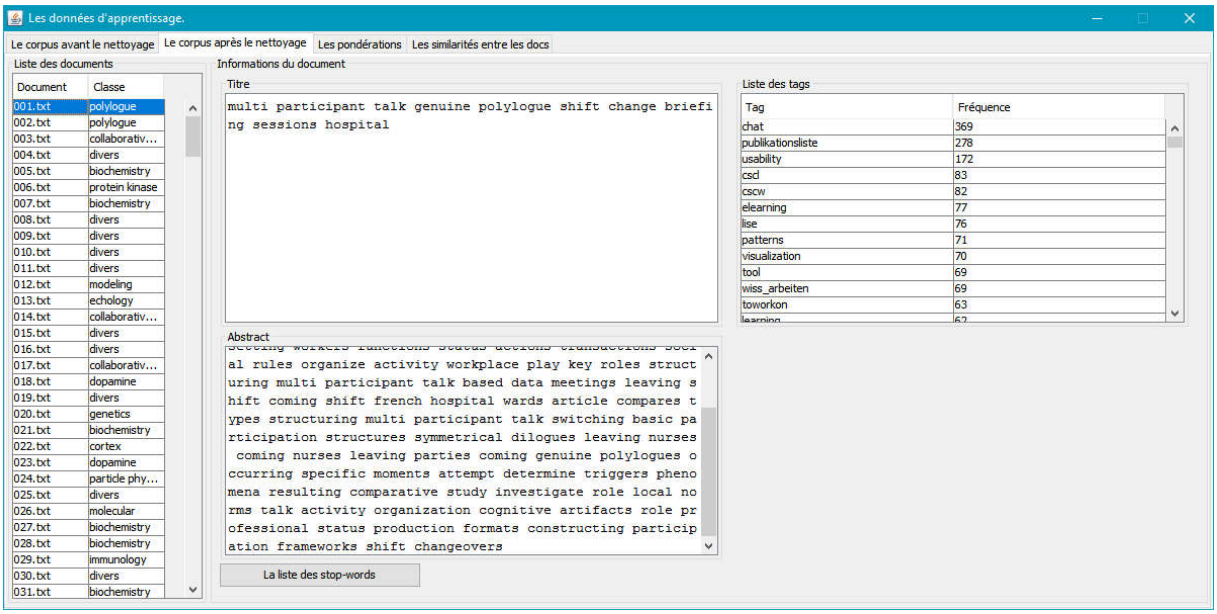


Figure 21 *: Interface d'apprentissage après le nettoyage.*

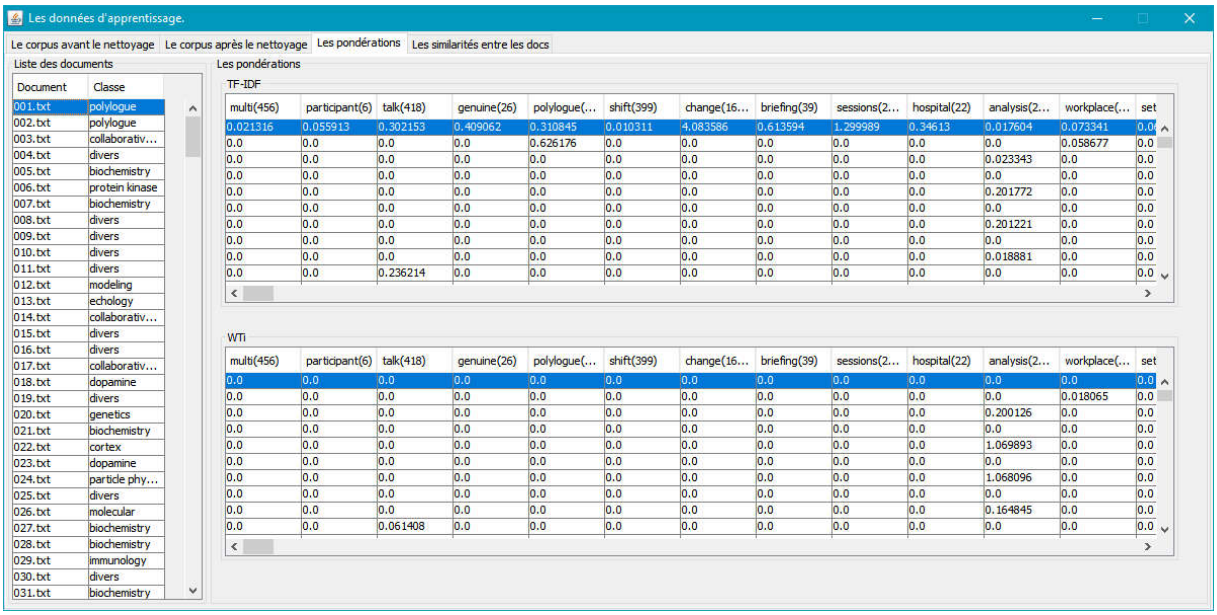


Figure 22 : *Vecteurs de termes pondérés du corpus.*

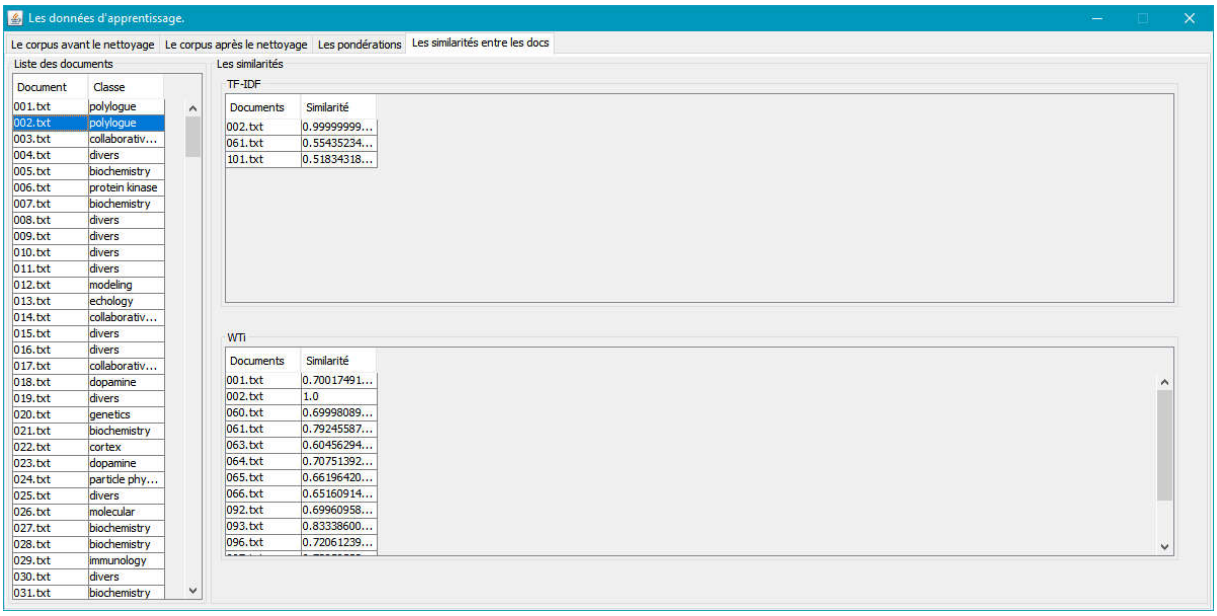


Figure 23 : *Interface de similarités entre les documents.*

Cette phase montre le nettoyage de corpus a chaque fois qu'on on ajoute un nouveau document :

a-Corpus avant nettoyage :

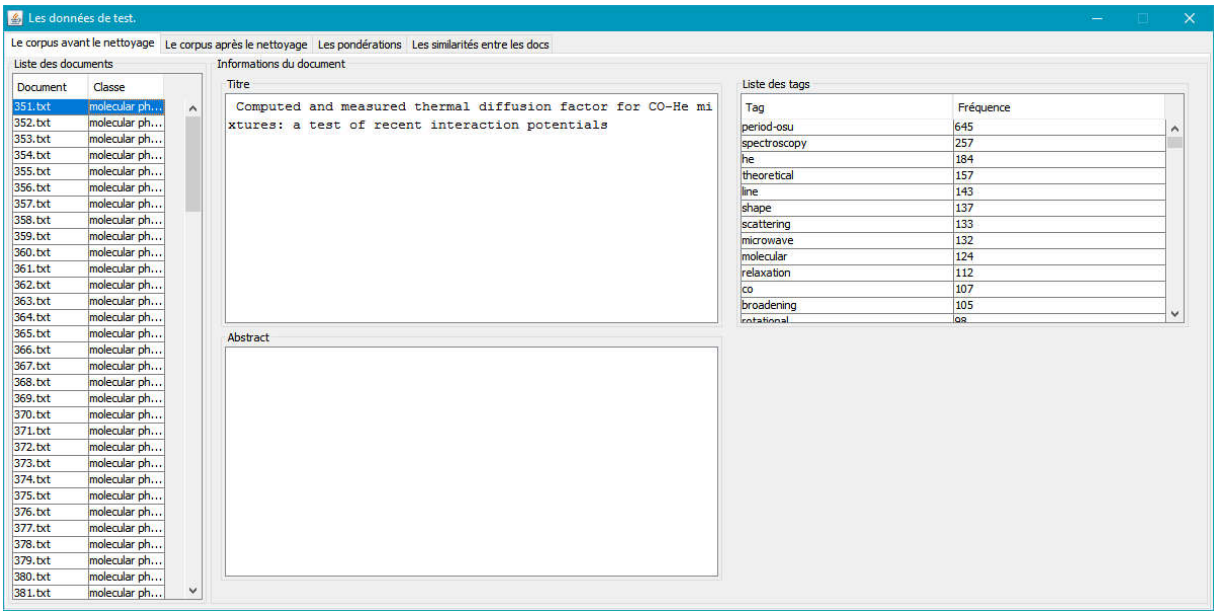


Figure 24 : *Le corpus avant le nettoyage.*

b-Corpus après le nettoyage

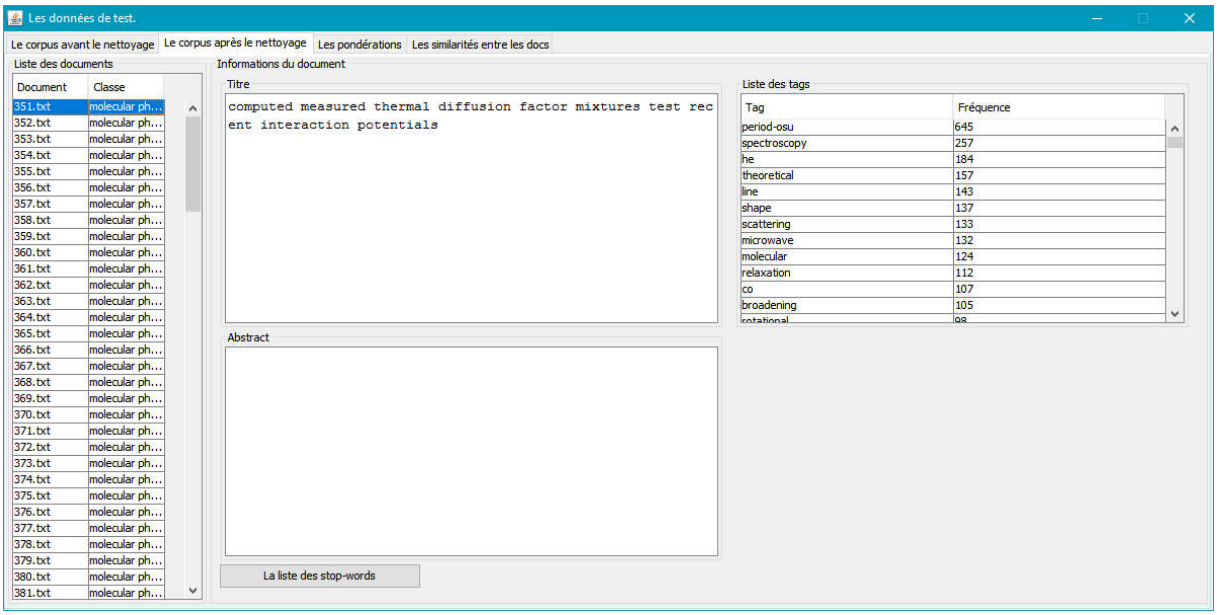


Figure 25 : *le corpus après le nettoyage.*

1. Affichage de document avant nettoyage dans les zones : titre et abstract ;
2. Affichage de document après nettoyage dans les zones : titre et abstract ;
3. Affichage de vecteur de pondération tf-idf lorsque nous cliquons sur la commande pondération.

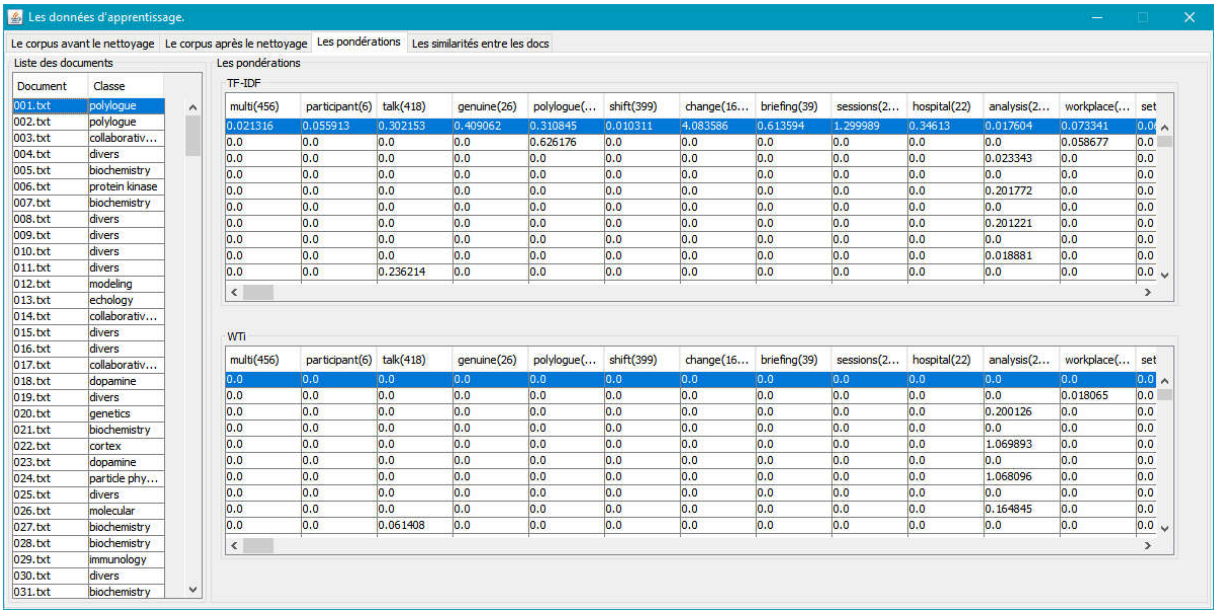


Figure 26 : *Pondération des vecteurs de termes.*

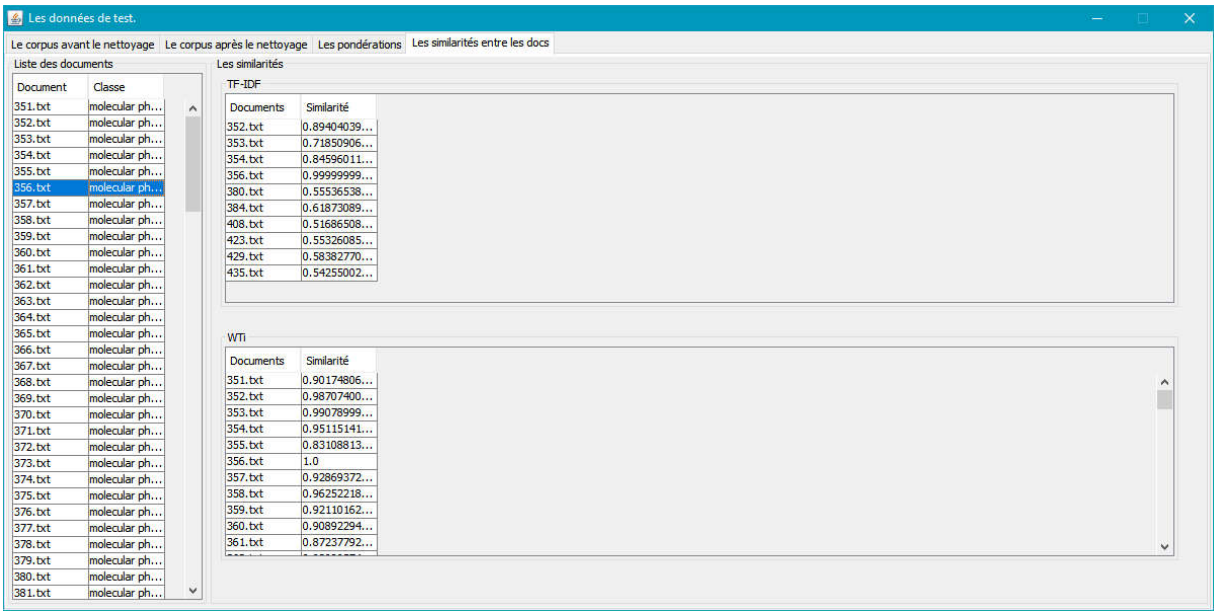


Figure 27 : *Les similarités entre les documents.*

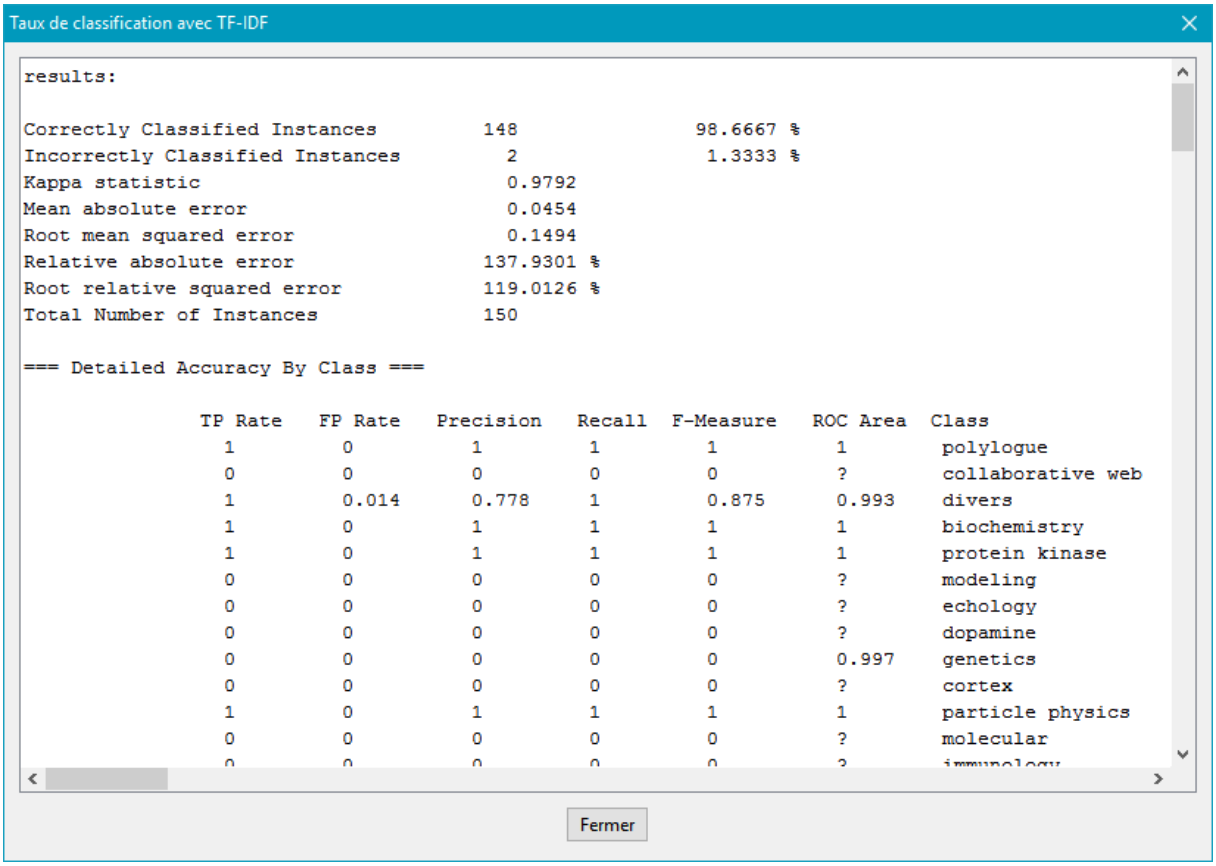


Figure 28 : *Taux de classification avec la formule tf-idf.*

- Quitter : pour quater l'application.
- Mode : pour choisir le mode de calcul le vecteur ***tf-idf*** (chargement des vecteurs existant ou bien refaire le calcul).

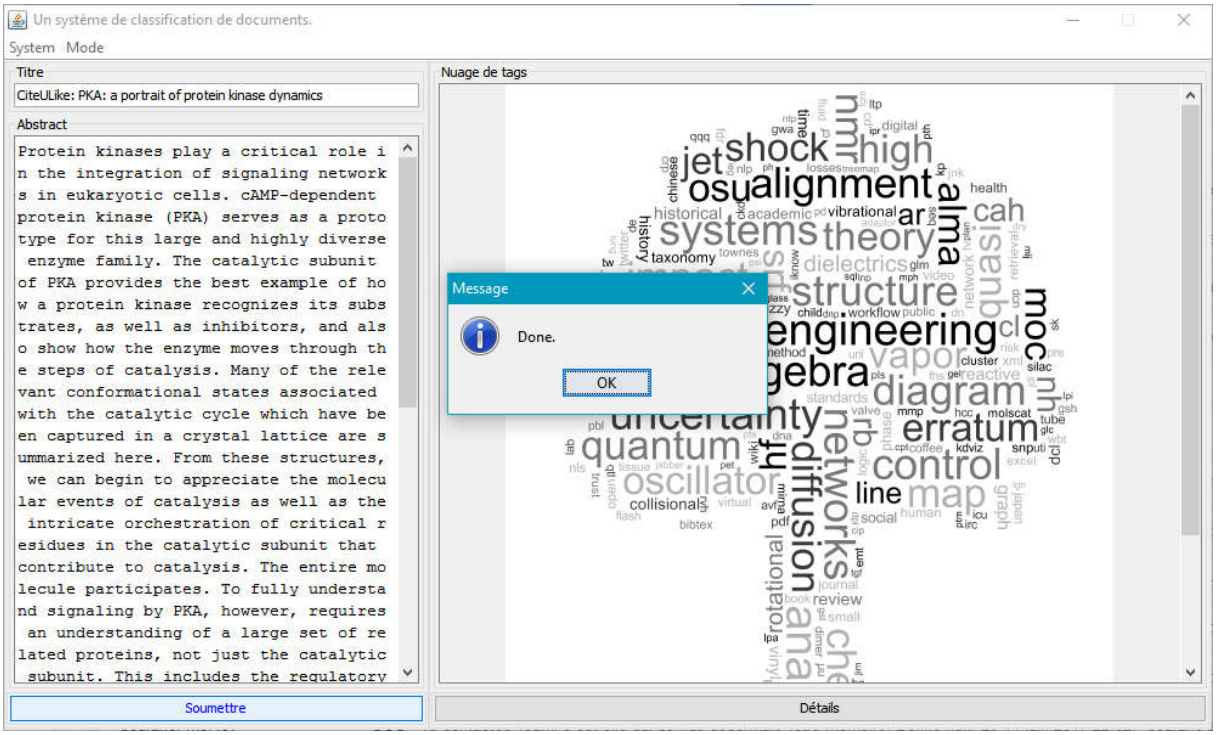


Figure 29 : *Interface affiche la classification du document.*

- Titre : champs de texte pour copier le titre d'un nouveau document.
- Abstract : champs de texte pour copier l'abstract d'un nouveau document.
- Soumettre : lancement de prétraitement du titre et abstract.
- Détail : pour accéder à l'interface de résultat et d'évaluation.

Donc une fois un nouveau titre et abstract sont soumis à notre système nous devons cliquer sur le bouton soumettre pour essayer de prédire sa thématique ou catégorie. Une fois la soumission terminée donc nous pouvons désormais cliquer sur le bouton détails pour voir les résultats obtenus.

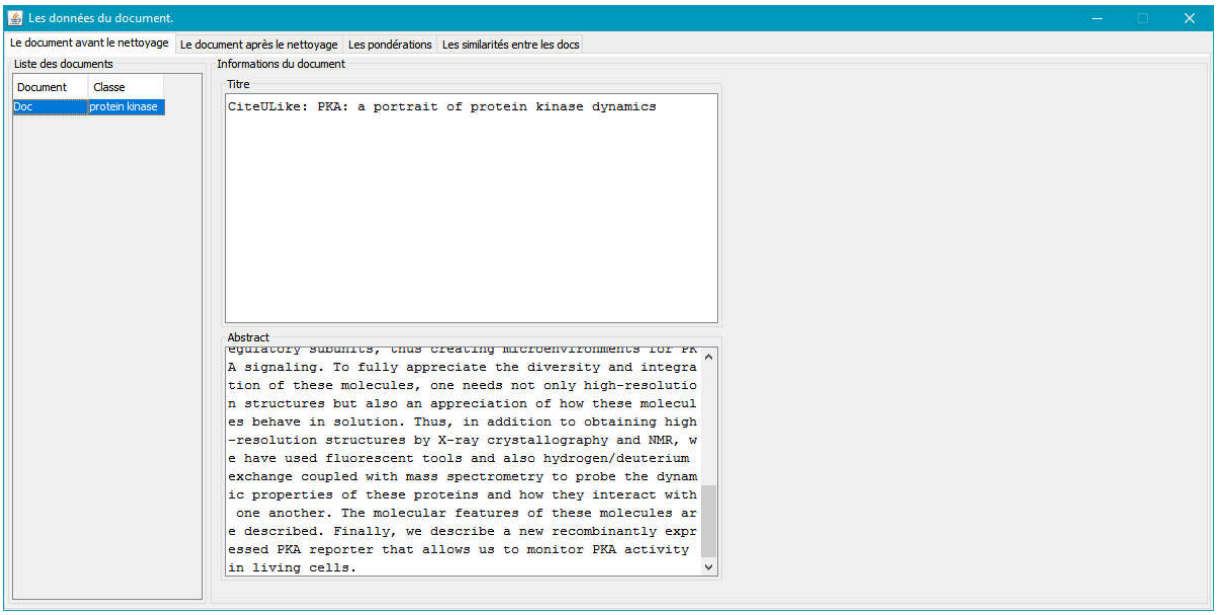


Figure 30 : *L'affichage de classe du document.*

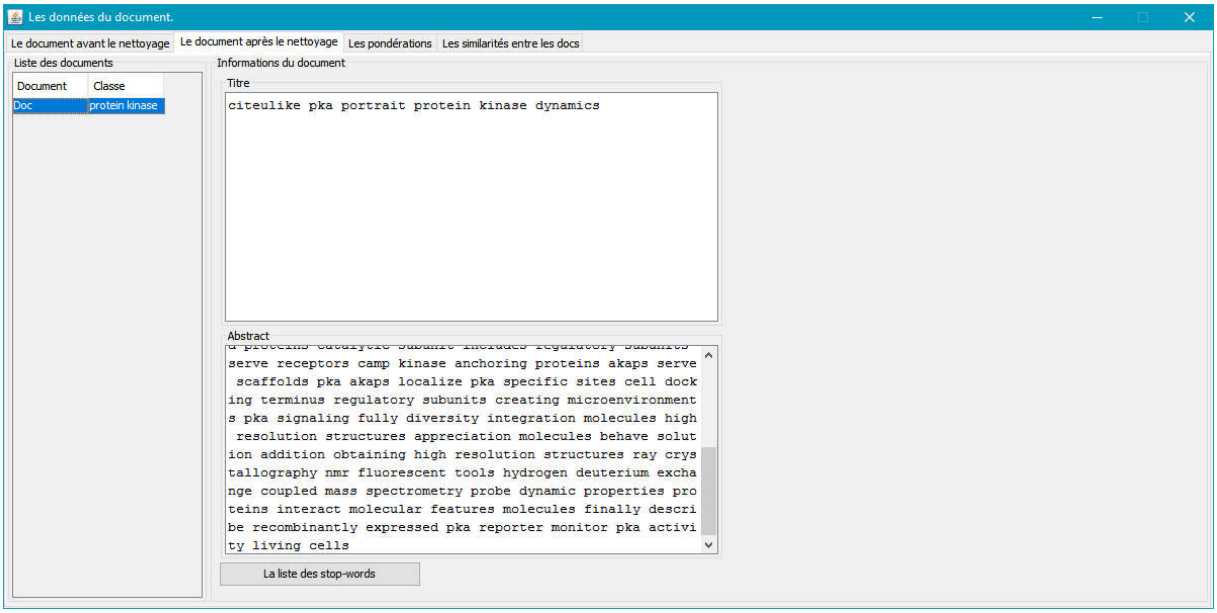


Figure 31 : *Interface de prétraitement d'un nouveau document.*

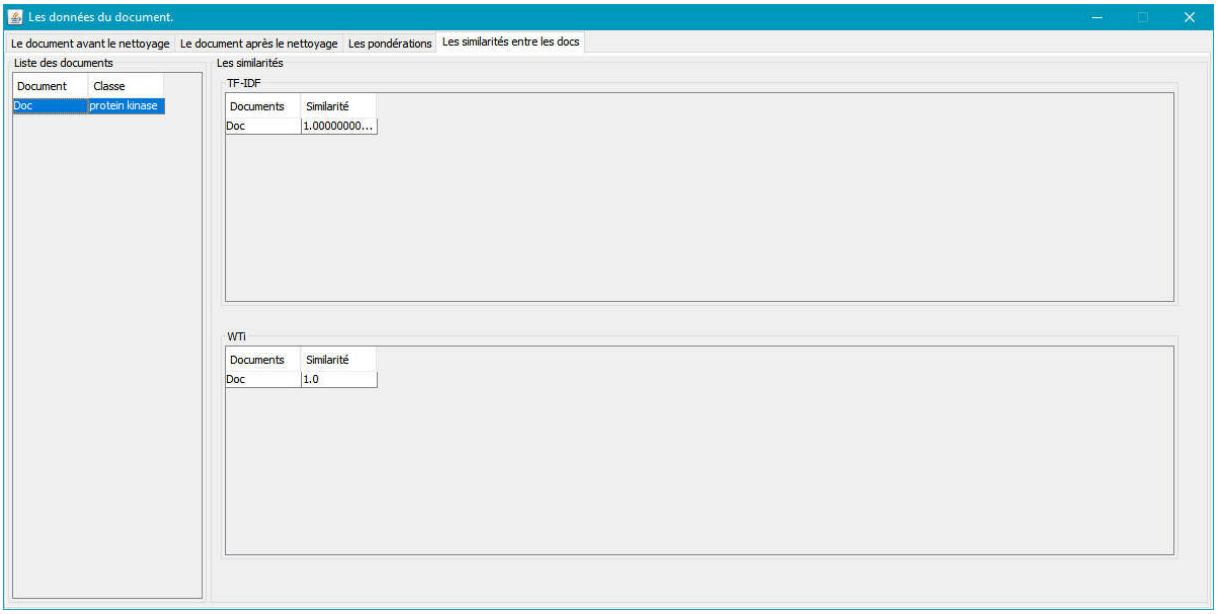


Figure 32 : *les résultats de calculs.*

Remarque :

Les fenêtres présentées par les figures 30, 31, 32 montrent les résultats de calculs attribués au nouveau texte soumis, elle montre par exemple sa classe prédite, son contenu avant et après nettoyage ainsi que son vecteur TF-IDF.

5. Expérimentation et Résultats

Vu à la grande taille du corpus, la partie prétraitement a été très longue notamment à la phase calcul des poids qui à duré environ 12 heures avec la formules ***tf-idf*** pour les 500 documents. Comme ce calcul se répète identiquement pour chaque test effectué, nous avons modifié l'algorithme pour stockés ces résultats dans un fichier et les charger dans la mémoire à chaque utilisation, cela nous a permis un gain de temps énorme, en effet les tests d'apprentissage et de classification ne nécessitaient plus que quelque minute.

Donc, cette nouvelle formule de pondération à montrer leur efficacité pour la réduction du temps de calcul et l'amélioration de taux de classification des documents**.**

- **Discussion des résultats :**

Dans cette section, nous discutons les résultats obtenus, on parle des taux de classification .

- Taux de classification

Nous avons utilisés les systèmes SVM pour la classification, et on obtenu un taux de classification = 98% à base des vecteurs ***tf-idf*** avec un corpus de test comportant 170 document. La classification de base ***tf-idf*** donne un taux d'erreur 0.02%, parce que le nombre de document traités appartenant à cette classe est petit.

Conclusion

Dans ce dernier chapitre, on a présenté une vue complète sur notre système à partir des différentes interfaces capturées, ainsi les outils utilisés pour l'implémentation.
L'exécution de système est bien déroulée, et il a fournit des résultats fiables comme prévue.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Conclusion Générale et Perspectives

Notre mémoire rentre dans le cadre de la Recherche d'Information. Les techniques traditionnelles de la RI représentent le contenu textuel d'un document (requête) par un ensemble de mots clés .Nous intéressons de récupérer les thématiques d'un document donné comme entré pour notre système.

Nous sommes intéressés à proposer des solutions permettant à mieux représenter le contenu textuel des documents et des requêtes pour arriver à le classer dans son meilleur domaine.

Donc, l'objectif principal de notre système est de déterminer automatiquement la catégorie d'appartenance d'un texte et de récupérer les thématiques d'un document donné comme entré pour notre système. Pour valider notre implémentation nous avons utilisé un corpus d'apprentissage téléchargé depuis le site de partage de publications scientifiques Citeulike. Les résultats obtenus sont assez satisfaisants, de efforts de prétraitement ont augmenté considérablement la qualité de la classification.

Les problèmes majeurs dans le domaine de la recherche et la catégorisation automatique des documents, se trouvent confrontés à des obstacles durant le processus de catégorisation tels que :

- La difficulté d'exploiter des collections de textes volumineux.
- La difficulté de choisir l'algorithme de catégorisation le plus performant.

Aussi, nous envisageons d'enrichir le corpus utilisé dans le but d'augmenter le taux de précision dans le cas des documents appartenant aux classes à nombre réduit de documents.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

- ✓ [1] Ouvrage Recherche d'information - Applications, modèles et algorithmes. Fouille de données, décisionnel et big data. Broché– 12 avril 2013 de Massih-Reza Amini (Auteur), Éric Gaussier (Auteur).
- ✓ [2] Thèse de "Doctorat en informatique thème « Recherche d'Information un model de langue combinant mots simple et mots composés » présenté par Me HAMMACHE Arezki université de tizi ouzou .
- ✓ [3] Thèse de master « Un modèle de reformulation des requêtes pour la recherche d'information sur le Web » – ABBASSI-MEFTAH.2016
- ✓ [4] Thèse de master « Un système de Classification et de la recherche de documents textuels sur le web collaboratif » université de Constantine.
- ✓ [5] Master Informatique M2 : Apprentissage pour la recherche d'information textuelle.
- ✓ [6] https://fr.wikipedia.org/wiki/Recherche_d%27information
- ✓ [7] http://snowball.tartarus.org/algorithmes/french/stemmer.html.
- ✓ [8] Httpm// snowbal .tartarus.org/algorithmes/french/stop.txt
- ✓ [9] http://snowball.tartarus.org/algorithms/english/stemmer.html
- ✓ [10] https://fr.wikipedia.org/wiki/Classification_et_cat%C3%A9gorisation_de_ documents.
- ✓ [11] https://wikimonde.com/article/Classification_et_cat%C3%A9gorisation_de_document
- ✓ [12] http//trec.nist.gov/data/reuters/reuters.html.
- ✓ [13] https://en.wikipedia.org/wiki/Unified_Modeling_Language
- ✓ [14] http://www.cs.waikato.ac.nz/ml/publications/1995/Garner95-WEKA.pdf
- ✓ [15] http://www.ims.uni-tuttgart.de/projekte /corplex/TreeTagger
- ✓ [16] www-connex.lip6.fr/~gallinar/
- ✓ [17] Miller D.H, Leek T., Schwartz R., 1999, A hidden Markov model information retrieval system, Sigir'99
- ✓ [18] Ogilvie P. Callan J., 2003, Combining document representations for known item search, Sigir'03
- ✓ [20] Piwowarski B. Gallinari P, A bayesian framework for xml information retrieval : Searching and learning with the

- [21] Inex collection. *Information Retrieval* , 8:655{681, 2005.
- [22] Appear Transactions on Information Systems.
- [23] Workshop of the initiative for the evaluation of XML retrieval (INEX), pages 84-94, 2005.
- [24] Vittaut N., Gallinari P., Machine Learning Ranking for Structured Information Retrieval, in Proc. ECIR'06
- [25] Baeza-Yates R, Ribeiro-Neto B., 1999, Modern Information Retrieval, Addison-Wesley
- [26] Manning D., Schütze H., 1999, Foundations of statistical natural language processing, MIT press
- [27] Christopher D. Manning, Prabhakar Raghavan and Heinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.
- [28] Hiemstra. D., Using language models for information retrieval. PhD thesis, University of Twente, 2001.
- [29] Hofmann T., Probabilistic latent semantic indexing, SIGIR 1999
- [30] Kazai G., Lalmas M., de Vries A.P., 2004, the overlap problem in content-oriented XML retrieval evaluation,
- [31] Mass Y. and Mandelbrod. M., 2005, Component ranking and automatic query renement, In INEX 2004, Lecture
- [32] Notes in Computer Science, volume 3493, pages 154 - 157. Springer-Verlag 2005.
- [33] Miller D.H, Leek T., Schwartz R., 1999, A hidden Markov model information retrieval system, Sigir'99
- [34] Ogilvie P. Callan J., 2003, Combining document representations for known item search, Sigir
- [35] Piwowarski B. Gallinari P, A bayesian framework for xml information retrieval : Searching and learning with the
- [36] Inex collection. *Information Retrieval* , 8:655{681, 2005.
- [37] Piwowarski B. Gallinari P. Dupret G., 2005, Precision recall with user modelling : Application to xml retrieval. To
- [38] Appear Transactions on Information Systems.
- [39] Robertson S., Zaragoza H., Taylor M., 2004, Simple BM extension to multiple weighted fields, CIKM'04
- [40] Sigurbjornsson B., Kamps J., and Rijke M. University of amsterdam at inex 2005. In Pre- Proceedings of the 4th
- [41] Workshop of the initiative for the evaluation of XML retrieval (INEX), pages 84-94, 2005.
- [42] Vittaut N., Gallinari P., Machine Learning Ranking for Structured Information Retrieval, in Proc. ECIR'06
- [43] Zaragoza H., Crasswell N., Taylor M. Saria S. Robertson S., Microsoft Cambridge at TREC-13 : Web and Hard tracks, NIST

RESUME

La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information, elle est aujourd'hui un champ pluridisciplinaire, intéressant même les sciences cognitives, elle a plusieurs domaines parmi ces derniers la catégorisation qui connait des améliorations et des innovations importantes dont le but est de faire apprendre à une machine comment attribuer un texte à la bonne catégorie. Face à l'accroissement du volume d'information disponible en ligne, la catégorisation automatique de texte s'impose de plus en plus comme une étape essentielle qui permet d'accélérer, cibler et d'améliorer la recherche d'information.

Dans ce mémoire nous présentons un système de prédiction de thématiques d'un document donné en entrée. Nous disposons Pour cela d'un ensemble de textes pour lesquels la catégorie est connue (corpus d'apprentissage et qui nous servent à entrainer un classifieur qu'il sera ensuite testé sur d'autres documents auxquels la catégorie est connue également (corpus de test). L'objectif de notre approche est d'améliorer les performances et l'efficacité des systèmes de recherche et de classification de textes pour mieux cibler l'information pertinente basant sur le pouvoir discriminatif des termes. Faire. A la fin nous obtiendrons un système qui est capable de prédire les thèmes enfouis dans un document donné.

Dans notre travail nous avons proposé d'adapter une formule de pondération à ce qui a permis de réduire le temps de calcul de vecteur de pondération et d'améliorer significativement le taux de classification.

Mots Clés: RI, SRI ,TF-IDF,SVM, classification automatique de texte, représentation vectorielle, mesures de similarité, Apprentissage.

ملخص

البحث عن المعلومات هو فرع من فروع علم الحاسوب الذي يركز على اكتساب وتنظيم وتخزين والبحث واختيار المعلومات، هو الآن حقل متعدد التخصصات، للاهتمام حتى العلم المعرفي، لديها العديد من المناطق بين التصنيف الأخير الذي يعرف التحسينات والابتكارات الهامة التي تهدف لتعليم آلة كيفية تعيين النص إلى الفئة الصحيحة. وفي مواجهة تزايد حجم المعلومات المتاحة على الإنترنت، التصنيف التلقائي للنص أصبحت أكثر وأكثر خطوة أساسية أن يسرع، والتركيز، وتحسين استرجاع المعلومات.

في هذه الورقة نقدم نظام للتنبؤ مواضيع المدخلات وثيقة معينة. لدينا ذلك لمجموعة من النصوص التي يعرف الفئة (الإحضار التدريب والتي نستخدمها لتدريب المصنف من شأنها أن يكون من الممكن اختبارها على الوثائق الأخرى التي يعرف الفئة أيضا (اختبار الإحضار). والهدف من نهجنا هو تحسين أداء وكفاءة نظم البحوث وتصنيف النص إلى أفضل المعلومات المستهدفة ذات الصلة على أساس القوة التمييزية للمصطلحات. ل. في النهاية نحن سوف تحصل على نظام قادر على التنبؤ المواضيع دفن في وثيقة معينة.

في عملنا اقترحنا على التكيف مع صيغة الترجيح لما تقليل الوقت الحوسبة الوزن النواقل وتحسن ملحوظ في معدل التصنيف.

كلمات البحث: RI، SRI، TF-IDF-ا، SVM، تصنيف التلقائي النص والتمثيل ناقلات والتدابير التشابه، التعلم.