République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire de Khenchela

Institut des Sciences et Technologies

Département de Mathématiques et Informatique

Ecole Doctorale

Sciences et **T**echnologies de l'**I**nformation et de la **C**ommunication

Option Systèmes d'Informations et de Connaissances

N° Ordre : .....
Réf : .....

Mémoire de Magister

En Informatique

THÈME

Apprentissage automatique et fusion d'informations

Application à l'extraction des connaissances des documents web

Présenté par

AZIZI Nabil

Devant le jury de soutenance composé de :

Mr. DJEDI Noureddine	Professeur	BISKRA	Président.
Mr. MOUSSAOUI Abdelouahab	Maître de conférence	SETIF	Rapporteur.
Mr. BABAHENINI Mohamed Chaouki	Maître de conférence	BISKRA	Examinateur.
Mr. CHERIF Foudil	Maître de conférence	BISKRA	Examinateur.

Promotion 2009 – 2010

Remerciement

En tout premier lieu, louange à dieu le tout puissant qui ma donné la volonté et la force de reprendre mes études après une rupture de longue durée.

Je tien à remercier Abdelouaheb Moussaoui, maîtres de conférences à l'université de Sétif d'avoir accepter d'être rapporteur de mon travail ainsi pour le temps et l'énergie qu'il a consacrés à ce travail, et également ses encouragements, son accompagnement, son soutien et ses conseils scientifiques et personnels durant ce travail.

Je remercie également le professeur Noureddine Djedi, les maîtres de conférence Mohamed Chaouki Babahenini et Foudil Cherif de l'université de Biskra d'avoir accepté de siéger à ce jury et juger mon travail.

Un grand merci à mon directeur Smail Boukhrissa pour son soutien et sa patience, qui m'a permis de mener à bien ce travail.

À ma famille, mon énorme soutien indéfectible et sans elle ce travail n'aurait pu être mené. Merci pour leur patience.

Table des matières

Liste des tableaux

Table des figures

Liste des Algorithmes

Introduction générale

Ces dernières années, l'accès à l'information textuelle a connu une évolution rapide, avec en particulier le développement de grandes bases de données textuelles et du web. En particulier, il est devenu important d'être capable de traiter d'énormes quantités de données textuelles, d'apporter des solutions diversifiées aux nouvelles demandes des utilisateurs, et d'automatiser les outils qui permettent d'extraire et d'exploiter l'information textuelle. Les méthodes classiques d'extraction de connaissances échouent la plupart du temps parce qu'elles n'utilisent en fait qu'une seule source d'information. De plus, la diversité de langues dans un même document, d'une part, et la diversité de sens qu'on peut avoir pour a même mot rendent l'extraction de connaissances, à partir de ce type de document, une tâche ardue et difficile.

A l'apparition de documents de type HTML, de nouveaux documents appelés documents semi structurés sont apparus. Ce type de document représente un compromis entre les données fortement structurées issues de base de données (données relationnelles par exemple) et les données faiblement structurées issues des communautés document numérique et recherche d'information (documents plats, images ... etc) [Denoyer, 2004].

Le format de représentation le plus utilisé par excellence est le format XML (eXtensible Markup Langage) qui se caractérise par sa simplicité son extensibilité et sa puissance de représentation de n'importe quels types de données.

Contexte de travail

Notre étude s'intéresse à la classification (catégorisation) des documents semi structuré XML. L'apprentissage automatique propose une gamme d'outils qui permettent d'avancer dans cette direction. C'est dans ce cadre que se situe notre travail qui vise à explorer le potentiel des techniques d'apprentissage pour répondre aux besoins de recherche et d'analyse d'information semi structuré comme la méthode à base de SVM, de réseaux de neurones, modèle bayésien,... etc.

Toutes ces méthodes s'accordent sur l'efficacité et la robustesse. Afin de tirer profit des avantages de chacune d'elles nous avons proposé une architecture de fusion d'information permettant

d'améliorer la classification des documents semi structurés en tenant compte de la structure et du contenu de ces documents.

Contribution

Notre contribution dans le cadre de la classification des documents XML se situe à deux niveaux :

1. Concernant les concepts de la classification de textes, des documents semi structurés et de fusion d'informations. Nous proposons une synthèse détaillée des différents concepts de ces domaines en les classant dans plusieurs catégories et en présentant les caractéristiques et méthodes principales de chacun d'eux.
2. Concernant notre apport direct au domaine, nous pouvons le résumer en trois points :
   (a) la proposition d'un nouveau modèle de représentation des document XML en tenant compte, à la fois, et de la structure et du contenu de ces document.
   (b) pour valider notre modèle, nous avons appliqué quelques algorithmes de classification en optant pour les plus adéquats.
   (c) finalement, nous avons proposé un schéma de fusion d'information qui a amélioré les performances de classification.

Organisation du mémoire

Ce manuscrit s'articule autour de quatre chapitres :

– Le premier chapitre est consacré, en premier lieu, à la définition de quelques notions importantes pour l'étude, comme l'extraction de connaissances dans les bases de données, le data mining et le text mining, avant de décrire, d'une manière plus approfondie, la catégorisation de textes (classification supervisée) qui est l'une des tâches du texte mining. Les différentes étapes de la catégorisation ainsi que les algorithmes d'apprentissage les plus utilisés font également office de ce même premier chapitre.
– Dans le deuxième chapitre nous nous sommes intéressé à la classification automatique de documents XML qui est un domaine de recherche très actif en particulier pour définir des modèles de représentations de documents qui étendent les modèles traditionnels (basé sur le contenu) en tenant compte de la structure du texte. Certaines méthodes de classification réduisent les documents XML à leur partie purement textuelle, tandis que d'autres prends en compte la structure qui véhicule une information très riche. Nous commencerons ce chapitre par une présentation des principes des documents et langage XML. Nous présentons, ensuite, une synthèse de quelques travaux de la classification de documents XML.

– Dans le troisième chapitre et afin de comprendre la théorie de la fusion d'informations. Nous commençons notre étude par des définitions de cette notion, nous présentons les différentes niveaux et stratégies de la fusion. Ensuite nous présenterons quelques méthodes et techniques de fusion qui nous ont paru les plus importantes dans la littérature parmi de nombreuses méthodes possibles.
– Le quatrième chapitre, quant à lui, est consacré à la présentation de notre contribution pour la classification des documents XML.

Chapitre 1

La fouille de données textuelles

1.1 Introduction

La fouille de données textuelles (*text mining*) est devenue un domaine de recherche d'une grande envergure, en raison de la quantité énorme d'information textuelle en format numérisée provenant de divers domaine (scientifiques, techniques, littéraires, journalistiques, historiques, échange web . . . etc). De plus en plus que nous somme submergés par cette information, il nous est de moins en moins possible de lire des informations qui nous intéressent. Par conséquence nous avons besoins des outils qui sont capable d'extraire des informations essentielles, de découvrir des connaissances non connus, non triviales, implicites, utiles et compréhensibles à partir d'une grande masse de données textuelle, sans recours à lire les textes disponibles, dans le but de nous aider à prendre les décisions adéquates.

Le *text mining* peut être vu comme une extension du data mining, cependant il est plus complexe puisque il traite des données floues et non structurées dans leurs natures.
Le *text mining* est un champ de recherche multidisciplinaire impliquant des domaines des domaines diverses comme : la recherche d'information, l'analyse textuelle, l'extraction d'information, le clustering, la catégorisation et l'apprentissage automatique . . . etc.

Nous allons commencer ce chapitre par des définitions de quelques notions importantes pour notre étude, à savoir l'extraction de connaissances dans des bases de données (section 1.2.1), le data mining et le text mining (section 1.2.2), ensuite nous présentons d'une manière plus approfondie l'une des tâches du texte mining qui est la catégorisation de textes (classification supervisé) (section 1.3), en détaillant les étapes de catégorisation ainsi que les algorithmes d'apprentissage les plus utilisés. Enfin nous concluons le chapitre par une étude comparative des méthodes de classification présentées (section 1.3.11).

1.2 Text Mining

Le text mining est le cœur du processus d'extraction de connaissances à partir de textes. il est considéré comme une instance de data mining appliquer à des données textuelles. Pour

bien comprendre cette notion, une définition de l'extraction de connaissances dans des bases de données fera l'objet de la section suivante.

1.2.1 Extraction de connaissances dans des bases de données (ECD)

Bien que le terme de fouille de données (*data mining*) désigne tout le processus de la découverte des connaissances, il ne constitue en fait, qu'une seule parmi plusieurs étapes du processus général ECBD (en anglais *Knowledge Discovery in Databases - KDD*).

Le *data mining* est défini comme *un processus non trivial consistant à identifier, dans les données, des schémas nouveaux, valides, potentiellement utiles et surtout compréhensibles et utilisables* [Fayyad *et al.*, 1996b].

L'*ECD* peut être vu comme : *" l'activité consistant à analyser un ensemble de données brutes pour en extraire des connaissances exploitables. Les connaissances sont des éléments qui possèdent une syntaxe et une sémantique, formalisées dans un langage de représentation de connaissances. Les connaissances sont manipulées dans un Système à Base de Connaissances (SBC) pour résoudre des problèmes et effectuer des raisonnements. Un raisonnement permet d'inférer de nouvelles connaissances à partir de connaissances existantes "* [Cherfi, 2004].

Le processus *ECD* comprend globalement trois phases [Laur, 2004] :

- **Préparation des données** : l'objectif de cette phase consiste à sélectionner uniquement les données potentiellement utiles de la base. L'ensemble des données est ensuite soumis à un pré-traitement, afin de gérer les données manquantes ou invalides (opération de nettoyage). L'étape suivante dans cette phase consiste à formater ces données, pour les rendre compréhensibles au processus de fouille de données (opérations de transformation et réduction).
- **Extraction** : en appliquant des techniques de fouilles de données (*Data mining*), l'objectif de cette phase est de mettre en évidence des caractéristiques et des modèles contenus intrinsèquement et implicitement dans les données. Il s'agit également de proposer des modèles ou motifs représentatifs du contenu de la base.
- **Interprétation des résultats** : le but de cette dernière phase est d'interpréter la connaissance extraite lors de l étape précédente, pour la rendre lisible et compréhensible par l'utilisateur et permettre ainsi de l intégrer dans le processus de décision.

Un système d'ECBD [Simon, 2000] s'articule autour de quatre composantes :

1. Une ou plusieurs bases de données et leurs systèmes de gestion. Un système d'ECBD doit être capable de traiter des masses de données volumineuses. Le passage à l'échelle d'une petite à une grande application doit se faire de façon transparente pour l'analyste ;

2. un système à base de connaissances qui permet, à la fois, la gestion des connaissances et la résolution de problèmes liés au domaine des données. Le SBC utilise une base de connaissances (par exemple une ontologie du domaine) qui est enrichie grâce aux nouvelles connaissances inférées par le SBC ;
3. un système de fouille de données (FdD) pouvant s'appuyer sur des techniques symboliques comme l'extraction des règles d'association, l'induction par des arbres de décision ou sur des techniques statistiques ;
4. une interface se chargeant des interactions avec l'analyste et de la visualisation des résultats. L'analyste est chargé de guider les recherches et de valider les connaissances extraites. Il est donc au centre de ces quatre composantes.

1.2.2 Fouille de textes (*text mining*) un paradigme de l'ECD

La fouille de textes, ou text mining, est introduite au milieu des années quatre-vingt-dix sous le terme Knowledge Discovery in Textual Databases (KDT).

Il existe plusieurs définitions dans la littératures, mais nous avons retenu celles qui nous sembles les plus intéressantes pour notre travail :

Définition 1 c'est «*le processus d'extraction de connaissances utiles, non triviales à partir des documents textuelles non structurées*» [Ah-hwee, 1999]

Définition 2 Le text mining est définit aussi comme le « *processus non trivial d'extraction d'informations implicites, précédemment inconnues, et potentiellement utiles, à partir de données textuelles non structurées dans de grandes collections de textes* » [Ibekwe-Sanjuan et Sanjuan, 2004].

Nous considérons la fouille de textes (FdT) comme un paradigme de l'ECD au sens où le processus de FdT prend modèle sur celui de l'ECD, c'est-à-dire que c'est une instance de l'ECBD appliquée aux textes.

Nous décrivons le processus de FdT par la figure 1.1 qui est calquée sur le schéma de l'ECBD présenté dans [Fayyad *et al.*, 1996a] et montre les différentes étapes de traitement dans un processus de FdT. Les données traitées sont constituées d'un ensemble de textes. Chaque texte est représenté par un ensemble de mots-clés. Cette représentation est stockée dans une base de données. Nous considérons un texte comme une entité porteuse d'une information qu'il faut préparer, représenter et organiser pour que nous puissions utiliser des outils de fouille de données et valider les résultats de la fouille.

La transformation des données textuelles en connaissances se compose donc de trois principales étapes [Cherfi, 2004] :

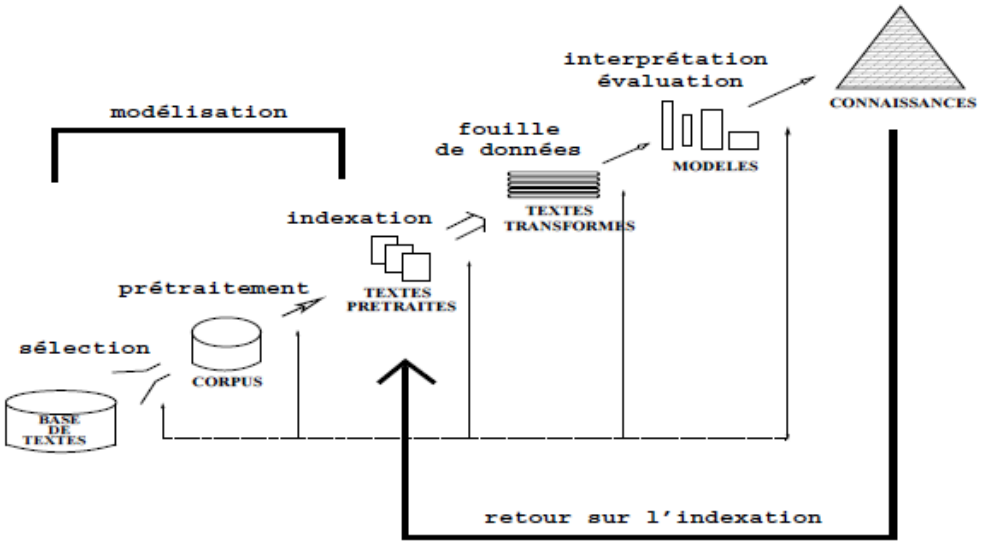


Figure 1.1 – *La chaîne de traitement pour le processus de fouille de textes*

1. **La modélisation du contenu des textes** qui permet d'extraire les données à partir d'une source textuelle. Nous nous appuyons sur une représentation de type, c-à-d qu'un texte est un ensemble de mots-clés, qui est une représentation également utilisée en recherche d'information. Cette représentation nous permet, par la suite d'appliquer les outils de data mining.
2. **Les outils de data mining** de la même façon que pour un processus d'ECBD, les outils de data mining constituent le module calculatoire d'un système de text mining. Les algorithmes de fouille de données que nous réutilisons et adaptons ont montré leur intérêt par la capacité à traiter de grandes masses de données, ce qui nous permet d'envisager de traiter les données très volumineuses extraites des textes.
3. **Le module d'analyse des résultats et leur validation** La contribution de humaine est indispensable pour les étapes d'analyse et la validation des connaissances potentielles extraites car ces deux étapes ne peuvent pas se faire de façon automatique. Le processus de text mining est semi-automatique, ce n'est qu'une fois les résultats validés qu'ils prennent le statut de connaissances. Ces connaissances peuvent alimenter une base de connaissances ou être exploitées à nouveau par le processus de text mining afin d'affiner la modélisation des textes.

L'étape de la modélisation du texte abouti à une représentation vectorielle, cette représentation est ensuite exploité pour [Agard, 2007] :

- **La catégorisation de textes (classification supervisé)** qui consiste à classer les documents en fonction de leurs contenus. Les catégories de classifications doivent cependant

être définies *a priori* par l'utilisateur. Après une phase d'apprentissage en utilisant des algorithme de data mining sur un ensemble de documents classifiés, il est possible de catégoriser de nouveaux documents.

- **Le clustering de textes (classification non supervisé)** qui consiste à regrouper les documents de contenus proches et organiser les documents en hiérarchies. Les documents étant représentés par des vecteurs, il suffit de définir des notions de distance de document à document et de document à un ensemble de documents pour être capable de retrouver tous les documents de contenu similaire et/ou classer ces documents selon des arborescences de proche en proche.
- **L'extarction des règles d'association** qui consiste à extraire des caractéristiques dans les textes : associations d'idées, structures communes à différents documents ... etc. Le vecteur de description peut révéler des associations entre les termes utilisés. Ces associations donnent une mesure de la force du lien entre deux (ou plusieurs) termes.
- **La recherche d'information** qui consiste à retrouver un document à partir de son contenu. Le vecteur de description de chaque document peut être facilement parcouru pour retrouver l'ensemble des documents pertinents vis-à-vis d'un ensemble de mots clés constituant une requête, les techniques d'indexation permettant une efficacité accrue. Cependant il est aussi possible de retrouver des documents similaires, même s'ils ne partagent pas forcément un vocabulaire similaire, si le vecteur s'appuie sur des dictionnaires (généralement thématiques).
- **L'extraction d'information** qui consiste à dresser des relations entre les personnes/-lieux/organisations/concepts ... etc.

  Nous avons vu précédemment (section 1.2.2) que le texte non structuré était représenté sous forme de vecteur (structuré), afin de pouvoir utiliser les méthodes traditionnelles du data mining. Il est cependant relativement aisé d'extraire de l'information enrichie des textes. Les auteurs, les adresses, les courriels, les dates, les citations ... etc, sont autant d'informations structurées dans les textes, qu'il suffit de reconnaitre. Ces informations, riches de signification, portent de l'information supplémentaire, généralement non ambigüe, qu'il faudra traiter de manière adaptée (par exemple : Citeseer traite les citations).

Parmi ces tâches, nous accentuons notre étude sur la catégorisation de texte qui sera présenter dans la section suivante.

1.3 Catégorisation de texte

L'étude de la catégorisation automatique des textes remonte au début des années soixante [Maron, 1961]. Puis, son utilisation principale était projetée pour l'indexation des documents scientifique moyennant d'un vocabulaire contrôlé. C'était seulement dans les année 1990 que ce champ soit en pleine maturité avec la disponibilité d'un nombre croissant des documents textuelles en format numérique et qui nécessite une organisation pour un usage plus facile.

Actuellement la catégorisation automatique de texte est appliquée dans une variété de domaine à savoir, le filtrage des Spams, la catégorisation de page Web, la génération automatique des méta-données, la détection du genre des textes, et à beaucoup d'autres.
il y a deux approches principales à la catégorisation des textes.

- La première est l'approche d'ingénierie cognitive dans laquelle la connaissance d'expert au sujet des catégories est directement encodée dans le système, soit d'une façon déclaratif, soit sous forme de règles de classification procédurales.
- L'autre est l'apprentissage automatique (*machine learning*) dans laquelle un processus inductif général établit un classifieur par apprentissage sur un ensemble d'exemples pré-classifiés.

Dans le domaine de gestion de document, les systèmes d'ingénierie cognitive surpassent habituellement les systèmes d'apprentissage automatique, malgré que l'écart en performance se rétrécisse. L'inconvénient principale de l'approche d'ingénierie cognitive est l'exigence d'un expert qualifié de connaissance ainsi que le cout considérable pour créer et maintenir les règles de connaissance. Par conséquent, la majeure partie des travaux récents sur la catégorisation est concentrée sur l'approche d'apprentissage automatique, qui exige seulement un ensemble d'exemples manuellement classifiés qui sont beaucoup moins coûteux pour les produire.

Les prochaines section seront inspirées du travail de [Sebastiani, 2002], qui synthétise les méthodes de classification de textes.

1.3.1 Définition

La catégorisation aussi appelée classification de texte consiste à chercher une liaison entre un ensemble de textes et un ensemble de catégories (*documents, classes*). Cette liaison est un modèle de prédiction, qui est estimée par une opération qui s'appelle apprentissage automatique (*machine learning*) sur un ensemble de textes préalablement étiquetés, dit ensemble d'apprentissage, à partir duquel nous estimons les paramètres du modèle de prédiction le plus performant possible.

D'une façon mathématique, la catégorisation de texte est la tâche d'assigner une valeur booléen pour chaque pair $(d_j, c_i) \in \mathcal{D} \times \mathcal{C}$ où $\mathcal{D}$ est le domaine des documents, et $\mathcal{C} = \{c_1, \ldots, c_{|c|}\}$

est un ensemble des catégories prédéfinies. La valeur V assigné à (d_j, c_i) indique la décision de classer d_j dans la classe c_i, tandis que la valeur F indique la décision de ne pas classer d_j dans la classe c_i. Plus formellement, cela revient à trouver d'une façon approximative la fonction inconnu $\breve{\Phi} :\in \mathcal{D} \times \mathcal{C} \rightarrow \{V, F\}$ (qui décrit comment les document doivent être classifiés) par le moyen de la fonction $\Phi :\in \mathcal{D} \times \mathcal{C} \rightarrow \{V, F\}$ dite le classifier (également règle, hypothèse ou modèle) tel que $\breve{\Phi}$ et Φ «coïncide autant que possible ». [Sebastiani, 2002]

1.3.2 La catégorisation classe unique *vs* multiclasses

- Dans la catégorisation multiclasses (*multilabel*), les catégories se chevauchent, et un document peut appartenir à plusieurs catégories.
- Dans la catégorisation classe unique (*single-label*), chaque document appartient à exactement une catégorie.
- La catégorisation *binaire* est un cas spécial de la catégorisation *single-label* dans laquelle le nombre de catégories est deux.

Le cas binaire est le plus important puisque c'est le plus simple, le plus commun et le plus souvent utilisé pour la démonstration des techniques de catégorisation. De plus la catégorisation classe unique est une simple généralisation de la catégorisation binaire. Dans le cas du catégorisation multiclasses, on peut transformer le problème en $|C|$ classifieurs binaire ($|C|$ est le nombre des catégories)

1.3.3 La catégorisation Articulé-Document *vs* Articulé-Catégorie

En général, les classifieurs sont employés de la façon suivante : étant donnée un document, le classifieur trouve toutes les catégories auxquelles le document appartient. Ceci s'appelle une catégorisation articulé-document (*Document-Pivoted*). Alternativement, il est possible qu'on ait besoin de trouver tous les documents qui devraient être classés dans une catégorie donnée. Ceci s'appelle une catégorisation articulé-catégorie (*Category-Pivoted*). La différence est significative seulement dans le cas de la non disponibilité immédiate de tous les documents ou toutes les catégories. Par exemple, dans la catégorisation «en ligne», les documents entrent un par un, ce qui implique que seule la catégorisation document-articulé est alors possible. D'autre part, si l'ensemble de catégories n'est pas fixé, et si les documents doivent être re-classifiés suivant les nouvelles catégories, alors la catégorisation articulé-catégorie est la plus appropriée.

1.3.4 La catégorisation Dure *vs* Ranking

Si un système conduit à une décision d'appartenance binaire $\{0, 1\}$ d'un document d_i à une classe c_j, on dit que le système fait une catégorisation dure. Le niveau de performance obtenu

par un tel système peut être insuffisant pour certains applications. Alors une approche semi-automatique dont laquelle le système classe les documents par ordre de pertinence pour une catégorie donnée (ou bien classe les catégories par ordre de pertinence pour un document donné) par une fonction de score de la manière suivante $SC : \mathcal{D} \times \mathcal{C} \rightarrow [0, 1]$. Ce système est appelé *Ranking*.

Différentes applications du *ranking* existant :

- La suggestion à un utilisateur pour la répartition d'experts compétents pour évaluer un projet.
- Le ranking de pages Web pour une thématique définie par un utilisateur.
- Le filtrage avec un réglage de seuil de tolérance, le seuil étant adapté par rapport au scores de ranking . . . etc.

1.3.5 Application de la catégorisation de texte

Depuis les travaux de [Maron, 1961], la catégorisation de textes est utilisée dans de nombreuses applications, parmi eux on peut distinguer trois applications principales à savoir :

1.3.5.1 Indexation de texte

Dans un système de recherche d'information (RI), chaque document dans une grande collection est assigné à un ou plusieurs termes clés décrivant son contenu. Puis, le système de RI peut rechercher les documents selon les requêtes de l'utilisateur, qui sont basées sur ces termes. Tous les termes clés appartiennent à un ensemble fini appelé un *vocabulaire controlé*, qui est souvent un thésaurus hiérarchique thématique tel que le thésaurus aérospatial de la NASA. La tâche d'assigner des mots-clés d'un *vocabulaire controlé* aux documents textuels s'appelle l'indexation de texte. Si les mots-clés sont vus comme catégories, alors l'indexation de texte est une instance du problème de la catégorisation de texte.

1.3.5.2 Routage et filtrage de texte

Le *routage* consiste à affecter un document à une catégorie parmi n. Par exemple, dans un journal, les annonces peuvent être classifiées par catégorie dans «personnel», «vente de voiture», «les immobiliers», et ainsi de suite.

Le *filtrage* consiste à déterminer si un document est pertinent ou non, il est considéré comme un routage binaire. La détection de *spams* (les courriers indésirables) pour ensuite les supprimer est l'exemple typique du filtrage.

1.3.5.3 Catégorisation hiérarchique des pages web

Un usage courant de la catégorisation de texte est la classification automatique des pages Web sous des catalogues hiérarchiques par des portails populaires d'Internet tels que Yahoo [1]. Ces catalogues sont très utiles pour une lecture rapide directe et pour limiter la recherche basée sur des requêtes des pages appartenant à un sujet particulier.

La catégorisation automatique de pages web a deux particularités essentielles :

1. La nature hypertextuelle des document
2. La structure hiérarchique de l'ensemble des catégories

1.3.6 Etapes de catégorisation de texte

Le processus de catégorisation selon [Jalam, 2003] intègre la construction d'un modèle de prédiction qui, en entrée, reçoit un texte et, en sortie, lui associe une ou plusieurs étiquettes. Pour identifier la catégorie ou la classe à laquelle un texte est associé, un ensemble d'étapes est habituellement suivies. Ces étapes concernent principalement la manière dont un texte est représenté, le choix de l'algorithme d'apprentissage à utiliser ainsi que l'évaluation des résultats obtenus pour garantir une bonne généralisation du modèle appris.

Le processus de catégorisation, intégrant la phase de classement de nouveaux textes, est résumé dans la figure 1.2. Il comporte deux phases :

1. **Apprentissage :** c'est la phase la plus importante du processus de catégorisation. Elle consiste à construire un modèle de prédiction, en suivant plusieurs étapes :
   (a) la disposition d'une collection de textes préalablement étiquetés (les catégories des document sont connus *a priori*) ;
   (b) à partir de cette collection, nous extrayons les k descripteurs (en anglais *features*, mots, ou termes) $(t_1; ..; t_k)$ les plus pertinents au sens du problème à résoudre ;
   (c) nous disposons alors d'un tableau «*descripteurs* $\times$ *individus* », et pour chaque texte nous connaissons la valeur de ses descripteurs et son étiquette ;
   (d) un algorithme d'apprentissage est appliqué sur ce tableau afin d'obtenir à un modèle de prédiction Φ.
2. **Le classement** d'un nouveau document d_x qui comprend deux étapes :
   (a) recherche puis pondération des occurrences $(t_1; ..; t_k)$ des termes extraits lors de la phase d'apprentissage dans le texte d_x à classer ;
   (b) application du modèle Φ sur ces occurrences afin de prédire l'étiquette de ce texte d_x.

1. http ://www.yahoo.fr

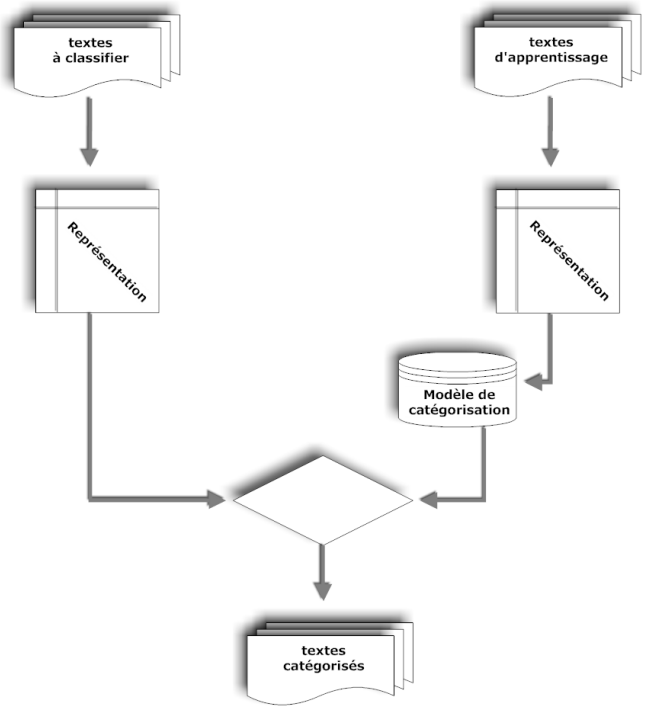


FIGURE 1.2 – *Processus de catégorisation de textes.*

1.3.7 Représentation des documents

Les algorithmes d'apprentissages ne peuvent pas traiter directement un document textuelle dans leur état brut. Donc, durant l'étape de pré-traitement les document sont convertis en représentation plus maniable. Typiquement, les documents sont représentés par des vecteurs de descripteurs ou termes (*features*). Pour la majorité des méthodes d'apprentissage, il faut transformer l'ensemble des textes en une matrice «Document-Termes»où chaque document est représenté dans l'espace des termes :

- un document d_j est un texte étiqueté lors de la phase d'apprentissage, et à classer dans la phase de prédiction.
- les descripteurs sont les termes t_k extraits lors de la phase d'apprentissage.

Le contenu de la matrice est un ensemble de poids w_{jk} du terme t_k dans le document d_j

$$M = \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_k \end{array} \\ \left(\begin{array}{cccc} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{jk} \end{array} \right) \begin{array}{c} d_1 \\ d_2 \\ \vdots \\ d_j \end{array} \end{array}$$

L'enjeu principal de la catégorisation de texte est est le choix des termes à utiliser ainsi que la manière de calculer les poids de ces termes ?

Bien qu'il existe plusieurs pistes pour choisir les termes (mots, lemmes, phrase, etc.), la piste couramment utilisé est le sac à mots (*bag-of-words*).

Les méthodes de donner des poids aux termes peuvent varier. La plus simple est les poids *binaire* dans laquelle le poids du terme est *un* s'il existe dans le document et *zéro* sinon.

$$w_{jk} = \begin{cases} 1 & \text{si } t_k \in d_j \\ 0 & \text{sinon.} \end{cases} \tag{1.1}$$

Des méthodes de pondération plus complexes sont possibles qui tiennent en compte les fréquences des mots dans le document, dans la catégorie, et dans la collection entière. Le principe de pondération par fréquence se résume en deux points : [Jalam, 2003]

1. Plus le terme t_k est fréquent dans un document d_j , plus il est en rapport avec le sujet de ce document.
2. Plus le terme t_k est fréquent dans la collection, moins il sera utilisé comme discriminant entre documents.

Parmi les méthodes de pondération les plus utilisé on peut citer :

1. **Le codage *tfidf*** , qui a été introduit dans le cadre du modèle vectoriel, et qui signifie en anglais «$termfrequency \times inversedocumentfrequency$ »

$$tfidf(t_k, d_j) = freq(t_k, d_j) \cdot \log(\frac{N}{docfreq(t_k)}) \tag{1.2}$$

   tel que $freq(t_k, d_j)$ est la fréquence du terme t_k dans le document d_j, N est le nombre total du documents dans la collection et $docfreq(t_k)$ le nombre nombre de documents qui contiennent le terme t_k.
2. **Le codage *tfc*** , pour prendre en compte la longueur des documents. Ce codage est introduit par la normalisation de *tfidf* en cosinus, afin de ne pas favoriser les documents les plus longs.

$$tfc(t_k, d_j) = \frac{tfidf(t_k, d_j)}{\sqrt{\sum_{s=1}^{N} tfidf(t_s, d_j)^2}} \tag{1.3}$$

1.3.8 Réduction de la dimension

Le nombre de différents mots est grand même dans les documents relativement petits tels que les articles ou les résumés de papier. Le nombre de différents mots dans de grandes collections de documents peut être énorme ; or la dimension de l'espace des termes de sac-à-mots pour une grande collection peut atteindre des centaines de milliers.

La plupart de ces mots sont non pertinents à la tâche de catégorisation et peuvent être abandonner sans aucun effet sur les performances du classifieur et peuvent même avoir comme conséquence l'amélioration dû à la réduction de bruit ainsi que le temps d'exécution des algorithmes d'apprentissage.

D'une façon plus formelle, la réduction de la dimension consiste à trouver un vecteur de nouveaux termes $\mathcal{T}'$ tel que $|\mathcal{T}'| \ll |\mathcal{T}|$.

[Sebastiani, 2002] classe ces techniques de deux façons selon qu'elles agissent localement ou globalement, et selon la nature des résultats de la sélection (sélection de termes ou d'une extraction de termes).

1.3.8.1 Réduction locale

Pour chaque catégories c_i un ensemble de termes $\mathcal{T}_i'$ avec $|\mathcal{T}_i'| \ll |\mathcal{T}|$ est choisi pour la classification sous c_i. Ceci signifie que différents sous-ensembles de $\vec{d_j}$ sont employés en travaillant avec les différentes catégories : c'est à dire que chaque catégorie c_i possède son propre ensemble de termes. Les valeurs typiques sont $10 \leq |\mathcal{T}_i'| \leq 50$. (voir [Apté *et al.*, 1994], [Sable et Hatzivassiloglou, 2000])

1.3.8.2 Réduction globale

Un ensemble de termes $\mathcal{T}'$ avec $|\mathcal{T}'| \ll |\mathcal{T}|$ est choisi pour la classification sous toutes les catégories $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$.

1.3.8.3 Sélection de termes

Les techniques utilisées pour choisir les termes pertinent à partir de la collection sont issues de la théorie de l'information et de l'algèbre linéaire, la table 1.1 ci-dessous nous montre quelque unes.

Les expériences prouvent que ces mesures peuvent réduire la dimension par un facteur de 100 sans perte de la qualité de catégorisation ou même avec une petite amélioration [Yang et Pedersen, 1997].

1.3.8.4 Extraction de termes

Une autre manière de réduire le nombre de dimensions est de créer un nouvel ensemble beaucoup plus petit de termes synthétique à partir de l'ensemble d'origine qui maximisent l'efficacité de classification. La raison de la synthèse des termes sert à remédier aux quelques problèmes du langage naturel tels que la synonymies, la polysémie et l'homonymies par une combinaison linéaire des mots qui ont un lien sémantique important.

fonction	*dénoté par*	*forme mathématique*
Facteur d'association DIA	$z(t_k, c_i)$	$P(c_i\|t_k)$
Gain d'information	$IG(t_k, c_i)$	$\sum_{c\in\{c_i,\bar{c}_i\}} \sum_{t\in\{t_i,\bar{t}_i\}} P(t,c) \cdot \log \frac{P(t,c)}{P(t)\cdot P(c)}$
Information mutuel	$MI(t_k, c_i)$	$\log \frac{P(c_i\|t_k)}{P(t)\cdot P(c)}$
Chi-square	$\chi^2(t_k, c_i)$	$\frac{\|Tr\| \cdot [P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i) - P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)]^2}{P(t_k)\cdot P(\bar{t}_k)\cdot P(c_i)\cdot P(\bar{c}_i)}$
Coefficient NGL	$NGL(t_k, c_i)$	$\frac{\sqrt{\|Tr\|} \cdot [P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i) - P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)]}{\sqrt{P(t_k)\cdot P(\bar{t}_k)\cdot P(c_i)\cdot P(\bar{c}_i)}}$
Score de pertinence	$RS(t_k, c_i)$	$\log \frac{P(t_k\|c_i)+d}{P(\bar{t}_k\|\bar{c}_i)+d}$
rapport des chances	$OR(t_k, c_i)$	$\frac{P(t_k\|c_i)\cdot(1-P(t_k\|\bar{c}_i))}{P(t_k\|\bar{c}_i)\cdot(1-P(t_k\|c_i))}$
Coefficient GSS	$GSS(t_k, c_i)$	$P(t_k,c_i)\cdot P(\bar{t}_k,\bar{c}_i) - P(t_k,\bar{c}_i)\cdot P(\bar{t}_k,c_i)$

Table 1.1 – *quelques fonctions de selection de termes utilisées en catégorisation de texte*

Parmi les approches d'extraction des termes, on peut citer, LSI «Latent Semantic Indexing »proposé par [Deerwester *et al.*, 1990] et «Terme clutering » [Lewis, 1992].

1.3.9 Evaluation des classifieurs

Puisque le problème de catégorisation des textes n'est pas suffisamment bien défini, les performances des classifieurs peuvent être seulement évaluée expérimentalement. Dans n'importe quelle expérience une collection de document préalablement étiqueté est exigée. Cette collection est divisée en deux parties :

1. **Ensemble d'apprentissage** (*training set*), est l'ensemble des documents utilisés pour apprendre un classifieur
2. **Ensemble de test** (*test set*), est l'ensemble des documents utilisés pour mesurer les performance d'un classifieur

Une autre méthode largement utilisée est la validation croisée (*cross-validation*). Dans cette méthode le corpus est divisé en n parties égales sur lesquelles un processus d'apprentissage et de test sont appliqués n fois. Chaque fois deux parties différentes sont utilisées pour l'apprentissage et le test. Le résultat final sera la moyenne des n parties.

1.3.9.1 Mesures des performances

Lors de l'évaluation d'un classifieurs à partir d'un ensemble de test, une **matrice de confusion** (*confusion matrix*) est construite, où chaque colonne de la matrice représente le nombre d'occurrences d'une classe estimée, tandis que chaque ligne représente le nombre d'occurrences

d'une classe réelle. A partir de cette matrice quatre nombres sont calculés pour chaque classe (voir la table 1.2).

1. le nombre de textes correctement classés comme appartenant à la classe i, noté VP_i (pour Vrai Positif).
2. le nombre de textes incorrectement classés comme appartenant à la classe i, noté FP_i (pour Faux Positif).
3. le nombre de textes incorrectement rejetés, noté FN_i (pour Faux Négatif).
4. le nombre de textes correctement rejetés, noté VN_i (pour Vrai Négatif).

Ces nombre sont représentés dans une table appelé table de contingence.

Expert \ Classifieur	c_i	$\neg c_i$
c_i	VP_i	FN_i
$\neg c_i$	FP_i	VN_i

Table 1.2 – *La table de contingence*

Afin d'évaluer un classifieur deux mesures issues de la recherche d'information sont utilisées, Rappel (*Recall*) et Précision (*Precision*). Un rappel ρ_i pour une catégorie c_i est défini comme le pourcentage des documents correctement classifiés parmi tous les documents appartenant à cette catégorie, et la précision π_i est le pourcentage des documents correctement classifiés parmi tous les documents qui ont été assignés à la catégorie par le classificateur.
Autrement dit :

$$\rho_i = \frac{VP_i}{VP_i + FP_i} \ , \ \pi_i = \frac{VP_i}{VP_i + FN_i} \tag{1.4}$$

Un système parfait fournira des valeurs de rappel et de précision égales à 1 (l'algorithme trouve la totalité des documents pertinents - rappel - et ne fait aucune erreur - précision), mais en réalité les algorithme de catégorisation sont plus ou moins précis, et plus ou moins pertinents. Pour trouver un compromis entre rappel-précision, une mesure nommée F-mesure (soit *F-measure* en anglais) combine ces deux valeurs dont la formule est la suivante :

$$F = \frac{2 \cdot (Rappel \cdot Prcision)}{(Rappel + Prcision)} \tag{1.5}$$

Dans le cadre multi-classes (ou i est supérieur à 1), les moyennes globales de la précision et du rappel sur l'ensemble des classes c_i peuvent être évaluées par la moyenne qui calcule d'abord la précision et le rappel sur chaque classe c_i suivie d'un calcul de la moyenne des précisions sur les n classes, on compte deux type de moyenne :

La macro-moyenne (*macro-average*) il s'agit de donner un poids égal 1 à toutes les classes, soit :

$$\rho_{global} = \frac{\sum_{i=1}^{|\mathcal{C}|} \rho_i}{|\mathcal{C}|} \qquad \text{et} \qquad \pi_{global} = \frac{\sum_{|\mathcal{C}|}^{i=1} \pi_i}{|\mathcal{C}|} \tag{1.6}$$

La micro-moyenne (*micro-average*) il s'agit de donner un poids proportionnel à la fréquence de chaque classe, soit

$$\rho_{global} = \frac{\sum_{i=1}^{|\mathcal{C}|} (VP_i + FN_i) \cdot \rho_i}{\sum_{i=1}^{|\mathcal{C}|} (VP_i + FN_i)} \qquad \text{et} \qquad \pi_{global} = \frac{\sum_{|\mathcal{C}|}^{i=1} (VP_i + FN_i) \cdot \pi_i}{\sum_{i=1}^{|\mathcal{C}|} (VP_i + FN_i)} \tag{1.7}$$

dont $(VP_i + FN_i)$ est le nombre de documents de la classe c_i

1.3.10 Les méthodes de catégorisation automatique

Beaucoup de méthodes sont utilisées dans la littérature pour la catégorisation de textes. Nous présentons, dans cette section, les méthodes les plus utilisées selon [Sebastiani, 2002]

1.3.10.1 Les classifieurs probabilistes

Dans un classifieur probabilistes, la $CSV(d_j, c_i)$ (*categorization status value*) est vue comme une probabilité conditionnelle $P(c_i|d_j)$ d'appartenance du document d_j représenté par un vecteur $\vec{d_j} = (w_{1j}, \cdots, w_{|\mathcal{T}|j})$ à la classe c_i, et pour calculer cette probabilité le théorème de Bayes est appliqué :

$$P(c_i|\vec{d_j}) = \frac{P(c_i)P(\vec{d_j}|c_i)}{P(\vec{d_j})} \tag{1.8}$$

Dans 1.8 $P(c_i)$ représente la probabilité qu'un document choisi aléatoirement soit appartient à la classe c_i, et $P(\vec{d_j})$ représente la probabilité de choisir un document $\vec{d_j}$ qui est constante pour toute les classe.

Pour estimer $P(\vec{d_j}|c_i)$ en supposant l'indépendance des termes du document, on obtient :

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|\mathcal{T}|} P(w_{kj}|c_i) \tag{1.9}$$

Les classifieurs résultant de cette hypothèse s'appellent les classifieurs « Bayésien Naïf », parce que l'hypothèse n'est jamais vérifiée et évidemment souvent fausse.

Lors de la phase d'apprentissage les probabilités $P(w_{kj}|c_i)$ sont estimées afin de les employer dans l'étape de classement d'un nouvel exemple selon la règle de bayes.

1.3.10.2 Les arbres de décision

Beaucoup de méthodes de catégorisation partagent quelques inconvénients, par exemple certains classifieurs ne peuvent pas être facilement compréhensibles par des humains (les classifieurs statistiques). En revanche les classifieurs symboliques, dont le classifieur d'arbre de décision est l'exemple le plus fameux, ne souffre pas de ce problème.

Un classifieur d'arbre de décision est une structure arborescente dont les nœuds internes représentent les termes, les deux branches partant de chaque nœud représente l'existence ou l'absence du terme de ce nœud dans le document, et les feuilles représentent les catégories (voir la figure 1.4).

Pour construire un arbre de décision l'algorithme d'apprentissage [Jalam, 2003] prend en entrée un échantillon Ω, comprenant N textes classés (d_j, c_i), et fournit en sortie un arbre de décision. L'algorithme procède de façon descendante : il part de la racine puis, récursivement, choisit l'étiquette des fils.

Il se résume dans les étape suivantes :

- L'algorithme d'arbres de décision construit une succession de partitions sur l'échantillon de données d'apprentissage. Les partitions sont de plus en plus fines. Le premier nœud contient toutes les données de l'échantillon avec leurs classes.
- On cherche, parmi les variables prédictives, celle qui donne la meilleure partition selon un critère de sélection des variables et on répète le processus de segmentation pour chaque nœud obtenu sans se préoccuper des autres nœuds. Si les variables prédictives sont discrètes, chaque variable peut engendrer une partition dont le nombre d'éléments dépend du nombre de valeurs que la variable peut prendre. Si les variables sont continues, elles doivent être discrétisées, soit *a priori*, soit sur chaque nœud.
- Un nœud est saturé s'il n'existe aucune variable (prédictive) qui permet de créer localement une partition qui améliore le critère utilisé.
- Le processus s'arrête quand tous les nœuds sont saturés.
- L'élagage qui consiste à simplifier un arbre de décision en coupant des branches. Il possède deux objectifs, l'un est de simplifier l'arbre de décision, l'autre est de diminuer le sur-apprentissage (augmenter la capacité de généralisation) et, par la même, diminuer le taux d'erreur. Il y'a deux possibilités soit élagage lors de la construction ou élagage après la construction

Algorithme 1: *Algorithme général d'apprentissage par arbres de décision*

Données : un échantillon Ω de S textes classés (d_j, c_i)

Entrées : arbre vide ; nœud courant : racine ; échantillon courant : Ω

```
répéter
    si le nœud courant est terminal alors
        étiqueter le nœud courant par une feuille portant le nom de cette classe;
    sinon
        choisir le meilleur attribut (terme) pour créer le sous-arbre;
    fin
    // nœud courant : un nœud non encore étudié et l'échantillon courant :
       échantillon atteignant le nœud courant
jusqu'à production d'un arbre de décision;
élaguer l'arbre de décision obtenu;
```

Résultat : arbre de décision élagué

Un arbre de décision classe un document en commençant de la racine de l'arbre et en se déplaçant successivement en bas par l'intermédiaire des branches dont les conditions sont satisfaites par le document jusqu'à ce qu'un nœud de feuille soit atteinte. Le document est alors assigné à la catégorie de la feuille. Chaque branche de l'arbre constitue une règle de décision de la forme `si condition alors conclusion`. L'ensemble de ces règles constitue le modèle de prédiction (voir la figure 1.3).

si	((*wheat* et *farm*)	ou
	(*wheat* et *commodity*)	ou
	(*bushels* et *export*)	ou
	(*wheat* et *tonnes*)	ou
	(*wheat* et *winter* et $\neg soft$))	Alors $WHEAT$ sinon $\neg WHEAT$

Figure 1.3 – *Modèle de prédiction produit par un arbre de décision sous la forme de règles*

Il y a un certain nombre d'algorithme d'apprentissage standard pour les arbres de décision, Parmi les plus populaires on peut citer **ID3** (employés par [Fuhr *et al.*, 1991]), **C4.5** (employé par [Cohen et Hirsh, 1998]), et **C5** (employé par [Li et Jain, 1998]).

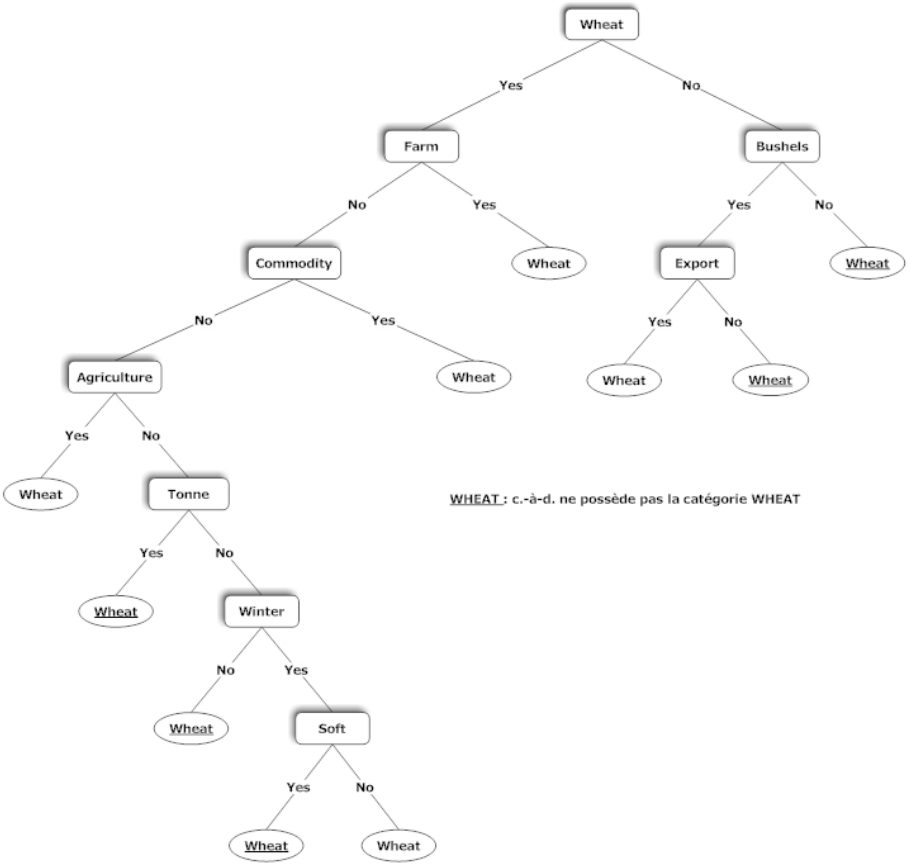


FIGURE 1.4 – *Exemple d'arbre de décision appliquée sur le corpus Reuters*

1.3.10.3 La méthode de Rocchio

La méthode de Rocchio est un classifieur linéaire proposé dans [Rocchio, 1971] pour améliorer les systèmes de recherche documentaires. Il représente les catégories par des profils «prototypiques». Un profil de la classe c_i est une liste de termes pondérés, dont la présence et l'absence discriminent au mieux cette classe c_i. Il se caractérise par sa simplicité et interpretabilité, car, pour un expert, ce profil prototype est plus compréhensible qu'un réseau de neurones par exemple. L'apprentissage de ce type de classifieur est souvent précédé par une sélection et une réduction de termes. L'adaptation en catégorisation de texte de la formule bien connue de Rocchio 1.10 a été proposée par [Hull, 1994], et a été employé par beaucoup d'auteurs depuis lors.

Ce classifieur s'appuie sur une représentation vectorielle des documents (voir 1.3.7). La méthode de Rocchio calcule un classificateur $\vec{c_i} = (w_{1i}, \cdots, w_{|\mathcal{T}|i})$ pour la catégorie c_i via la formule :

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{POS_i} - \gamma \cdot \sum_{d_j \in NEG_i} \frac{w_{kj}}{NEG_i} \tag{1.10}$$

Où w_{kj} est le poids de t_k dans le document d_j, $POS_i = \{d_j \in Tr | \Phi(d_j, c_i) = vrai\}$, $NEG_i =$

$\{d_j \in Tr | \Phi(d_j, c_i) = faux\}$, Tr est le corpus d'apprentissage.
β et γ sont deux paramètres choisis selon l'importance accordée aux deux ensembles POS_i et NEG_i. Par exemple si β prend la valeur 1 et γ la valeur 0, le profil prototype c_i est le barycentres des exemples positifs. Le rôle des exemples négatifs est habituellement réduit par l'affectation des valeurs élevées pour β et des valeur faible pour γ, [Cohen et Singer, 1996] utilisent $\beta = 16$ et $\gamma = 4$.

[Singhal *et al.*, 1997] propose une amélioration par la prise en compte seulement d'une partie ($NPOS_i$, *near-positives*) des exemples de l'ensemble NEG_i qui sont les plus proches aux exemples positifs, soit :

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{POS_i} - \gamma \cdot \sum_{d_j \in NPOS_i} \frac{w_{kj}}{NPOS_i} \tag{1.11}$$

Le classement de nouveaux documents s'opère en calculant la distance euclidienne entre la représentation vectorielle du document et celle de chacune des classes ; le document est assigné à la classe la plus proche.

Cette méthode est tout à fait facile à implémenter, et est également assez efficace quand la séparation des classes soit linéaire, mais elle est peu adaptée dans les problème non linéairement séparable [Lewis *et al.*, 1996].

1.3.10.4 Réseau de neurones

Un classifieur de textes réseau de neurones est un réseau des unités, où l'unité d'entrer représente les termes, les unités de sortie représente les catégories et les poids sur les bords reliant des unités représentent des relations de dépendance. Pour classifier un document d_j les poids de termes w_{kj} sont chargés dans les unités d'entrée ; l'activation de ces unités est propagée en avant par le réseau, et les valeurs des unités de sortie détermine la décision de catégorisation.

Une manière typique d'entrainer ce réseau est la rétro-propagation, par laquelle les poids de termes d'un document d'apprentissage soient chargés dans les unités d'entrée, et si une classification fausse se produit l'erreur rétro-propager afin de changer les paramètres du réseau et éliminer ou minimiser l'erreur, cette méthode est dite rétro-propagation du gradient de l'erreur.

Le type le plus simple de classifieur réseau de neurones est le perceptron qui est un classifieur linéaire avec deux couches entré et sortie, en revanche un réseau de neurones non linéaire est un réseau avec une ou plusieurs couches additionnelles, dont RBF (*Radial Basis Function*) est l'exemple le plus célèbre qui est constitué uniquement de 3 couches.

les expériences ont montré que les réseaux non linéaires ont apporter une très de petite -ou non- amélioration par rapport aux réseau linéaires dans la tâche de catégorisation des textes.

1.3.10.5 Classifieurs à base d'exemples

Les classifieurs à base d'exemples n'établissent pas une représentations déclaratives explicites des catégories mais se fondent sur le calcul directe de la similarité entre le document à classifier et les documents d'apprentissage. Ces méthodes se sont aussi appelées *lazy learners*, puisque la phase d'apprentissage est réduite à un stockage des exemples d'apprentissage.
[Creecy *et al.*, 1992] est le premier qui a appliqué les méthodes à base d'exemples à la catégorisation de textes.
[Aha *et al.*, 1991] montre qu'un algorithme d'apprentissage à base d'exemples est caractérisé par

1. une fonction de similarité,
2. une fonction de sélection des exemples typiques,
3. une fonction de classement qui détermine de quelle manière un nouvel exemple est lié aux exemples appris.

Le classifieur à base d'exemples le plus connu est les *k-plus proches voisins* (*KNN, k-nearest neighbor*). Il procède par la prédiction de classe d'un texte t en fonction des k textes les plus proches voisins déjà étiquetés en mémoire.

La méthode ne nécessite pas de phase d'apprentissage ; c'est l'échantillon d'apprentissage, associé à une fonction de distance et à une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle. Le modèle est constitué des trois éléments : 1) l'échantillon d'apprentissage, 2) la distance et 3) la méthode de combinaison des voisins.
L'algorithme 2 montre comment classer un nouvel exemple par la méthode *k-PPV*

Algorithme 2: *Algorithme de classification par k-PPV*

Données : un échantillon de l textes classés en $C = c_1, c_2, \cdots, c_n$ classes
Entrées : le nombre k de voisins
1 **pour** *chaque texte* ***t*** **faire**
2 transformer le texte t en vecteur $t = (x_1, x_2, \cdots, x_m)$;
3 déterminer les k plus proches textes du texte t selon une métrique de distance;
4 combiner les classes de ces k exemples en une classe c;
5 **fin**
Résultat : le texte t associé à la classe c

La distance entre un texte et ses voisins se fait via une métrique de distance. Cette métrique peut être la métrique de Minkowski :

$$d_p(a,b) = \sqrt{\left(\sum_i |a_i - b_i|^p\right)^{\frac{1}{p}}} \tag{1.12}$$

Selon la valeur de p, on retrouve plusieurs distances connues :

- Si $p = 1$ cette distance est la distance de Manhattan définie par

$$d_m(a,b) = |a_1 - b_1| + |a_2 - b_2| + \cdots + |a_n - b_n| \tag{1.13}$$

- Si $p = 2$ c'est la distance euclidienne définie par

$$Modleprobabiliste(NaveBayes) d_e(a,b) = \sqrt{\sum_i (a_i - b_i)^2} \tag{1.14}$$

- Si $p = \infty$ c'est la distance de Chebyshev définie par

$$d_c(a,b) = \max\{|a_1 - b_1|, |a_2 - b_2|, \cdots, |a_n - b_n|\} \tag{1.15}$$

L'emploi de k voisins, au lieu d'un seul, assure une plus grande robustesse à la prédiction. Toutefois, la valeur de k peut changer les performances du modèle, comme cela est présenté en figure (1.5).

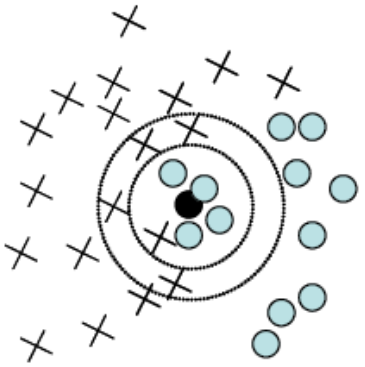


Figure 1.5 – *Choix de k influence la décision : pour $k = 5$, la décision est de classer l'exemple « noir » dans la classe « ronds ». Pour $k = 9$, la décision est de le classer en tant que « croix »*

Une première façon de combiner les k classes des k plus proches voisins est le vote majoritaire. Elle consiste simplement à prendre la classe majoritaire. Une seconde façon est le vote majoritaire pondéré. Chaque vote, c'est-à-dire la classe d'un des k plus proches voisins, est pondéré : soit x_i le voisin considéré, le poids de $c(x_1)$ est inversement proportionnel à la distance entre l'enregistrement à classer et x_i.
voici quelques limites qui affectent la catégorisation de textes [Breiman *et al.*, 1984] :

1. les algorithmes de type plus proche voisin sont longs en phase de généralisation, puisqu'ils sauvegardent tous les exemples de la phase d'entraînement ;
2. ils sont sensibles au bruit sur les variables prédictives ;

3. ils sont sensibles au choix de la fonction de similarité de l'algorithme.

1.3.10.6 Machine à Vecteurs de Support

Les Machine à Vecteurs de Support (*SVM - Support Vector Machine*) ont été introduites par [Vapnik, 1995], et en catégorisation de texte par [Joachims, 1998]. Il s'agit donc d'une classe récente de méthodes d'apprentissage automatique.

Le SVM revient à chercher un hyperplan dont la distance entre les exemples d'apprentissage positifs et négatifs est maximale. L'hyperplan optimal séparant les points de deux classes est celui qui passe « au milieu » de ces classes, c'est-à-dire dont la distance aux points les plus proches est maximale. Ces exemples les plus proches qui suffisent à déterminer cet hyperplan sont appelés *vecteurs de support*, ou encore exemples critiques. La distance séparant l'hyperplan de ces points est appelée « marge ».

1.3.10.6.1 Cas des classes linéairement séparables : Soit S un ensemble de l points linéairement séparables , $S = \{x_i \in \mathbb{R}^n | i = 1, \cdots, l\}$. Chaque point x_i appartient à une classe $y_i \in \{-1, +1\}$. Un hyperplan sépare l'ensemble S selon les deux classes. Cet hyperplan séparateur est défini en fonction du vecteur de poids w (vecteur normal à l'hyperplan), b et $\|w\|$ (la norme euclidienne de w) ; l'hyperplan vérifie l'équation [Jalam, 2003] :

$$w \cdot x + b = 0 \tag{1.16}$$

$$\text{tel que} \begin{cases} w \cdot x_i + b \geq +1 & \text{si } y_i = +1 \\ w \cdot x_i + b \leq +1 & \text{si } y_i = -1 \end{cases} \tag{1.17}$$

pour $i = 1, 2, \cdots, l$; où le produit scalaire $w \cdot x$ est calculé comme

$$w \cdot x = \sum_j w_j x_j \text{pour } j = 1, \cdots, n \tag{1.18}$$

Pour cet hyperplan, la marge vaut $1/\|w\|$, et donc la recherche de l'hyperplan optimal revient à minimiser $\|w\|$, soit à résoudre le problème suivant :

$$\begin{cases} \text{minimiser} & \frac{1}{2}\|w\|^2 \\ \text{sous les contraintes} & \begin{cases} w \cdot x_i + b \geq +1 & \text{si } y_i = +1 \\ w \cdot x_i + b \leq +1 & \text{si } y_i = -1 \end{cases} \text{ pour } i = 1, 2, \cdots, l \end{cases} \tag{1.19}$$

Les inégalités 3.29 peuvent être résumé ainsi : $y_i \cdot h(x_i, w, b) \geq 1, i = 1, \cdots, l$

En écrivant le lagrangien, on montre que la solution f s'écrit sous la forme

$$h(x) = w_0 \cdot x + b = \sum_i \alpha_i x_i x + b \tag{1.20}$$

où α_i est le multiplicateur de Lagrange associé à l'exemple i. Les x_i qui interviennent dans la solution sont nommés vecteurs de support. On notera l'ensemble de ces points $SV = x_i$ pour $i = 1, \cdots, m$ avec $m \leq l$. Ce sont les points de S les plus proches de l'hyperplan qui sont suffisent à déterminer cet hyperplan optimal (voir la figure 1.6)

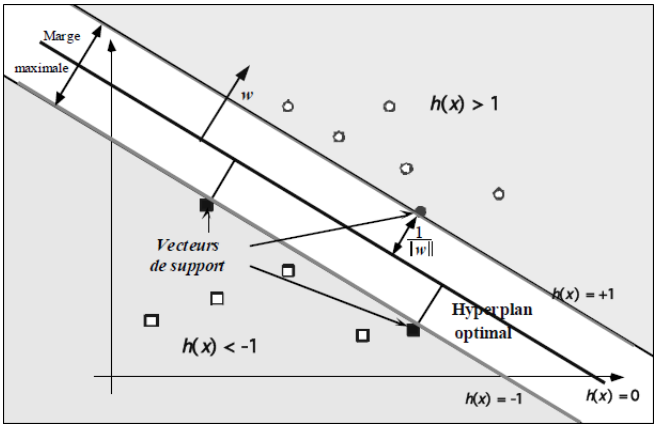


Figure 1.6 – *L'hyperplan optimal de la methode SVM*

1.3.10.6.2 Cas des classes non séparables : cette approche s'étend au cas où les données ne sont pas séparables grâce à des variables d'écart qui permettent à certains points de se situer du mauvais coté de la frontière. De plus, les *SVM* s'étendent très élégamment pour construire des modèles non linéaires en enrichissant l'espace de représentation. Pour cela, on peut substituer le produit scalaire dans la formule 1.20 par une *fonction noyau symétrique* $K(x, y)$ et obtenir un comportement non linéaire, la table 1.3 montre quelques exemples de noyaux.

La solution s'exprime sous la forme : $h(x) = \sum_i \alpha_i x_i x + b$		
Fonction noyau	**Forme fonctionnelle**	**Commentaire**
- polynomiale	$K(x, y) = (x.y + c)^n$	La puissance n est déterminée *a priori* par l'utilisateur.
- fonctions à base radiale	$K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$	L'écart type σ^2, commun à tous les noyaux, est spécifié *a priori* par l'utilisateur.
- fonctions sigmoïdes	$K(x, y) = \tanh((a(x \cdot y) - b))$	Le théorème de Mercer n'est vérifié que pour certaines valeurs de a et b.

Table 1.3 – *Les fonctions noyau les plus courantes avec leurs paramètres.*

La méthode des SVM est aisée d'emploi, et a montré une très grande performance par rapport aux autres méthodes indépendamment de la dimension de l'espace des termes. Cette méthode est applicable pour des tâches de classification à deux classes, mais il existe des extensions pour la classification multiclasse.

1.3.11 Comparaison entre classifieurs

[Yang, 1999] propose deux méthodes pour comparer les classifieurs et les évaluer. Cette comparaison peut être directe, ou indirecte :

1. **une comparaison directe** : il s'agit de l'utilisation de plusieurs méthodes par le même auteur ; de cette manière, le découpage et les mesures sont identiques pour toutes les méthodes. [Yang, 1999] comparent les machines à vecteurs supports, les plus proches voisins, les réseaux de neurones, une combinaison linéaire, et des réseaux bayésiens. [Dumais *et al.*, 1998] proposent une série de comparaisons en mettant en compétition une variante de l'algorithme de Rocchio (appelée find similar), des arbres de décision, des réseaux bayésiens et des machines à vecteurs supports. Cette méthode de comparaison est la plus crédible de point de vue scientifique.
2. **une comparaison indirecte** : soient Φ' et Φ'' deux classifieurs. Ces deux classifieurs peuvent être comparés si deux conditions sont réunies :
   (a) les deux classifieurs sont testés par différents groupes de chercheurs (même avec de conditions d'expérimentation différentes) sur deux collections Ω' et Ω'' respectivement ;
   (b) un ou plusieurs classifieurs de « références » ,$\bar{\Phi}_1, \bar{\Phi}_1, \cdots, \bar{\Phi}_m$ sont testés sur les deux collections Ω' et Ω'' par des comparaisons directes ; ceci donne une idée du niveau de difficulté d'apprentissage sur chaque collection.

Les comparaisons précédentes sont aboutées aux résultats suivantes :

1. Les classifieurs par combinaison de décisions, les SVM, les méthodes à base d'exemples donnent les meilleurs résultats.
2. Les réseaux de neurones, les classifieurs « en ligne » donnent des résultats inférieurs à ceux des précédents.
3. Les classifieurs linéaires tels que la méthode de Rocchio donnent souvent de mauvais résultats.

1.4 Conclusion

Dans ce chapitre nous avons présenté des notions importantes pour notre étude, tels que l'extraction des connaissances dans des bases de données, le data mining, et le text mining. Nous

avons vu que le text mining est plus complexe que le data mining, en raison de la nature non structurée des documents textuelles.

Ensuite nous avons effectué une étude approfondie de l'une des tâche de text mining qui est la catégorisation de textes, en détaillant chacune de ces trois étapes à savoir, la représentation, la construction du modèle et l'évaluation.

Les étapes représentation et construction du modèle, affectent souvent les performances de la classification. La première par le choix des termes pertinents, et la deuxième par le choix de la méthode d'apprentissage.

Toutes ces notion vont être une base pour la compréhension des prochains chapitres.

Dans le chapitre suivant nous allons aborder la problématique des document semi-structurés, et en particulier les documents XML qui feront l'objet de notre étude.

Chapitre 2

Classification des documents XML

2.1 introduction

De plus en plus d'informations sont présentées sur le Web et il est difficile d'accéder à la page que l'on cherche. L'une des causes de ce problème est que le Web n'a pas de schéma *a priori* et qu'il est donc impossible d'utiliser les algorithmes classiques des bases de données pour interroger des sites internet.

Parallèlement aux documents de type HTML, de nouveaux documents appelés documents semi structurés sont apparus. Ce type de document représente un compromis entre les données fortement structurées issues de la communauté BD (données relationnelles par exemple) et les données faiblement structurées issues des communautés document numérique et RI (documents plats, images ... etc).

Le format de représentation le plus utilisé par excellence est le format XML (eXtensible Markup Language) qui se caractérise par sa simplicité, extensibilité et puissance de représentation de n'importe quelles types de données.

Dans ce chapitre nous nous intéressons à la classification automatique de documents XML qui est un domaine de recherche très actif en particulier pour définir des modèles de représentations de documents qui étendent les modèles traditionnels (basés sur le contenu) en tenant compte de la structure du texte.

Certaines méthodes de classification réduisent les documents XML à leur partie purement textuelle, tandis que d'autres prends en compte la structure qui véhicule une information très riche.

Nous allons commencer ce chapitre par une présentation des principes des documents et langage XML (section 2.2). Nous présentons, ensuite, une synthèse de quelques travaux de la classification de documents XML selon qu'ils traitent le contenu, la structure ou les deux informations en même temps, (section 2.3).

2.2 Les documents XML

2.2.1 Le langage XML

XML (eXtensible Markup Language) est un standard mis en place par le *World Wide Web Consortium* (W3C)[1]. "*Il définit une syntaxe générique utilisée pour marquer les données avec un balisage simple et lisible par les humains.*" [Tannier, 2006]

La structure est représentée en XML par des éléments, contenant des attributs, du texte ou d'autres éléments. Les éléments ne peuvent pas se chevaucher. Le choix du nom des éléments structurants et des attributs, ainsi que l'organisation des éléments entre eux, sont laissés a la libre volonté de l'auteur. C'est pourquoi on dit que le langage XML est générique.

Contrairement à HTML, XML est extensible (balises - tags - non fixées), peut définir une structure de données complexe (à plusieurs niveaux d'imbrication) et la conformité structurelle de ses documents peut être validée. Ses utilisateurs peuvent choisir librement les attributs et les noms des balises (parmi les caractères unicode) qui qualifient sémantiquement et non graphiquement leurs données.

XML n'est pas réellement un langage, c'est un métalangage à balise, utilisé pour définir des langages de description (e.g., XML-Schema, XUpdate, etc.). Un document XML est composé de blocs de textes structurés par des balises de début et de fin, par exemple, `<booktitle>XML Classification</booktitle>`. Ainsi, un document XML peut non seulement contenir des données, mais aussi leurs structures. En utilisant XML, le document transporte donc de l'information à propos des données qu'il contient. Ces données sont dans le format texte et ainsi, les documents XML sont indépendants des plate-formes.

Un exemple de document XML, représentant des informations sur des mémoires présents dans une bibliothèque, est donné à la figure 2.1. L'organisation particulière d'un document XML (notamment l'imbrication des éléments sans possibilité de chevauchement) permet de représenter celui-ci sous forme arborescente, comme le montre l'exemple de la figure 2.2.
Nous pouvons illustrer par cet exemple les bases de la terminologie XML :

- "`Bibliothèque`", "`mémoire`", "`titre`", etc. sont des *noms* ou des *types de balises* ;
- `<mémoire>` et `</mémoire>` sont des *balises* (respectivement balises de début et de fin d'élément) ;
- Les balises de début et de fin ainsi que leur contenu (texte et éléments insérés entre les balises) constituent un *élément*, aussi appelé *nœud* ou sous-arbre ;
- *id="1"* est un attribut (id) ayant la valeur "1".
- Les parties purement textuelles ("*Azizi Nabil*", "*2010*", etc.) sont des éléments textuels.
- L'élément `chapitres` est le *parent* des éléments `chapitre`, qui sont donc ses *enfants*. On dit

1. http ://www.w3.org/

également qu'un parent *contient* un enfant. Par extension, l'élément `mémoire` est l'ancêtre des éléments `chapitre` (et `titre`, etc.) qui sont ses *descendants*.

– L'élément `Bibliothèque` est l'élément *racine*, il est l'ancêtre de tous les autres éléments. Par la même analogie avec les arbres, on dit que les éléments situes au plus bas de l'arborescence (`titre`, `chapitre`, etc.) sont des feuilles.

```
<Bibliothèque>
 <mémoire id="1">
   <titre>classification des documents XML</titre>
   <auteur>Azizi Nabil</auteur>
   <année>2010</année>
   <encadreur>Moussaoui Abdelouaheb</encadreur>
   <chapitres>
      <chapitre>Chapitre 1</chapitre>
      <chapitre>Chapitre 2</chapitre>
      <chapitre>Chapitre 3</chapitre>
   </chapitres>
 </mémoire>
 <mémoire id="2">
  ...
 </mémoire>
...
</Bibliothèque>
```

Figure 2.1 – *Exemple de document XML représenté sous forme textuelle.*

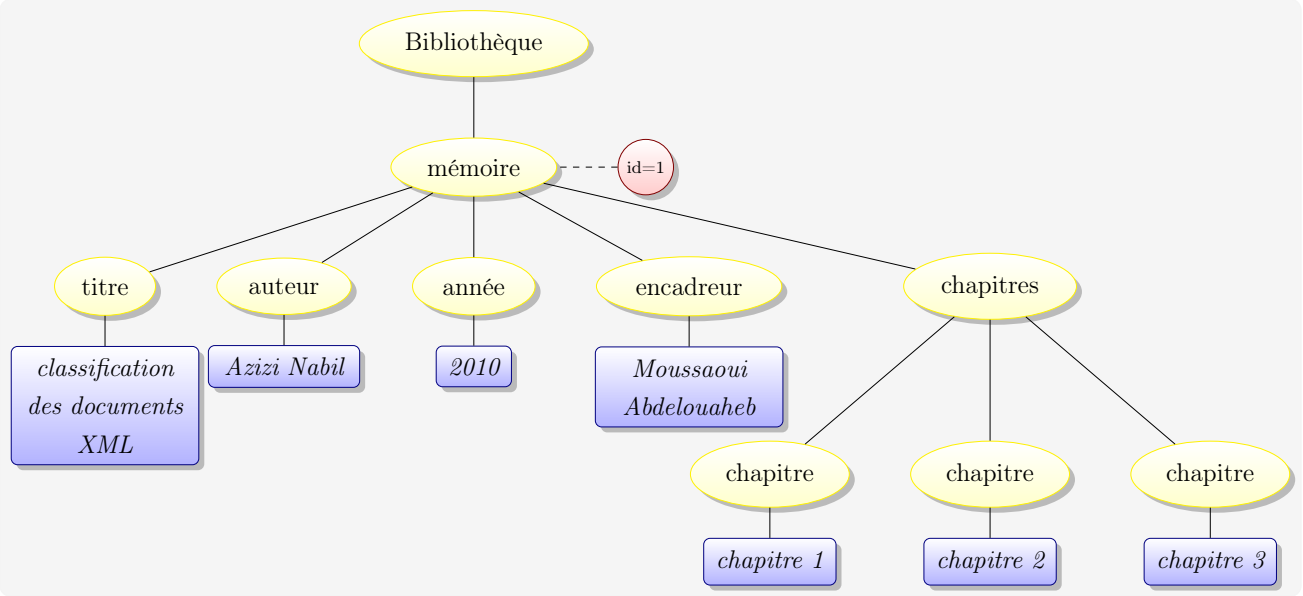


Figure 2.2 – *Exemple de document XML représenté sous forme arborescente.*

Nous verrons par la suite quelques concepts concernant les document XML [Courbis, 2002] : les documents bien formés (correct vis-à-vis de la syntaxe XML), les documents valides (conforme

à une définition de structure) et la définition de la structure des documents XML par des DTD ou des schéma XML.

2.2.2 Document bien formé

Pour qu'un document XML soit Bien formé (*well formed document*) il doit respecté la syntaxe du langage XML définie par le W3C :

- tout document doit commencer par un entête (prolog) - indiquant, au minimum, le numéro de version d'XML - et n'avoir qu'un seul élément, la racine de la structure arborescente des données ;
- toute utilisation de fragment de contenu (entité) externe est proscrite à moins qu'une définition de structure du document (DTD) ne soit précisée (par l'ajout de `<!DOCTYPE...>` entre l'entête et l'unique élément). Toute définition récursive d'entité est aussi interdite ;
- tout nom d'élément ou d'attribut doit commencer par une lettre ou un souligné (underscore) suivi, optionnellement, par des lettres, chiffres, traits d'union, points, deux points ou soulignés. La chaîne xml, quelle que soit sa casse, en début des noms est interdite (réservée pour de futurs usages) ;
- toute balise ouverte doit être fermée, soit par `/>` si l'élément est vide, soit par la balise fermante correspondante (`</nomBalise>`) ;
- ces deux balises (ouvrante et fermante) doivent être correctement imbriquées entre les balises de l'élément parent (pas de recouvrement) et de casse identique ;
- tout attribut ne doit apparaître qu'une seule fois dans un élément, doit avoir sa valeur entourée par des guillemets et de type chaîne non *parsable* (`CDATA`) si aucune DTD n'est précisée ;

L'exemple de la figure 2.3 montre un document XML bien-formé avec un entête (première ligne) et une seule racine `bibliographie` qui n'a qu'un livre, *le dragon*.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- voila un commentaire non utile -->
3   <bibliographie>
4    <livre surnom="le dragon" année="1986" titre="Compilers: Principle, Techniques and Tools"
         édiRef="7">
5     <auteur>Aho, Alfred</auteur>
6     <auteur>Sethi, Ravi</auteur>
7     <auteur>Ullman, Jeffrey</auteur>
8    </livre>
9    <éditeur nom="Addison Wesley" id="7" site="http://www.aw.com"/>
10  </bibliographie>
```

Figure 2.3 – *Exemple de document bien formé.*

2.2.3 Document valide

Il est aussi possible de définir la structure des données et donc de spécifier un langage métier ou vocabulaire adapté à ses besoins avec une DTD ou un XML Schema. De cette manière, toute erreur de structure des documents peut être détectée, avant tout traitement par les applications, lors de l'analyse syntaxique. Un document qui est bien formé et qui est conforme à sa définition de structure est dit **valide**.

2.2.4 Les langages de définitions de structure : DTD et XMLSchema

Une DTD ou un XML Schema décrit la structure des données des documents XML et sert à la validation des documents lors de l'analyse syntaxique. Les parseurs XML validants détectent ainsi les erreurs de syntaxe (document mal formé) et de contenu (document non valide) avant tout traitement par des applications. Cette définition de structure indique les éléments pouvant appartenir à un document ainsi que leur ordre, leur nombre, leur contenu et leurs attributs.

2.2.4.1 DTD

Une DTD est une grammaire principalement composée de déclarations d'éléments et d'attributs. Une déclaration d'élément indique son nom ainsi que le modèle de son contenu qui peut être soit :

- vide. Souvent de tels éléments possèdent des attributs ;

```
<!ELEMENT monElement1 EMPTY>
```

- du texte (`PCDATA`) ou du texte et des éléments mélangés et d'occurrence infinie. Le texte (données d'un `PCDATA`) doit avoir un contenu parsable (par exemple, le caractère `<` doit être remplacé par &lt ;) ;

```
<!ELEMENT monElement2 (#PCDATA)>
<!ELEMENT monElement3 (#PCDATA|elem1|elem2|elem3)*>
```

- des éléments. Il est possible d'indiquer que le fils de l'élément est n'importe quel autre élément déclaré (ANY), que c'est une séquence (de caractère séparateur ,) ou un choix (séparateur |) d'éléments. Pour ces deux derniers formats, il est possible d'indiquer, sous forme EBNF (*Extensible Backus Naur Form*), des contraintes d'occurrence des éléments fils et d'imbriquer d'autres séquences ou choix. Par exemple, un élément `monElement5` (dont la structure est donnée ci-dessous) aura comme fils, dans l'ordre, un élément `elem1` puis peut-être un `elem2` et enfin un à plusieurs `elem3`. Un élément `monElement7` aura un `elem1` puis, zéro à plusieurs `elem2` ou séquences d'un `elem3` suivi de un à plusieurs `elem1`.

```
<!ELEMENT monElement4 ANY>
<!ELEMENT monElement5 (elem1, elem2?, elem3+)>
<!ELEMENT monElement6 (elem1 | elem2* | elem3)>
<!ELEMENT monElement7 (elem1, (elem2 | (elem3, elem1+))*)>
```

Une déclaration d'attributs indique le nom de l'élément auquel ils appartiennent, puis la liste des attributs avec pour chacun son nom, l'énumération des valeurs possibles ou son type, et une indication s'il est requis (`#REQUIRED`), optionnel (`#OPTIONAL`), ou constant (`#FIXED`, sa valeur doit dans ce cas être précisée) ou la valeur par défaut à lui attribuer s'il n'est pas défini. Les types d'attributs les plus courants sont :

- `CDATA` désigne les données textuelles qui ne doivent pas être traitées par l'analyseur,
- `ID` qui indique que cet attribut définit une clé unique pour référencer l'élément ;
- `IDREF` qui indique que cet attribut est une clé qui référence un élément existant (plusieurs éléments peuvent être référencés avec le type `IDREFS`).

L'exemple ci-dessous indique que l'élément `monElement1` a quatre attributs : un requis qui est la clé identifiant l'élément, un constant de valeur `maValeur` et deux optionnels dont l'un doit avoir sa valeur comprise parmi celles de l'énumération.

```
<!ATTLIST monElement1 monAttr1 ID #REQUIRED
                      monAttr2 CDATA #FIXED 'maValeur'
                      monAttr3 (val1|val2|val3) #OPTIONAL>
<!ATTLIST monElement1 monAttr4 CDATA #OPTIONAL>
```

Document valide par rapport à une DTD : Voici, par exemple, les principales règles de validité d'un document par rapport à une DTD :

- Tout élément présent dans le document doit avoir été déclaré, une et une seule fois, dans la DTD, être correctement positionné et avoir un contenu conforme (nombre, type et ordonnancement des fils corrects) vis-à-vis de cette déclaration ;
- Tout attribut d'un élément doit avoir été déclaré et avoir une valeur conforme au type indiqué. Par exemple, cette valeur doit être une de celles proposées si c'est une énumération. Un élément ne peut avoir qu'un attribut d'identification (de type `ID`)requis ou optionnel à valeur unique (clé). Toute clé référencée par un attribut (de type `IDREF` ou `IDREFS`) doit avoir été définie auparavant. Un attribut requis doit être initialisé dans chaque occurrence de l'élément dont il dépend.
- Toute entité doit être définie dans la DTD avant d'être référencée. Les entités générales ne peuvent être utilisées que dans les documents et les entités paramètres dans les DTDs. Si le texte de remplacement d'une entité paramètre contient une balise (ouvrante), il devra aussi avoir la balise inverse (fermante) ; idem pour les parenthèses et les symboles d'inclusion paramétrée (<![,]]>). De plus, si ce texte est utilisé pour décrire le contenu d'un élément de type séquence, choix ou texte mélangé à des éléments, il doit au moins contenir un caractère non blanc et ne pas commencer ou finir par les caractères séparateurs de ces types de contenu (|, ,).
- La racine des données doit correspondre au nom donné dans la ligne du `DOCTYPE`, ligne indiquant quelle est la DTD à utiliser.

L'exemple de la figure 2.4 présente la DTD externe validant le document de la figure 2.5. Un document peut aussi contenir une partie (avec de possibles déclarations surchargées d'éléments, d'entités ou d'attributs) ou l'ensemble de la DTD (DTD interne) qui est prioritaire vis-à-vis de la DTD externe.

```
<!ELEMENT bibliographie (livre+, éditeur*)>
<!ELEMENT livre (auteur+)>
<!ELEMENT auteur (#PCDATA)>
<!ELEMENT éditeur EMPTY>
<!ATTLIST livre titre CDATA #REQUIRED surnom CDATA #IMPLIED isbn CDATA #IMPLIED année CDATA #
    IMPLIED édiRef IDREF #IMPLIED>
<!ATTLIST éditeur id ID #REQUIRED nom CDATA #REQUIRED site CDATA #IMPLIED adresse CDATA #
    IMPLIED>
```

Figure 2.4 – *Exemple de DTD externe.*

```
<?xml version="1.0" standalone="no" encoding="UTF-8"?>
<!DOCTYPE bibliographie SYSTEM="http://www-sop.inria.fr/oasis/personnel/Carine.Courbis/these/
    biblio.dtd">
<!-- voila un commentaire non utile -->
<bibliographie>
 <livre surnom="le dragon" année="1986" titre="Compilers: Principle, Techniques and Tools"
     édiRef="7">
  <auteur>Aho, Alfred</auteur>
  <auteur>Sethi, Ravi</auteur>
  <auteur>Ullman, Jeffrey</auteur>
 </livre>
 <éditeur nom="Addison Wesley" id="7" site="http://www.aw.com"/>
</bibliographie>
```

Figure 2.5 – *Exemple de document valide par rapport à la DTD de la figure 2.4.*

Bien que la DTD soit l'un des langages de schémas les plus utilisés , elle a des limitations, parmi lesquelles on peut énumérer :

- Une DTD définit très peu de types de données pour la validation du contenu. En fait, les DTD ont seulement le type texte (i.e. chaîne de caractères).
- Il est impossible d'intégrer d'autres définitions existantes puisque le support des *namespaces* différenciant les langages n'est pas supporté.
- Le langage n'est pas exprimé en XML, ce qui est en contradiction avec la forme même du langage défini.

2.2.4.2 XML Schema

Du fait des limitations énumérées ci-dessus, d'autres formalisme (Comme XML-Schema) ont été proposés. Ce formalisme, XML-Schema, semble avoir émergé et est maintenant adopté par la

plupart des industriels et des outils commerciaux liés à XML. Son contenu peut ainsi être traité par les outils standards liés à XML. Il se compose principalement de déclarations d'éléments et de définitions de types. Les caractéristiques principales de XML Schema sont :

- **Syntaxe XML :** Son contenu peut ainsi être traité par les outils standards liés à XML. Si la déclarations d'éléments et de définitions de types sont filles de l'élément racine `<xsd:schema ...>`, elles sont dites globales et réutilisables ; les autres sont dites locales.
- **Typage des données et intervalle du nombre d'occurrences précis :** Il existe deux sortes de type : les simples qui n'ont pas d'attribut ou de sous-élément et les autres, les complexes. Un type simple représente une feuille sans attribut dans l'arbre. Un type atomique est soit un type prédéfini, soit la redéfinition/restriction d'un type prédéfini.
- **Factorisation de contenu ou de déclarations d'attributs :** Il existe le même mécanisme que les entités paramètres des DTDs pour factoriser la définition d'une séquence, d'un choix ou de `all` avec l'élément `group` ou d'un ensemble d'attributs avec l'élément `attributeGroup :`. Ces groupes (qui pourront être référencés à l'aide de l'attribut `ref`) améliorent la lisibilité et les mises à jour.
- **Dérivation des types possible :** Cette caractéristique de restriction ou d'extension (héritage) de type est l'un des autres atouts de XML Schema. La restriction d'un type simple consiste à indiquer des contraintes ou *facets*. La restriction d'un type complexe consiste à réduire le nombre d'occurrences des éléments, non à supprimer les éléments d'un certain type.
- **Substitution d'élément, éléments et types abstraits :** Il existe aussi un mécanisme pour substituer un élément (dit élément de tête) par un autre qui doit avoir le même type ou un type dérivé et l'attribut `substitutionGroup` initialisé avec le nom de l'élément de tête. Ainsi, l'élément de tête et ses possibles remplaçants peuvent être utilisés de façon interchangeable dans le document.
- **Gestion des espaces de nom :** Pour éviter tout conflit de nom, il est préférable de limiter la portée des définitions et déclarations dans un espace de nom donné. Le formalisme XML Schema est lui bien adapté à cette utilisation d'espaces de nom. Il est possible d'indiquer l'espace de nom du langage cible (attribut `targetNamespace` de l'élément `schema`) pour les définitions et déclarations globales. Ce espace peut aussi être utilisé pour qualifier les éléments et les attributs locaux (selon les valeurs des attributs `elementFormDefault` et `attributFormDefault` de l'élément `schema`, et `form`).
- **Inclusion, redéfinition ou importation d'autres XML Schemas :** Un ou plusieurs XML Schemas peuvent être inclus (`include`) dans un nouveau XML Schema à condition qu'ils aient le même espace de nom que ce dernier. Il est aussi possible d'inclure un XML Schema (de même espace de nom) mais de redéfinir (`redefine`) certaines de ses définitions globales. Un XML Schema d'un autre espace de nom peut être utilisé par importation

(`import`). Dans ce cas, ses définitions n'appartiendront pas à l'espace de nom du langage cible contrairement à une inclusion.

- **Mécanismes d'unicité de valeur, de clés primaires et étrangères :** La valeur d'un élément, d'un attribut ou d'un n-uplet d'éléments ou/et d'attributs peut être déclarée comme unique (`unique`) dans un certain contexte et non pour tout le document comme avec `ID`. Une expression XPath simplifiée permet de sélectionner les éléments (le contexte) dans lesquels la valeur doit être unique. Le même mécanisme est utilisé pour définir des clés primaires (`key`) et des clés étrangères (`keyref`), utiles pour les bases de données.

L'exemple de la figure 2.6 présente un XML Schema équivalent à la DTD de la figure 2.5 mais plus précis en terme de type des valeurs atomiques. Il utilise aussi la notion d'espace de nom et de clés primaires et étrangères à la place de `ID` et `IDREF`.

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
            xmlns:bib="ftp://ftp-sop.inria.fr/oasis/publications/"
            targetNamespace="ftp://ftp-sop.inria.fr/oasis/publications/"
            elementFormDefault="qualified">
<xsd:complexType name="biblio_type">
 <xsd:sequence>
  <xsd:element ref="bib:livre" maxOccurs="unbounded"/>
  <xsd:element ref="bib:éditeur" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="livre_type">
 <xsd:sequence>
  <xsd:element ref="bib:auteur" maxOccurs="unbounded"/>
 </xsd:sequence>
 <xsd:attribute name="titre" type="xsd:string" use="required"/>
 <xsd:attribute name="surnom" type="xsd:string"/>
 <xsd:attribute name="isbn" type="xsd:string"/>
 <xsd:attribute name="année" type="xsd:gYear"/>
 <xsd:attribute name="édiRef" type="unsignedShort" use="required"/>
</xsd:complexType>
<xsd:complexType name="éditeur_type">
 <xsd:attribute name="nom" type="xsd:string" use="required"/>
 <xsd:attribute name="id" type="unsignedShort" use="required"/>
 <xsd:attribute name="site" type="xsd:anyURI"/>
 <xsd:attribute name="adresse" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="bibliographie" type="bib:biblio_type">
 <xsd:keyref name="foo" refer="bib:cleEdi">
  <xsd:selector xpath="bib:livre"/>
  <xsd:field xpath="@édiRef"/>
 </xsd:keyref>
 <xsd:key name="cleEdi">
  <xsd:selector xpath="bib:éditeur"/>
  <xsd:field xpath="@id"/>
 </xsd:key>
</xsd:element>
<xsd:element name="livre" type="bib:livre_type"/>
<xsd:element name="auteur" type="xsd:string"/>
<xsd:element name="éditeur" type="bib:éditeur_type"/>
</xsd:schema>
```

Figure 2.6 – *Exemple de XML Schema.*

2.2.5 Traitement de document XML

XML permet de définir la structure du document uniquement, ce qui permet d'une part de pouvoir définir séparément la présentation de ce document, d'autre part d'être capable de récupérer les données présentes dans le document pour les utiliser. Toutefois la récupération des données encapsulées dans le document nécessite un outil appelé analyseur syntaxique (en anglais *parser*), permettant de parcourir le document et d'en extraire les informations qu'il contient.

L'analyseur syntaxique (`parseur`) est un outil logiciel permettant de parcourir un document et d'en extraire des informations. Dans le cas de XML (on parle alors de *parseur* XML), l'analyseur permet de créer une structure hiérarchique contenant les données contenues dans le document XML. On distingue deux types de *parseurs* XML :

- les parseurs validants (*validating*) permettant de vérifier qu'un document XML est conforme à sa DTD.
- les parseurs non validants (*non-validating*) se contentant de vérifier que le document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base).

Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document. On distingue actuellement deux types d'approches :

- Les API (*Application Programming Interface*) utilisant une approche hiérarchique : les analyseurs utilisant cette technique construisent une structure hiérarchique contenant des objets représentant les éléments du document, et dont les méthodes permettent d'accéder aux propriétés. La principale API utilisant cette approche est DOM (*Document Object Model*)
- Les API basés sur un mode évènementiel permettent de réagir à des évènements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application utilisant cette API. SAX (*Simple API for XML*) est la principale interface utilisant l'aspect évènementiel

2.2.5.1 DOM

Le DOM est une spécification est une spécification du *W3C*[2] définissant la structure d'un document sous forme d'une hiérarchie d'objets, afin de simplifier l'accès aux éléments constitutifs du document. Plus exactement : le DOM est un langage normalisé d'interface (*API*), indépendant de toute plateforme et de tout langage, permettant à une application de parcourir la structure du document et d'agir dynamiquement sur celui-ci. Ainsi Javascript et ECMAScript utilisent DOM pour naviguer au sein du document HTML, ce qui leur permet par exemple de pouvoir récupérer le contenu d'un formulaire, le modifier, ...
DOM se divise en deux spécifications :

2. http ://www.w3.org/DOM/

- La spécification **DOM level 1** (*DOM niveau 1*) se séparant en deux catégories
  - *Core DOM level 1* : La spécification pour les documents en général (dont XML)
  - *HTML DOM level 1* : La spécification retenant uniquement les méthodes applicables à HTML
- La spécification **DOM level 2** ajoutant de nouvelles fonctionnalités comme la prise en compte des feuilles de style CSS dans la hiérarchie d'objets.

2.2.5.2 SAX

SAX est une API basée sur un modèle évènementiel, cela signifie que SAX permet de déclencher des évènements au cours de l'analyse du document XML. Une application utilisant SAX implémente généralement des gestionnaires d'évènements, lui permettant d'effectuer des opérations selon le type d'élément rencontré.
Soit le document XML suivant :

```
<personne>
 <nom>Azizi</nom>
 <prenom>Nabil</prenom>
</personne>
```

Une interface évènementielle telle que l'API SAX permet de créer des évènements à partir de la lecture du document ci-dessus. Les évènements générés seront :

```
start document
start element: personne
start element: nom
characters: Azizi
end element: nom
start element: prenom
characters: Nabil
end element: prenom
end element: personne
end document
```

Ainsi, une application basée sur SAX peut gérer uniquement les éléments dont elle a besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document.
L'API SAX définit les quatre interfaces suivantes :

- **DocumentHandler** possédant des méthodes renvoyant des évènements relatifs au document :
  - *startDocument()* renvoyant un évènement lié à l'ouverture du document
  - *startElement()* renvoyant un évènement lié à la rencontre d'un nouvel élément
  - *characters()* renvoyant les caractères rencontrés
  - *endElement()* renvoyant un évènement lié à la fin d'un élément
  - *endDocument()* renvoyant un évènement lié à la fermeture du document
- **ErrorHandler** possédant des méthodes renvoyant des évènements relatifs aux erreurs ou aux avertissements

- **DTDHandler** renvoie des évènements relatifs à la lecture de la DTD du document XML
- **EntityResolver** permet de renvoyer une URL lorsqu'une URI est rencontrée

2.2.5.3 Comparaisons de DOM et SAX

Les analyseurs utilisant l'interface DOM souffrent du fait que cette API impose de construire un arbre en mémoire contenant l'intégralité des éléments du document en mémoire quelle que soit l'application. Ainsi pour de gros documents (dont la taille est proche de la quantité de mémoire présente sur la machine) DOM devient insuffisant. De plus, cela rend l'utilisation de DOM lente, c'est la raison pour laquelle la norme DOM est généralement peu respectée car chaque éditeur l'implémente selon ses besoins, ce qui provoque l'apparition de nombreux *parseurs* utilisant des interfaces propriétaires...
Ainsi de plus en plus d'applications se tournent vers des API évènementielles telles que SAX, permettant de traiter uniquement ce qui est nécessaire.

2.3 Classification des documents XML

L'accès aux documents semi-structurés y compris les document XML pose le problème de l'hétérogénéité sémantique de la structure (semantic heterogeneity) [Denoyer, 2004]. Par exemple, dans un contexte éditorial, deux éditeurs différents de revues scientifiques n'utiliseront pas la même DTD alors que ces mêmes revues peuvent traiter de domaines identiques. Comment dans ce cas définir les unités d'information que l'on recherche. De façon générale, et quelque soient les efforts de normalisation ou de recommandation entrepris, il y aura une grande diversité et variabilité dans la structure des documents. Nous allons par la suite distinguer deux cas :

- Le premier concerne le fait qu'une même information peut-être représentée avec des structures différentes, en particulier par des auteurs différents. La figure 2.7 illustre sous forme XML ce phénomène. Dans ce cas, l'adresse est représentée soit de manière très structurée, soit de manière peu structurée. Ce phénomène est très réel avec le format XML qui laisse la liberté à un utilisateur de concevoir son propre modèle de représentation. Par ailleurs, même si la structure a été fixée *a priori* , il apparaît que différents auteurs de documents vont tout de même utiliser cette structure « à leur manière ».

```xml
<Adresse>
 <Rue>115 rue de belleville</Rue>
 <Ville>PARIS</Ville>
 <CodePostal>75020</CodePostal>
</Adresse>
```

```xml
<Adresse>
 <Numero>115</Numero>
 <Voie>belleville</Voie>
 <Departement>75</Departement>
 <Arrondissement>20</Arrondissement>
 <Ville>PARIS</Ville>
 <Pays>France</Pays>
</Adresse>
```

FIGURE 2.7 – *Deux représentations structurelles différentes d'une même information.*

- Le second concerne le fait que la structure d'un document n'est pas obligatoirement informative ou partiellement informative pour la tâche fixée. La figure 2.8 illustre un même document. La première version du document nous renseigne fortement sur le thème abordé tandis que la seconde, qui correspond à un format d'impression, n'est pas informative pour la tâche de classification.

```xml
<Document>
 <Titre>Thèse de dotorat</Titre>
 <MotsCles>Apprentissage,RI </MotsCles>
 <Resume>
   Cette thèse aborde le problème de ...
 </Resume>
</Document>
```

```xml
<Page>
 <Centré>
  <Grand>
   Thèse de doctorat
  </Grand>
 </Centré>
 <Ligne>
  <Italique>
   Apprentissage, RI
  </Italique>
 </Ligne>
 <Paragraphe>
   Cette thèse aborde le problème...
 </Paragraphe>
</Page>
```

FIGURE 2.8 – *Deux représentations structurelles d'un même document La représentation de gauche est informative pour la tâche de classification tandis que la deuxième l'est beaucoup moins.*

Dans les section qui suivent nous présentons quelque travaux qui traitent la problématique de classification des documents XML, en les divisant en trois approches, les travaux qui prennent que la structure du document en compte, ceux qui ne prennent que le contenu en compte et ceux qui traitent le contenu et la structure du document XML.

2.3.1 Classification selon le contenu

Dans la problématique de catégorisation des document XML, la prise en compte du contenu seulement revient à une tache de catégorisation de textes non structurés présenté au chapitre 1. Parmi les travaux qui s'intéressent au contenu seulement des document XML on énumère deux articles présenter à INEX XML Mining Track 2007 et 2008.

[Meenakshi *et al.*, 2007] proposent une méthode de classification (catégorisation) basé sur la création des profils en utilisant la fréquence de documents de catégorie négative (NCD negative category document frequency) et la fréquence moyenne de document des termes de descriptions initiales dans les articles Wikipédia [3]. Les auteurs ont montré les inconvénients de l'utilisation de la moyenne de TF et la moyenne de TF*IDF de ne pas considérer la distribution des termes entre les catégories négatives et positives.
NCD qui est indiqué dans la formule 2.1, réduit le poids d'un terme en fonction de sa répartition par catégories négatives.

$$ncd(t) = \begin{cases} \log(1 + ncf/ndf) & \text{si } t \in \text{documents négatives} \\ 1 & \text{si } t \notin \text{documents négatives} \end{cases} \tag{2.1}$$

où,
ncf = nombre de catégories négatives dont le terme apparait
ndf = nombre de documents négatives dont le terme apparait

Les auteurs ont observé que chaque article de Wikipédia commence par une première description, qui énonce clairement le sujet de l'article et contient les termes qui peuvent être employées pour distinguer les catégories. ils ont crée un autre profil basé sur la fréquence de document et la fréquence de document inverse (DF*IDF). La pondération des descripteurs (termes) utilisée est TF*NCD et les documents de tests ont été représentées en TF*IDF.
Ils ont utilisé aussi deux type de mesure de similarité pour mesurer la similarité entre les article Wikipédia de l'ensemble de test, à savoir la similarité cosinus (pondération des termes) et la

3. http ://www.wikipedia.org/

similarité fractionnaire qui est indiquée par la formule 2.2.

$$Fraction(CD, d) = \begin{cases} \frac{\alpha}{\gamma} & \text{si } \{d\} - \{CD\} \neq \phi \\ \alpha & \text{si } \{d\} - \{CD\} = \phi \end{cases} \tag{2.2a}$$

où

$$\begin{array}{ll} \alpha = \sum_{k=1}^{p} w_k * v_k & \text{si } terme_k \in CD \text{ et } D \\ \gamma = \sum_{k=1}^{p} v_k & \text{si } terme_k \notin CD \text{ et } terme_k \in D \end{array} \tag{2.2b}$$

- w_k le poids du $terme_k$ dans CD
- v_k le poids du $terme_k$ dans le document d
- p le nombre de termes dans CD et d

L'évaluations a été effectuée sur le corpus **INEX 2007** [Denoyer et Gallinari, 2008] [4] qui est constitué d'environ 96 000 documents XML extraits à partir du Wikipédia et divisé en deux parties 50% pour l'apprentissage et 50% pour le test. Les résultats obtenus sont montrés dans le tableau suivant :

	rappel Micro moyenne	**rappel Macro moyenne**
Profils $TF * NCD$ et similarité fractionnaire	0.774598902805093	0.714838847158184
Profils $TF * NCD$ et similarité cosinus	0.773170479246455	0.734183147200625
Profils $TF * NCD$ et similarité cosinus avec 5% des poids pour les description initiales	0.78008487734189	0.757502564330129

Le meilleur des trois résultats présentés combine les scores de similarité des deux méthodes, l'une fondée sur l'article en entier et l' autre basée sur la similarité de la description initiale dans un article Wikipédia. Dans cette méthode, les auteurs ont donné 95% des poids pour la similarité de les profils NCD avec le contenu entier des documents Wikipédia, et 5% des poids pour la similarité avec les descriptions initiales.

[Géry *et al.*, 2008] proposent une méthode de sélection des termes (étape de pré-traitement) et qui sera une plateforme pour les travaux qui vont prendre en compte la structures des documents.
sa représentation de documents et basé sur le modèle vectoriel (VSM), notamment le pouvoir discriminatoire d'un terme (TF*IDF).

4. http ://www-connex.lip6.fr/ denoyer/wikipediaXML/

Les auteurs ont introduit deux critère CC et CCE afin de sélectionner un sous ensemble T qui fournit une description des documents appartenant à une même catégorie.

Étant donné df_j^k le nombre de documents dans la catégorie C_k dont le terme t_j apparaît et f_j^k la fréquence des document appartenant à C_k en comprenant le terme t_j :

$$df_j^k = |\{d_i \in C_k : t_j \in d_i\}|, k \in \{1, \cdots, r\} \tag{2.3}$$

$$f_j^k = \frac{df_j^k}{|C_k|} \tag{2.4}$$

La fréquence f_j^k dépend du nombre de documents de C_k contenant t_j, d'autre part ils considèrent que le nombre de documents dont le terme t_j apparaît dans toutes les catégories sauf C_k, donc le premier critère couverture de catégorie noté CC (en Anglais Category Coverage), est calculé comme suit :

$$CC_j^k = \frac{(f_j^k)^2}{df_j} \tag{2.5}$$

Si la valeur de la CC_j^k est élevé, alors t_j est un terme représentatif de la catégorie C_k. La fréquence f_j^k ne prend pas en compte le nombre d'occurrence du terme t_j dans la catégorie C_k, pour cela il a introduit p_j^k qui représente la fréquence d'un terme dans une catégorie et une mesure utilisée en théorie d'information appelée *entropie* qui évalue la pureté des catégories pour un terme t_j. Dans le contexte de la catégorisation de textes il mesure le pouvoir discriminatoire des termes. L'entropie de Shannon E_j du terme t_j est donnée par :

$$E_j = -\sum_{k=1}^{r} (p_j^k) * \log(p_j^k) \tag{2.6}$$

Si l'entropie est minimale alors le terme présente un bon pouvoir discriminatoire pour la tache de catégorisation, en revanche si l'entropie est maximale alors le terme apparaît dans toutes les catégories par la même fréquence, donc il n'est pas représentatif.
Un deuxième critère est proposé noté CCE (Category Coverage Entropy) qui combine f_j^k de CC et l'entropie par la formule suivante :

$$CCE_j^k = (\alpha * f_j^k) + (1 - \alpha) * (\frac{E_j}{\max E}) \tag{2.7}$$

Pour chaque catégorie, un sous-ensemble des termes de T correspondant à la valeur la plus élevée du critère. Ensuite, l'indexe est défini comme l'union de ces sous-ensembles.
Pour l'étape de la catégorisation l'auteur a utilisé la représentation vectoriel $tf * idf$ et SVM comme algorithme de classification.

Les indexe utilisés sont indiqués dans la table suivante (notons que les critères CC et CCE sont utilisés pour la sélection des termes) :

Indexe	nombre de termes
I	77697
CC_{100}	1051
CC_{10000}	75181
CCE_{100}	909
CCE_{10000}	77580

Table 2.1 – *La taille des indexes*

Les résultat de l'évaluation sur le corpus **INEX 2007** ont montré que la sélection de 10 000 meilleurs termes pour chaque classe en utilisant CCE et CC, a donné des bons rappels, 0.7874 pour CCE et 0.7876 pour CC, qui sont classés parmi les trois meilleurs résultats obtenus dans XML Mining challenge.

2.3.2 Classification selon la structure

Parmi les travaux qui manipule la structure des documents XML

[Knijf, 2006] a proposé un algorithme de classification (FAT-miner) basé sur les arbres d'attributs fréquents qui est considéré comme une instance des méthodes d'arbres fréquents mining (frequent tree mining). Ces derniers sont des techniques de data mining capables d'exploiter l'information de la structure des document semi structurés dont plusieurs travaux ont été menés notamment pour le clustering des documents XML ([Termier *et al.*, 2002], [Zaki, 2002]). Selon l'auteur ces méthodes présentent des inconvénients tels que la négligence des attributs associés avec la structure qui véhicule une information potentiellement utile.
Pour induire un sous arbre d'attribut fréquents (voir 2.9) les conditions suivantes doivent être vérifiés :

1. préserver les étiquetés des nœuds.
2. préserver l'ordre parmi les enfants de mêmes parents.
3. préserver la relation parent-enfant.
4. préserver les attributs.

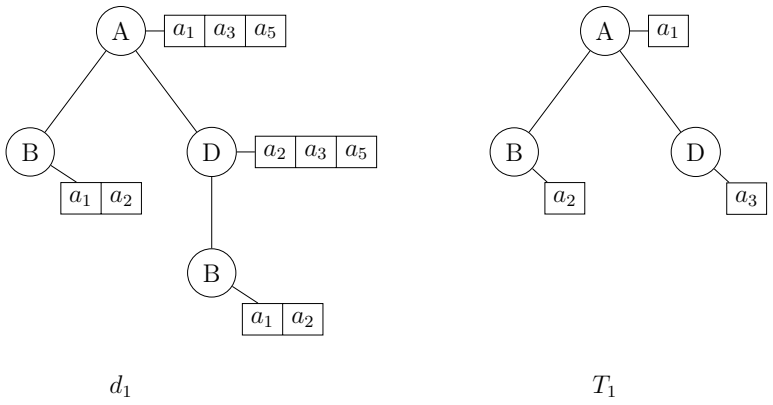


FIGURE 2.9 – *T_1 est un sous arbre du document d_1.*

Le principe de classification et de trouver des sous arbres discriminants des classe, c.à.d de chercher les arbres qui se produisent souvent dans une classe et rarement dans n'importe quelle autre classe. La méthode naïve pour calculer ces modèles est de calculer tous les sous-arbres, et de choisir parmi ces modèles précédemment calculés ceux qui sont fréquents. Cependant, c'est infaisable d'un point de vue informatique, puisque le nombre de sous-arbres d'un arbre T est exponentiel.
Le principe de l'algorithme **FAT-miner** est d'utiliser une fonction de support pour réduire le nombre de recherches. Le **FAT-miner** est divisé en deux parties globale qui manipule le document entier, et locale qui doit être exécuter pour chaque nœud des sous arbre.

La classification des documents XML suit les étapes ci-dessous :

1. Calculer les patterns fréquents pour les différente classes de l'ensemble d'apprentissage.
2. Sélectionner parmi ces patterns les plus pertinents qui seront utilisés pour apprendre le modèle de classification (phase apprentissage).
3. Évaluer le modèle de classification sur l'ensemble de test (phase d'évaluation).

L'évaluation a été réalisé sur **Wikipedia SC** [Denoyer *et al.*, 2007] qui est une collection de documents XML proposée dans *INEX 2006 chalenge* et qui comprend 150 000 documents XML réparties en 60 catégories dont 75 000 documents pour l'apprentissage et le reste pour le test.

	avec attributs	sans attributs
Micro moyenne F1	0.479802	0.338321
Macro moyenne F1	0.527043	0.338920

TABLE 2.2 – *Resultat de classification F1 Micro et Macro moyennes*

Les résultats indiqués dans le tableau ci-dessus montrent clairement l'apport de l'utilisations des attributs.

[Garboni *et al.*, 2005] présente une technique de classification supervisée des document XML basé sur la structure seulement dont chaque document est vu comme un arbre étiqueté et ordonné représenté par leurs tags.

La méthode est fondé sur la découverte de la structure basée sur le pattern séquentiel mining (la fouille des patterns séquentiels). Les auteurs ont utilisés une technique de transformation d'un arbre XML en une séquence. En commençant par l'association d'un identifieur pour chaque nœud de l'arbre (mapping), ensuite les transformer en une séquence suivant leurs profondeur (réduction) (voir la figure 2.10).

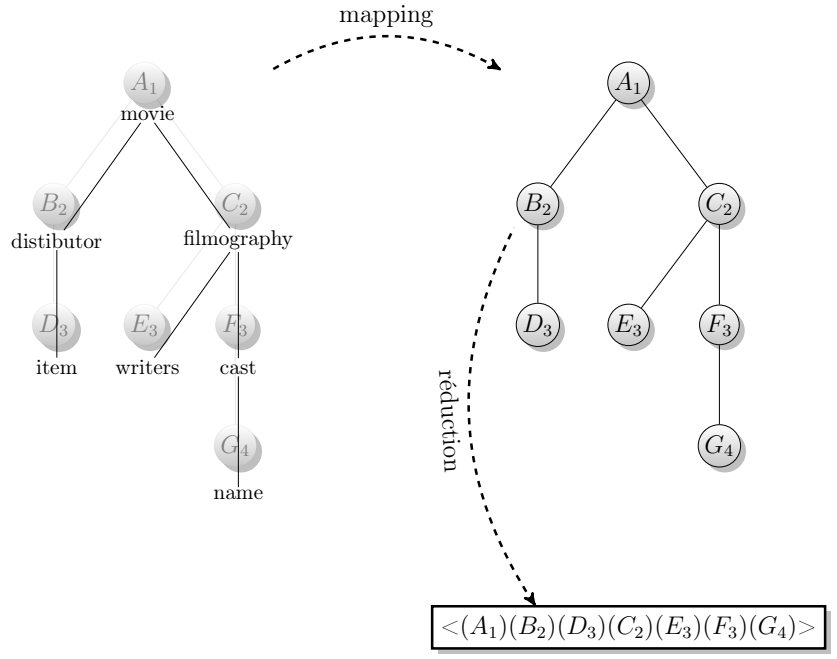


Figure 2.10 – *Transformation d'un arbre XML en une séquence*

La méthode est est résumée par les étapes suivantes (figure 2.11) :

1. une étape de nettoyage (pré-traitement) est effectuée. les tags non pertinents sont enlevés. Un tag qui est fréquent dans la collection est considéré comme non pertinent puisque il n'aide pas dans la séparation des documents.
2. dans cette étape chaque document XML est transformé en une séquence de tags (voir la figure 2.10). Ensuite une étape de data mining (plus précisément Pattern Séquentiel Mining basé sur les règles d'association) est exécutée pour chaque classe de la collection d'apprentissage dans le but d'extraire un ensemble de patterns séquentiels qui représente chaque classe.
3. la dernière étape consiste au classement des documents de tests par le matching entre la représentation d'un document XML et l'ensemble de pattern séquentiel de chaque classe en utilisant une mesure de distance définie par la formule 2.8.

$$score(D_i, C_j) = \frac{\sum_{k=1}^{|C_j|} \frac{i \times |LCS(D_i, SP_{C_j}(k))|}{|SP_{C_j}(k)|}}{|C_j|} \quad (2.8)$$

Où

- $LCS(D_i, SP_{C_j}(k))$ est la sous-séquence commune la plus longue entre le document et le k^{ime} pattern séquentiel de la classe C_j.
- i est un paramètre qui dépend de la longueur la sous-séquence commune entre le document et la classe.

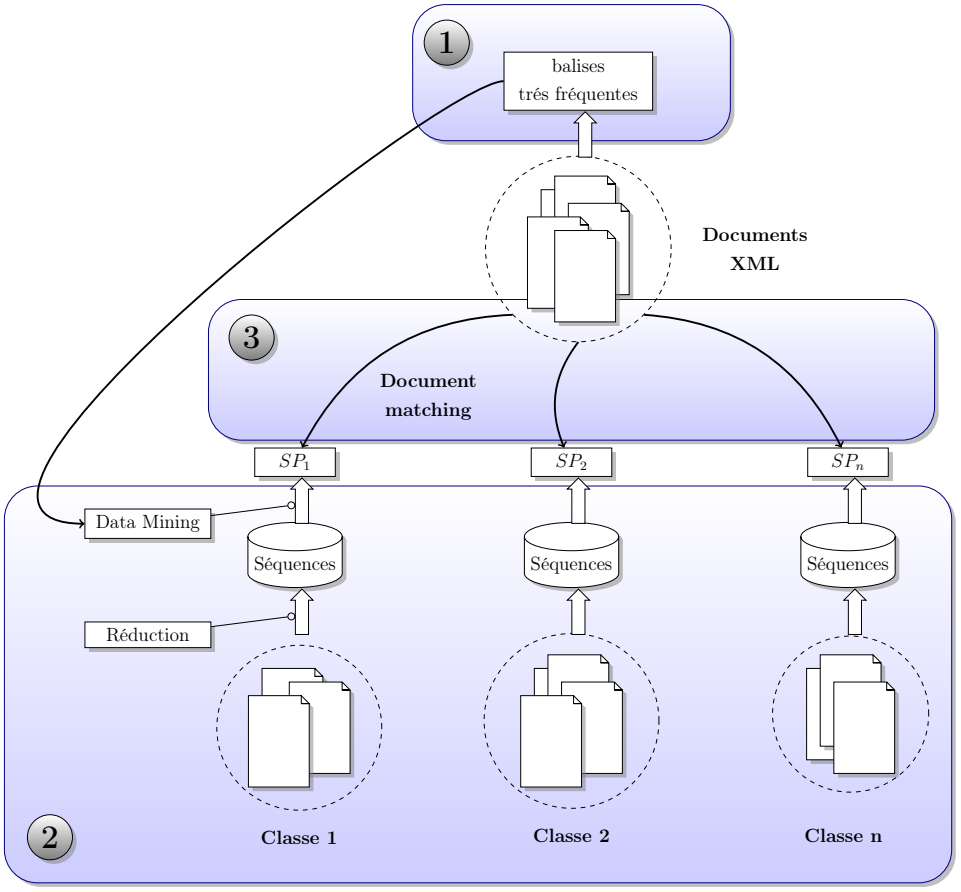


Figure 2.11 – *Le modèle de classification des documents XML* [Garboni *et al.*, 2005].

L'évaluation de la méthode a été effectuée sur le corpus **Movie SO** [Denoyer *et al.*, 2007] qui est une collection de documents XML hétérogènes décrivant les Films. Cette collection contient 9 643 documents et 11 catégories pré-définis. l'algorithme **Porter** est utilisé pour le *stemming* (radicalisation) des tags.
Les auteurs ont réalisé des **Rappels** moyennes entre 0.8 et 0.95 qui est une résultat assez bonne, surtout qu'ils n'ont utilisés qu'une partie du document XML (la structure).

L'autre point fort de cette méthode est la réduction de complexité de traitement des documents arborescentes par une transformation linéaire (représentation séquentiel).

Dans [Candillier *et al.*, 2005] les auteurs ont étudié différents type de représentation pour la manipulation des documents XML. Ils ont proposé de transformer les arbres en ensembles d'attributs-valeurs.

La représentation la plus simple d'un arbre XML est *bag-of-tags* (sac à balises) [Doucet et Ahonen-Myka, 2002] dont les auteurs s'inspirent pour proposer des descripteurs ou termes plus riches :

- l'ensemble des relation *père-fils* (le domaine de l'ensemble est les paires de balises marquant les nœuds père et fils).
- l'ensemble des relations *prochain frère* (le domaine de l'ensemble est les paires de balises marquant les nœuds et leurs frères).
- l'ensemble des chemins distincts (y compris les sous chemins) en commençant par la racine (le domaine de l'ensemble est les séquences de balises marquant les nœuds).

Par la suite un attribut est crée pour chaque nouveau descripteur (terme) dans l'ensemble d'apprentissage avec une valeur qui correspond aux nombres d'occurrences de ce terme dans un document donné. Finalement les auteurs ont créé autant de nouveaux attributs que de positions absolues de nœuds dans l'arbre qui sont identifies par une séquence de nombre (**0** représente la racine, **0.0** représente son premier fils, **0.2** son deuxième fils et ainsi de suite). Pour chaque identifieur de de position d'un nœud un valeur est attribué qui correspond aux nombre de fils de ce nœud dans le document.

Une telle représentation permet de distinguer :

- deux ensembles de documents utilisant différents balises, dont le nombre de certaines balises sont différents.
- deux ensembles de documents dans lesquels les relations (père-fils, prochain frère) ne sont pas similaire.
- deux ensembles de documents dans lesquels le nombre de fils d'un nœud de position donné sont différents.

Cependant, cette représentation génère une grande dimension d'attributs qui nécessite un algorithme adéquat pour ce type de problème.

Pour cette raison les auteurs ont choisis deux algorithmes à savoir, l'arbre de décision $C5$ qui peut manipuler une grande base d'attributs et effectuer une sélection de termes pendant la phase d'apprentissage et SSC (clustering de sous-espace) qui est un algorithme de clustering et a été adapté pour la classification.

L'évaluation de la méthode a été effectuée sur le corpus **Movie SO** qui est réparti en cinq sous collections de documents XML *inex-s, m-db-s-0, m-db-s-1, m-db-s-2* et *m-db-s-3*.

collection	nombre de tags	nombre de relations père-fils	nombre de relations prochain frère	nombre de positions des nœuds	nombre de chemins	total
inex-s	150	1038	827	2475	3674	8164
m-db-s-0	197	2172	419	6575	320	9683
m-db-s-1	197	6477	5617	9159	16772	38222
m-db-s-2	196	8953	7455	9183	25628	51415
m-db-s-3	199	10639	9557	8537	37576	66508

TABLE 2.3 – *Nombre d'attributs générés pour chaque collection*

collection	micro-rappel	macro-rappel
inex-s	0.941	0.958
m-db-s-0	0.968	0.960
m-db-s-1	0.966	0.956
m-db-s-2	0.942	0.932
m-db-s-3	0.947	0.935

TABLE 2.4 – *Micro et macro rappel de l'algorithme C5*

L'adaptation de l'algorithme SSC pour la classification est appliqué seulement sur la collection *m-db-s-0* et réalise des résultats intéressants avec micro-rappel de 0.906 et un micro-rappel de 0.924.
Le résultat de la classification par SSC est un arbre de décision (figure 2.12) tel que :

- $S2$, $S3$, $S4$ et $S5$ représente des probabilités de modèle basé sur la relation prochain frère concernant respectivement les classes 2, 3, 4 et 5.
- $P6$ et $P11$ représente des probabilités de modèle basé sur les chemins concernant les classes 6 et 11.

Bien que la table 2.4 montre de bons résultats, les algorithmes sont appliqués sur des base relativement petites (de l'ordre de 200 balises). Si la collection utilisée a une grande dimension alors le nombre d'attributs générés sera très grand et par conséquence la méthode proposée sera plus complexe voire irréalisable, puisque plus ce nombre est élevé, plus le volume de calcul est important.

[Wisniewski *et al.*, 2005] a proposé un modèle génératif stochastique de documents semi structurés basé sur les réseaux bayésiens pour le calcul de la probabilité de la structure des documents arborescents.
Étant donné un document XML d, $|d|$ noté le nombre de ses nœuds. Chaque nœud n_i est composé d'une étiquette s_i et d'un contenu t_i et correspond à une entité structurelle du document (un

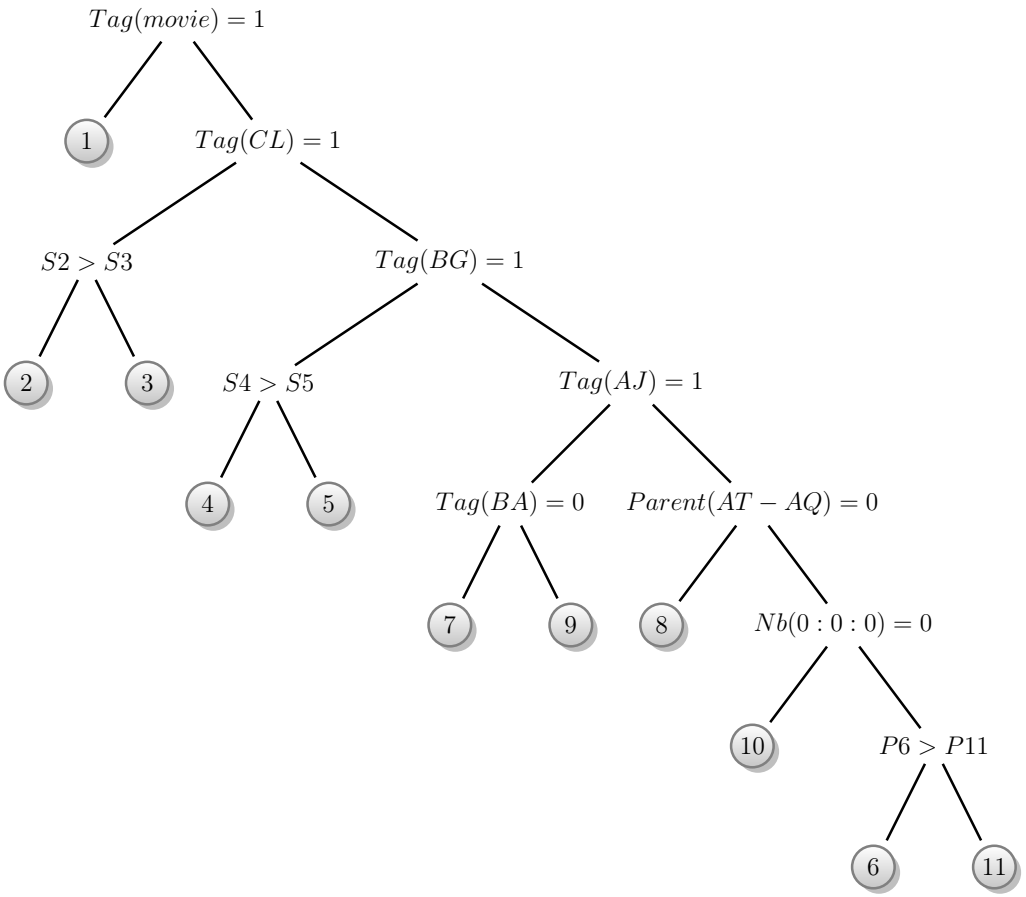


FIGURE 2.12 – *Arbre obtenu par la classification de la base m-db-s-0*

paragraphe, un titre...). Soit Λ l'ensemble des étiquettes possibles. Le processus génératif consiste à calculer la probabilité conditionnelle d'un document (en utilisant un modèle de paramètres θ).

$$P(d|\theta) = P(s_1, ..., s_{|d|}|\theta) \tag{2.9}$$

Soit $(s_1, \cdots, s_{|d|})$ l'ensemble des nœuds de structure d'un document d. La structure du document est considérée modéliser par un réseau bayésien de N variables aléatoires $X_1, \cdots, X_N$. Les arcs du réseau seront modélisés par la fonction $pa(X_i)$ qui renvoie l'ensemble des parents de la variables X_i dans le réseau. Deux types de variables ont été distinguées :

- les variables $S_1, \cdots, S_{|d|}$ qui représentent les nœuds du document modélisé.
- les variables $Y_1, \cdots, Y_{N-|d|}$ permettant de modéliser les dépendances entre les nœuds du documents.

Ainsi, l'ensemble des variables s'écrit $(X_1, \cdots, X_N) = (S_1, \cdots, S_{|d|}, Y_1, \cdots, Y_{N-|d|})$.
Pour représenté la structure, différents deux modèles ont été proposés :

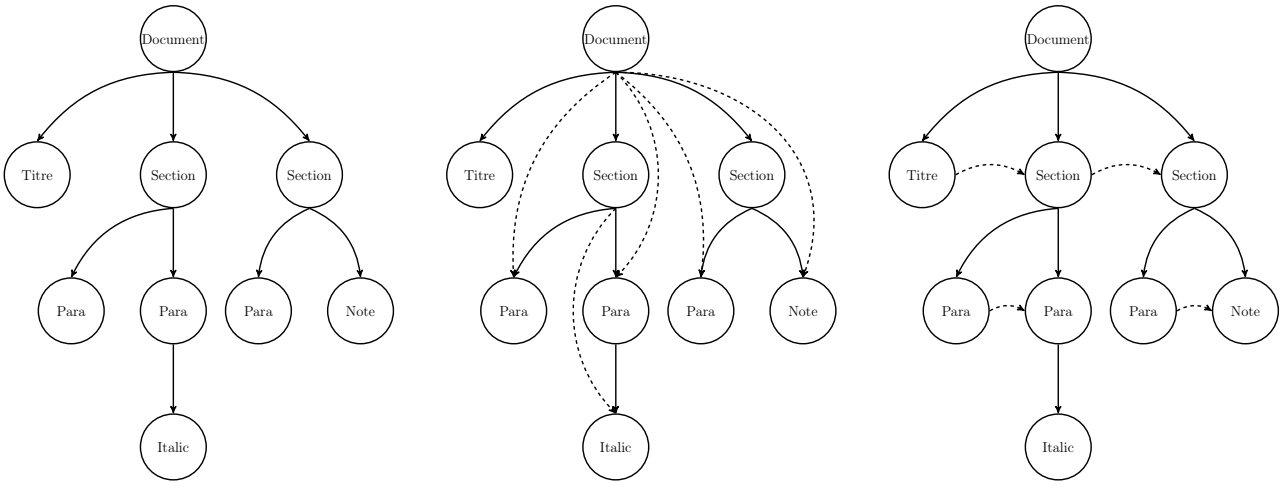
- ***Modèles de structure de type 1 :*** dans cette famille de modèle les liens entre les nœuds ont été considérés. Ils correspondent à des réseaux simple pour lesquels les variables aléatoires sont les entités structurelles du document ($N = |d|$). (voir la figure 2.13) et

formalisés (pour le modèle père) par :

$$P(d|\theta) = \prod_{i=1}^{|d|} P(S_i = s_i)|pa(S_i)|\theta) = \prod_{i=1}^{|d|} P(S_i = s_i)|pre(S_i), \theta) \tag{2.10}$$

– ***Modèle grammaire*** qui est un modèle plus complexe vise à modéliser la dépendance entre un nœud et l'ensemble de ses fils et permet d'extraire, sous forme d'une DTD probabiliste, un représentant structurel d'un corpus (voir la figure 2.14). La probabilité structurelle pour un document sera :

$$P(d|\theta) = P(s_1, \cdots, s_{|d|}|\theta) = \prod_{i=1}^{|d|} P(enfants(n_i) = s_i)|s_i, \theta) \tag{2.11}$$



(a) Modèle père **(b)** Modèle père-père **(c)** Modèle père-frère

FIGURE 2.13 – *Le modèle de classification des documents XML :*

La phase d'apprentissage pour apprendre les paramètres des modèles a été effectuée par l'algorithme CEM qui se résume en trois étapes :

Étape E : Dans cette étape, on calcule, pour chaque document d et pour chaque classe i, la probabilité $P(d|\theta_{c_i})$selon les modèles présentés.

Étape C : L'étape C permet d'attribuer chaque document du corpus à la classe qui maximise la probabilité $P(d|\theta_{c_i})$.

Étape M : les paramètres des différentes classes vont être mis à jour en fonction du résultat de la classification obtenue lors de l'étape précédente. Cette mise à jour est faite en maximisant la log-vraisemblance ,

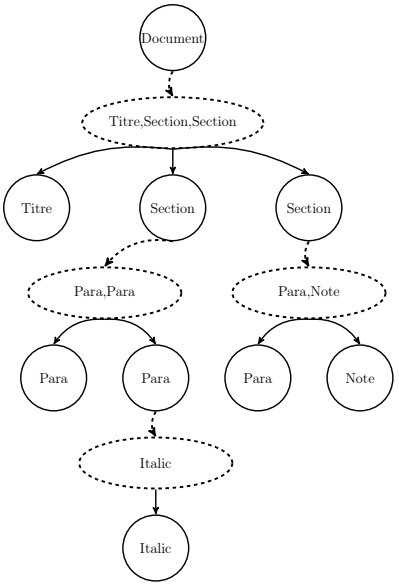


FIGURE 2.14 – *Le modèle grammaire.*

Pour évaluer ces modèles les auteurs ont utilisés deux mesures (*entropie croisée*,et *pureté*) qui sont généralement utilisées pour évaluer les système de clusering. L'évaluation a été effectuée sur le corpus **IEEE** d'INEX qui comprend 12 108 documents XML. Ils s'ont aperçu que les modèles simple (modèle parent, modèle père-frère) ne permettent pas d'obtenir, sur ce corpus, de meilleurs résultats que le modèle simple Naïve Bayes. Par contre, le modèle grammaire montre de meilleures performances, notamment lorsque le nombre de classes est faible. d'autres expériences sur un corpus simulé montrent que le modèle grammaire est capable de retrouver les différentes DTD apparaissant dans un corpus, sans toutefois réaliser une inférence de DTD.

2.3.3 Classification selon le contenu et la structure

A notre connaissance, peu de travaux dans la littérature qui tirent avantage des deux composantes porteuses d'information d'un document XML à savoir la structure et le contenu. La tendance récente est d'utiliser ces deux informations s'avère une très bon piste de recherche. Nous commençons par le travail de :

[Ghosh et Mitra, 2008] proposent une méthode de classification des documents XML basé sur des noyaux SVM composés (Composite SVM Kernels) en combinant la similarité entre les structures et les contenus des document XML, par l'application de la méthode SVM séparément sur le contenu représenté par le modèle vectoriel $tf * idf$ et la structure, ensuite combiner les résultat par une combinaison linéaire, tel que les poids de la combinaison sont déterminés par une heuristique basé sur l'entropie de la distribution de la fréquence des unité d'indexe dans les documents.

Pour représenter la structure, les auteurs ont procédés à une extraction des chemins textuelles qui commence par la racine jusqu'au feuilles dont le contenu est un texte. Ces chemins sont appelés *Tpaths* et constituent les indexes du modèle vectorielle. Deux représentation possibles de la structure sont abordées :

- *la similarité cosinus* dans laquelle le score des éléments de vecteur est $tf * idf$.
- *la similarité booléenne* dans laquelle le score des éléments de vecteur est binaire, pour indiquer la présence ou l'absence du *Tpath* dans un document.

L'entropie de la distribution de fréquence d'un document est définit par :

$$H = -\sum_i p_i \log(p_i) \tag{2.12}$$

p_i est un valeur normalisé dans l'intervalle $[0, 1]$ qui représente la fréquence des documents contenant un terme t_i.
Étant donné H_S représentant l'entropie de la distribution de la fréquence de document des *Tpaths* dans l'ensemble d'apprentissage et H_C l'entropie de la distribution de la fréquence de document des termes du contenu. Le noyau composé est définit par la formule :

$$\kappa_{compos}(x_i, x_j) = \frac{H_S}{H_S + H_C}\kappa_{structure}(x_i, x_j) + \frac{H_S}{H_S + H_C}\kappa_{contenu}(x_i, x_j) \tag{2.13}$$

L'évaluation était effectuée sur le corpus **IEEE SC** (INEX 1.3) qui comprend **18** catégories et **12 108** documents XML dont **6 053** sont utilisés pour l'apprentissage et le reste pour le test. Les résultats obtenus sont indiqués dans le tableau suivant :

noyau SVM utilisé	Precision		Rapell		F1	
	micro	macro	micro	macro	micro	macro
noyaux structure seulement						
TPath Cosinus	0.597	0.629	0.584	0.516	0.562	0.532
TPath Booléen	0.567	0.578	0.568	0.499	0.543	0.512
Utilsé SVM sur le modele sec-à-mots de contenu de documents						
sac-à-mots Cosinus	0.832	0.819	0.821	0.752	0.819	0.774
noyaux composés Structure-Contenu						
TPath Cosinus & Texte Cosinus	0.862	0.886	0.857	0.842	0.857	0.861
TPath Booléen & Texte Cosinus	0.867	0.891	0.862	0.846	0.861	0.864

Table 2.5 – *Résultats d'évaluation*

Les résultats ci-dessus montre l'efficacité de la technique utilisée par rapports à d'autres méthodes. La combinaison des noyaux peut être vue comme une méthode de fusion linéaire d'information, qui est similaire à notre travail. Le point fort de cette proposition est l'heuristique utilisée pour la détermination des poids de combinaison, qui sont proches aux poids idéaux

obtenus par expérimentation, ainsi que l'utilisation de SVM, qui donne de meilleurs résultats de classification et peut traiter des grosses collections de documents.

[de Campos *et al.*, 2007] ont proposé une méthode de classification probabiliste des documents XML basée sur le réseau bayésien appliquer sur une transformation de document structuré en un document plat qui prendre en compte le contenu et la structure.

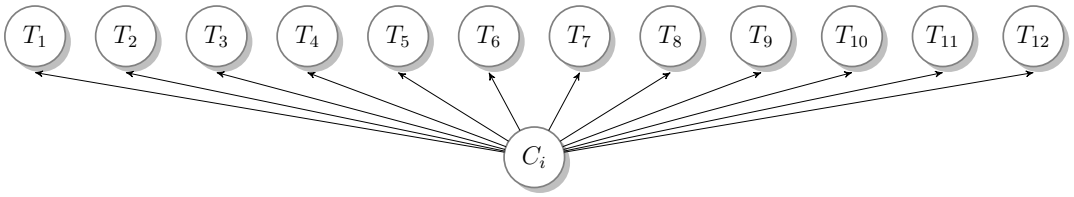


Figure 2.15 – *Le classifieur naïve Bayes.*

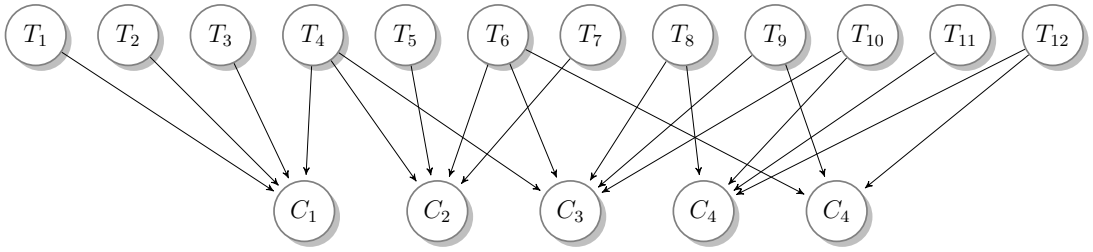


Figure 2.16 – *Le classifieur porte OU (OR gate).*

La méthode est appelée le **classifieur réseau bayésien porte OU** (*OR Gate Bayesian Network Classifier*). Il est basé sur la topologie de réseau à deux couches :

- **la couche des termes** (*les causes*) chaque terme t_k apparaissant dans l'ensemble d'apprentissage est associé à un variable binaire T_k prenant son valeur dans l'ensemble $\{t_k, \bar{t}_k\}$.
- **la couche des classe** (*les effets*) chaque classe c_i est associé à un variable binaire C_i prenant son valeur dans l'ensemble $\{c_i, \bar{c}_i\}$.

Ensuite les deux couches sont liées par des arcs allant de chaque terme aux classes dans lesquelles il apparaît dans l'ensemble d'apprentissage.
L'information quantitative associée pour ce réseau est les probabilités conditionnelles $p(c_i|pa(C_i))$, tel que $Pa(C_i)$ est l'ensemble des parents du variable C_i et $pa(C_i)$ est n'importe quelle configuration possible de l'ensembles des parents.

$$p(c_i|pa(C_i)) = 1 - \prod_{T_k \in R(pa(C_i))} (1 - w(T_k, C_i)), \qquad p(\bar{c}_i|pa(C_i)) = 1 - p(c_i|pa(C_i)) \tag{2.14}$$

où $R(pa(C_i))$ est la sous-ensemble des parents qui sont instanciés a ses valeurs t_k dans la configuration $pa(C_i)$. $w(T_k, C_i)$ est un poids qui représente la probabilité de l'occurrence de *la cause* T_k seul, qui fait que *l'effet* est vrai (i.e., forcer la classe c_i à se produire).

Les poids $w(T_k, C_i)$ ont été estimés en utilisant vraisemblance maximum (*maximum likelihood*) par les deux formules :

$$w(T_k, C_i) = \frac{N_{ik}}{N_k} \tag{2.15}$$

$$w(T_k, C_i) = \frac{N_{ik}}{N_k} \times \prod_{h \neq k} \frac{(N_i - N_{ih})N}{(N - N_h)N_i} \tag{2.16}$$

Étant donné un document d_j à classifier, les variable T_i dont les terme apparaissant dans le document sont instanciées t_i et toutes les autres variables T_h sont instanciées au valeur $\bar{t}_h$

$$p(c_i|d_j) = 1 - \prod_{T_k \in Pa(C_i)} (1 - w(T_k, C_i) \times p(t_k|d_j)) = 1 - \prod_{T_k \in Pa(C_i) \cap d_j} (1 - w(T_k, C_i)) \tag{2.17}$$

où N est le nombre total des mots dans les documents d'apprentissage.

Pour représenter les documents XML, les auteurs ont présentés cinq méthodes de transformation d'un document semi-structuré en un document plat :

1. **Texte seulement** la structure des documents XML est omis.
2. **Adding** la structure est ajouté au contenu textuel.
3. **Tagging** elle est similaire à la représentation précédente mais considère q'un terme apparus dans des chemins différents n'est pas le même.
4. **Structure seulement** le contenu des documents XML est omis.
5. **Réplication de texte** un poids n est associé à chaque balise qui représente son importance, ensuite le contenu de cette balise est dupliqué n fois.

L'évaluation a été effectuée sur le corpus **INEX 2007** afin de trouver la meilleur combinaison de classifieur (bayésien naïve, classifieur *OR gate*) et représentation (les cinq représentations ci-dessus). Les cinq meilleurs résultats sont montrés dans le tableau suivant :

classifieur	**Représentation**	**sélection**	**micro rappel**	**macro rappel**
Bayésien naïve	texte seulement	pas de sélection	0.77630	0.58536
Bayésien naïve	réplication (id = 2)	pas de sélection	0.78107	0.6373
Or gate	réplication (id=8)	Information mutuelle (MI)	0.75097	0.61973
Or gate	réplication (id=5)	Information mutuelle (MI)	0.75354	0.61298
Or gate	texte seulement	≥ 2	0.78998	0.76054

Table 2.6 – *Les résultats d'évaluation*

La cinquième combinaison en utilisant la sélection (≥ 2) [5] donne la meilleur résultats, mais on constate que la structure n'est pas prise en compte.

5. les termes apparaissant dans plus de 2 documents

[Yang et Zhang, 2007] proposent une nouvelle représentation des documents XML, appelé Modèle à vecteur de lien structuré (*Structured link vector model* (SLVM)), qui est une amélioration du modèle vectoriel conventionnel (VSM) de [Salton et Mcgill, 1983] (*bag of words*), afin de représenter la structure des documents XML.
SLVM représente un document XML doc_x, comme une matrice Δ_x, où les ligne sont des termes du document, et les colonnes sont les termes de la structures (les balises).

$$\Delta_x = \left[\Delta_{x(1)}, \Delta_{x(2)}, \cdots\cdots, \Delta_{x(m)}\right] \tag{2.18}$$

Où, m est le nombre des balises distincts et $\Delta_{x(i)}$ est la représentation vectorielle du i^{ime} élément de la structure. On peut écrire la formule 2.18 comme :

$$\Delta_{x(i,j)} = TF(w_j, doc_x.e) \cdot IDF(w_j) \tag{2.19}$$

Soit un document XML illustré par la figure 2.18 :

```xml
<article>
   <title>Ontology Enabled Web Search</name>
   <author>John</author>
   <conference>Web Intelligence</conference>
</article>
```

Figure 2.17 – *Un exemple de document XML.*

$$d_x = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{matrix}$$

Figure 2.18 – *La représentation vectorielle du document de la figure 2.18.*

$$\Delta_x = \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{title} & \text{author} & \text{conference} \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{matrix}$$

Figure 2.19 – *La représentation SLVM du document de la figure 2.18.*

$$\tilde{\Delta}_x = \begin{array}{c} \begin{array}{ccc} \text{title} & \text{author} & \text{conference} \end{array} \\ \left(\begin{array}{ccc} 0.5 & 0 & 0 \\ 0.5 & 0 & 0 \\ 0.5 & 0 & \sqrt{2}/2 \\ 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sqrt{2}/2 \end{array} \right) \end{array} \begin{array}{l} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{array}$$

FIGURE 2.20 – *La représentation SLVM normalisée du document de la figure 2.18.*

Les auteurs ont aussi défini une nouvelle mesure de similarité entre les documents XML par la formule suivante :

$$sim(doc_x, doc_y) = \sum_{1}^{n} \Delta^{n}_{x(i)}{}^{T} M_e \Delta_{y(i)} \tag{2.20}$$

où M_e est une matrice de $m \times m$ (m est le nombre de balises), appelée ***matrice de similarité d'élément***.
Il ont appliqué ensuite la méthode SVM, avec un noyau personnalisé, qui est définie par la formule 2.20, sur la matrice normalisée $\widetilde{\Delta}_{x(i,j)}$.

L'évaluation de la méthode, qui a été effectué sur le corpus **INEX 2007**, réalise un *micro rappel* de 0.8722 et un macro rappel de 0.8390. Cette méthode innovatrice, que se soit dans la représentation des documents XML ou l'application de la méthode SVM, réalise des performances nettement plus avancées que les autres méthodes.

Enfin, et pour terminer cette partie, nous allons résumer les travaux présenté auparavant, par le tableau récapitulatif 2.7, qui donne un vu général sur les ressemblances et les divergences de ces travaux, les résultats obtenus ainsi que les corpus utilisés.

2.4 Conclusion

L'extraction des connaissances à partir de documents semi-structurés, particulièrement à partir les document XML, est un domaine de recherche très actif. La structure d'un document XML véhicule une information très riche, cependant elle est difficile à analyser, en raison de la complexité des méthodes traitant les structures arborescentes.

Afin de bien comprendre la notion des document XML, nous avons commencé ce chapitre par une partie introductive, qui décrit le langage XML en se focalisant sur sa puissance et sa

flexibilité de représentation de la structure. Ensuite nous avons réalisé une synthèse de quelques travaux de recherche. Elle nous a permet de distinguer trois approches :

- La première vise à exploiter le contenu d'un document XML en ignorant la structure. Les deux travaux présentés s'intéressent à des nouvelles techniques de classification et de réductions de dimension des documents textuels.
- La deuxième approche prend en compte seulement la structure du document, dont les travaux visent à réduire la complexité des document XML par des transformation de la structure arborescente.
- La troisième approche est la plus complexe, puisque l'objectif est de bénéficier d'information issue de la structure et du contenu d'un document XML, en les combinons par divers méthodes.

Lors de cette synthèse nous avons remarqué que le modèle de représentation vectoriel est le plus souvent utilisé : soit en transformant le document XML pour qui'il s'adapte à ce modèle, soit en l'améliorant afin de prendre en compte la structure du document.

Chaque méthode de classification présente des avantage et des limites, et afin de tirer partie des avantages de chacune, nous avons pensé à utiliser la théorie de la fusion d'information.

Dans le chapitre suivant, nous allons introduire cette théorie, leurs principes et méthodes.

Auteurs	Contenu	Structure	Méthode de représentation	Méthode de classification	meilleur résultat	Corpus
[Meenakshi *et al.*, 2007]	*Oui*	*Non*	la fréquence de documents de catégorie négative (NCD)	similarité cosinus, similarité fractionnaire	Micro rappel = 0.780 Macro rappel = 0.757	**INEX 2007** (96 000 docs)
[Géry *et al.*, 2008]	*Oui*	*Non*	vectoriel par sélection proposé (*CC*, *CCE*)	SVM	Macro rappel = 0.7876	**INEX 2008** (114 336 docs)
[Knijf, 2006]	*Non*	*Oui*	les sous arbres d'attributs fréquents	pattern mining	Micro F1 = 0.480 Macro F1 = 0.527	**Wikipedia SC** INEX 2006 (150 000 docs)
[Garboni *et al.*, 2005]	*Non*	*Oui*	transformer les sous arbres fréquents en une séquence	règles d'association + mesure de distance	Macro rappel entre 0.8 et 0.95	**Movie SO** INEX (9 463 docs)
[Candillier *et al.*, 2005]	*Non*	*Oui*	ensembles d'attributs-valeurs	arbre de décision, *SSC*	Micro rappel = 0.968 Macro rappel = 0.960	**Movie SO** INEX (9 463 docs)
[Wisniewski *et al.*, 2005]	*Non*	*Oui*	modèle probabiliste	réseau bayésien		**IEEE SC** (INEX 1.3) (12 108 docs)
[Ghosh et Mitra, 2008]	*Oui*	*Oui*	vectoriel (*TPaths*) + vectoriel (Contenu textuel)	SVM à noyau composé	Micro rappel = 0.862 Macro rappel = 0.846	**IEEE SC** (INEX 1.3) (12 108 docs)
[de Campos *et al.*, 2007]	*Oui*	*Oui*	Transformer le document XML en texte plat (vectoriel)	Réseau bayésien, bayésien naïf	Micro rappel = 0.790 Macro rappel = 0.760	**INEX 2007** (96 000 docs)
[Yang et Zhang, 2007]	*Oui*	*Oui*	Modèle à vecteur de lien structuré (SLVM)	SVM avec noyau personnalisé	Micro rappel = 0.872 Macro rappel = 0.839	**INEX 2007** (96 000 docs)

Table 2.7 – *Un tableau récapitulatif du synthèse des travaux de recherche*

Chapitre 3

La fusion d'informations

3.1 introduction

Bien que le concept de fusion d'informations ait explosé relativement récemment dans la littérature scientifique, il est utilisé depuis fort longtemps dans la nature : ainsi les animaux eux-mêmes exploitent l'information auditive et visuelle pour se protéger contre d'éventuels risques. De même, pour localiser un emplacement quelconque, ils combinent plusieurs images visuelles des endroits avoisinant le lieu considéré.

La fusion d'informations est apparue afin de gérer des quantités très importantes de données multisources dans le domaine militaire. Depuis quelques années des méthodes de fusion ont été adaptées et développées pour des applications en traitement du signal et plus particulièrement pour la classification.

Dans ce chapitre et afin de comprendre la théorie de la fusion d'informations. Nous commençons notre étude par la définition de la notion de cette notion, nous allons présenter les différentes niveaux et stratégies de la fusion. Nous présenterons, par la suite, quelques méthodes ou techniques de fusion qui nous ont paru les plus importantes dans la littérature parmi des nombreuses méthodes possibles.

3.2 La fusion d'informations

Il existe plusieurs définition de la fusion d'informations, varie d'un scientifique à l'autre. Nous avons choisi les plus simples.
Le groupe européen SEE (Société d'électricité et d'électronique), la branche française de l'Institute of Electric and Electronics Engineers (IEEE), et la branche européenne de l'International Society for Photogrammetry and Remote Sensing (ISPRS) ont proposé la définition suivante :

Définition1 *la fusion d'informations constitue un cadre formel dans lequel s'expriment les moyens et techniques permettant l'alliance des données provenant de sources diverses* [Arif, 2005].

Définition2 *la fusion d'informations consiste à combiner des informations issues de plusieurs sources afin d'aider à la prise de décision* [Le Cadre *et al.*, 2003].

3.2.1 Niveaux de fusion

On peut trouver dans la littérature plusieurs manières de classer les différentes étapes ou types de fusion. Cette différence provient principalement du niveau où l'opération de fusion est accomplie. Nous avons choisi celle qui convient au domaine de la fusion de classifieurs.
Dans [Xu *et al.*, 1992] trois niveaux de fusion sont décrits de la manière suivante :

(i) **Fusion au niveau de la représentation :** un classifieur e attribue à chaque classe des scores traduisant la similarité calculée entre la forme entrante x et les autres formes de la classe ou en donnant un certain degré de ressemblance entre telle ou telle classe.

(ii) **Fusion au niveau des rangs :** un classifieur e donne les rangs de toutes les classes, le premier rang étant réservé à son choix préféré.

(iii) **Fusion au niveau de la décision :** un classifieur e indique sa décision d'une classe ou parfois un sous-ensemble de classes.

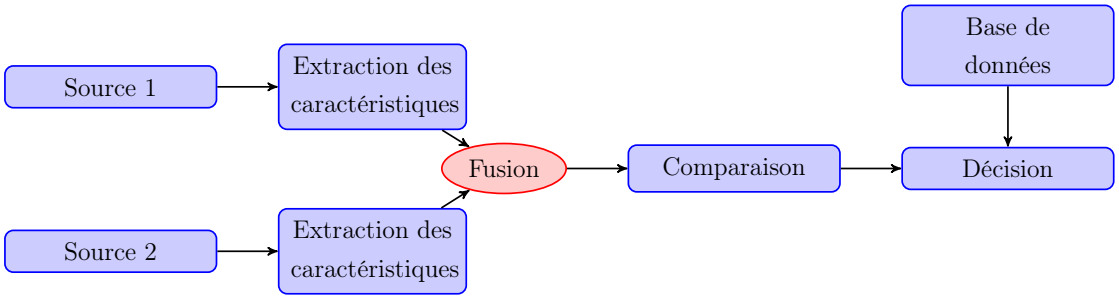


Figure 3.1 – *fusion au niveau des caractéristiques (bas niveau)*

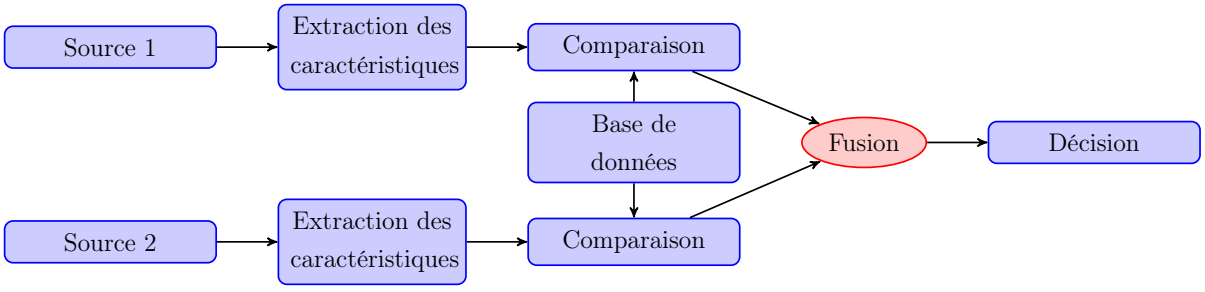


Figure 3.2 – *fusion au niveau représentation ou niveau des rangs (niveau intermédiaire)*

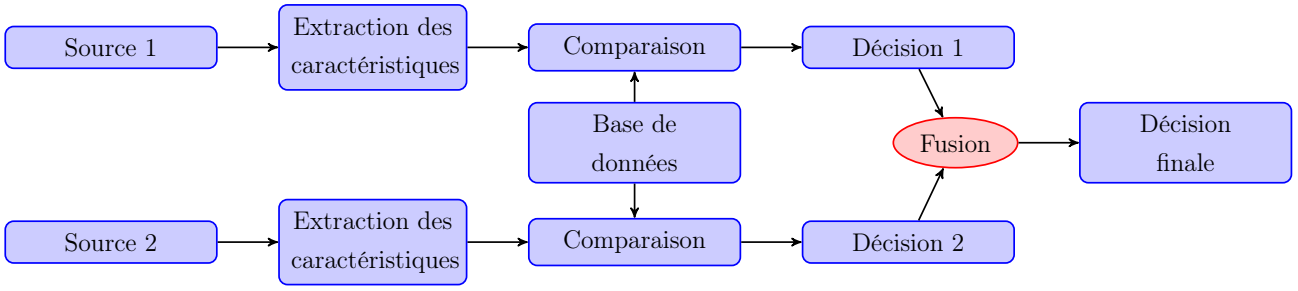


FIGURE 3.3 – *fusion au niveau décision (haut niveau)*

L'avantage de la fusion bas niveau est qu'elle permet de combiner des données avant qu'elles ne soient déformées par les procédures de traitement et d'analyse. Par contre, la fusion aux niveaux intermédiaire et haut permet de se rapprocher davantage du raisonnement de l'expert [Arif, 2005].
Dans la communauté de la recherche documentaire, la fusion d'informations s'appelle parfois la combinaison de doutes, et la fusion de décisions s'appelle parfois la combinaison de données [Wald, 2002].

3.2.2 Stratégies de fusion

Les stratégies ou architectures de fusion décrivent l'ensemble des sources, la manière dont elles sont assemblées et les techniques mathématiques ou statistiques pour le traitement. La multiplication des travaux sur la fusion dans les domaines différents (imagerie, intelligence artificielle et reconnaissance de formes, etc.) a entraîné la mise au point de nombreux schémas traitant les données de manières différentes [Arif, 2005].
Les stratégies de fusion proposées peuvent être regroupées en trois approches principales [Zouari, 2004]

Approche séquentielle

La combinaison séquentielle, appelée également combinaison série, est organisée en niveaux successifs de décision permettant de réduire progressivement le nombre de classes possibles. Dans chaque niveau, il existe un seul classifieur qui prend en compte la réponse fournie par le classifieur placé en amont afin de traiter les rejets ou confirmer la décision obtenue sur la forme qui lui est présentée (figure 3.4).

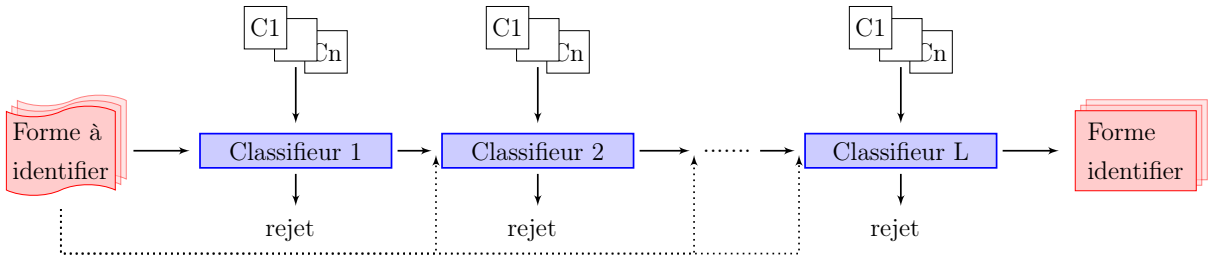


FIGURE 3.4 – *Combinaison séquentielle de classifieurs*

Une telle approche peut être vue comme un filtrage progressif des décisions dans la mesure où elle permet de diminuer au fur et à mesure l'ambigüité sur la classe proposée. Cela permet généralement de diminuer le taux d'erreur globale de la chaine de reconnaissance. Néanmoins, une combinaison de ce type demeure particulièrement sensible à l'ordre dans lequel sont placés les classifieurs. En effet, même s'ils ne nécessitent pas d'être les plus performants, les premiers classifieurs invoqués doivent être robustes, c'est-à-dire que la solution réelle de la forme à identifier doit apparaître dans les listes successives quelle que soit leur taille. En cas de mauvaise décision du premier classifieur, placé en amont de la série des classifieurs utilisés, l'erreur va se propager de façon irrévocable. Il faudra donc choisir judicieusement le premier classifieur afin d'éviter - autant que possible - l'apparition d'une telle situation. La combinaison séquentielle suppose donc une certaine connaissance *a priori* du comportement de chacun des classifieurs. Notons que dans cette approche, chaque classifieur est réglé en fonction du classifieur placé en amont de la chaîne. Une simple modification du premier classifieur peut provoquer un ré-paramétrage (ré-apprentissage) des classifieurs suivants.

Approche parallèle

A la différence de l'approche séquentielle, l'approche parallèle laisse dans un premier temps les différents classifieurs opérer indépendamment les uns des autres puis fusionne leurs réponses respectives. Cette fusion est faite soit de manière démocratique, dans le sens où elle ne favorise aucun classifieur par rapport à un autre, soit au contraire dirigée et, dans ce cas, on attribue à la réponse de chaque classifieur un poids en fonction de ses performances. L'ordre d'exécution des classifieurs n'intervient pas dans cette approche. La figure 3.5 fournit une représentation de la combinaison parallèle de classifieurs.

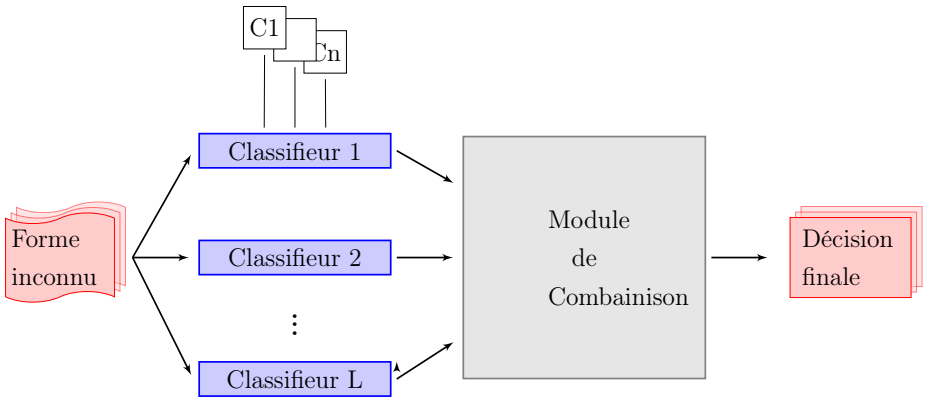


FIGURE 3.5 – *Combinaison parallèle de classifieurs*

L'inconvénient majeur de l'approche parallèle est qu'elle nécessite l'activation de tous les classifieurs du système qui doivent participer de manière concurrente et indépendante. Par contre, la décision finale est prise avec le maximum de connaissances mises à disposition par chaque classifieur. Dès lors se posent les problèmes de précision des informations fournies par les classifieurs et de la confiance qu'on peut accorder à chacun d'eux.

Approche hybride

L'approche hybride consiste à combiner à la fois des architectures séquentielles et parallèles afin de tirer pleinement avantage de chacun des classifieurs utilisés. La figure 3.6 présente un exemple de combinaison hybride dans laquelle on combine un classifieur en série avec deux classifieurs en parallèle.

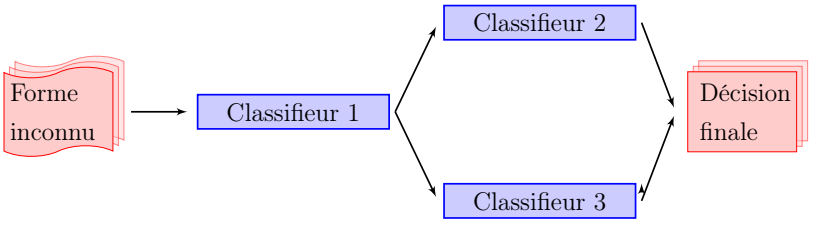


FIGURE 3.6 – *Combinaison hybride de classifieurs*

Ce type d'approche permet de générer de nombreux schémas de coopération qui peuvent rapidement devenir complexes et ont besoin d'optimisation. Il illustre les deux aspects de la combinaison qui sont d'une part la réduction de l'ensemble des classes possibles et d'autres part la recherche d'un consensus entre les classifieurs afin d'aboutir à une décision unique.

En prenant en compte les avantages mentionnés précédemment, nous nous intéressons tout au long des prochaines sections à la fusion haute niveau, avec une combinaison parallèle de classifieurs.

3.3 méthodes de fusion de classfieurs

Les principales méthodes pour la fusion d'informations sont : le principe du vote majoritaire, l'approche bayésienne, les méthodes issues de la théorie des possibilités et des ensembles flous et celles issues de la théorie de Dempter-Shafer. A ces méthodes s'ajoutent un grand nombre de techniques qui peuvent être vues comme de la fusions d'informations.
Avant de rentrer dans les détails des méthodes de Fusion, il est nécessaire de rappeler ce qu'on entend généralement par classifieur dans le cadre de la combinaison des classifieurs.

Définition d'un classifieur

Étant donné un ensemble de classes C_i, $i \in [1, \cdots, N]$. Dans le cas le plus général, on peut associer à la forme à reconnaître x un vecteur de degré d'appartenance $D(x)$ tel que [Zouari, 2004] :

$$D(x) = \begin{bmatrix} D^1(x) \\ D^2(x) \\ \vdots \\ D^N(x) \end{bmatrix} \tag{3.1}$$

avec $D^i(x) = D\{x \in C_i\}$. Dans ce cas, x peut appartenir à plusieurs classes si $D^i(x) \neq 0$. Toutefois, dans la majorité des problèmes de classification, on a affaire à une classification exclusive dans laquelle une forme ne peut appartenir qu'à une seule classe. On a alors $D^i(x) = \delta_{i,j}$ tel que :

$$\delta_{ij} = \begin{cases} 1 & \text{si } j = i \\ 0 & \text{si } j \neq i \end{cases} \tag{3.2}$$

C_i est alors souvent appelée "la vraie classe" :

$$D(x) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.3}$$

Selon [Xu *et al.*, 1992], cette réponse peut être divisée en trois catégories suivant le niveau d'information apporté par le classifieur :

type classe :

$$e_j(x) = C_i, i \in \{1, \cdots, N\} \tag{3.4}$$

C_i est la classe attribuée par le classifieur e_j à x. Dans ce cas, l'avis du classifieur est binaire. On peut alors représenter la réponse du classifieur par un vecteur binaire dans lequel '**1**' indique la classe proposée par le classifieur. Un classifieur peut aussi produire un ensemble de classes. Il considère alors que la forme x appartient à une des classes de cet ensemble sans donner d'autres informations permettant de discriminer les classes.

type rang :

$$e_j(x) = (r_{1,j}, r_{2,j}, \cdots, r_{N,j}) \tag{3.5}$$

$r_{i,j}$ est le rang attribué à la classe C_i par le classifieur e_j. Il s'agit d'un classement sur les classes. Le classifieur indique ce classement en fournissant en sortie un vecteur des rangs de taille N. La classe placée au premier rang de la liste proposée par le classifieur est considéré comme la plus probable pour la forme x et la classe du dernier rang est la moins probable.

type mesure :

$$e_j(x) = (m_{1,j}, m_{2,j}, \cdots, m_{N,j}) \tag{3.6}$$

$m_{i,j}$ est la mesure attribuée à la classe i par le classifieur e_j. Elle indique le niveau de confiance du classifieur dans sa proposition. La sortie du classifieur est donc un vecteur de mesures de taille N. Cette mesure, normalise ou non, peut être une distance, une probabilité a posteriori, une valeur de confiance, un score, une fonction de croyance, une possibilité, une crédibilité, une mesure floue, ... etc.

Chaque type de sortie (**classe**, **rang** ou **mesure**) correspond à un niveau d'information différent fourni par le classifieur. La sortie de type classe est la plus simple mais la moins riche en information. La sortie de type rang reflète l'ordre de préférence des propositions fournies par le classifieur. La sortie de type mesure est la plus riche en information puisqu'elle reflète le niveau de confiance du classifieur dans ses propositions.

Lorsque les sorties sont de type rang ou de type mesure, on peut évidemment les transformer en type classe (avec perte d'informations)(figure 3.7). Ceci consiste à tenir compte uniquement de la première solution de la liste proposée par chaque classifieur. Pour le type rang, il suffit de choisir la classe qui est placée au premier rang. Les autres classes ne seront pas prises en compte. Pour le type mesure, il suffit de choisir la classe ayant la meilleure mesure (valeur minimale ou maximale, selon que la mesure est croissante ou décroissante).

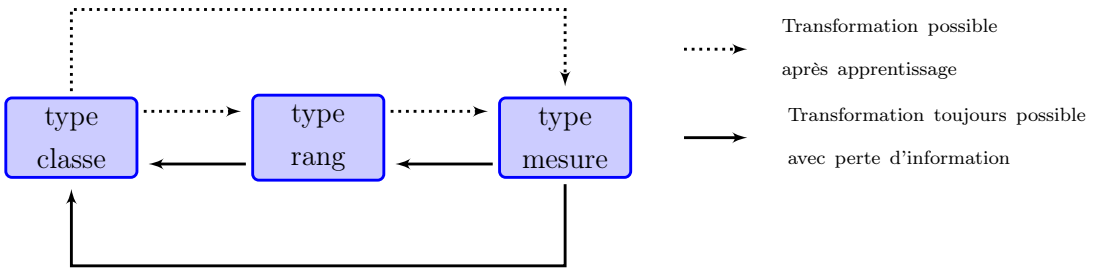


FIGURE 3.7 – *Transformation des sorties de classifieurs*

Le problème de fusion de classifieurs se pose de la manière suivante [Zouari, 2004] : étant donné un ensemble de L classifieurs, participant de manière indépendante au même problème de classification, comment peut-on élaborer une réponse finale à partir des résultats de ces classifieurs ? Ce problème nécessite l'utilisation d'un module de fusion pour élaborer une décision finale.
Considérons un système composé d'un ensemble de L classifieurs. Pour reconnaître une forme x, chaque classifieur e_j produit une réponse sous forme d'un vecteur

$$e_j(x) = [e_{1,j}(x), e_{2,j}(x), \cdots, e_{N,j}(x)]^T \tag{3.7}$$

La composante $e_{i,j}(x)$ indique que le classifieur e_j a attribué à la forme x la classe C_i parmi l'ensemble de N classes possibles ($i = 1, \cdots, N$). Cette réponse peut être de type **classe**, **rang** ou **mesure**. Nous pouvons représenter toutes les décisions des classifieurs sous la forme d'une matrice $e_j(x) = [e_1(x), e_2(x), \cdots, e_L(x)]$ qui représente les réponses de l'ensemble des classifieurs pour une forme x. Chaque ligne i de cette matrice représente la réponse de tous les classifieurs concernant la classe C_i. Chaque colonne j représente le vecteur de réponse d'un classifieur e_j . Il s'agit de l'espace intermédiaire de caractéristiques du module de combinaison E.

$$MD(x) = \begin{pmatrix} e_{1,1}(x) & \cdots & e_{1,j}(x) & \cdots & e_{1,L}(x) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{i,1}(x) & \cdots & e_{i,j}(x) & \cdots & e_{i,L}(x) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ e_{N,1}(x) & \cdots & e_{N,j}(x) & \cdots & e_{N,L}(x) \end{pmatrix} \tag{3.8}$$

La décision de fusion peut alors être obtenue par :

$$E(x) = \mathbf{f}(e_1, e_2, \cdots, e_L) \tag{3.9}$$

où $\mathbf{f}$ est appelé opérateur de combinaison ou d'agrégation.

3.3.1 Théorie des probabilités : approche bayésienne

La notion de probabilité fut introduite par Pascal (1623-1662) afin de traduire le hasard dans le jeu. On distingue aujourd'hui différentes approches telles que, les modèles théoriques, la fréquence d'occurrence d'un évènement et une définition dite subjective. Cette dernière probabilité est appelée mesure de confiance Bayésienne [Arif, 2005].
L'avantage majeur des probabilités est qu'elles reposent sur de solides bases mathématiques, et qu'il existe une grande variété de méthodes d'apprentissage pour estimer les lois de probabilité. Les méthodes bayésiennes sont utilisées pour fusionner des sorties de classifieurs exprimées en probabilités a posteriori.
L'utilisation de la théorie de Bayes consiste à déterminer la classe C_i pour laquelle la probabilité a posteriori $P(C_i|e_1 = C_1, \cdots, e_L = C_L)$ est maximum, c'est-à-dire :

$$E(x) = C_i \text{ si } P(C_i|e_1 = C_1, \cdots, e_L = C_L) = \max_{m=1}^{N} P(C_m|e_1 = C_1, \cdots, e_L = C_L) \tag{3.10}$$

Pour estimer les probabilités a posteriori, plusieurs études supposent que les classifieurs sont indépendants [Xu *et al.*, 1992]. Sous cette hypothèse, la probabilité a posteriori s'écrit :

$$P(C_i|e_1 = C_1, \cdots, e_L = C_L) = P(C_i) \prod_{l=1}^{L} \frac{P(C_i|e_l = C_l)}{P(C_i)} \tag{3.11}$$

Les probabilités $P(C_i|e_l = C_l)$ peuvent être déterminées à partir de la matrice de confusion obtenue pour les classifieurs sur une base d'apprentissage. Si on note $n^j_{C_i,C_l}$ le nombre d'éléments de cette base pour lesquels le classifieur e_j attribue les éléments de la classe C_i à la classe C_l, et $n^j_{.,C_l}$ le nombre total d'éléments attribués par le classifier e_j à la classe C_l, alors on peut écrire :

$$P(C_i|e_j = C_l) = \frac{n^j_{C_i,C_l}}{n^j_{.,C_l}} \tag{3.12}$$

3.3.2 Théorie de l'évidence (Dempster-Shafer)

La théorie de l'évidence, proposée par Glenn Shafer en 1976 [Shafer, 1976], prenant appui sur les bases formulées par Dempster [Dempster, 1967], est aussi appelée « théorie de Dempster-Shafer » ou « théorie des croyances ». Dans son essai, Shafer propose une nouvelle interprétation des travaux de Dempster qui identifie les probabilités inférieures à des degrés de croyance tout en conservant les règles de combinaison de ces degrés de croyance. La théorie Bayésienne est alors envisagée comme un cas particulier de cette théorie. Cette théorie permet de gérer les situations d'ignorance ce qui n'est pas le cas dans le cadre de la théorie des probabilités. La modélisation

des informations se fait à l'aide de fonctions de croyance. Une fois les fonctions de croyance obtenues, la fusion est réalisée par l'intermédiaire de la règle de combinaison de Dempster.

Au contraire de la théorie des probabilités, l'utilisation de la théorie de l'évidence ne nécessite pas de connaissance *a priori* sur le problème à traiter d'une part et permet d'autre part de répartir la croyance non seulement sur les hypothèses élémentaires mais aussi sur des compositions d'hypothèses.

Principe

Soit $\Omega = \{H_1, H_2, H_3, \cdots, H_M\}$ l'ensemble des propositions possibles du problème posé (dans le cas de classification M est l'ensemble des classes), appelé cadre de discernement ou corps évidentiel. On suppose que le cadre de discernement est exhaustif et que les hypothèses sont exclusives. Cette notion est aussi appelée monde fermé (closed world). Toutefois, il est possible de s'affranchir de cette condition en admettant que l'ensemble Ω est un cadre de discernement non exhaustif. Cette approche est alors appelée hypothèse du monde ouvert (open world) [Smets, 1990]. Nous préférons garder l'hypothèse d'exhaustivité de Ω. A partir de cet ensemble Ω, on définit un ensemble noté 2^Ω, l'ensemble des 2^Ω parties A de Ω. Cet ensemble est défini de la manière suivante :

$$2^\Omega = \{A | A \subseteq \Omega\} = \{\phi, \{H_1\}, \cdots, \{H_m\}, \{H_1, H_2\}, \cdots, \Omega\}. \tag{3.13}$$

Cet ensemble contient les hypothèses singletons de Ω, toutes les disjonctions possibles de ces hypothèses ainsi que l'ensemble vide. Par la suite, nous noterons H_n une hypothèse singleton, et A une proposition désignant indifféremment une hypothèse ou une disjonction d'hypothèses. Une information (qui peut être issue d'un capteur, d'un agent, d'un expert,...) traduisant une opinion sur l'état d'un système est caractérisée par des degrés de croyance dans les différentes hypothèses. Ces degrés de croyance peuvent être décrits par une *fonction de croyance* ou *d'allocation de masse* notée m qui est définie par :

$$m : 2^\Omega \rightarrow [0, 1] \tag{3.14}$$

Cette fonction vérifie les propriétés suivantes :

$$\begin{cases} m(\phi) = 0 \\ \sum_{A \subseteq \Omega} m(A) = 1 \end{cases} \tag{3.15}$$

La quantité $m(A)$ est la part de croyance placée strictement sur A. La quantité se différencie d'une probabilité par le fait que la totalité de la croyance est répartie non seulement sur les hypothèses singletons H_n mais aussi sur les hypothèses composites A. On peut alors accorder une partie de la croyance à une proposition, et ainsi affecter à l'ensemble des hypothèses contenues dans la proposition une croyance à la réalisation de chacune d'entre-elles sans prendre parti pour

l'une d'elles précisément.
Les sous-ensembles de 2^{Ω} dont la masse est non nulle, sont appelés *éléments focaux*. Une situation d'ignorance totale est donnée par $m(\Omega) = 1$ et de certitude totale (sur une hypothèse singleton) par $m(H_n) = 1$.

La fonction d'allocation de masse m permet de construire la notion de *Crédibilité* (croyance) et de *Plausibilité* :

La fonction de Crédibilité est définie par :

$$\begin{cases} Bel(\phi) = 0 \\ Bel(A) = \sum_{B \subseteq A} m(B), \forall A \subseteq \Omega, A \neq \phi \end{cases} \tag{3.16}$$

$Bel(A)$ regroupe l'ensemble des croyances apportées par les éléments qui composent A. Elle correspond à la quantité d'information qui est tout entière contenue dans le sous ensemble considéré.

La fonction de plausibilité, notée Pl, exprime que plus un élément est vrai, moins son contraire l'est. Si A est une proposition, $Pl(A)$ mesure combien l'information apportée par une source ne contredit pas A. Elle est définie :

$$\begin{cases} Pl(\phi) = 0 \\ Pl(A) = \sum_{B \cap A \neq \phi} m(B), \forall A \subseteq \Omega, A \neq \phi \end{cases} \tag{3.17}$$

La plausibilité de A est également reliée à la crédibilité du complémentaire de A. Elle correspond à toute l'information ne créditant pas la véracité du complémentaire de A.

$$Pl(A) = 1 - Bel(\bar{A}) \tag{3.18}$$

Combinaison des croyances

Si nous disposons de plusieurs sources d'information, et de fonctions d'allocation de masse, le problème est de combiner les informations. Par exemple pour deux sources S_1 et S_2 produisant les masses m_1 et m_2, le vecteur de masse m par fusion ($m = m_1 \oplus m_2$), d'après la règle orthogonale de Dempster, est donné par :

$$m(H) = \frac{1}{1-K} \sum_{A \cap B = H} m_1(A) \cdot m_2(B) \qquad \forall H \subseteq \Omega, \text{ et } H \neq \phi \tag{3.19}$$

Où,

$$K = \sum_{A \cap B = \phi} m_1(A) \cdot m_2(B) \tag{3.20}$$

K est la masse de croyance conflictuelle existant entre les fonctions de croyance à combiner.

Si $K = 0$, les sources sont en parfait accord mais si K est égal à 1, elles sont en conflit total. Alors, les informations ne peuvent pas être fusionnées par cette méthode.

La décision se fait en faveur de la classe qui a, soit la plus grande crédibilité (choix le plus pessimiste), soit la plus grande plausibilité (choix le plus optimiste) [Bloch, 1996].

3.3.3 Théorie des possibilités (approche possibiliste)

Après avoir introduit la théorie des ensembles flous [Zadeh, 1965], en 1978, Zadeh a proposé la théorie des possibilités à partir de sa théorie des sous ensembles flous [Zadeh, 1978], puis développée par Dubois et Prade [Dubois et Prade, 1985].

L'intérêt de la fusion par le biais de la théorie des possibilités est de pouvoir combiner des informations plus ou moins précises et fiables, de différentes origines, afin de fournir une information globale de meilleure qualité. Les données sont représentées sous forme numérique continue, en s'appuyant sur la théorie des ensembles flous [Ploix *et al.*, 2008].

La théorie des possibilités considère certaines situations plus ou moins possibles par rapport à d'autres. Elle modélise, non pas un degré de croyance ou de vérité, mais plutôt la préférence que l'on a pour une proposition, c'est-à-dire un moyen de dire dans quelle mesure la réalisation d'un évènement est possible et dans quelle mesure on en est certain. Dans cette théorie, on formalise donc ces deux évaluations subjectives à travers une mesure de *possibilité* et une mesure de *nécessité*. Ces deux mesures prennent leurs valeurs dans l'intervalle [0,1]. Un évènement est tout à fait possible si la mesure de sa possibilité est égale à 1, et impossible si celle-ci est nulle [Arif, 2005].

Principes

Nous rappelons ici quelques principes fondamentaux de cette théorie [Arif, 2005].

Mesure de possibilité

Soit un référentiel U. On définit une mesure de **possibilité** Π sur l'ensemble $P(U)$ des parties de U par les trois axiomes fondamentaux suivants :

i- on peut définir la possibilité d'un évènement par un coefficient (degré de possibilité) compris entre 0 et 1 :

$$\Pi : P(U) \rightarrow [0, 1] \tag{3.21}$$

ii- l'ensemble de référence U est complètement possible ou complètement compatible avec la connaissance disponible ; on lui attribue alors le degré de possibilité maximum 1. A

l'inverse l'ensemble vide reçoit le degré zéro :

$$\begin{aligned}\Pi(U) &= 1 \\ \Pi(\phi) &= 0\end{aligned} \tag{3.22}$$

iii- la possibilité d'un évènement formé par une collection d'éléments est égale au degré de possibilité de la valeur préférée parmi ses éléments, dans le sens où la valeur préférée est celle possédant le plus grand degré de possibilité :

$$A_i \in P(U), \Pi(\cup_i A_i) = \sup_i \Pi(A_i) \tag{3.23}$$

Mesure de nécessité

La mesure de nécessité N est une mesure duale de la mesure de possibilité. $N(A)$ indique le degré avec lequel la réalisation d'un évènement A est certaine. En effet, la possibilité s'avère insuffisante pour représenter l'information sur un évènement puisqu'elle nous informe uniquement sur l'occurrence de cet évènement. On peut constater que deux évènements A et son contraire $\bar{A}$ peuvent être tous les deux possibles, ce qui correspond à une situation d'ignorance totale. C'est pour lever cette ambiguïté que cette mesure complémentaire a été introduite. On dit qu'un évènement est certain si son évènement contraire est impossible, d'où le lien entre les deux mesures :

$$\forall A \in P(U), N(A) = 1 - \Pi(\bar{A}) \tag{3.24}$$

Distribution de possibilités

Pour éviter de travailler sur l'ensemble $P(U)$ des parties de U, on définit une fonction directement sur U qu'on appelle distribution de possibilité. Elle associe à chaque élément de U une valeur dans [0,1]. Afin que la distribution soit normalisée, il suffit qu'il existe un élément de U qui soit complètement possible :

$$\exists x_0 \in U, \pi(x_0) = 1 \tag{3.25}$$

La distribution de possibilité est directement liée à la notion de mesure de possibilité par :

$$\forall A \in P(U), \Pi(A) = \sup\{\pi(x), x \in A\} \tag{3.26}$$

L'expression 3.26 peut être étendue au cas où A est un sous-ensemble flou (caractérisé par sa fonction d'appartenance $\mu_A(x)$) :

$$\Pi(A) = \sup_{x \in U}(\min(\mu_A(x), \pi(x))) \tag{3.27}$$

Opérateurs de fusion

Le principe général est de trouver les zones d'accord et de désaccord entre les sources, pour en extraire une ou des informations suffisamment fiables. Il n'existe pas d'unique combinaison satisfaisante pour tous les problèmes. Toute la difficulté dans le choix d'un opérateur consiste à chercher un bon compromis entre un résultat précis mais sûrement faux et un résultat certain mais trop imprécis [Ploix *et al.*, 2008].

Les deux opérateurs fondamentaux en théorie des possibilités sont :

- le mode conjonctif (« et » logique), qui représente un comportement sévère (notion généralisée d'intersection d'ensembles). Dans cette catégorie se trouvent les normes triangulaires, ou **t-normes**,
- le mode disjonctif (« ou » logique), qui représente un comportement indulgent (notion généralisée d'union d'ensembles). Dans cette catégorie se trouvent les conormes triangulaires, ou **t-conormes**.

Le tableau 3.1 montre quelques exemples des t-normes et t-conormes les plus couramment utilisées, et la figure 3.8 montre un exemple de trois distributions à fusionner, et le résultat de la fusion en utilisant d'une part la t-norme et d'autre part la t-conorme de Zadeh.

nom	**t-norme**	**t-conorme duale**
Zadeh	$\min(\pi_1, \pi_2)$	$\max(\pi_1, \pi_2)$
Probabiliste	$\pi_1 \cdot \pi_2$	$\pi_1 + \pi_2 - \pi_1 \cdot \pi_2$
Lukasiewicz	$\max(0, \pi_1 + \pi_2 - 1)$	$\max(1, \pi_1 + \pi_2)$

Table 3.1 – Exemples de T-normes et t-conormes les plus courantes.

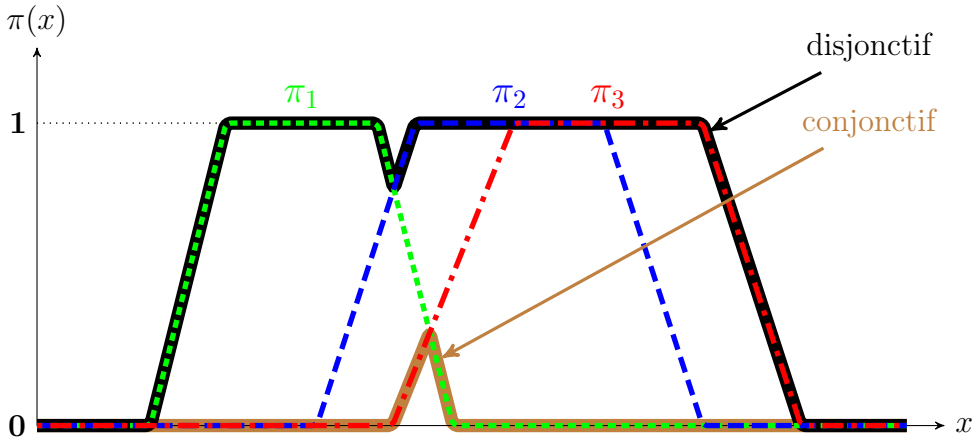


Figure 3.8 – *T-norme et t-conorme de Zadeh combinant 3 distributions.*

3.3.4 Méthodes des votes

Il y a maintenant plus de 200 ans que le principe du vote a été formalisé par le mathématicien et philosophe Condorcet [Condorcet, 1785]. Depuis, plusieurs méthodes de vote ont été proposées. Ces méthodes sont les plus simples à mettre en œuvre ; elles sont sans apprentissage et non paramétriques. Elles sont surtout utilisées dans les élections.

Dans le cadre de la fusion au niveau de la décision, les méthodes de vote consistent à interpréter chaque sortie d'un classifieur comme un vote pour l'une des classes possibles. La classe ayant un nombre de votes majoritaire ou supérieur à un seuil préfixé est retenue comme décision finale. Les votes des classifieurs ne sont pas pondérés et chaque classe reçoit autant de votes qu'il y a de classifieurs à combiner. La plupart de ces méthodes ne nécessitent qu'un seul niveau de décision [Zouari, 2004].

Le principe du vote est une méthode de combinaison particulièrement adaptée aux décisions de type symbolique (type classe).
Notons $S_j(x) = i$ le fait que la source S_j attribue la classe C_i à l'observation x. Nous supposons ici que les classes C_i sont exclusives. A chaque source nous associons la fonction indicatrice [Martin, 2005] :

$$M_i^j(x) = \begin{cases} 1 & \text{si } S_j(x) = i, \\ 0 & \text{sinon} \end{cases} \tag{3.28}$$

La combinaison des sources s'écrit par :

$$M_k^E(x) = \sum_{j=1}^{m} M_k^j(x) \tag{3.29}$$

pour tout k. L'opérateur de combinaison est donc associatif et commutatif. La règle du vote majoritaire consiste à choisir la décision prise par le maximum de sources, c'est-à-dire le maximum de M_k^E. Cependant cette règle simple n'admet pas toujours de solutions dans l'ensemble des classes $D = \{C_1, \cdots, C_n\}$. En effet, par exemple si le nombre de sources m est paire et que $\frac{m}{2}$ sources décident C_{i1} et $\frac{m}{2}$ autres sources disent C_{i2} , ou encore dans le cas où chaque source affecte à x une classe différente. Nous sommes donc obligé d'ajouter une classe C_{n+1} qui représente l'incertitude totale liée au conflit des sources sous l'hypothèse de l'exhaustivité des classes $C_{n+1} = \{C_1, \cdots, C_n\}$. La décision finale de l'expert prise par cette règle s'écrit donc par :

$$E(x) = \begin{cases} k & \text{si } \max_k M_k^E(x), \\ m+1 & \text{sinon} \end{cases} \tag{3.30}$$

Cette règle est cependant peut satisfaisante dans les cas où deux sources donnent le maximum

pour des classes différentes. La règle la plus employée est la règle du vote majoritaire absolu. Soit r donné par :

$$r = \begin{cases} \frac{m}{2} & \text{si } m \text{ est pair,} \\ \frac{m+1}{2} & \text{si } m \text{ est impair} \end{cases} \tag{3.31}$$

La règle du vote majoritaire absolu s'écrit par :

$$E(x) = \begin{cases} k & \text{si } \max_k M_k^E(x) \geq r, \\ n+1 & \text{sinon} \end{cases} \tag{3.32}$$

A partir de cette règle il a été démontré [Lam et Suen, 1997] que plusieurs résultats prouvant que la méthode du vote permet d'obtenir de meilleurs performances que toutes les sources prises séparément, sous des hypothèses d'indépendance statistique des sources et de même probabilité, et ceci est d'autant plus vrai que m est impaire.

Il est possible de généraliser le principe du vote majoritaire afin de supprimer le conflit. Au lieu de combiner les réponses des sources par une somme simple comme dans l'équation 3.29, l'idée est d'employer une somme pondérée :

$$M_k^E(x) = \sum_{j=1}^{m} \alpha_j^k M_k^j(x) \tag{3.33}$$

où $\sum_{j=1}^{m} \sum_{k=1}^{n} \alpha_j^k = 1$. Les poids α_j^k représentent la fiabilité d'une source pour une décision donnée, et l'estimation de ces poids peut se faire à partir des taux normalisés de réussite pour chaque classe et chaque classifieur. Notons qu'alors nous introduisons une connaissance *a priori* non nécessaire précédemment. Les différentes règles de décision possibles peuvent se résumer par la formule suivante :

$$E(x) = \begin{cases} k & \text{si } \max_i M_i^E(x) \geq cm + b(x), \\ n+1 & \text{sinon} \end{cases} \tag{3.34}$$

où c est une constante de [0,1] et $b(x)$ est une fonction de $M_k^E(x)$.

3.3.5 Méthodes de rangs (Borda Count)

Il s'agit d'une procédure de vote proposée en 1770 par Jean Charles de Borda dans laquelle chaque voteur arrange les N candidats selon sa préférence en attribuant un nombre de points à chacun. Le candidat préféré reçoit $N-1$ points, le candidat suivant reçoit $N-2$, ainsi de suite. Le nombre de points du candidat placé à la fin de la liste est 0. La règle de Borda offre donc un seul ensemble de rangs " échelle ", $N-1, N-2, \cdots, 0$. Certains auteurs attribuent les rangs autrement : N points à la classe placée en tête (top) de chaque liste de classifieurs, $N-1$ à la

classe suivante et 1 à la dernière classe. Pour un problème à deux classes, le Borda Count est un vote à la majorité. Pour une classe particulière $C_i(i = 1, ..., N)$, le Borda Count $BC(C_i)$ est la somme de tous les rangs proposés par les classifieurs pour cette classe [Zouari, 2004].

$$BC(C_i) = \sum_{i=1}^{L} r_{i,j} \tag{3.35}$$

La méthode de Borda a deux variantes [Van-Erp et Schomaker, 2000]. La méthode de Borda Médiane est une variante très proche de la méthode de Borda originale. Au lieu de sommer les votes relatifs aux indices des différentes classes dans les classifieurs, on en extrait la médiane.

$$BC(C_i) = \begin{cases} \frac{r_{i,\frac{L}{2}} + r_{i,\frac{L}{2}+1}}{2} & \text{si } L \text{ est pair} \\ r_{i,\frac{L+1}{2}} & \text{si } L \text{ est impair} \end{cases} \tag{3.36}$$

La méthode de Nanson exécute M – 1 fois la méthode de Borda originale sur les classifieurs. A chaque itération, les occurrences de la classe qui a obtenu le moins de votes sont supprimées des données à traiter à l'itération suivante. Cette méthode s'arrête lorsqu'il n'y a plus qu'une seule classe à traiter.

3.3.6 Réseaux de neurones

En général, un réseau de neurones est fondamentalement un classifieur, il réalise un travail de classification pendant la phase d'apprentissage, et de classement lors de la reconnaissance. cependant il pourra être utilisé pour réaliser la fusion de données.

Les classifieurs dans l'ensemble sont au début appris sur la base entière ou sur des parties de cette base. Ensuite, une base de validation est classée en utilisant cet ensemble de classifieurs. La réponse à chaque forme est considérée comme une nouvelle donnée à apprendre par le réseau de neurones ce qui lui permet d'apprendre le comportement des classifieurs. Chaque forme de la base de test est alors classée au début par l'ensemble de classifieurs dont les sorties sont ensuite combinées par le réseau de neurones pour fournir la décision finale. Les sorties des classifieurs sont donc considérées pour le classifieur de fusion comme un nouvel ensemble de caractéristiques de chaque forme de test [Zouari, 2004].

Les classifieurs de réseaux de neurones fournissent une intégration efficace de différents types de données. L'approche non paramétrique qu'ils mettent en œuvre permet l'agrégation de différentes sources provenant d'un vecteur de données sans qu'il soit besoin de faire l'hypothèse d'une distribution probabiliste spécifique des données à fusionner [Arif, 2005]. Un exemple de fusion multi sources par l'approche connexionniste est décrit dans [Simone *et al.*, 2002]. Leur difficulté de mise en œuvre réside dans :

– le choix du modèle (architecture) ; a) variables d'entrée à choisir (étude de la sensibilité), si

le nombre de variables d'entrée constituant du bruit est trop grand. b) nombres de neurones sur la/les couches cachées.
- l'apprentissage, (a) choix de l'algorithme ; apprentissage par descente de gradient ; problème des minima locaux. (b) réglage des paramètres.
- Les données ; beaucoup de données, données complètes, pas de points aberrants, données bien reparties dans l'espace des entrées.
- temps de traitement, car les applications opérationnelles peuvent nécessiter des réseaux à plusieurs milliers de neurones.

3.3.7 Méthodes fixes

Le principe de base derrière les méthodes fixes est le suivant [Zouari, 2004] :
les classifieurs sont indépendants et estiment des probabilités a posteriori (adapté au classifieur de type mesure) des classes. Ainsi, pour reconnaître une forme x, on utilise une règle de décision $E(x)$ qui revient à choisir la classe C_i pour laquelle la probabilité a posteriori P_i est la plus élevée :

$$E(x) = \begin{cases} C_i & \text{si } \max_{i=1}^{N} P_i = \max_{m=1}^{N} P_m \\ \text{rejet sinon} \end{cases} \tag{3.37}$$

La probabilité a posteriori P_m peut être calculée par l'une des règles suivantes :

1. **La règle maximum**
$$P_m = \max_{j=1}^{L} m_i^j \tag{3.38}$$
2. **La règle minimum**
$$P_m = \min_{j=1}^{L} m_i^j \tag{3.39}$$
3. **La règle produit**
$$P_m = \prod_{j=1}^{L} m_i^j \tag{3.40}$$
4. **La règle linéaire**
$$P_m = \lambda \sum_{j=1}^{L} m_i^j \tag{3.41}$$

Les deux premières règles sont connues sous le nom d'*opérateurs d'ordre statique*. La règle maximum consiste à choisir la classe pour laquelle la probabilité $m_{i,j}$ est la plus élevée. Si le classifieur qui propose cette confiance a une mauvaise performance alors la règle maximum n'est pas fiable. La règle minimum consiste à choisir le classifieur qui propose la probabilité la plus faible.

La règle de produit donne de bonnes performances si les classifieurs sont indépendants. Cependant, la règle du produit est sensible aux erreurs estimées par les classifieurs (dues à la présence de bruit dans les données ou à l'utilisation de base d'apprentissage de petite taille). Il s'agit de l'effet de veto causé par l'existence d'une faible probabilité (proche de 0) produites par au moins l'un des classifieurs à combiner.
Dans la règle linéaire, λ est une constante. Ainsi, pour $\lambda = 1$, on obtient la règle de somme. Si $\lambda = \frac{1}{L}$ alors la règle est la moyenne simple.

3.3.8 Comparaison des méthodes de combinaison

Le problème du choix entre les méthodes paramétriques (en plus des sorties des classifieurs, des paramètres doivent être fournis par apprentissage) et non paramétriques (seules les sorties de classifieurs sont nécessaire) a été posé récemment [Roli *et al.*, 2002] [Duin, 2002]. D'après [Duin, 2002], la combinaison paramétrique est plus intéressante que la combinaison non paramétrique surtout lorsque la base d'apprentissage est à la fois informative et représentative. Pourtant, ce sont les méthodes de combinaison non paramétriques qui sont les plus utilisées par les chercheurs. Cela est du au fait que ces méthodes sont simples à mettre en œuvre et n'utilisent pas de traitements supplémentaires (apprentissage).

La combinaison des sorties de type classe a reçu de la part des chercheurs une attention plus grande que la combinaison des sorties de type mesure car elles sont très simples à appliquer et à analyser expérimentalement et même théoriquement. Même si les méthodes de type mesure semblent *a priori* plus séduisantes (quand on peut les utiliser) puisqu'elles exploitent toute l'information fournie par le classifieur, cela ne veut pas dire qu'elles sont toujours plus intéressantes que les méthodes de type rang [Zouari, 2004]. Par exemple [Parker, 2001] a montrée que les méthodes de type rang peuvent être plus performantes que les méthodes de type classe et mesure.

3.4 Conclusion

Dans ce chapitre nous avons essayé de comprendre la notion de la fusion d'information. Nous avons mis l'accent sur les niveau, les stratégies et techniques de fusion les plus utilisés.

Les méthodes de combinaison que nous avons présenté montrent la diversité des approches qui s'offrent au concepteur de systèmes de reconnaissance à plusieurs classifieurs. Ces méthodes se distinguent essentiellement par le niveau d'information en sortie qu'apporte chacun des classifieurs. Au niveau classe, la sortie de chaque classifieur est une étiquette. Au niveau rang, chaque classifieur fournit en sortie une liste ordonnée de solutions. Au niveau mesure, les classifieurs proposent en plus des mesures reflétant la confiance qu'ils ont dans les classes. La combinaison

est alors différente si on dispose d'un classifieur produisant des mesures (riche en information), ou seulement des classifieurs donnant des classes (pauvre en information).

Dans le prochain chapitre, nous allons exploité les synthèses présentées auparavant comme base afin de décrire notre proposition dans la problématique de fusion de classifieurs des document XML.

Chapitre 4

Contribution et validation

4.1 introduction

Afin de tirer profit de l'étude bibliographique poussée sur les méthodes d'apprentissage, la classification des document XML et les méthodes de fusion d'information, présenté dans les chapitres précédents. Nous allons consacrer ce dernier chapitre à la description de notre contribution pour répondre au problème de classification des document XML.

Notre contribution se situé en trois niveaux :

- en premier niveau, nous avons proposé une nouvelle méthode de représentation de document XML en tenant compte de leurs structures et leurs contenus.
- dans le deuxième niveau de la contribution, nous avons opté pour trois méthodes d'apprentissage automatique selon qu'elles soient adéquates à la classification de textes et qu'elles renvoient des sorties de type mesure.
- finalement, nous avons proposé une méthode de fusion. Afin de la valider nous avons fait une comparaison objective par rapport à d'autres méthodes en présentant les résultats obtenus.

4.2 Représentation des documents XML

D'après ce qu'on a vu dans le chapitre 1 et le chapitre 2, la représentation d'un document occupe un rang primordiale. Parmi les méthodes de représentation, il y en a celles qui traitent directement les arbre XML pour déterminer la similarité entre eux (sous-arbres fréquents par exemple), et d'autres qui transforment un document XML en un vecteur dont les cordonnées sont les termes extraient à partir du corpus d'apprentissage (le modèle vectoriel).

Le modèle de représentation que nous allons proposé est basé sur l'un des modèles de [de Campos *et al.*, 2007] en l'occurrence "*text replication*".

Parmi les lacunes que nous avons décelé dans ce modèle, le fait qu'il ne s'intéresse pas à la position de la balise dans l'arbre bien qu'elle véhicule une information importante qui doit être exploitée. Pour remédier à ce problème nous allons au long de cette section présenter une méthode de représentation des document XML qui utilise en plus des valeurs désignant l'importance de la balise dans le corpus, la profondeur dans l'arbre XML comme information supplémentaire.

4.2.1 Principe

Le principe de notre modèle consiste à transformer un document XML en un texte plat pour qu'on puisse par la suite en appliquer facilement le modèle vectoriel ou sac-à-mots (*chapitre*1.3.7).

Le modèle vectoriel qui est présenté en détail dans le chapitre 1 est un modèle simple, adéquat pour les documents textuelles et donne souvent des résultats satisfaisants par rapport à d'autres modèles plus sophistiqués [Sebastiani, 2002].

Cette méthode de représentation prend en compte la connaissance *a priori* sur la structure d'un document XML, c-à-d qu'elle assigne une valeur entier pour chaque balise textuelle (son contenu est un texte) proportionnelle à son importance dans le document (plus la valeur est supérieur, plus la balise est importante) et sa profondeur (niveau) dans l'arbre XML (plus le niveau est supérieur - proche de la racine -, plus la balise est importante). Par la suite le texte sera répliqué en fonction de la valeur et la profondeur de la balise.

Soit un document XML tiré du corpus INEX 2007 (figure 4.1). L'extraction du contenu textuel à partir de ce document donne le texte suivant :

« The Green Futures of Tycho 0 The Green Futures of Tycho is a 1981 science fiction novel for young audience by William Sleator. The book explores the effect of excessive parent expectations on the future of their children. Plot The main character is,Tycho an 11 yearold boy. Each child in his family is named after a famous artist or scientist and their parent's expect them to life up to their names. Tycho himself is named after Tycho Brahe, the Danish astronomer (the author also has a brother named Tycho). He who finds a pocket sized time machinein the family's garden. He immediately uses it to change some things from the past and to visit the future. But as he travels more and more he realizes that he is turning into something horrible and it becomes a race against time to save all of his family and himself. »

Pour comprendre le principe de la méthode de représentation, nous avons pris comme exemple la combinaison des valeurs de réplication suivante :

```
name               3
title              2
conversionwarning  0
emph5              0
```

```
<?xml version="1.0" encoding="utf-8"?>
<article>
<name id="1495676">The Green Futures of Tycho</name>
<conversionwarning>0</conversionwarning>
<body>
 <emph5>The Green Futures of Tycho</emph5>is a 1981
 <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="26787.xml">science fiction</collectionlink>
 <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="21882.xml">novel</collectionlink>for young audience by
 <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="386786.xml">William  Sleator</ collectionlink>. The book explores the effect of
excessive parent expectations on the future of their children.
 <section>
  <title>Plot</title>
  <templateSpoiler NAME="Spoiler"> </templateSpoiler>
  <p>The main character is, Tycho an 11 year-old boy. Each child in his family is named after a
famous artist or scientist and their parents expect them to life up to their names. Tycho
    himself is
named after
    <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="30027.xml">Tycho Brahe</collectionlink>, the
    <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="76972.xml">Danish</collectionlink>
    <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="580.xml">astronomer</collectionlink>
                (the author also has a brother named Tycho). He who finds a pocket sized
    <collectionlink xmlns
        :xlink="http://www.w3.org/1999/xlink" xlink:type="simple" xlink:href="31591.xml">time
machine</collectionlink> in the family s garden. He immediately uses it to change some things
    from
the past and to visit the future. But as he travels more and more he realizes that he is
    turning
into something horrible and it becomes a race against time to save all of his family and
    himself.
  </p>
  <templateSf_book_stub NAME="Sf-book-stub"> </templateSf_book_stub>
</section> </body>
</article>
```

Figure 4.1 – *Document 1495676.xml du corpus INEX 2007 .*

Notons que le reste des balises vont prendre la valeur 1.

L'arbre de ce document comprend 4 niveau, par conséquent la balise `name` par exemple sera d'une profondeur de 4 et la balise `title` d'une profondeur de 2. Par contre la balise `collectionlink` se produit à plusieurs niveau de l'arbre et donc ces niveau ne seront pas traité d'une façon similaire, à chaque fois son contenu textuel sera répliqué en fonction de sa profondeur multiplié à sa valeur d'importance.

La réplication de texte du document de la figure 4.1 donne le résultat suivant :

«The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of T des document XMLycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho The Green Futures of Tycho is a 1981 is a 1981 is a 1981 is a 1981 science fiction science fiction science fiction novel novel novel for young audience by for young audience by for young audience by for young audience by

William Sleator William Sleator William Sleator. The book explores the effect of excessive parent expectations on the future of their children. . The book explores the effect of excessive parent expectations on the future of their children. . The book explores the effect of excessive parent expectations on the future of their children. . The book explores the effect of excessive parent expectations on the future of their children. Plot Plot Plot Plot The main character is, Tycho an 11 year-old boy. Each child in his family is named after a famous artist or scientist and their parent's expect them to life up to their names. Tycho himself is named after The main character is, Tycho an 11 year-old boy. Each child in his family is named after a famous artist or scientist and their parent's expect them to life up to their names. Tycho himself is named after Tycho Brahe , the , the Danish astronomer (the author also has a brother named Tycho). He who finds a pocket sized (the author also has a brother named Tycho). He who finds a pocket sized time machine in the family's garden. He immediately uses it to change some things from the past and to visit the future. But as he travels more and more he realizes that he is turning into something horrible and it becomes a race against time to save all of his family and himself. in the family's garden. He immediately uses it to change some things from the past and to visit the future. But as he travels more and more he realizes that he is turning into something horrible and it becomes a race against time to save all of his family and himself.
»

L'algorithme 3 qui suit décrit la méthode de représentation des documents XML proposée, qui permet de transformer l'ensemble des exemples XML en générant un ensemble des exemples textuels :

Algorithme 3: *Algorithme proposé de réplication de texte d'un document XML*

Données : un échantillon de l documents XML classés en $C = \{c_1, c_2, \cdots, c_n\}$
Entrées : Assigné un valeur v pour chaque balise b désignant son importance
pour *chaque document* ***d*** **faire**
  pour *chaque balise* ***b*** *du document* ***d*** **faire**
    si *la balise courante est textuelle* **alors**
      déterminer la hauteur h de la balise b;
      dupliquer le contenu textuel de la balise $v \times h$ fois;
      `// la valeur` v `peut être nulle`
    fin
  fin
fin
Résultat : un échantillon de l documents XML transformer en texte classés en $C = \{c_1, c_2, \cdots, c_n\}$

	Total	Apprentissage	Test
Nombre de documents	96,611	48,306	48,305
Nombre de nœuds interne	≈ 9 M	4,505,141	4,487,819
Nombre de mots distincts	446,916 (dépend de prétraitement)		
Nombre de mots	33,944,462	17,261,996	16,682,466
La longueur moyenne des documents	351.4	357.3	345.5
Nombre de balises distincts	≈ 5,800		
La taille de corpus	≈ 720 Mbytes	≈ 360 Mbytes	≈ 360 Mbytes

Table 4.1 – *Statistiques du corpus.*

4.3 Le corpus INEX 2007

Le corpus INEX 2007 est un corpus qui est utilisé dans les deux tâches de classification et clustering. Il est composé d'environ 96 000 document XML extraits à partir du corpus Wikipédia XML et s'est divisé en deux parties [Denoyer et Gallinari, 2008] :

- la partie d'apprentissage qui est composée de 50% des documents
- la partie de test qui est composée du reste des documents.

Id	Catégorie	Taille
2112299	portail :Loi	24213
1597184	portail :littérature	16929
1484914	portail :Sports et jeux	14595
1620218	portail :Politique	1355
1480358	portail :Art	7624
1886386	portail :physique	5149
3091788	portail :Christianisme	4671
2773006	portail :Chimie	4567
1685758	portail :Histoire	3246
3091127	portail :Spiritualité	2704
2914908	portail :Sexualité	2402
2879927	portail :Guerre	2217
1507239	portail :Aviation	1217
2328885	portail :Formula One	1188
1486363	portail :Astronomie	1105
1895383	portail :Trains	953
2314377	portail :Université	605
2635947	portail :Comics	600
2257163	portail :Pornographie	458
2263642	portail :Writing	412
1474166	portail :Musique	401

Table 4.2 – *Description des différentes catégories.*

Le corpus est téléchargeable du site web de XML Mining [1].

Les documents sont organisés dans 21 catégories qui correspondent à différents portails de Wikipédia et correspondent fondamentalement aux catégories thématiques. Le corpus a été construit d'une façon d'être assez grand pour des applications réelles et assez petit pour permettre à l'évaluation des nouveau système sans perdre du temps pour développer des logiciels complexes.

Les tables 4.1 et4.2 donnent quelques statistiques au sujet du corpus complet.

Notons que les catégories ne sont pas bien équilibrées : quelques catégories sont grandes comme par exemple le *portail :loi* qui se compose d'environ 25% des documents, tandis que d'autres portails ont une très petite taille. *portail :musique* par exemple se compose d'environ 0.5% des documents. Ce corpus est intéressent parce que quelques modèles tendent à apprendre seulement sur de grandes catégories et le corpus aidera à mesurer la capacité des modèles de traiter de petites classes. D'ailleurs, le corpus a été construit afin de proposer certaines catégories ambiguës comme par exemple le *portail :Christianisé* et le *portail :Spiritualité*.

4.4 La classification

Le processus de classification est composé de trois étapes principales que nous allons détailler dans les sections suivantes :

4.4.1 Prétraitement

Pour manipuler les documents XML, nous avons opter pour l'API JDOM [2] qui est une API du langage Java développée indépendamment de Sun Microsystems, elle permet de manipuler des donnés XML plus simplement qu'avec les API classiques qui ne sont pas spécifique à Java. Son utilisation est pratique pour tout développeur Java et repose sur les API XML DOM et SAX vus en chapitre 2 [CYNOBER, 2005].

En se basant sur ces deux API, *JDom* permet donc de construire des documents XML, de naviguer dans leur structure, d'ajouter, de modifier, ou de supprimer leur contenu.

Après avoir appliquer le parser présenté par l'algorithme 3, en utilisant la combinaison suivante :

1. http://xmlmining.lip6.fr
2. http://www.jdom.org/dist/binary/

name	10
title	7
caption	6
collectionlink	4
unknownlink	4
wikipedialink	4
weblink	4
emph2	3
emph3	3
emph4	3
emph5	3
conversionwarning	0
languagelink	0

Nous allons obtenir un ensemble de textes préclassifiers, et donc la tâche revient à une tâche de text mining.

Pour permettre la manipulation de cette ensemble de textes par Weka, celles ci doivent être réparties sur un ensemble de sous-répertoires, où chaque sous-répertoire représente la classes et son contenu (l'ensemble de textes assignées à cette classe).

```
 ...
|
+- textes_examples
   |
   +- classe1
   |  |
   |  + fichier1.txt
   |  |
   |  + fichier2.txt
   |  |
   |  ...
   |
   +- class2
   |  |
   |  + un_autre_fichier1.txt
   |  |
   |  + un_autre_fichier2.txt
   |  |
   |  ...
```

Notons que les valeur assignées aux différentes balises que nous avons proposé sont déterminées après une étude du corpus pour sélectionner les plus importantes entre eux.

Afin de manipuler l'ensemble de textes générés et réaliser une transformation $tf \times idf$ (représentation vectoriel), nous avons utilisé le logiciel libre de Data mining Weka écrit en java [3] qui :

3. http://www.cs.waikato.ac.nz/ml/weka/

– fournit des algorithmes de data mining et d'apprentissage automatique
– permet le prétraitement et la visualisation des données

Parmi ses fonctionnalité, Weka fournit La classe *StringToWordVector* permettant la transformation $tf \times idf$.

Pour réduire le nombre de termes générés, une étape de filtrage sur le texte est nécessaire. Dans cette étape, nous utilisons les méthodes classiques de réduction du dimension, à savoir :

– Définir une liste de mots vides (*stopword*), qui est un ensemble de termes qui n'ont aucun rôle à jouer dans la classification et donc ne doivent pas être prisent en compte. Par exemple les pronoms, les auxiliaires, ... etc.
– Radicaliser (*stemming*) les termes du texte, ceci consiste à les remplacer par leur racine. Pour cela, nous avons utilisé l'outil *SnowballStemmer* qui est très connu dans le domaine de recherche d'information (IR). Cet algorithme supprime les suffixes des mots en anglais. La suppression des suffixes permet de réduire la taille et la complexité des données pour améliorer les performances. Le principale avantage de cet algorithme est sa rapidité de traitement de grand volume de données, mais il présente tout de même quelques limitations. En effet, le Stemmer transforme un terme en une racine qui peut ne pas être sa racine linguistique. Par exemple, le mot `References` est normalement transformé en `Refer`, par contre, le mot `titles` sera quant à lui transformé en `titl` (sans e), qui n'est pas vraiment la racine de ce mot.

4.4.2 L'application des algorithmes

Notre objectif est de choisir des méthodes de classification qui sont adéquates pour la catégorisation de textes et qui permettre en outre de générés des sortie de type mesure (chapitre 3). Dans notre cas les sortie sont des vecteurs désignant les probabilités d'appartenance à chaque classe.

$$e(x) = (p_{c_1}, p_{c_2}, \cdots, p_{c_{21}}) \tag{4.1}$$

L'avantage de ce type de sortie est qu'il peut être facilement se transformer en d'autre types. Néanmoins, il y a peu d'algorithmes qui génèrent des sorties de type mesure.

4.4.2.1 SVM

Nous avons choisi cette méthode pour deux raisons [Joachims, 1998] :

– c'est une méthode robuste en catégorisation de textes, et surpasse en performance les autres méthodes existantes.
– capable de traiter une grande dimension, sans avoir besoin d'une étape de sélection de termes.

LibSVM[4] est une bibliothèque de SVM. Sont but est de fournir un outil qui facilite l'utilisation de la méthode SVM.
L'outil *LibSVM* présente plusieurs avantages, parmi eux on peut cité :

- la génération des sorties probabiliste.
- il fournit plusieurs méthodes et plusieurs noyaux de SVM.
- la catégorisation multiclasse (à l'origine SVM est un classifieur binaire).

Pour plus de détail sur cet outil consulter [Chang et Lin, 2001]. et pour des détails sur l'intégration de l'outil *LibSVM* dans Weka consulter [EL-Manzalawy, 2005].

4.4.2.2 Naïve bayes

Il s'agit ici d' :

- un classifieur simple et robuste
- une méthode dont performances pour la tâche de classification font aujourd'hui un modèle de référence notamment en classification de textes.
- une méthode peu réaliste, due à l'hypothèse d'indépendance.

Malgré que le modèle Naïve bayes semble peu réaliste, il donne de bons résultats pour la classification des documents textuels [Joachims, 1998].
Pour construire un classifieur Naïve bayes, nous avons opter pour le modèle multinomional qui donne souvent des bon résultats et peut traiter des grosse volume de collections textuelles [Mccallum et Nigam, 1998]

4.4.2.3 KNN

Le KNN est un méthode qui ne nécessite pas d'apprentissage, mais tous les calculs doivent être effectués lors de la classification. La conséquence directe de cette caractéristique est l'importance d'avoir un espace mémoire conséquent pour stocker l'échantillon.
Les performances de la méthode dépendent du deux facteurs :

- choix de la distance
- le nombre de voisins.

4.4.3 L'évaluation

Afin de valider notre méthode de représentation, nous allons appliquer en premier temps les trois algorithmes d'apprentissage précédents sur le contenu seul des document XML, ils vont servir de référence pour juger de la qualité de notre modèle en prenant en compte le contenu et la structure conformément à l'algorithme 3. Nous avons obtenu les résultats suivants :

4. http ://www.csie.ntu.edu.tw/c̃jlin/libsvm/

4.4.3.1 *SVM* sur le contenu seul

```
Correctly Classified Instances       38414               79.5239 %
Incorrectly Classified Instances      9891               20.4761 %
Kappa statistic                          0.7619
Mean absolute error                      0.0279
Root mean squared error                  0.1197
Relative absolute error                 33.764  %
Root relative squared error             58.9225 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.731     0.002      0.656     0.731     0.692      0.992    1474166
                 0.622     0.02       0.733     0.622     0.673      0.95     1480358
                 0.91      0.012      0.931     0.91      0.92       0.989    1484914
                 0.901     0.002      0.874     0.901     0.887      0.998    1486363
                 0.823     0.003      0.799     0.823     0.811      0.997    1507239
                 0.839     0.04       0.818     0.839     0.828      0.971    1597184
                 0.777     0.003      0.797     0.777     0.787      0.986    1620218
                 0.528     0.007      0.721     0.528     0.609      0.943    1685758
                 0.637     0.012      0.763     0.637     0.694      0.947    1886386
                 0.779     0.001      0.863     0.779     0.819      0.997    1895383
                 0.831     0.1        0.735     0.831     0.78       0.938    2112299
                 0.942     0.001      0.887     0.942     0.913      1        2257163
                 0.813     0.001      0.77      0.813     0.791      0.983    2263642
                 0.578     0.004      0.483     0.578     0.526      0.987    2314377
                 0.958     0.001      0.958     0.958     0.958      1        2328885
                 0.602     0.002      0.656     0.602     0.628      0.995    2635947
                 0.853     0.005      0.888     0.853     0.87       0.991    2773006
                 0.798     0.006      0.75      0.798     0.773      0.987    2879927
                 0.573     0.007      0.684     0.573     0.623      0.947    2914908
                 0.699     0.009      0.691     0.699     0.695      0.977    3091127
                 0.833     0.007      0.848     0.833     0.84       0.986    3091788
Weighted Avg.    0.795     0.038      0.796     0.795     0.793      0.965

=== Confusion Matrix ===

     a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r     s     t     u   <-- classified as
   147     3     1     0     0     3     0     0     0     0    43     0     2     0     0     0     0     0     1     0     1 |     a = 1474166
    13  2414    77     3     2   483    18    54    62     5   580     5    10     7     3     9     4    33    38    31    33 |     b = 1480358
     2    32  6613     2     6    55    10    14    30     6   429     0     1    14     1     1     0    11    17     8    15 |     c = 1484914
     0     0     0   500     4     3     0     4    11     0    32     0     1     0     0     0     0     0     0     0     0 |     d = 1486363
     0     0     0     1   508     1     1     0     5     0    95     0     0     0     0     0     0     6     0     0     0 |     e = 1507239
    12   346    27     0    12  7064    12    30    49     1   525     4     9    16     0    16     5    52    82    87    69 |     f = 1597184
     5     3    12     1     5     8   513     7     2     1    46     0     6     4     3     0     1    11     1    25     6 |     g = 1620218
     5    43    30    12     3    94    16   838    39     4   401     2     5     8     0     0     1    44    16     7    20 |     h = 1685758
     2    29    10    27    15   149     5    25  1694     2   414     0     5    19     0     5   115     4    23   111     5 |     i = 1886386
     0     0     0     0     0     0     0     1     0   377   106     0     0     0     0     0     0     0     0     0     0 |     j = 1895383
    25   323   257    11    40   374    28   163   214    41 10062     8     8    71    14    52   103    83    92    59    77 |     k = 2112299
     0     2     1     0     0     6     0     0     0     0     4   227     0     0     0     0     0     0     1     0     0 |     l = 2257163
     0     0     0     0     0    11     0     1     2     0    25     0   187     2     0     0     0     0     0     1     1 |     m = 2263642
     0     1     3     1     0    20     0     0     2     0    83     0     1   159     0     0     1     0     0     0     4 |     n = 2314377
     0     0     0     0     0     0     0     0     0     0    25     0     0     0   566     0     0     0     0     0     0 |     o = 2328885
     0     3     1     0     0    90     0     0     0     0    27     0     0     0     0   183     0     0     0     0     0 |     p = 2635947
     0     6     1     2     1     8     0     4    72     0   224     0     1    13     2     0  1973     0     4     1     2 |     q = 2773006
     1    11     1     1    36    39     5    13     1     0    99     0     0     0     1     0     0   902    11     1     8 |     r = 2879927
    11    45    61     2     1   117    10     4     4     0   165    10     0     2     0     9    16    30   698    20    14 |     s = 2914908
     0    14     4     6     3    67    17     3    29     0   139     0     2     8     0     4     3     5    18   929    78 |     t = 3091127
     1    17     4     3     0    48     9     1     4     0   170     0     5     6     1     0     0    22    18    65  1860 |     u = 3091788
```

4.4.3.2 *SVM* sur le contenu et la structure

```
Correctly Classified Instances       38518               79.7392 %
Incorrectly Classified Instances      9787               20.2608 %
Kappa statistic                          0.7644
Mean absolute error                      0.0277
Root mean squared error                  0.1192
Relative absolute error                 33.6042 %
Root relative squared error             58.6785 %
Total Number of Instances            48305

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

```
                 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                   0.726     0.002      0.646     0.726     0.684      0.992    1474166
                   0.632     0.02       0.736     0.632     0.68       0.952    1480358
                   0.911     0.012      0.931     0.911     0.921      0.989    1484914
                   0.912     0.001      0.877     0.912     0.894      0.999    1486363
                   0.822     0.003      0.803     0.822     0.813      0.997    1507239
                   0.84      0.037      0.826     0.84      0.833      0.972    1597184
                   0.789     0.003      0.804     0.789     0.797      0.986    1620218
                   0.538     0.007      0.723     0.538     0.617      0.945    1685758
                   0.637     0.011      0.765     0.637     0.695      0.946    1886386
                   0.777     0.001      0.866     0.777     0.819      0.997    1895383
                   0.834     0.1        0.737     0.834     0.782      0.938    2112299
                   0.938     0.001      0.886     0.938     0.911      1        2257163
                   0.813     0.001      0.763     0.813     0.787      0.986    2263642
                   0.582     0.004      0.479     0.582     0.525      0.987    2314377
                   0.954     0.001      0.954     0.954     0.954      1        2328885
                   0.622     0.002      0.652     0.622     0.636      0.995    2635947
                   0.847     0.005      0.888     0.847     0.867      0.991    2773006
                   0.798     0.007      0.744     0.798     0.77       0.986    2879927
                   0.576     0.007      0.688     0.576     0.627      0.947    2914908
                   0.696     0.009      0.689     0.696     0.693      0.976    3091127
                   0.833     0.007      0.851     0.833     0.842      0.985    3091788
Weighted Avg.      0.797     0.037      0.798     0.797     0.796      0.965

=== Confusion Matrix ===

     a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r     s     t     u   <-- classified as
   146     4     1     0     0     3     0     0     0     0    43     0     2     0     0     0     0     0     1     0     1 |     a = 1474166
    14  2454    79     4     2   431    13    50    56     5   592     5    10     7     3    13     4    34    43    33    32 |     b = 1480358
     2    34  6619     2     6    59    10    21    31     5   406     0     1    16     2     1     1    12    17     9    13 |     c = 1484914
     0     0     0   506     4     3     0     3    11     0    27     0     1     0     0     0     0     0     0     0     0 |     d = 1486363
     0     0     0     0   507     1     0     0     4     0    97     0     0     0     0     0     0     8     0     0     0 |     e = 1507239
    12   356    28     0    13  7071    11    31    54     1   506     3     9    17     0    16     4    53    81    86    66 |     f = 1597184
     5     3    13     0     5     8   521     7     3     1    45     0     5     3     3     0     0     7     1    25     5 |     g = 1620218
     5    44    30    11     2    89    16   854    39     4   381     2     8     8     0     0     0    49    16     7    23 |     h = 1685758
     3    29     9    28    15   140     5    27  1693     2   419     0     5    22     0     5   118     4    20   108     7 |     i = 1886386
     0     0     0     0     0     0     0     0     0   376   108     0     0     0     0     0     0     0     0     0     0 |     j = 1895383
    25   316   252    12    37   364    29   158   207    40 10093     8     8    72    15    53   101    86    87    62    80 |     k = 2112299
     0     1     1     0     0     6     0     0     0     0     6   226     0     0     0     0     0     0     1     0     0 |     l = 2257163
     0     0     0     0     0    14     0     1     2     0    23     0   187     1     0     0     0     0     0     1     1 |     m = 2263642
     0     2     2     1     0    21     0     0     2     0    79     0     1   160     0     0     3     0     0     0     4 |     n = 2314377
     0     0     0     0     0     0     0     0     0     0    27     0     0     0   564     0     0     0     0     0     0 |     o = 2328885
     0     2     2     0     0    87     0     0     0     0    24     0     0     0     0   189     0     0     0     0     0 |     p = 2635947
     0     5     1     2     1     7     0     4    75     0   234     0     1    13     2     0  1961     0     4     1     3 |     q = 2773006
     1     8     1     1    35    33     4    17     1     0   107     0     0     0     1     0     0   902    10     1     8 |     r = 2879927
    12    46    62     2     1   112    11     3     4     0   166    11     0     3     0     9    14    30   702    19    12 |     s = 2914908
     0    15     3     5     3    67    17     4    28     0   149     0     2     7     0     4     2     5    21   925    72 |     t = 3091127
     1    17     4     3     0    46    11     1     4     0   170     0     5     5     1     0     0    23    16    65  1862 |     u = 3091788
```

4.4.3.3 *Bayes Multinominal* sur le contenu seul

```
Correctly Classified Instances       32543               67.3698 %
Incorrectly Classified Instances     15762               32.6302 %
Kappa statistic                          0.6361
Mean absolute error                      0.0312
Root mean squared error                  0.1738
Relative absolute error                 37.7703 %
Root relative squared error             85.5431 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                   0.886     0.016      0.188     0.886     0.31       0.974    1474166
                   0.586     0.056      0.478     0.586     0.527      0.866    1480358
                   0.866     0.026      0.853     0.866     0.859      0.957    1484914
                   0.84      0.001      0.917     0.84      0.877      0.964    1486363
                   0.925     0.009      0.569     0.925     0.705      0.984    1507239
                   0.638     0.03       0.816     0.638     0.716      0.927    1597184
                   0.768     0.009      0.546     0.768     0.639      0.952    1620218
                   0.554     0.021      0.472     0.554     0.51       0.87     1685758
                   0.673     0.03       0.569     0.673     0.617      0.904    1886386
                   0.942     0.013      0.421     0.942     0.582      0.985    1895383
                   0.469     0.029      0.845     0.469     0.603      0.849    2112299
                   0.967     0.004      0.566     0.967     0.714      0.995    2257163
```

```
                 0.922     0.004      0.546     0.922     0.686      0.99     2263642
                 0.804     0.02       0.189     0.804     0.306      0.968    2314377
                 0.927     0.001      0.927     0.927     0.927      0.986    2328885
                 0.895     0.012      0.313     0.895     0.463      0.987    2635947
                 0.857     0.007      0.864     0.857     0.86       0.975    2773006
                 0.827     0.013      0.604     0.827     0.698      0.949    2879927
                 0.571     0.013      0.531     0.571     0.55       0.882    2914908
                 0.797     0.024      0.479     0.797     0.599      0.951    3091127
                 0.861     0.015      0.74      0.861     0.796      0.965    3091788
Weighted Avg.    0.674     0.026      0.734     0.674     0.681      0.912

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r    s    t    u   <-- classified as
  178    9    1    0    0    0    0    0    0    0    5    0    4    0    0    0    0    1    3    0    0 |    a = 1474166
   95 2277  117    4    8  311   18  190  156   16  197   40   22   49    1   85    4   49   74  100   71 |    b = 1480358
   17   86 6290    4   18  140   33   79  115   34  245    6   16   45    2   19    2   24   25   49   18 |    c = 1484914
    1    2    3  466   12    2    0    9   40    7    4    0    0    4    0    1    0    2    1    1    0 |    d = 1486363
    1    2    0    2  571    0    0    1   13    0   18    0    0    0    0    0    1    7    1    0    0 |    e = 1507239
  213 1086   31    1   61 5370   23   60   81   11  105   75   60   96    1  236    2  159  200  401  146 |    f = 1597184
    4    8   30    0    4    4  507    6    4    1    6    0   10    5    2    0    2   12    7   36   12 |    g = 1620218
    7   96   29    5   12   34   97  880   60   21  110    8   10   30    0    5    2   79   29   35   39 |    h = 1685758
   24   85   14    8   24  148    5   64 1789   11   95    5   19   50    0   13  122    2   13  156   12 |    i = 1886386
    0    0    0    0    2    1    1    0    0  456   22    0    0    0    0    0    0    0    1    0    1 |    j = 1895383
  356  913  779   11  248  386  139  528  681  512 5678   17   20  604   35  221  149  213  200  189  226 |    k = 2112299
    0    4    0    0    0    0    0    0    0    0    1  233    0    0    0    0    0    0    3    0    0 |    l = 2257163
    0    2    0    0    1    3    1    1    3    0    0    0  212    3    0    0    0    0    0    3    1 |    m = 2263642
    0    0    1    0    0    5    1    4    9    0   14    0    1  221    0    0    3    0    2    4   10 |    n = 2314377
    0    0    2    0    2    0    0    0    2    0   37    0    0    0  548    0    0    0    0    0    0 |    o = 2328885
    0    1    0    0    0   27    0    0    0    0    3    0    0    0    0  272    0    0    0    1    0 |    p = 2635947
    1    9    0    5    2    5    1   20  138    6   81    0    7   26    0    0 1982    6   13   10    2 |    q = 2773006
    2   13    6    0   37   21   32   12    2    4   24    0    0    0    1    0    0  935   11    3   27 |    r = 2879927
   45  114   63    1    0   54   32    0    7    0   17   24    0   13    0   11   21   37  696   40   44 |    s = 2914908
    2   16    2    1    2   49   23    5   37    1   17    1    5   15    0    7    4    3   13 1059   67 |    t = 3091127
    3   36    4    0    0   22   15    4    5    4   44    3    2    8    1    0    0   20   18  122 1923 |    u = 3091788
```

4.4.3.4 *Bayes Multinominal* sur le contenu et la structure

```
Correctly Classified Instances       33186               68.701  %
Incorrectly Classified Instances     15119               31.299  %
Kappa statistic                          0.6503
Mean absolute error                      0.0299
Root mean squared error                  0.1702
Relative absolute error                 36.2026 %
Root relative squared error             83.7728 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.905     0.013      0.229     0.905     0.365      0.978    1474166
                 0.607     0.055      0.491     0.607     0.543      0.876    1480358
                 0.867     0.026      0.856     0.867     0.861      0.959    1484914
                 0.877     0.001      0.905     0.877     0.891      0.976    1486363
                 0.925     0.009      0.577     0.925     0.711      0.985    1507239
                 0.668     0.03       0.827     0.668     0.739      0.934    1597184
                 0.797     0.008      0.584     0.797     0.674      0.957    1620218
                 0.563     0.02       0.485     0.563     0.521      0.875    1685758
                 0.676     0.029      0.576     0.676     0.622      0.906    1886386
                 0.942     0.012      0.449     0.942     0.608      0.986    1895383
                 0.486     0.029      0.847     0.486     0.617      0.858    2112299
                 0.971     0.003      0.589     0.971     0.734      0.996    2257163
                 0.913     0.004      0.544     0.913     0.682      0.994    2263642
                 0.785     0.018      0.198     0.785     0.316      0.97     2314377
                 0.932     0.001      0.945     0.932     0.939      0.987    2328885
                 0.898     0.011      0.335     0.898     0.488      0.987    2635947
                 0.857     0.007      0.862     0.857     0.859      0.976    2773006
                 0.842     0.014      0.598     0.842     0.699      0.95     2879927
                 0.576     0.013      0.526     0.576     0.55       0.889    2914908
                 0.807     0.022      0.504     0.807     0.62       0.954    3091127
                 0.862     0.014      0.747     0.862     0.8        0.966    3091788
Weighted Avg.    0.687     0.026      0.741     0.687     0.693      0.918

=== Confusion Matrix ===
```

```
    a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r    s    t    u   <-- classified as
  182    5    1    0    0    0    1    0    0    0    1    0    4    0    0    0    0    1    6    0    0 |    a = 1474166
   67 2356  124    2    5  268   27  191  154   12  196   40   21   43    1   80    4   63   73   91   66 |    b = 1480358
   10   75 6299    4   16  137   34   75  112   34  264    4   22   47    0   17    2   25   29   45   16 |    c = 1484914
    0    0    0  487   10    1    0    7   34    5    2    0    1    2    0    1    0    4    1    0    0 |    d = 1486363
    1    2    0    3  571    0    2    1   14    0   15    0    0    0    0    0    1    7    0    0    0 |    e = 1507239
  155 1050   24    2   57 5620   24   52   77    7  106   59   59   89    0  204    0  139  204  341  149 |    f = 1597184
    4    7   27    0    4    3  526    7    5    1    5    0    7    4    2    0    1   10    3   35    9 |    g = 1620218
    8   94   28    9   11   38   86  894   63   21  111   10    9   24    0    1    2   88   21   33   37 |    h = 1685758
   22   77   13   12   28  148    5   58 1797   10   93    4   16   45    0   11  123    2   26  159   10 |    i = 1886386
    0    0    0    0    1    1    1    0    0  456   23    0    0    0    0    0    0    0    1    0    1 |    j = 1895383
  304  931  762   12  243  394  115  516  663  453 5881   16   23  564   27  211  157  229  205  174  225 |    k = 2112299
    0    2    0    0    0    1    0    0    0    0    1  234    0    0    0    0    0    0    3    0    0 |    l = 2257163
    0    1    0    0    0    5    1    2    3    0    0    0  210    3    0    0    0    0    1    3    1 |    m = 2263642
    0    0    1    0    0    7    2    4    7    0   18    0    0  216    0    0    6    0    2    4    8 |    n = 2314377
    0    0    1    0    3    0    0    0    2    0   34    0    0    0  551    0    0    0    0    0    0 |    o = 2328885
    0    1    1    0    0   25    0    0    0    0    3    0    0    0    0  273    0    0    0    1    0 |    p = 2635947
    0   10    2    5    2    8    1   17  139    6   81    0    7   23    0    0 1983    6   12    9    3 |    q = 2773006
    1   14    6    1   36   18   22   10    2    5   27    0    0    0    1    0    0  951   14    1   21 |    r = 2879927
   39  115   64    1    0   55   26    3    8    0   20   26    0   14    0   10   18   40  702   40   38 |    s = 2914908
    1   15    1    0    2   48   15    4   36    1   20    1    4   12    0    7    4    3   15 1072   68 |    t = 3091127
    2   40    5    0    0   21   13    3    6    4   43    3    3    6    1    0    0   23   16  120 1925 |    u = 3091788
```

4.4.3.5 *KNN* sur le contenu seul

```
Correctly Classified Instances       28051               58.0706 %
Incorrectly Classified Instances     20254               41.9294 %
Kappa statistic                          0.5124
Mean absolute error                      0.0425
Root mean squared error                  0.1851
Relative absolute error                 51.483  %
Root relative squared error             91.1032 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.607     0          0.865     0.607     0.713      0.786    1474166
                 0.391     0.058      0.37      0.391     0.38       0.661    1480358
                 0.795     0.052      0.729     0.795     0.76       0.889    1484914
                 0.787     0.001      0.881     0.787     0.832      0.895    1486363
                 0.438     0.001      0.844     0.438     0.576      0.703    1507239
                 0.599     0.065      0.661     0.599     0.628      0.783    1597184
                 0.324     0.003      0.568     0.324     0.413      0.642    1620218
                 0.375     0.013      0.503     0.375     0.43       0.687    1685758
                 0.369     0.042      0.336     0.369     0.352      0.673    1886386
                 0.535     0.001      0.878     0.535     0.665      0.78     1895383
                 0.653     0.155      0.585     0.653     0.617      0.798    2112299
                 0.515     0          0.919     0.515     0.66       0.823    2257163
                 0.543     0          0.868     0.543     0.668      0.796    2263642
                 0.171     0.001      0.456     0.171     0.249      0.585    2314377
                 0.878     0          0.977     0.878     0.925      0.943    2328885
                 0.224     0.002      0.436     0.224     0.296      0.61     2635947
                 0.63      0.018      0.638     0.63      0.634      0.858    2773006
                 0.519     0.005      0.726     0.519     0.605      0.809    2879927
                 0.235     0.008      0.446     0.235     0.308      0.6      2914908
                 0.528     0.049      0.233     0.528     0.324      0.768    3091127
                 0.454     0.013      0.634     0.454     0.529      0.756    3091788
Weighted Avg.    0.581     0.069      0.598     0.581     0.582      0.78

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r    s    t    u   <-- classified as
  122   10    8    0    0    7    0    2    8    0   32    0    0    0    0    0    3    0    1    5    3 |    a = 1474166
    2 1519  237    7    0  718   10   67  178    0  766    3    2    1    0    3   53   18   31  203   66 |    b = 1480358
    2  235 5776    2    2  151    6   46  124    0  651    0    1    8    0    1   46    5   41  138   32 |    c = 1484914
    0   14    5  437    1    4    3    6   17    0   52    0    0    0    0    0    4    0    1   11    0 |    d = 1486363
    0   27   30    0  270   13    0    7   38    0  157    0    1    1    0    0   28    8    0   34    3 |    e = 1507239
    4  622  363   11    0 5040   16   73  380    5 1058    0    5    9    0   69   88   78  102  402   93 |    f = 1597184
    1   45   55    1    0   48  214   35   51    3  135    0    1    1    1    0    8    5   10   33   13 |    g = 1620218
    1  123   82    2    4  106   18  596   92    2  406    0    1    1    0    0   25   16   16   69   28 |    h = 1685758
    2  197  164   11    3  160    4   55  981    0  574    0    2    6    0    2  144    7   16  316   15 |    i = 1886386
    0   33   16    0    0   20    3    2   17  259  108    0    0    0    0    0    7    0    1    9    9 |    j = 1895383
    7  731  738   17   16  662   36  164  602   23 7906    2    0   21   11    9  228   41   53  652  186 |    k = 2112299
```

```
    0   35   10    0    0   27    0    3    6    0   26  124    0    0    0    0    1    0    0    3    6 |    l = 2257163
    0   10   13    0    0   26    1    2    5    0   31    0  125    1    0    0    2    2    3    8    1 |    m = 2263642
    0   26   27    0    0   28    1    3   11    0  100    0    0   47    0    0    8    0    0   11   13 |    n = 2314377
    0    4    7    0    0    4    0    2    9    0   37    0    0    0  519    0    3    2    1    3    0 |    o = 2328885
    0   27   14    0    0   62    2    3   20    0   72    0    2    0    0   68   13    3    7   10    1 |    p = 2635947
    0   72   69    2    0   36    2   18  105    1  365    0    1    1    0    0 1457    1    4  168   12 |    q = 2773006
    0   52   26    1   24   57   10   34   27    1  171    0    0    1    0    0   95  587   11   14   19 |    r = 2879927
    0  122   94    2    0  143   12   34   65    1  277    6    0    2    0    3   32   12  287   91   36 |    s = 2914908
    0   69   61    1    0  127   16    8   78    0  173    0    2    1    0    1   15    6   20  702   49 |    t = 3091127
    0  129  129    2    0  189   23   26  106    0  408    0    1    2    0    0   23   18   38  125 1015 |    u = 3091788
```

4.4.3.6 *KNN* sur le contenu et la structure

```
Correctly Classified Instances       27864               57.6835 %
Incorrectly Classified Instances     20441               42.3165 %
Kappa statistic                          0.5113
Mean absolute error                      0.0425
Root mean squared error                  0.1863
Relative absolute error                 51.4859 %
Root relative squared error             91.7222 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.617     0.001      0.827     0.617     0.707      0.809    1474166
                 0.406     0.063      0.361     0.406     0.382      0.659    1480358
                 0.802     0.054      0.726     0.802     0.762      0.895    1484914
                 0.777     0.001      0.913     0.777     0.839      0.888    1486363
                 0.423     0.001      0.853     0.423     0.566      0.69     1507239
                 0.584     0.055      0.691     0.584     0.633      0.783    1597184
                 0.361     0.003      0.589     0.361     0.447      0.662    1620218
                 0.381     0.012      0.522     0.381     0.44       0.677    1685758
                 0.39      0.055      0.292     0.39      0.334      0.682    1886386
                 0.539     0.001      0.891     0.539     0.672      0.763    1895383
                 0.629     0.131      0.616     0.629     0.622      0.78     2112299
                 0.535     0          0.902     0.535     0.672      0.818    2257163
                 0.5       0          0.846     0.5       0.628      0.796    2263642
                 0.156     0.001      0.43      0.156     0.229      0.577    2314377
                 0.885     0          0.979     0.885     0.93       0.952    2328885
                 0.243     0.002      0.44      0.243     0.314      0.616    2635947
                 0.622     0.021      0.595     0.622     0.608      0.836    2773006
                 0.535     0.005      0.725     0.535     0.616      0.797    2879927
                 0.263     0.014      0.325     0.263     0.291      0.619    2914908
                 0.536     0.052      0.224     0.536     0.316      0.769    3091127
                 0.453     0.013      0.633     0.453     0.528      0.756    3091788
Weighted Avg.    0.577     0.063      0.603     0.577     0.582      0.776

=== Confusion Matrix ===

    a    b    c    d    e    f    g    h    i    j    k    l    m    n    o    p    q    r    s    t    u   <-- classified as
  124    9    6    0    0    4    0    1    9    0   31    0    0    0    0    0    4    0    5    5    3 |    a = 1474166
    4 1575  242    4    0  635    6   69  246    1  636    3    3    0    0    4   72   14   76  237   57 |    b = 1480358
    1  246 5828    1    0   88    9   48  161    0  559    0    2   11    0    1   59    8   71  143   31 |    c = 1484914
    0   20    8  431    1    2    3    5   22    0   46    0    0    0    0    0    2    0    1   13    1 |    d = 1486363
    0   35   31    1  261   17    0    9   49    0  133    0    0    1    1    0   36    7    1   31    4 |    e = 1507239
    6  671  373    4    0 4918   20   74  469    4  830    1    5    6    0   69  120   85  204  463   96 |    f = 1597184
    0   46   59    2    1   38  238   24   55    2  110    0    1    1    1    0    7    7   14   37   17 |    g = 1620218
    1  136   80    2    3   92   17  605  114    2  359    0    0    3    0    2   33   20   22   71   26 |    h = 1685758
    1  200  176   13    5  145    7   53 1036    0  518    0    3    5    0    2  158    7   21  294   15 |    i = 1886386
    0   36   22    0    0   12    2    3   31  261   92    0    0    0    0    0    5    0    0   11    9 |    j = 1895383
   10  783  741    9   11  555   33  157  814   21 7613    3    0   23    9   12  277   37  119  682  196 |    k = 2112299
    0   38    8    0    0   20    0    2    5    0   27  129    0    0    0    0    1    0    0    5    6 |    l = 2257163
    0   15   15    0    0   29    3    3    9    0   26    0  115    0    0    0    3    1    4    6    1 |    m = 2263642
    1   28   25    0    0   27    0    2    9    0   97    0    1   43    0    0   12    0    0   14   16 |    n = 2314377
    0    5    9    0    0    2    0    2   12    0   29    0    0    0  523    0    2    0    3    3    1 |    o = 2328885
    0   25   15    0    0   52    2    1   29    0   66    0    2    0    0   74   12    3   12   10    1 |    p = 2635947
    0   84   73    0    0   28    6   19  128    1  357    0    1    0    0    0 1439    1    7  156   14 |    q = 2773006
    1   61   29    1   24   40    8   26   42    1  149    0    0    1    0    0   97  605   11   18   16 |    r = 2879927
    1  125   95    1    0  121   13   24   88    0  238    7    0    5    0    4   39   10  321   96   31 |    s = 2914908
    0   76   64    1    0  121   15    8   90    0  143    0    2    0    0    0   18    4   28  712   47 |    t = 3091127
    0  154  126    2    0  167   22   24  134    0  306    0    1    1    0    0   24   26   69  165 1013 |    u = 3091788
```

Les résultats obtenus sont résumés dans la figure suivante :

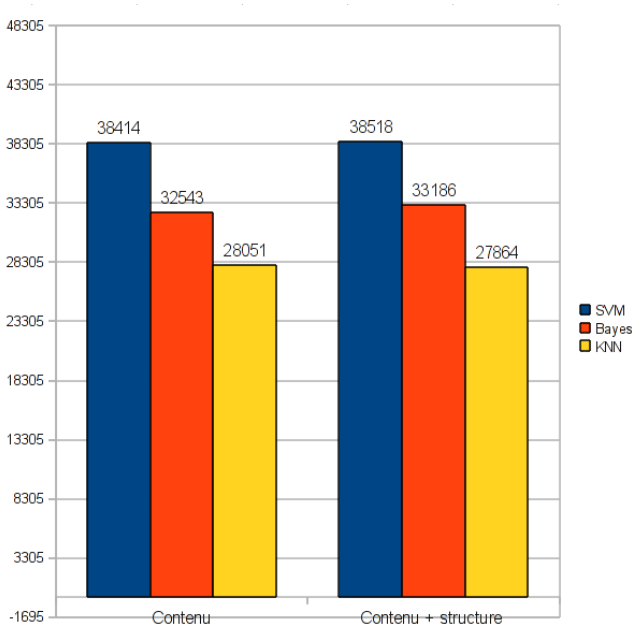


Figure 4.2 – *Un histogramme qui représente le nombre de documents correctement classifiés des divers méthodes de classification appliquées sur une représentation basée sur le contenu seul et notre représentation basée sur le contenu et la structure.*

4.5 La fusion

Nous avons vu que la tâche de catégorisation de texte est de construire une approximation d'une fonction inconnue $\varphi : D \times C \rightarrow \{V, F\}$, où φ s'appelle un classifieur. Cependant, au lieu d'attribuer une sortie de type $\{V, F\}$ (classe) à un document d_i, il est préférable d'en attribuer une mesure qui représente une distribution de probabilités d'appartenances à chaque classe comme suit $\varphi(d_i) = \{m_1, m_2, \cdots, m_{|C|}\}$ (voir la section 3.3).
La figure 4.3 illustre la décision du classifieur *SVM* pour le document `Instance 1` du corpus de test :

Instance 1 : { 0.21762597970361017 , 0.060231464308183456 , 0.0065703352969177665 , 0.001128332850800984 , 8.688918433369702E-4 , 0.019957071638665537 , 0.0032895557394132875 , 0.015106178723183102 , 0.0033535985565545214 , 7.062909727684878E-4 , 0.6556855592385595 , 8.110027229785687E-4 , 7.052136801597756E-4 , 0.001064776067813498 , 5.18330911976195E-4 , 8.123066144190994E-4 , 9.96627186383671E-4 , 8.990419782137952E-4 , 0.0036252626251512825 , 0.002025985236722724 }

Figure 4.3 – *Décision de SVM pour `Instance 1`*

La solution pour amélioré le taux de reconnaissance de l'ensemble des classifieurs est de les combiner en utilisant la théorie de fusion d'information (chapitre 3). Parmi les méthodes de

fusion adéquates à ce type de sortie nous allons opter pour la méthode de fusion linéaire en raison de sa simplicité à mettre en œuvre.

4.5.1 La fusion linéaire

La méthode de fusion linéaire est l'une des méthodes fixes (*chapitre* 3.3.7) particulièrement adaptée aux décisions de type mesures. Son principe est le suivant :

Étant donné un ensemble de $\{j = 1, \cdots, L\}$ classifieurs dont la décision pour un document d_i est représentée par le vecteur $P_j(d_i)$:

$$P_j(d_i) = \{m_1(j), m_2(j), \cdots, m_{|}C|(j)\} \tag{4.2}$$

La combinaison de $\{i = 1, \cdots, |C|\}$ probabilités est donnée par la formule suivante :

$$P_{m_i} = \lambda \sum_{j=1}^{L} m_i \tag{4.3}$$

La décision finale revient à choisir la classe C_i pour laquelle la probabilité P_{m_i} est la plus élevée.

Le problème pour appliquer cette méthode de fusion est comment déterminer les poids idéals. Nous avons pour cela réalisé plusieurs expériences en variant chaque fois la valeur de λ de chaque classifieurs. Nous avons obtenu la combinaison suivante :

SVM	$\lambda = \frac{3}{5}$
Bayes	$\lambda = \frac{1}{5}$
KNN	$\lambda = \frac{1}{5}$

Nous constatons que ces valeurs obtenu par expérience sont en accord avec la fiabilité des classifieurs. Par la suite nous présentons les résultats de fusion linéaire avec la combinaison des λ précédente :

```
=== Summary === By Azizi

Correctly Classified Instances        38857      80.4409 %
Incorrectly Classified Instances       9448      19.5591 %
Kappa statistic                        0.7738
Total Number of Instances             48305

=== Detailed Accuracy By Class === By Azizi

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.776     0.002     0.624      0.776     0.692     0.000     1474166
                 0.662     0.026     0.698      0.662     0.679     0.000     1480358
                 0.916     0.013     0.926      0.916     0.921     0.000     1484914
                 0.919     0.001     0.916      0.919     0.917     0.000     1486363
                 0.833     0.002     0.82       0.833     0.826     0.000     1507239
                 0.832     0.034     0.842      0.832     0.837     0.000     1597184
                 0.8       0.003     0.795      0.8       0.798     0.000     1620218
                 0.576     0.007     0.725      0.576     0.642     0.000     1685758
                 0.666     0.014     0.741      0.666     0.701     0.000     1886386
                 0.835     0.001     0.867      0.835     0.851     0.000     1895383
                 0.814     0.084     0.778      0.814     0.795     0.000     2112299
```

```
                 0.967     0.001     0.889     0.967     0.926     0.000     2257163
                 0.843     0.001     0.798     0.843     0.82      0.000     2263642
                 0.611     0.003     0.506     0.611     0.554     0.000     2314377
                 0.963     0.0       0.968     0.963     0.965     0.000     2328885
                 0.734     0.003     0.648     0.734     0.688     0.000     2635947
                 0.861     0.006     0.886     0.861     0.873     0.000     2773006
                 0.836     0.007     0.734     0.836     0.782     0.000     2879927
                 0.595     0.007     0.689     0.595     0.638     0.000     2914908
                 0.775     0.013     0.637     0.775     0.699     0.000     3091127
                 0.85      0.008     0.841     0.85      0.845     0.000     3091788
Weighted Avg.    0.804     0.034     0.805     0.804     0.804

=== Confusion Matrix === By Azizi

      a      b      c      d      e      f      g      h      i      j      k      l      m      n      o      p      q      r      s      t      u   <-- classified as
    156      3      1      0      0      2      0      0      0      0     33      0      3      0      0      0      0      0      3      0      0 |     a = 1474166
     17   2570     78      2      0    367     19     64     84      6    475      5      8      8      2     10      4     36     37     51     41 |     b = 1480358
      1     40   6658      0      4     66      9     20     35      8    345      0      3     20      0      0      0     16     12     17     13 |     c = 1484914
      0      0      0    510      4      1      0      3     17      0     19      0      0      0      0      0      0      0      1      0      0 |     d = 1486363
      1      0      0      0    514      0      0      0      5      0     88      0      0      0      0      0      1      8      0      0      0 |     e = 1507239
     15    450     26      0     10   7000     13     27     47      1    411      3      8     22      0     23      2     63     90    136     71 |     f = 1597184
      5      6     16      0      3      5    528      5      3      2     32      0      5      0      2      0      1      9      1     33      4 |     g = 1620218
      5     66     28      8      3     64     17    914     45      5    310      2      3      7      0      0      1     55     15     13     27 |     h = 1685758
      5     46     11     20     12    143      4     32   1770      2    302      0      4     22      0      6    112      3     22    137      6 |     i = 1886386
      0      0      0      0      0      0      0      0      0    404     79      0      0      0      0      0      0      0      1      0      0 |     j = 1895383
     30    375    289      7     40    376     33    175    251     38   9855      7      6     63     13     69    110     92     90     91     95 |     k = 2112299
      0      2      0      0      0      1      0      0      0      0      5    233      0      0      0      0      0      0      0      0      0 |     l = 2257163
      0      1      0      0      0     12      1      1      2      0     16      0    194      0      0      0      0      0      2      0      1 |     m = 2263642
      0      0      4      0      0     14      0      3      2      0     70      0      1    168      0      0      6      0      1      1      5 |     n = 2314377
      0      0      0      0      0      0      0      0      1      0     21      0      0      0    569      0      0      0      0      0      0 |     o = 2328885
      0      4      1      0      0     58      0      0      0      0     15      0      1      0      0    223      0      0      0      2      0 |     p = 2635947
      0      5      1      2      0      7      0      2     82      0    192      0      1     13      1      0   1993      0      9      2      4 |     q = 2773006
      1     14      3      1     35     22      5      8      1      0     72      0      0      0      1      0      0    945     11      1     10 |     r = 2879927
     12     64     67      2      0     85     13      5      6      0    132     12      0      2      0      8     16     32    725     21     17 |     s = 2914908
      0     15      2      2      2     47     13      2     32      0     86      0      2      3      0      5      3      5     15   1030     65 |     t = 3091127
      2     23      4      3      0     43      9      0      5      0    116      0      4      4      0      0      1     23     17     82   1898 |     u = 3091788
```

Nous allons appliquer aussi deux autres méthodes de fusion suivantes qui seront des références de comparaison :

4.5.2 La fusion évidentiel

Dans cette théorie la première difficulté est le choix de la fonction de masse (*chapitre* 3.3.2). Plusieurs approches ont été proposées et nous retiendrons celle de [Xu *et al.*, 1992] qui implique la considération de 2^{Ω} fonctions de masse aux seuls éléments focaux possibles $m_k(c_{i_k})$, $m_k(\neg c_{i_k})$ et $m_k(\Omega)$:

$$\begin{cases} m_k(c_{i_k}) = \tau_{rec}^k \\ m_k(\neg c_{i_k}) = \tau_{err}^k \\ m_k(\Omega) = 1 - \tau_{rec}^k - \tau_{err}^k \end{cases} \tag{4.4}$$

Notons que dans notre cas $m_k(\Omega)$ est toujours égale à zéro puisque le taux de rejet est nul.

La combinaison des trois classifieurs se fait par la somme orthogonale de Dempster donnée par la formule suivante :

$$m(c_i) = m_1(c_i) \oplus m_2(c_i) \oplus m_3(c_i) = \frac{1}{K} \sum_{A \cap B \cap C = c_i} m_1(A) \cdot m_2(B) \cdot m_3(C) \tag{4.5}$$

Où,

$$K = \sum_{A \cap B \cap C \neq \phi} m_1(A) \cdot m_2(B) \cdot m_3(C) \tag{4.6}$$

La décision finale est donnée en faveur de la classe qui maximise la masse de croyance.

En appliquant cette méthode nous avons obtenu les résultats suivants :

```
 === Summary === By Azizi

Correctly Classified Instances        38585      79.8779 %
Incorrectly Classified Instances       9720      20.1221 %
Kappa statistic                        0.7666
Total Number of Instances              48305

=== Detailed Accuracy By Class ===By Azizi

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.791     0.002     0.668      0.791     0.724     0.000     1474166
                 0.645     0.024     0.709      0.645     0.675     0.000     1480358
                 0.913     0.014     0.922      0.913     0.918     0.000     1484914
                 0.915     0.001     0.904      0.915     0.91      0.000     1486363
                 0.823     0.002     0.822      0.823     0.823     0.000     1507239
                 0.833     0.037     0.833      0.833     0.833     0.000     1597184
                 0.779     0.002     0.815      0.779     0.796     0.000     1620218
                 0.548     0.007     0.734      0.548     0.628     0.000     1685758
                 0.653     0.013     0.749      0.653     0.698     0.000     1886386
                 0.829     0.001     0.876      0.829     0.851     0.000     1895383
                 0.819     0.101     0.749      0.819     0.782     0.000     2112299
                 0.946     0.001     0.894      0.946     0.919     0.000     2257163
                 0.813     0.001     0.792      0.813     0.803     0.000     2263642
                 0.604     0.003     0.508      0.604     0.551     0.000     2314377
                 0.954     0.0       0.967      0.954     0.961     0.000     2328885
                 0.717     0.003     0.616      0.717     0.663     0.000     2635947
                 0.85      0.005     0.888      0.85      0.869     0.000     2773006
                 0.817     0.007     0.749      0.817     0.781     0.000     2879927
                 0.577     0.007     0.695      0.577     0.63      0.000     2914908
                 0.735     0.011     0.663      0.735     0.697     0.000     3091127
                 0.842     0.007     0.852      0.842     0.847     0.000     3091788
Weighted Avg.    0.799     0.038     0.8        0.799     0.798

=== Confusion Matrix === By Azizi

    a     b     c     d     e     f     g     h     i     j     k     l     m     n     o     p     q     r     s     t     u   <-- classified as
  159     4     1     0     0     3     0     0     0     0    30     0     2     0     0     0     0     0     1     0     1 |    a = 1474166
   14  2504    83     2     1   397    13    53    65     4   561     5     9     7     2    11     5    34    39    42    33 |    b = 1480358
    0    33  6637     2     5    63     5    20    36     5   393     0     1    16     0     1     0    11    15    11    13 |    c = 1484914
    0     0     0   508     4     3     0     3    18     0    18     0     1     0     0     0     0     0     0     0     0 |    d = 1486363
    0     0     0     0   508     1     0     0     4     0    94     0     0     0     0     0     1     9     0     0     0 |    e = 1507239
   11   418    24     0    10  7015    11    23    43     1   465     3     7    18     0    52     2    56    77   117    65 |    f = 1597184
    4     4    24     0     3     8   514     8     3     1    42     0     4     3     3     0     0     7     1    26     5 |    g = 1620218
    5    53    28     8     2    79    17   871    42     4   364     2     8     8     0     1     0    49    16    10    21 |    h = 1685758
    3    37    10    22    12   134     4    25  1735     2   385     0     4    20     0     5   113     3    23   115     7 |    i = 1886386
    0     0     0     0     0     0     0     0     0   401    82     0     0     0     0     0     0     0     0     0     1 |    j = 1895383
   28   360   319     9    35   382    30   157   244    39  9919     7     6    66    12    53   105    85    82    80    87 |    k = 2112299
    0     1     1     0     0     5     0     0     0     0     5   228     0     0     0     0     0     0     1     0     0 |    l = 2257163
    0     0     0     0     0    14     1     1     2     0    21     0   187     1     0     0     0     0     1     1     1 |    m = 2263642
    0     2     1     1     0    18     0     1     2     0    73     0     1   166     0     0     4     0     0     1     5 |    n = 2314377
    0     0     0     0     0     0     0     0     1     0    26     0     0     0   564     0     0     0     0     0     0 |    o = 2328885
    0     3     0     0     0    60     0     0     0     0    23     0     0     0     0   218     0     0     0     0     0 |    p = 2635947
    0     6     1     2     1     7     0     3    83     1   225     0     0     8     0     0  1968     0     5     1     3 |    q = 2773006
    1    10     1     1    35    29     7    13     1     0    88     0     0     0     1     0     0   923    11     1     8 |    r = 2879927
   12    55    63     2     0   103    10     4     5     0   157    10     0     2     0     9    16    31   703    21    16 |    s = 2914908
    0    17     2     3     2    57    11     4    26     0   130     0     1     7     0     4     2     5    20   977    61 |    t = 3091127
    1    25     4     2     0    43     8     0     5     0   149     0     5     5     1     0     0    20    16    70  1880 |    u = 3091788
```

4.5.3 La fusion probabiliste

Nous avons vu dans le chapitre 3.3.1 que cette méthode de fusion est basé sur l'estimation de la probabilité *a posteriori* qui s'écrit :

$$P(C_i|e_1 = C_1, \cdots, e_L = C_L) = P(C_i) \prod_{l=1}^{L} \frac{P(C_i|e_l = C_l)}{P(C_i)} \tag{4.7}$$

Pour une implémentation pratique [Xu *et al.*, 1992] a proposé une approximation de 4.7 par la formule suivante :

$$bel(i) = \eta \prod_{k=1}^{K} P(x \in C_i|e_k(x) = j_k) \tag{4.8}$$

où η est un constant qui assure que $\sum_{i=1}^{M} bel(i) = 1$. C'est-à-dire, nous avons

$$\frac{1}{\eta} = \sum_{i=1}^{M} \prod_{k=1}^{K} P(x \in C_i|e_k(x) = j_k) \tag{4.9}$$

Les probabilités $P(x \in C_i|e_k(x) = j_k)$ sont déterminées à partir de la matrice de confusion par la formule :

$$P(x \in C_i|e_k(x) = j_k) = \frac{n_{ij}^{(k)}}{n_{.j}^{(k)}} = \frac{n_{ij}^{(k)}}{\sum_{i=1}^{M} n_{ij}^{(k)}}, i = 1, \cdots, M \tag{4.10}$$

La matrice de confusion pour un classifieur k est donnée par :

$$PT_k = \begin{pmatrix} n_{11}^{(k)} & n_{12}^{(k)} & \cdots & n_{1M}^{(k)} \\ n_{21}^{(k)} & n_{22}^{(k)} & \cdots & n_{2M}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ n_{M1}^{(k)} & n_{M2}^{(k)} & \cdots & n_{MM}^{(k)} \end{pmatrix} \tag{4.11}$$

Finalement la décision finale est déterminer par l'indice de la classe qui maximise la probabilité *a posteriori*.
Notons que M est le nombre des classes et K est le nombre des classifieurs.

Après avoir appliquer cette méthode de fusion pour notre cas, nous avons obtenu les résultats suivants :

```
 === Summary === By Azizi

Correctly Classified Instances        37395      77.4143 %
Incorrectly Classified Instances      10910      22.5857 %
Kappa statistic                       0.7332
Total Number of Instances             48305

=== Detailed Accuracy By Class === By Azizi

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.537     0.0       0.991      0.537     0.697     0.000     1474166
                 0.564     0.016     0.753      0.564     0.645     0.000     1480358
```

	0.9	0.013	0.926	0.9	0.913	0.000	1484914
	0.894	0.0	0.963	0.894	0.927	0.000	1486363
	0.653	0.001	0.856	0.653	0.741	0.000	1507239
	0.866	0.056	0.775	0.866	0.818	0.000	1597184
	0.741	0.001	0.878	0.741	0.804	0.000	1620218
	0.404	0.003	0.822	0.404	0.541	0.000	1685758
	0.621	0.011	0.773	0.621	0.689	0.000	1886386
	0.479	0.0	0.932	0.479	0.633	0.000	1895383
	0.867	0.183	0.652	0.867	0.744	0.000	2112299
	0.929	0.0	0.953	0.929	0.941	0.000	2257163
	0.665	0.0	0.9	0.665	0.765	0.000	2263642
	0.12	0.0	0.805	0.12	0.209	0.000	2314377
	0.944	0.0	0.977	0.944	0.96	0.000	2328885
	0.118	0.0	0.837	0.118	0.207	0.000	2635947
	0.837	0.005	0.898	0.837	0.866	0.000	2773006
	0.68	0.004	0.805	0.68	0.737	0.000	2879927
	0.399	0.002	0.835	0.399	0.54	0.000	2914908
	0.636	0.007	0.72	0.636	0.675	0.000	3091127
	0.816	0.006	0.874	0.816	0.844	0.000	3091788
Weighted Avg.	0.774	0.06	0.789	0.774	0.767		

=== Confusion Matrix === By Azizi

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	<-- classified as
108	2	1	0	0	5	0	0	0	0	82	0	2	0	0	0	0	0	1	0	0	a = 1474166
0	2190	74	0	0	603	6	32	47	0	834	1	4	0	0	1	4	24	10	28	26	b = 1480358
0	26	6538	0	2	71	4	7	29	0	554	0	1	2	0	0	0	10	6	8	9	c = 1484914
0	0	0	496	2	3	0	3	17	0	34	0	0	0	0	0	0	0	0	0	0	d = 1486363
0	0	0	0	403	1	0	0	4	0	203	0	0	0	0	0	1	5	0	0	0	e = 1507239
0	282	20	0	4	7293	5	14	36	0	615	2	1	1	0	3	1	23	22	49	47	f = 1597184
0	3	25	0	1	15	489	4	3	0	85	0	1	0	2	0	0	4	0	25	3	g = 1620218
0	39	28	6	1	106	10	641	33	0	648	1	2	0	0	0	0	41	6	6	20	h = 1685758
0	21	9	10	7	185	3	16	1652	0	558	0	3	0	0	0	100	0	6	83	6	i = 1886386
0	0	0	0	0	0	0	0	0	232	252	0	0	0	0	0	0	0	0	0	0	j = 1895383
0	268	297	2	16	445	19	56	194	17	10490	2	1	5	11	3	94	42	22	53	68	k = 2112299
0	1	1	0	0	7	0	0	0	0	8	224	0	0	0	0	0	0	0	0	0	l = 2257163
0	0	0	0	0	35	0	0	2	0	39	0	153	0	0	0	0	0	0	0	1	m = 2263642
0	0	1	0	0	19	0	1	2	0	213	0	0	33	0	0	3	0	0	0	3	n = 2314377
0	0	0	0	0	0	0	0	0	0	33	0	0	0	558	0	0	0	0	0	0	o = 2328885
0	1	0	0	0	119	0	0	0	0	148	0	0	0	0	36	0	0	0	0	0	p = 2635947
0	2	0	0	0	8	0	0	78	0	284	0	0	0	0	0	1937	0	3	1	1	q = 2773006
1	7	1	0	33	61	6	4	1	0	234	0	0	0	0	0	0	768	5	1	8	r = 2879927
0	45	61	1	0	188	5	1	2	0	363	5	0	0	0	0	16	18	486	16	12	s = 2914908
0	7	2	0	2	169	6	1	32	0	193	0	1	0	0	0	2	3	7	845	59	t = 3091127
0	16	4	0	0	78	4	0	5	0	221	0	1	0	0	0	0	16	8	58	1823	u = 3091788

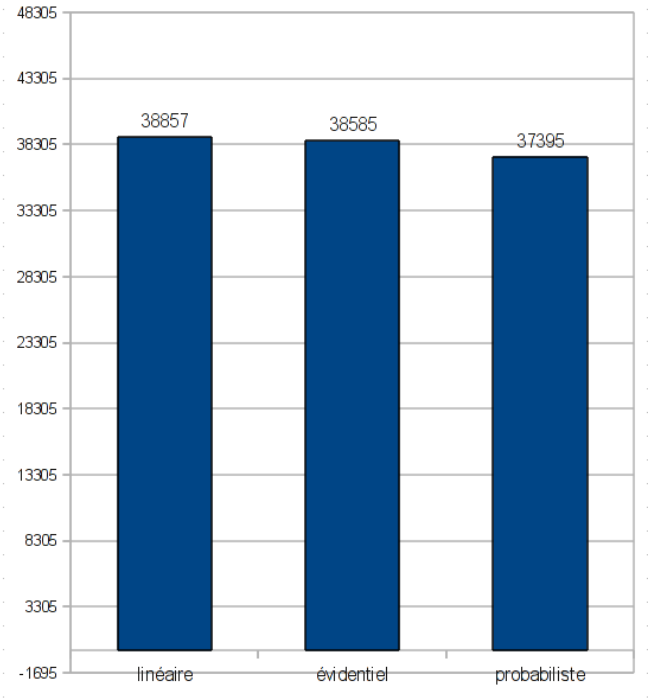


Figure 4.4 – *Un histogramme qui représente le nombre de documents correctement classifiés des divers méthodes de fusion.*

4.6 Discussion

Les résultats obtenus par notre modèle sur le corpus **INEX 2007** sont des résultats encourageants. Cette section propose de discuter les différents choix effectués pour chacune des étapes afin de moduler la signification de nos résultats.

- La première étape du processus consiste à définir une représentation du corpus par des termes. Nous avons choisi ici la représentation basée sur les mots. Cette étape est constitue de deux points principaux :
  - l'extraction du contenu d'un document XML, dans laquelle nous avons proposé un algorithme d'extraction (algo. 3) basé sur le parseur *JDom*. Cette algorithme permet la transformation d'un document XML en un document texte plat qui facilite par la suite son traitement.
  - la réduction de la dimension du corpus, dans laquelle nous avons appliquer deux méthode de sélection de termes à savoir le *stemming* et la suppression des *stop words*. la sélection des termes est intéressante puisqu'elle permet de réduire le nombre des mots distincts de **446 916** à **8 298** et par conséquence réduire la complexité de traitement.
- La deuxième étape concerne l'apprentissage, dans laquelle nous avons opter pour trois algorithmes (*SVM*, *KNN* et *Bayes multinominal*). Ce choix était motivé par les performances de ces algorithmes ainsi que leur capacité à traiter des grosse collections.

	SVM	Bayes	KNN
contenu seul	0.793	0.681	0.582
contenu et structure	0.796	0.693	0.582

TABLE 4.3 – *valeurs de F-mesure des différentes méthodes de classification*

A l'exception de *KNN*, les résultats obtenus, dans cette partie qui sont illustrer dans la table 4.3, montrent clairement que cette méthode de représentation des documents XML apporte plus d'efficacité par rapport à la représentation qui ne tient compte que du contenu du document seul.
Le résultat médiocre de *KNN* peut être dû aux paramètres de distance et de nombres de K voisins qui affectent les performances de la méthode.

- La dernière étape consiste à proposer un schéma de fusion afin d'améliorer le processus de classification. Nous avons en premier temps appliqué un schéma de fusion linéaire pondéré (la pondération a été choisie selon la fiabilité du classifieur. 3/5 pour *SVM* et 1/5 pour les autres) selon la formule suivante :

$$p_m = \frac{3m_i^{svm} + m_i^{bayes} + m_i^{knn}}{5} \tag{4.12}$$

Ensuite et afin de valider notre méthode nous avons appliqué deux autres méthodes de fusion qui sont souvent utilisées pour la combinaisons des classifieurs. Le tableau 4.4 montre les résultats obtenus lors de l'application des trois méthodes de fusion :

1. La méthode linéaire pondéré. (non paramétrique et sortie de classifieurs de type mesure)
2. La méthode probabiliste (bayésienne) selon [Xu *et al.*, 1992]. (paramétrique et sortie de classifieurs de type classe)
3. Le méthode de Dempster-Shafer (évidentiel) selon [Xu *et al.*, 1992]. (paramétrique et sortie de classifieurs de type classe)

	linéaire	probabiliste	évidentiel
F-mesure	0.804	0.767	0.798

TABLE 4.4 – *valeurs de F-mesure des différentes méthodes de fusion*

4.7 Conclusion

L'objet de ce chapitre est proposé une architecture de fusion d'information qui permet d'améliorer la classification des documents semi structurés en tenant compte de la structure et le contenu de ces documents.

Nous avons introduit une nouvelle méthode de représentation des document XML. Cette méthode est basé sur la transformation de la structure arborescente en texte plat selon deux critères, l'importance de la balise et son niveau dans l'arbre.

Nous avons illustré notre procédé par une application sur un corpus réel (**INEX 2007**). Nous avons décrit chaque étape de notre processus et présenté les résultats de nos expériences en appliquant trois algorithmes d'apprentissage.

Afin de perfectionner les performances de notre méthode de classification des documents XML, nous avons proposé un schéma de fusion dans lequel nous avons combiné les trois classifieurs selon une méthode linéaire.

Pour valider cette méthode nous avons fait une comparaison par rapport à deux autres méthodes de fusion, souvent utilisées pour la combinaisons des classifieurs. Les résultats obtenus sont très satisfaisantes et montrent l'efficacité de notre approche.

Conclusion générale

Le travail présenté dans ce mémoire se situe dans le contexte du text mining, et plus particulièrement dans le cadre de classification des documents XML.

Dans ce contexte, nous avons commencer notre étude par un premier chapitre dont l'objectif est la présentation détaillé de la notion de classification (catégorisation) de texte. Nous avons vu que le processus de classification comporte généralement trois étapes : la représentation, le choix de classifieurs et l'évaluation du modèle. Le bon choix des termes et la méthode d'apprentissage est primordial puisque ils affectent souvent les performances de la classification. L'étape d'évaluation permet la généralisation des modèles et l'estimation des performances de classification.

Au cours du deuxième chapitre nous avons vus que la classification des document XML n'est qu'une extension de la classification des documents plats en essayant de prendre en considération la notion de structure. Cette notion qui a ramené plus de sémantique au contenu des documents a également suscité un ensemble de traitements supplémentaires. La synthèse des travaux de recherche dans cette direction nous a permet de distinguer trois approches : la première vise à exploiter le contenu d'un document XML en ignorant la structure c'est donc revient à une tâche de classification de texte, la deuxième s'intéresse à la structure des documents et la troisième qui bénéficié des deux informations. Cette dernière nous a paru la plus objective.

Lors des synthèses nous avons constaté que chaque méthode de classification présente des avantages et des inconvénients. La solution convenable pour tiré partie des avantages des ces méthodes est de combiner les classifieurs afin de d'améliorer les performances de classification. En conséquence nous avons présenté dans le troisième chapitre la notion de la fusion. Nous avons mis l'accent sur les méthodes de fusion les plus utilisés dans la littérature. Ces méthodes se distinguent essentiellement par le niveau d'information en sortie qu'apporte chacun des classifieurs.

Le dernier chapitre a été consacré à notre contribution. En premier temps, nous avons proposé un nouveau modèle de représentation des documents XML qui prend en considération la structure et le contenu de ces documents. Ce modèle est basé sur la notion de réplication du contenu des balises textuelles en fonction de leurs importances et leurs profondeurs dans l'arbre. Nous avons ensuite appliqué trois algorithmes de classification en l'occurrence *SVM*, *KNN* et *Bayes multinominal*. Nous avons constaté que la phase de pré-traitement des documents représente une amélioration conséquente des résultats obtenus sur la collection **INEX 2007**.

Dans un deuxième temps, nous avons appliqué une simple méthode de fusion linéaire. Cette méthode non paramétrique (qui n'a besoin que des sorties de classifieurs), montre clairement que la simplicité peut apporter des résultats très satisfaisantes par rapport aux méthodes plus complexe.

Perspectives

Les perspectives envisageables à nos travaux portent sur les points suivants :

- étendre notre modèle de représentation pour la prise en compte des attributs des documents XML.
- penser à appliquer des méthodes de classification adéquates à la nature multilingues de l'information présenté sur le web.
- proposé un schéma de fusion de bas niveau pour la prise en compte du contenu, structure, attribut et la langue des document XML.

Bibliographie

[Agard, 2007] Agard, B. (2007). Extraction de connaissances à partir de données textuelles – vue d'ensemble. *In 10ème Colloque National AIP PRIMECA*.

[Ah-hwee, 1999] Ah-hwee, T. (1999). Text mining : The state of the art and the challenges. *In In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, pages 65–70.

[Aha *et al.*, 1991] Aha, D. W., Kibler, D. et Albert, M. K. (1991). Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66. http://dx.doi.org/10.1023/A:1022689900470.

[Apté *et al.*, 1994] Apté, C., Damerau, F. J. et Weiss, S. M. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3): 233–251. http://www.acm.org/pubs/articles/journals/tois/1994-12-3/p233-apte/p233-apte.pdf.

[Arif, 2005] Arif, M. (2005). *Fusion de Données : Ultime Etape de Reconnaissance de Formes, Applications à l'Identification et à l'Authentification*. Thèse de doctorat, Université François Rabelais, Tours.

[Bloch, 1996] Bloch, I. (1996). Some aspects of demspter-shafer evidence theory for classification of multi-modality medical images taking partial volume effect into account. *Pattern Recognition Letters*, 17:905–919.

[Breiman *et al.*, 1984] Breiman, L., Friedman, J. H., Olshen, R. A. et Stone, C. J. (1984). *Classification and Regression Trees*. Chapman & Hall, New York, NY.

[Candillier *et al.*, 2005] Candillier, L., Tellier, I. et Torre, F. (2005). Transforming xml trees for efficient classification and clustering. *In INEX*, pages 469–480.

[Chang et Lin, 2001] Chang, C.-C. et Lin, C.-J. (2001). Libsvm - a library for support vector machines. The Weka classifier works with version 2.82 of LIBSVM.

[Cherfi, 2004] Cherfi, H. (2004). *Étude et réalisation d'un système d'extraction de connaissances à partir de textes*. Thèse de doctorat, Université Henri Poincaré - Nancy 1, Nancy.

[Cohen et Hirsh, 1998] Cohen, W. W. et Hirsh, H. (1998). Joins that generalize : Text classification using whirl. *In In Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, pages 169–173.

[Cohen et Singer, 1996] Cohen, W. W. et Singer, Y. (1996). Context-sensitive learning methods for text categorization. *In SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–315, New York, NY, USA. ACM. http://doi.acm.org/10.1145/243199.243278.

[Condorcet, 1785] Condorcet, M. (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie Royale, Paris.

[Courbis, 2002] Courbis, C. (2002). *Contribution à la programmation générative, application dans le générateur Smarttools : technologie XML, programmation par aspects et composants*. Thèse de doctorat, Université Sophia-Antipolis, Nice.

[Creecy *et al.*, 1992] Creecy, R. H., Masand, B. M., Smith, S. J. et Waltz, D. L. (1992). Trading mips and memory for knowledge engineering. *Commun. ACM*, 35(8):48–64. http://doi.acm.org/10.1145/135226.135228.

[CYNOBER, 2005] CYNOBER, N. (2005). Manipuler des données xml avec java et jdom.

[de Campos *et al.*, 2007] de Campos, L. M., Fernández-Luna, J. M., Huete, J. F. et Romero, A. E. (2007). Probabilistic methods for structured document classification at inex'07. *In INEX*, pages 195–206.

[Deerwester *et al.*, 1990] Deerwester, S., Dumais, S., Landauer, T., Furnas, G. et Harshman, R. (1990). Indexing by latent semantic anlysis. *Journal of the American Society of Information science*, 416(6):391–407.

[Dempster, 1967] Dempster, A. (1967). Upper and lower probabilities induced by multi-valued mapping. *Annals of Mathematical Statistics*, pages 325–339.

[Denoyer, 2004] Denoyer, L. (2004). *Apprentissage et inférence statistique dans les bases de documents structurés : Application aux corpus de documents textuels*. Thèse de doctorat, Université Paris 6, Paris.

[Denoyer et Gallinari, 2008] Denoyer, L. et Gallinari, P. (2008). Report on the xml mining track at inex 2007 categorization and clustering of xml documents. *ACM SIGIR FORUM*, pages 22–28.

[Denoyer *et al.*, 2007] Denoyer, L., Gallinari, P. et Vercoustre, A.-M. (2007). Report on the xml mining track at inex 2005 and inex 2006 categorization and clustering of xml documents.

[Doucet et Ahonen-Myka, 2002] Doucet, A. et Ahonen-Myka, H. (2002). Naïve clustering of a large xml document collection. *In 1st Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX'02)*, Schloss Dagstuhl, Germany.

[Dubois et Prade, 1985] Dubois, D. et Prade, H. (1985). A review of fuzzy sets aggregation connectives. *Information Sciences*, 36:85–121.

[Duin, 2002] Duin, R. (2002). The combining classifier : to train or not to train ? *volume II of 16th International Conference on Pattern Recognition (ICPR), IEEE Computer Society Press*, pages 765–770.

[Dumais *et al.*, 1998] Dumais, S., Platt, J., Heckerman, D. et Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *In CIKM '98 : Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA. ACM.

[EL-Manzalawy, 2005] EL-Manzalawy, Y. (2005). Wlsvm. You don't need to include the WLSVM package in the CLASSPATH.

[Fayyad *et al.*, 1996a] Fayyad, U., Piatetsky-shapiro, G. et Smyth, P. (1996a). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54.

[Fayyad *et al.*, 1996b] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. et Uthurusamy, R., éditeurs (1996b). *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.

[Fuhr *et al.*, 1991] Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., Tzeras, K., Darmstadt, T. H., Informatik, F. et Knorz, G. (1991). Air/x - a rule-based multistage indexing system for large subject fields. *In Proceedings of RIAO'91*, pages 606–623.

[Garboni *et al.*, 2005] Garboni, C., Masseglia, F. et Trousse, B. (2005). Sequential pattern mining for structure-based xml document classification. *In Workshop of the INitiative for the Evaluation of XML Retrieval (INEX'05)*, pages 458–468.

[Ghosh et Mitra, 2008] Ghosh, S. et Mitra, P. (2008). Combining content and structure similarity for xml document classification using composite svm kernels. *In ICPR08*, pages 1–4.

[Géry *et al.*, 2008] Géry, M., Largeron, C. et Moulin, C. (2008). Ujm at inex 2008 xml mining track. *In INEX 2008 Workshop Pre-Proceedings*, pages 384–390.

[Hull, 1994] Hull, D. (1994). Improving text retrieval for the routing problem using latent semantic indexing. *In SIGIR '94 : Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–291, New York, NY, USA. Springer-Verlag New York, Inc.

[Ibekwe-Sanjuan et Sanjuan, 2004] Ibekwe-Sanjuan, F. et Sanjuan, E. (2004). Ingénierie linguistique et fouille de textes. *Veille stratégique, scientifique et technologique (VSST'2004)*.

[Jalam, 2003] Jalam, R. (2003). *Apprentissage automatique et catégorisation de textes multilingues*. Thèse de doctorat, Université LUMIERE LYON2.

[Joachims, 1998] Joachims, T. (1998). Text categorization with suport vector machines : Learning with many relevant features. *In ECML '98 : Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, London, UK. Springer-Verlag.

[Knijf, 2006] Knijf, J. D. (2006). Fat-cat : Frequent attributes tree based classification. *In Workshop of the INitiative for the Evaluation of XML Retrieval (INEX'06)*, pages 485–496. http://dx.doi.org/10.1007/978-3-540-73888-6_45.

[Lam et Suen, 1997] Lam, L. et Suen, C. (1997). Application of majority voting to pattern recognition : An analysis of its behavior and performance. *IEEE Transactions on Systems, Man Cybernetics*, 27(5):553–568.

[Laur, 2004] Laur, P. A. (2004). *Données semi structurées : Découverte, Maintenance et Analyse de Tendances*. Thèse de doctorat, Université MONTPELLIER II, Montpellier.

[Le Cadre *et al.*, 2003] Le Cadre, J.-P., Nimier, V. et Reynaud, R. (2003). Fusion en traitement du signal. *In* Bloch, I., éditeur : *Fusion d'informations en traitement du signal et des images*, Traité IC2, chapitre 2, pages 29–52. Hermès.

[Lewis, 1992] Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. *In SIGIR '92 : Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50, New York, NY, USA. ACM. http://doi.acm.org/10.1145/133160.133172.

[Lewis *et al.*, 1996] Lewis, D. D., Schapire, R. E., Callan, J. P. et Papka, R. (1996). Training algorithms for linear text classifiers. *In SIGIR '96 : Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306, New York, NY, USA. ACM. http://doi.acm.org/10.1145/243199.243277.

[Li et Jain, 1998] Li, Y. et Jain, A. K. (1998). Classification of text documents. *Pattern Recognition, International Conference on*, 2:1295. http://doi.ieeecomputersociety.org/10.1109/ICPR.1998.711938.

[Maron, 1961] Maron, M. (1961). Automatic indexing : an experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417. http://www.acm.org/pubs/articles/journals/jacm/1961-8-3/p404-maron/p404-maron.pdf.

[Martin, 2005] Martin, A. (2005). Fusion d'informations haut niveau. application à la classification d'images sonar. *In Atelier : Fouille de Données Complexes, EGC'05*.

[Mccallum et Nigam, 1998] Mccallum, A. et Nigam, K. (1998). A comparison of event models for naive bayes text classification. *In AAAI-98 Workshop on 'Learning for Text Categorization'*.

[Meenakshi *et al.*, 2007] Meenakshi, S. M., Lakshmi, K. et Mukherjee, S. (2007). A categorization approach for wikipedia collection based on negative category information and initial descriptions. *In Pre-Proceedings of INEX 2007*, pages 212–214.

[Parker, 2001] Parker, J. (2001). Rank and response combination from confusion matrix data. *Information Fusion*, 2:113–120.

[Ploix *et al.*, 2008] Ploix, M.-A., Garnier, V. et Moysan, J. (2008). Théorie des possibilités appliquée à l'end du béton. *In Possibility theory applied to NDE of concrete, Congrès COFREND*, Toulouse.

[Rocchio, 1971] Rocchio, J. (1971). Relevance feedback in information retrieval. *In* Salton, G., éditeur : *The SMART Retrieval System - Experiments in Automatic Document Processing*, page 313–323, Englewood, Cliffs, New Jersey. Prentice Hall.

[Roli *et al.*, 2002] Roli, F., Fumera, G. et Kittler, J. (2002). Fixed and trained combiners for fusion of imbalanced pattern classifiers. *In In 5th International Conference on Information Fusion,*, Annapolis (Washington) USA.

[Sable et Hatzivassiloglou, 2000] Sable, C. L. et Hatzivassiloglou, V. (2000). Text-based approaches for non-topical image categorization. *International Journal of Digital Libraries*, 3:261–275.

[Salton et Mcgill, 1983] Salton, G. et Mcgill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47. www.cs.umu.se/education/examina/Rapporter/ShoaibTariq.pdf.

[Shafer, 1976] Shafer, G. (1976). A mathematical theory of evidence. *Princeton University Press*.

[Simon, 2000] Simon, A. (2000). *Outils classificatoires par objets pour l'extraction de connaissances dans les bases de données*. Thèse de doctorat, Université Henri Poincaré-Nancy 1, Nancy.

[Simone *et al.*, 2002] Simone, G., Farina, A., Morabito, F. C., Serpico, S. B. et Bruzzone, L. (2002). Image fusion techniques for remote sensing applications. *Information Fusion*, 3(1):3–15.

[Singhal *et al.*, 1997] Singhal, A., Mitra, M. et Buckley, C. (1997). Learning routing queries in a query zone. *SIGIR Forum*, 31(SI):25–32. http://doi.acm.org/10.1145/278459.258530.

[Smets, 1990] Smets, P. (1990). The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 12(5):447–458.

[Tannier, 2006] Tannier, X. (2006). *Extraction et recherche d'information en langage naturel dans les documents semi-structurés*. Thèse de doctorat, Université Jean Monnet, Saint-Etienne.

[Termier *et al.*, 2002] Termier, A., Rousset, M.-C. et Sebag, M. (2002). Treefinder : a first step towards xml data mining. *Data Mining, IEEE International Conference on*, 0:450. http://doi.ieeecomputersociety.org/10.1109/ICDM.2002.1183987.

[Van-Erp et Schomaker, 2000] Van-Erp, M. et Schomaker, L. (2000). Variants of the borda count method for combining ranked classifieur hypotheses. *In In L. Schomaker and L. Vuurpijl, editors, 7th International Workshop on Frontiers in Handwriting Recognition*, pages 443–452, Amsterdam.

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

[Wald, 2002] Wald, L. (2002). Data fusion, definition and architectures, fusion of images of different spatial resolutions. *Presses de l'Ecole des Mines Paris*.

[Wisniewski *et al.*, 2005] Wisniewski, G., Denoyer, L. et Gallinari, P. (2005). Classification automatique de documents structurés. application au corpus d'arbres étiquetés de type XML. *In 2ème Conférence en Recherche d'Informations et Applications (CORIA'05)*, Grenoble.

[Xu *et al.*, 1992] Xu, L., Krzyzak, A. et Suen, C. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418–435.

[Yang et Zhang, 2007] Yang, J. et Zhang, F. (2007). Xml document classification using extended vsm. *In INEX*, pages 234–244.

[Yang, 1999] Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88.

[Yang et Pedersen, 1997] Yang, Y. et Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. *In* Fisher, D. H., éditeur : *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US. Morgan Kaufmann Publishers, San Francisco, US. www.cs.cmu.edu/~yiming/papers.yy/ml97.ps.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information Control*, 8:338–353.

[Zadeh, 1978] Zadeh, L. (1978). Fuzzy sets as a basis for possibility theory. *Fuzzy Sets and Systems*, 1:3–28.

[Zaki, 2002] Zaki, M. J. (2002). Efficiently mining frequent trees in a forest. *In KDD '02 : Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, New York, NY, USA. ACM.

[Zouari, 2004] Zouari, H. K. (2004). *Contribution à l'évaluation des méthodes de combinaison parallèle de classifieurs par simulation*. Thèse de doctorat, Université, Rouen.

RESUME :

Ces dernières années, l'accès à l'information textuelle a connu une évolution rapide, avec en particulier le développement de grandes bases de données textuelles et du web. En particulier, il est devenu important d'être capable de traiter d'énormes quantités de données textuelles, d'apporter des solutions diversifiées aux nouvelles demandes des utilisateurs, et d'automatiser les outils qui permettent d'extraire et d'exploiter l'information textuelle. Les méthodes classiques d'extraction de connaissances échouent la plupart du temps parce qu'elles n'utilisent en fait qu'une seule source d'information. De plus, la diversité de langues dans un même document, d'une part, et la diversité de sens qu'on peut avoir pour le même mot rendent l'extraction de connaissances, à partir de ce type de document, une tâche ardue et difficile.

Parallèlement aux documents de type HTML, de nouveaux documents appelés documents semi structurés sont apparus. Ce type de document représente un compromis entre les données fortement structurées issues de la communauté base de données (données relationnelles par exemple) et les données faiblement structurées issues des communautés document numérique et recherche d'information (documents plats, images . . . etc) [Denoyer, 2004].

Le format de représentation le plus utilisé par excellence est le format XML (eXtensible Markup Langage) qui se caractérise par sa simplicité, extensibilité et puissance de représentation de n'importe quels types de données.

Notre étude s'intéresse à la classification (catégorisation) des documents semi structuré XML. L'apprentissage automatique propose une gamme d'outils qui permettent d'avancer dans cette direction. C'est dans ce cadre que se situe notre travail qui vise à explorer le potentiel des techniques d'apprentissage pour répondre aux besoins de recherche et d'analyse d'information semi structuré comme la méthode à base de SVM, de réseaux de neurones, modèle bayésien,...etc.

Toutes ces méthodes s'accordent sur l'efficacité et la robustesse et afin de tirer profit des avantages de chacune d'elles nous allons proposé une architecture de fusion d'information qui permet d'améliorer la classification des documents semi structurés en tenant compte de la structure et le contenu de ces documents.

Mots-clés : Fouille de données, Document semi-structurés, ECD, Apprentissage automatique, Fusion d'information.

ملخـــــص :

شهدت السنوات الأخيرة ، تطوا كبيرا في الوصول إلى المعلومات النصية ، خصوصا مع تطوير قواعد بيانات نصوص كبيرة وشبكـــة الويـــب. لقد أصبح مـــن المهم على وجه الخصوص أن نكون قادرين على التعامل مـــــع كم هائل مـــــن البيانات النصية ، وتوفير حلول مختلفة للطلبات الجديدة للمستخدمين، وجعل الأدوات التي تمكننا من استخراج واستغلال المعلومات النصية آلية. الأساليب التقليدية لاستخراج المعرفة تفشل في غالب الأحيان لأن معظمها لا يستخدم ســـــوى مصـــــدر وحيد للمعلـــومات. علاوة على ذلك، تنوع اللغات في الوثيقة الواحدة ، من ناحية، وتعدد المعاني للكلمـــــــة الواحدة من ناحية أخرى يجعل استخراج المعرفة من هذا النوع مـــن الوثائق مهمـــــة بالغة الصعوبة.

بالتوازي مع الوثائق من نوع HTML ، ظهرت وثائق جديدة تدعى الوثائق الشبه منظمة . هذا النوع مـــن الوثائق يمثل حلا وسطا بين البيانات المنظمـــة للغاية مـــن مجتمع قواعد البيانات (مثل البيانات العلائقية) والبيانات ضعيفة التنظيم مـــن مجتمعات الوثائـــق الرقميـــة واســـترجاع المعلومـــات (وثائق مسطحة، صور... الخ).

التنسيق التمثيلي المستخـــدم بكثـــرة وبامتياز هو تنسيق XML الذي يتميز بالبساطة ، والتدرجية ، وقوة التمثيل لأي نـــــــوع من البيانات. دراستنا تركـــز على تصنيـــف الوثائق شبه المنظمة XML. التعلم الآلي يقدم لنا مجموعة من الأدوات لتحقيق تقدم في هذا الاتجاه. في هذا السياق يكمـــن عملنا، الذي يهدف إلى استكشاف تقنيات التعلم الآلي لتلبية احتياجات البحث والتحليل للمعلومات شبه المنظمة، مثل طريقة SVM ، الشبكات العصبية, النموذج البايزي...الخ.

جميع هذه الأساليب تتفق على الفعالية والمتانة ، وللإستفادة مـــن محاسن كل واحدة منها سنقترح بنية لدمج المعلومات التي ستمكننا من تحسين تصنيف الوثائق شبه المنظمة مع الأخذ بعين الاعتبار هيكل ومحتوى هذه الوثائق.

كلمات مفتاحية : تنقيب البيانات، الوثائق شبه المنظمة، إستخراج المعارف من البيانات، التعلم الآلي، دمج المعلومات