République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Abbas Laghrour Khenchela
Institut des Sciences et Technologies
Département de Mathématiques et Informatique

Ecole Doctorale

Sciences et Technologies de l'Information et de la Communication

Option : Systèmes d'Informations et de Connaissance

N° Ordre : .....
Réf : .....

Mémoire

Présenté par

Makhlouf LEDMI

Pour obtenir le grade de

Magister en Informatique

THÈME

Classification Automatique des documents XML

Soutenue à Khenchela le 17/10/2010

en présence d'un jury composé de :

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

Promotion 2009/2010

Remerciements

C'est ici l'occasion pour moi de remercier les personnes qui ont participé, à travers leurs conseils et leur encouragements, à réaliser ce travail.

Je remercie infiniment Allah de m'avoir donner le pouvoir d'atteindre ce stade.

Ce mémoire n'aurait pas été possible sans mon encadrant. Je l'en remercie vivement, Mr Abdelouahab Moussaoui Maître de conférence de l'université de Sétif, pour avoir accepté la direction de mon mémoire et pour la confiance qu'il n'a cessée de me renouveler.

Je remercie en premier lieu les membres du jury pour avoir accepté de rapporter et examiner ce travail : Mr Noureddine DJEDI Professeur de l'université de Biskra, Mr Mohamed Chaouki BABAHENINI et Mr Foudil CHERIF Maîtres de conférence de l'université de Biskra.

Je remercie du fond du cœur ma famille et ma femme sans le soutien desquelles je n'aurais jamais songé à me lancer dans cette aventure. Je vous aime et vous serai éternellement reconnaissant.

Je tiens également à remercier mes amis et mes collègues de l'école doctorale, surtout : Nabil Azizi, Abdelouaheb Khiari, les deux Mohamed (Khergag et Boussalem), Mourad Attia, Belkacem Iskhar, Hadj Madani, Ahmed Zahrouri.

Enfin, je remercie toute personne qui m'a aider à réaliser ce travail de loin ou de près, plus particulièrement Farid Ledmi, Djamel Kheddouma, Bachir Takouachet, le personnel de la direction de l'éducation - Khenchela et le personnel de la direction de Jeunesse et de Sport-Khenchela.

Table des matières

Liste des tableaux

Table des figures

Liste des Algorithmes

Introduction Générale

La quantité d'information accessible aujourd'hui est telle que les outils, même sophistiqués, utilisés pour rechercher l'information dans les documents et pages Web ne suffisent plus : il faut maintenant pouvoir "*découvrir*" une information non explicitement contenue dans ces documents, afin de présenter une vue synthétique de grande quantité d'information. C'est précisément l'objectif de ce qu'on appelle la *fouille de texte*. Celle-ci utilise différentes techniques, extraire une information structurée dans du texte libre, regrouper les documents dans des classes existantes ou émergentes, afin d'agréger ou de synthétiser l'information contenue dans une large collection.

De plus, l'apparition des données semi structurées de type XML ou HTML a considérablement modifié le cadre habituel de la Recherche d'Information (RI). En effet, la notion même d'unité d'information est aujourd'hui complètement remise en cause et il est donc nécessaire d'une part d'adapter les modèles pour prendre en compte ce nouveau type de documents et d'autre part il faut s'intéresser aux nouvelles problématiques qui émergent.

Contexte de travail

Notre travail se situe dans le contexte de la **Catégorisation de Textes (TC)** et de **l'Ingénierie des Connaissances (IC)**.

La **Catégorisation de Textes (TC)** consiste à analyser de nouveaux documents et à les affecter, en fonction de leurs caractéristiques ou attributs, à telle ou telle classe prédéfinie. Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des documents à partir de certains traits descriptifs. Les documents sont considérés comme des ensembles d'unités atomiques et indépendantes. Dans la pratique, les documents possèdent une structure interne et sont parfois inter-reliés (c'est le cas des pages HTML et les documents XML). Notre travail se situe dans la perspective d'une **TC** semi-structurée. L'objectif d'un modèle de Catégorisation de Textes Semi-structurés est de prendre en compte la structure des documents, ce qui nécessite de s'interroger sur la sémantique d'une telle structure, et donc des relations entre les différents éléments, pour pouvoir comprendre son impact sur la description de l'information.

L'objectif de **l'Ingénierie des Connaissances (IC)** est de proposer des concepts, des méthodes et des techniques permettant d'acquérir, de modéliser et de formaliser des connaissances pour les mobiliser dans une activité individuelle ou collective au sein d'une organisation ou d'une communauté. Avec la vision du Web Sémantique, qui permet une utilisation

des connaissances formalisées en plus du contenu informel du Web classique, le Web passera du Web de "liens" au Web de "sens". L'IC retrouve de ce fait toute sa place au sein du Web et les ontologies formelles deviennent le concept pivot.

C'est dans ce contexte, nous explorons la prise en compte de la sémantique et de la structure des documents dans une tâche de catégorisation de textes.

Problématique

Notre problématique de catégorisation a comme élément pivot le **texte** :

- Nous nous intéressons à la prise en compte des informations explicites autour des éléments textuels, à savoir les éléments structurels des documents XML (balises),
- Ainsi qu'aux informations implicites (dans la mesure où le texte n'est pas annoté par une représentation de son sens.), à savoir la sémantique des éléments structurels et textuels.

Contribution

Notre contribution concerne les deux domaines : **TC** et Web Sémantique.

Notre contribution dans le cadre de la TC semi-structurée se situe à deux niveaux :

- Etendre la représentation vectorielle classique de Salton aux documents XML, ce qui nous amène à redéfinir les unités (termes) qui permettent une représentation permettant la prise en compte à la fois de structure et de contenu.
- Choisir une méthode de catégorisation adéquate adaptée à la classification des documents XML.

Notre contribution dans le cadre du Web sémantique se situe à plusieurs niveaux :

- Nous proposons de relier les concepts de l'ontologie, que nous définissons, aux termes structurels du corpus utilisé.
- Nous proposons aussi, de relier les termes textuels à leur sens en utilisant Wordnet comme ressource sémantique pour modéliser la notion de voisinage sémantique à l'aide d'un calcul de similarité entre termes.

Plan du mémoire

Ce mémoire se décompose en quatre (04) chapitres : l'état de l'art (chapitre de 1 à 3) et un chapitre où nous présentons nos propositions et nos expérimentations (chapitre 4).

Chapitre 1 : Extraction des connaissances à partir de textes L'objectif de ce chapitre est de présenter un état de l'art des techniques de fouille de données textuelles et plus précisément les techniques de catégorisation de textes " plats ".

Chapitre 2 : Classification des documents XML L'objectif de ce chapitre est d'introduire la notion de structure explicite et plus spécifiquement XML. Après un bref aperçu de la technologie XML, nous présentons les approches de classification semi-structurée proposées dans la littérature, en mettant l'accent sur celles qui prennent en compte à la fois le contenu des documents et/ou leurs structures et se rapprochent de ce fait de notre travail.

Chapitre 3 : Ontologie et classification de textes Ce chapitre présente un état de l'art de l'ontologie en présentant sa définition, son cycle de vie, méthodologies et outils de construction. Nous nous intéressons particulièrement à l'utilisation des ontologies dans le cadre de la catégorisation de textes.

Chapitre 4 : Contribution et Validation Ce chapitre présente notre contribution, nous donnons une vue générale du corpus INEX utilisé, ensuite nous présentons l'architecture globale de notre système en définissant l'approche utilisée pour représenter les documents XML, ainsi que la définition de notre ontologie et la façon de l'intégrer dans le processus de catégorisation. Enfin, nous présentons les résultats d'expérimentation du système sur le corpus de test.

Enfin, nous terminons par une conclusion générale et nous présentons quelques perspectives que nous voyons pour notre travail.

Extraction de connaissances à partir de textes

1.1 Introduction

L'analyse et le traitement d'informations textuelles sont devenus un enjeu majeur avec l'explosion du Web : la plupart des informations accessibles est sous forme textuelle (bibliothèques électroniques, pages HTML, forums de discussion, etc.). De plus, il ne suffit plus d'analyser , mais aussi d'extraire des informations cachées et utiles dans ces données : « *il ne suffit plus de lire des lignes de textes mais aussi de lire ce qui est écrit entre ces lignes* ».

L'extraction de Connaissances à partir de Données (ECD) désigne le processus global de découverte de connaissances qui permet de passer de données brutes à des connaissances alors que la fouille de données n'est qu'une étape de l'ECD au cours de la quelle un modèle est construit.

La fouille de donnée (souvent appelée « data mining ») est l'exploration et l'analyse de grandes quantités de données afin d'y découvrir de l'information implicite. Cette information peut être de différente nature, par exemple on recherchera des règles d'association, une classification ou une segmentation de population.

Il existe également une branche spécialisée de la fouille de données qui prend part à l'analyse de textes libres : la fouille de données textuelles ou « text mining ». La fouille de texte vise à faciliter l'extraction des connaissances « cachées » dans les documents. Ce domaine de recherche tente de mettre à profit la surabondance d'informations textuelles en utilisant des techniques d'informatique linguistique,de datamining, d'apprentissage automatique et de statistiques.

1.2 Extraction des connaissances à partir de données

1.2.1 Définition

Initialement proposée par [Shapiro et Frawley, 1991], la définition de l'ECD a été enrichie au cours du temps et celle proposés dans [Fayyad *et al.*, 1996] est maintenant consensuelle :

Définition 1 (L'extraction des connaissances) *L'extraction des connaissances à partir de bases de données est un processus non trivial qui construit un modèle valide, nouveau, potentiellement utile et au final compréhensible, à partir de données.*

L'ECD peut se décomposer en de nombreuses étapes plus au moins complexes mais la figure 1.1 en donne une vision synthétique. Parmi les grandes étapes de l'ECD, on peut distinguer :

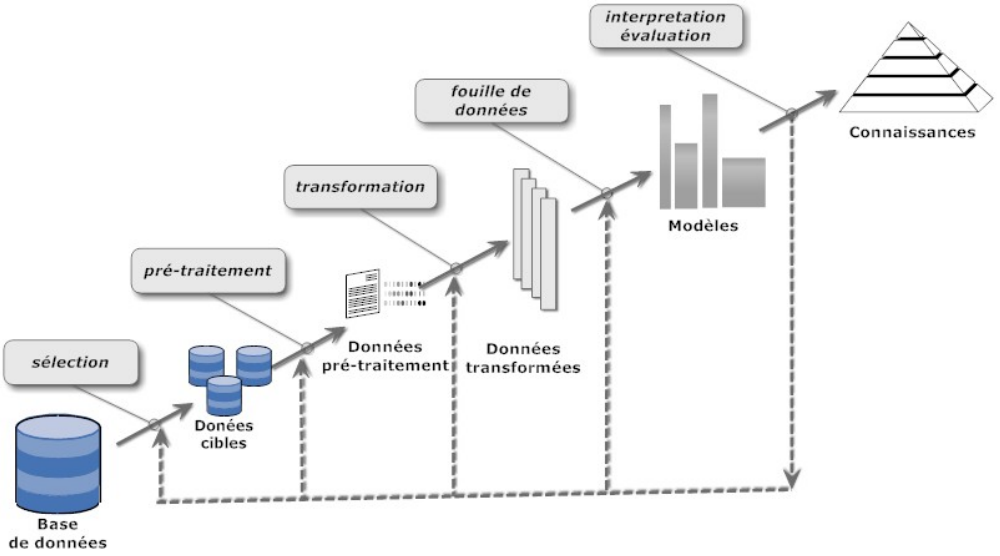

Figure 1.1 – Schéma globale de l'ECD.

- La **sélection** qui crée un ensemble de données à étudier .
- Le **prétraitement** qui vise à enlever le bruit et à définir une stratégie pour traiter les données manquantes.
- La **transformation** où l'on recherche les meilleures structures pour représenter les données.
- La **fouille de données** est la définition des différentes tâches comme : classification, recherche de modèles, et la définition des paramètres appropriés.
- L'**interprétation** et l'évaluation pendant laquelle les patrons extraits sont analysés. La connaissance qui est ainsi extraite est alors stockée dans la base de connaissances.

1.2.2 Les six pôles de l'ECD

On s'intéresse à la découverte de connaissances, on a donc besoin de techniques d'exploration des données (de façon supervisée ou non) pour trouver des formes "intéressantes" qui aident à expliciter une information auparavant cachée dans les données. Les problèmes fondamentaux de la découverte de connaissances sont donc : la représentation des connaissances, la sélection des attributs, la prise en compte des données manquantes, bruitées et rares, la découverte de formes « intéressantes » et « utiles ».

Pour résoudre ces problèmes, l'ECD fait appel aux différentes techniques existantes comme ASA [1], Statistiques, Analyse de Données, réseaux Neuronaux, les systèmes de gestion des BD (SGBD) et Systèmes de Visualisation. Ces techniques ont cependant été construites par des communautés scientifiques très différentes, pour de très bonnes raisons. Avec des préoccupations si différentes, il est tout à fait normal que les domaines scientifiques ci-dessus aient évolué dans des directions différentes. La première difficulté de l'ECD est qu'elle exige une utilisation coordonnée de ces six domaines scientifiques. On ne peut pas sérieusement prétendre qu'il faut « simplement les assembler » [Kodratoff, 1998].

En particulier, ces domaines ont choisi des représentations des connaissances qui rendent compte de différents aspects de la réalité et qui sont presque impossibles à réconcilier sans perdre ce qui fait la force du domaine concerné. Il s'en suit que, d'un point de vue technique, l'ECD peut être vue comme se trouvant au centre d'un hexagone formé de ses sciences parentes, comme illustré par la figure la figure 1.2

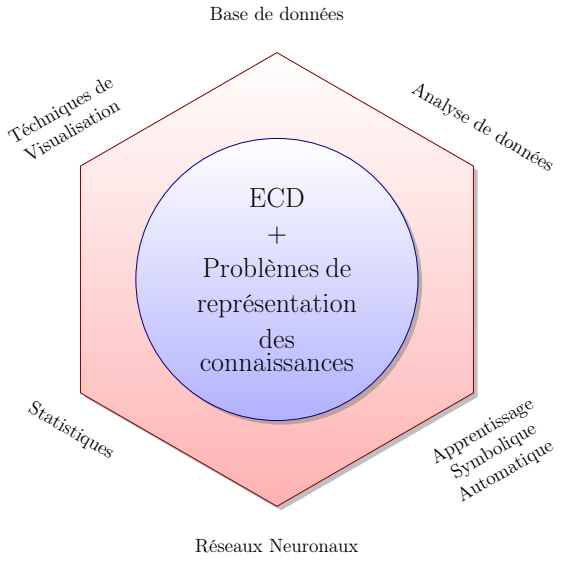

Figure 1.2 – Six pôles del'ECD.

1.2.3 Les tâches de data mining

Le data mining n'est pas le remède miracle capable de résoudre toutes les difficultés ou besoins de l'entreprise mais résout une multitude de problèmes d'ordre intellectuel. Entre autres : Classification, prédiction, estimation, association et Segmentation [Delorme, 2002].

Afin de lever toute ambiguïté sur des termes qui peuvent paraître similaires, il semble raisonnable de les définir :

1. Aprentissage Symbolique Automatique

La classification : Elle consiste à examiner les caractéristiques d'un individu et lui attribuer une classe, la classe est un champ particulier à valeurs discrètes. Des exemples de tâche de classification sont :

- Attribuer ou non un prêt à un client,
- Etablir un diagnostic,
- Accepter ou refuser un retrait dans un distributeur,
- Attribuer un sujet principal à un article de presse.

La classification est souvent réalisée comme une étape préliminaire dans le processus de Data Mining. Elle porte sur le regroupement d'enregistrements, d'observations ou de cas en groupe d'objets similaires dans des classes. Les techniques de classification sont par exemple utilisées lors d'opérations de *mailing* pour cibler la bonne population et éviter ainsi un nombre trop important de réponses ou une réponse vide. De la même manière, cette démarche peut permettre de déterminer, pour une banque, si un prêt peut être accordé, en fonction de la classe d'appartenance d'un client.

L'estimation : Elle consiste à estimer la valeur d'un champ à valeurs continues à partir des caractéristiques d'un objet. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâche d'estimation sont :

- Noter un candidat à un prêt ; cette estimation pourra trouvera une application pour attribuer un prêt (classification), par exemple, en fixant un seuil d'attribution,
- Estimer les revenus d'un client.

Un des intérêts de l'estimation est de pouvoir ordonner les résultats pour ne retenir, si on le désire, que les n meilleures valeurs. Cette technique est souvent utilisée en marketing, combinée à d'autres, pour proposer des offres aux meilleurs clients potentiels. Enfin, il est facile de mesurer la position d'un élément dans sa classe si celui ci a été estimé, ce qui peut être particulièrement important pour les cas limitrophes.

La prédiction : Cela consiste à estimer une valeur future. En général, les valeurs connues sont classées chronologiquement. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des précédentes. Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâche de prédiction sont :

- Prédire les valeurs futures d'actions,
- Prédire au vu de leurs actions passées les départs de clients.

L'association : Cette tâche, plus connue comme *l'analyse du panier de la ménagère*, consiste à déterminer les variables qui sont associées. L'exemple type est la détermination des articles (le poisson et les boissons gazeuses, lait pour nourrissant et les couches bébé ,... etc) qui se retrouvent ensemble sur un même ticket de supermarché. Cette tâche peut être effectuée pour identifier des opportunités de vente croisée et concevoir des groupements

attractifs de produit.

La segmentation (Clustering) : Il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation.

1.3 Extraction des connaissances à partir de textes

La fouille de textes, traduit du terme anglais *text mining*, est apparu dans la deuxième moitié des années 90, en écho à des travaux réalisés depuis les années 80 sur des bases de données. Ce n'est que vers 1995 que l'usage des termes *Knowledge Discovery from Databases* et *Data Mining* se précise.

Les travaux sur les textes bénéficient donc de ces années d'expérience sur les bases de données. En 1995, Feldman [Feldman et Dagan, 1995] introduit le terme de KDT : *Knowledge Discovery in Texts* . Nous calquons donc notre définition de l'extraction des connaissances à partir de textes sur celle de l'extraction de connaissances à partir de bases de données :

Définition 2 (L'extraction de connaissances à partir de textes) : *L'extracton de connaissances à partir de textes est un processus non trivial qui construit un modèle de connaissances valide, nouveau, potentiellement utile et au final compréhensible à partir de textes bruts.*

Puisque nous définissons un processus interactif, nous définissons l'utilisateur d'un système de fouille de textes comme étant un expert du domaine de spécialité et qui à besoin de répertorier, d'accéder ou de raisonner sur la connaissance pour une application donnée. Il est donc capable de valider ou d'invalider une information et de formuler des préférences quant à la structure finale ds connaissances.

1.3.1 Fouille de textes et fouille de données

L'extraction des connaissances à partir de données ou à partir de textes, ont deux objectifs très proches qui exploitent des méthodes et outils très semblables notamment au niveau de l'étape de fouille. Il existe pourtant des différences significatives entre ces deux processus. Ainsi, dans le cadre du traitement de données, l'*ECD* utilise généralement des données brutes qui ont été stockées dans une base de données (souvent relationnelle) et les modéliser en une base de données. Cette modélisation, lorsqu'il s'agit de textes est beaucoup plus complexe. Les textes font référence à des concepts, à des relations entre ces concepts et à

des évènements qu'il est parfois difficile d'identifier [Toussaint, 2004].

D'autre part, la fouille de textes présente ses propres spécificités : les documents peuvent être plus ou moins structurés, la phase de prétraitement joue un grand rôle,... etc. Il est courant et souvent intéressant de prendre en compte l'ordre des mots, et il est maintenant établi que des ressources provenant du traitement automatique des langues et/ou de la linguistique sont nécessaires pour apporter des résultats applicatifs. Beaucoup de méthodes de fouille de textes incluent des traitements d'analyse prélexicale (i.e., traitement des chiffres), d'analyse lexicale telle que l'élimination de « mots vides », morphologie, analyse syntaxique (i.e., détermination des groupes nominaux), analyse sémantique incluant des apports linguistiques aussi bien que des particularités des textes étudiés [N. et B., 2004].

Une des caractéristiques premières des textes, notamment dans des domaines scientifiques techniques est qu'ils font trés largement appel à des concepts. Le mot, unité sémantiquement trop pauvre, a été supplanté par la notion de termes que l'on peut associer à un concept dans une ontologie. Plus la dimension des données est importante, plus il sera difficile d'analyser une classification. Or compte tenu des variations dans la langue pour désigner un concept, il est nécessaire de regrouper des formulations différentes d'un même concept *(taux de chômage/ taux de chômeurs)* ou des concepts différents en un concept plus générique pour éviter une trop grande dispersion des données à fouiller.

L'identification des événements ou des états dans un texte est également une tâche difficile alors qu'elle est souvent implicite dans les bases de données car réalisée au préalable par l'analyste. Ainsi une base de données où sont décrits des individus qui ont des propriétés telles que ***blond, 1,40m*** décrit implicitement des états *avoir les cheuveux de couleur blonde, avoir une taille de 1,40 m.*

Dans le cadre du texte, la construction du sens d'un texte fait appel à de nombreuses étapes, il est de plus en plus facile de se constituer une base de textes sur support électroniques mais le contexte dans lequel ce texte a été produit, la finalité de son auteur, échappent généralement à celui qui le récupère.

1.3.2 Tâche de la fouille de texte

Parmi les techniques empruntées à la fouille de données on retrouve les trois types de tâches principaux :

- la segmentation appelée dans le domaine de l'analyse de texte : regroupement ou « clustering »,
- l'extraction des règles d'association,
- la classification de groupes de textes dans des catégories définies.

L'extraction d'associations découvre les associations entre les différents concepts et exprime les résultats en tant que règles de la forme $A \Rightarrow B[support, confiance]$.

Soit $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ un ensemble de termes, et $\mathcal{K} = \{k_1, k_2, \ldots, k_m\}$ un ensembles de termes-clés asociés à ces textes. Un règle d'association est une implication pondérée de la forme $A \Rightarrow B$ ou $A = \{t_1, t_2, \ldots, t_p\}$ (*la premisse*) et $B = \{t_{p+1}, t_{p+2}, \ldots, t_q\}$ (*la conclusion*). La règle $A \Rightarrow B$ s'interprète comme le fait que tous les textes contenant les termes-clés $\{t_1, t_2, \cdots, t_p\}$ ont tendance aussi de contenir les termes-clés $\{t_{p+1}, t_{p+2}, \ldots, t_q\}$ avec une certaine probabilité donnée par la confiance de la règle. Plusieurs algorithmes permettant de mettre en œuvre un tel processus d'extraction de règles à partir des motifs fréquents, par exemple *Apriori* et *Close*.

Le support et la confiance sont deux mesures associés aux règles d'associations qui sont exploités par les algorithmes d'extraction des règles pour réduire la complexité de l'extraction. Le support d'une règle est donné par le nombre de textes contenant à la fois les termes de A et de B (Ce nombre peut être normalisé par le nombre total de textes). La confiance d'une règle est le rapport entre le nombre de textes contenant l'ensemble des termes $A \bigcap B$ où $A \bigcap B$ désigne l'ensemble des terme-clés qui sont à la fois dans A et B, et le nombre de textes contenant A [Cherfi *et al.*, 2005].

Le clustering de textes consiste à séparer des documents textuels en plusieurs catégories distinctes, selon leur contenu. Chaque catégorie regroupe donc des documents au contenu similaire. Généralement, un document présente des signes d'appartenance plus marqués pour une certaine classe. Par contre, ça ne veut pas dire que le document n'est apparenté à aucune autre catégorie. Il est donc possible d'utiliser une classification stricte, c'est à dire que chaque document n'appartient qu'à une classe, où les documents peuvent faire partie de plus d'une catégorie.

La catégorisation de textes vise à classer un texte dans la bonne catégorie en se basant sur son contenu. Habituellement, les catégories font référence aux sujets des textes, mais pour des applications particulières, elles peuvent prendre d'autres formes. En effet, on peut résoudre, par des techniques de catégorisation, des problèmes tels que l'identification de la langue d'un document, le filtrage de courrier électronique pertinent ou indésirable, ou encore la désambiguïsation de termes. Un autre aspect du problème qui varie selon les applications est la présence ou non d'une contrainte concernant le nombre de catégories assignables à un document donné. Il se peut qu'on désire qu'un même texte ne soit associé qu'à une seule catégorie ou bien on peut permettre que plusieurs catégories accueillent un même document.

1.4 La Catégorisation de texte

1.4.1 Définition

La catégorisation de textes consiste à identifier un lien fonctionnel entre un ensemble de catégories (ou étiquettes ou encore sujets) et un ensemble de textes. Usuellement, ce

lien fonctionnel, plus communément appelé **modèle prédictif**, est estimé à l'aide de méthodes d'apprentissage supervisé. Un ensemble de textes pré-étiquetés, dénommé ensemble d'apprentissage, est nécessaire afin d'estimer les paramètres du modèle le plus efficace, c'est-à-dire celui ayant le plus faible taux d'erreur pour un nombre constant de paramètres [Clech et Zighed, 2004].

Formellement, la Catégorisation de textes consiste en l'attribution d'une valeur booléenne à chaque paire $\langle d_j, c_i \rangle \in D \times C$ où :
D est un domaine des documents, et $C = \{c_1, \ldots, c_{|C|}\}$ est un ensemble de catégories prédéfinies.
Une valeur de T attribuée à la paire $\langle d_j, c_i \rangle$ indique une décision de déposer d_j sous c_i, et une valeur de F indique une décision de ne pas déposer d_j sous c_i. Plus formellement, la tâche consiste à approximer une fonction cible

$$\breve{\Phi} : D \times C \to \{T, F\}$$

(qui décrit la façon dont les documents doivent être classifiés) par le biais d'une fonction

$$\Phi : D \times C \to \{T, F\}$$

appelée le classificateur de telle sorte que $\breve{\Phi}$ et Φ coïncident autant que possible [Sebastiani, 2002].

1.4.2 Applications de la Catégorisation de texte

La catégorisation de texte est actuellement appliquée dans de nombreux et différents contextes : l'indexation de documents basée sur un lexique, le filtrage de documents, la génération automatique de méta-données, la suppression de l'ambigüité du sens des mots, le peuplement des catalogues hiérarchique de ressources Web, et en général toutes les applications nécessitant l'organisation de documents ou le traitement sélectif et l'adaptation de document [Sebastiani, 2002]

1.4.2.1 Indexation automatiques pour les systèmes de Recherche d'Information

Les premières applications concernées étaient l'indexation automatique pour les systèmes de Recherche d'Information(IR) booléens. Les premières recherches dans le domaine ont été effectuées par Borko et Bernick [Borko et Bernick, 1963], Gray et Harley [Gray et Harley, 1971]. A chaque document est attribué un ou plusieurs mots ou expressions clés décrivant son contenu, ces mots et expressions clés appartiennent à un ensemble fini appelé **dictionnaire contrôlé**, souvent composé d'un thésaurus thématique hiérarchique (par exemple, le thésaurus de NASA pour la discipline aéronautique, ou le thésaurus de MESH pour la médecine). Habituellement, cette attribution este effectuée par des indexeurs manuels, et c'est donc une activité coûteuse. Divers classifieurs de texte explicitement conçus pour l'indexation de document sont été décrit dans la littérature [Fuhr et Knorz, 1984], [Robertson et Harding, 1984]

L'indexation automatique utilisant les dictionnaires est étroitement liée à la génération automatique de méta-données. Dans les bibliothèques numériques, nous sommes souvent plus intéressés par le marquage des documents par des méta-données qui les décrivent sous différents aspects (par exemple, date de création, type de document ou le format, disponibilité,...etc.). Le rôle de certaines de ces méta-données est de décrire la sémantique du document de la signification des codes bibliographiques, des mots-clés ou des phrases-clés.

1.4.2.2 Organisation de documents

L'indexation avec un vocabulaire contrôlé est un exemple de la problématique générale d'organisation du document. Le plus souvent, de nombreux autres problèmes relatifs à l'organisation et au classement du document, que ce soit pour des organisations personnelles ou la structuration d'un document de base d'entreprise, peuvent être réglées par les techniques de **TC** [2]. Dans les bureaux d'un journal, par exemple, les annonces doivent être classées dans les catégories telles que les rencontres, voitures à vendre, immobilier,...etc., avant les publications. Les journaux avec un grand nombre d'annonces bénéficieraient d'un système automatique qui pourrait choisir pour une annonce la catégorie donnée la plus appropriée. D'autres applications possibles sont les applications d'organisation des brevets en catégories pour rendre leur recherche plus facile [Larkey, 1999], le classement automatique des articles de journaux sous les sections appropriées (par exemple, la politique, événements, styles de vie,etc.), ou le regroupement automatique en sessions des papiers de conférence [Sebastiani, 2002].

1.4.2.3 Filtrage de Textes

Une autre application des techniques de **TC** est le Filtrage de Textes *(ang : Text Filtering -TF)*. Le Filtrage de Textes est l'activité de classification d'un flux de documents expédiés de manière asynchrone par un producteur d'information à destination d'un consommateur d'information [Belkin et Croft, 1992]. Un cas typique est une situation dans laquelle le producteur est une agence de presse et le consommateur est un journal [Hayes *et al.*, 1990]. Dans ce cas, le système de filtrage doit empêcher la livraison de documents qui n'intéressent pas le consommateur. Le filtrage peut être considéré comme un cas de **TC** de l'étiquetage, c'est la classification des documents en deux catégories disjointes, la catégorie "pertinents"et la catégorie "non pertinents".
En outre, un système de filtrage peut également classer les documents jugés pertinents pour le consommateur en catégories thématiques, en classant par exemple à part les articles de sport pour un journal de sport. Tous les articles de sports devraient être classés en fonction du sport qu'ils traitent, de manière à permettre aux journalistes spécialisés dans des sports individuels d'accéder uniquement aux documents les concernant. De même, un système de filtrage des mails peut filtrer les spam ainsi que classer les messages dans des catégories thématiques pour l'utilisateur [Androutsopoulos *et al.*, 2000]. Un système de filtrage peut être installé chez le producteur d'information, dans ce cas il doit envoyer les documents seulement

2. Text Categorisation

à des consommateurs intéressés, ou chez tous les consommateurs. Dans ce cas il doit bloquer la livraison de documents jugés sans intérêt pour le consommateur. Dans le premier cas, le système construit et met à jour un « profil » pour chaque consommateur [Liddy *et al.*, 1994], alors que dans le dernier cas un seul profil est nécessaire.

1.4.2.4 Word Sense Disambiguation

Les techniques de **TC** permettent également de lever l'ambigüité sur le sens des mots *(ang : Word Sense Disambiguation - **WSD**)*. La **WSD** est l'activité de recherche dans un texte des sens des mots ambigus. Un seul mot peut avoir plusieurs significations. La tâche du système **WSD** est donc de décider de quel des sens il s'agit. La **WSD** est très importante pour de nombreuses applications, y compris le traitement du langage naturel et l'indexation des documents par le sens des mots. La **WSD** peut être considérée comme une tâche de [Gale *et al.*, 1992], [Escudero *et al.*, 2000] si nous considérons le contexte d'occurrence des mots comme un document et le sens du mot comme une catégorie. La **WSD** est juste un exemple du problème plus général consistant à lever les ambiguïtés du langage naturel, un des problèmes les plus importants en linguistique computationnelle.

1.4.3 L'apprentissage automatique

Dans les années 80, l'approche la plus populaire pour la création des classifieurs automatique de documents a consisté à construire manuellement un système expert capable de prendre des décisions de **TC**. Un tel système d'expertise était composé généralement d'un ensemble de règles logiques définies manuellement par une catégorie, du type,

$$\textbf{if } \langle DNF formula \rangle \textbf{ then } \langle category \rangle$$

DNF *(ang : disjonctive forme normale)* est une disjonction de propositions conjonctives. Le document est classé dans la $\langle category \rangle$ si et seulement s'il est en accord avec la formule, donc s'il est en accord avec au moins une des propositions. L'inconvénient de cette approche est que les règles doivent être définies manuellement par un ingénieur des connaissances à l'aide d'un expert du domaine. Si l'ensemble des catégories est mis à jour, ces deux professionnels doivent intervenir à nouveau, et si le classifieur est adapté à un tout autre domaine (c'est-à-dire, ensemble de catégories), des experts d'un domaine différent doivent intervenir et le travail doit être repris à partir de zéro.

Depuis le début des années 90, l'approche de **ML** [3] pour le besoin de **TC** a gagné en popularité et a fini par devenir l'approche dominante [Mitchell, 1999]. Dans cette approche, un processus inductif (également appelé l'apprentissage) construit automatiquement un classifieur pour une catégorie c_i en observant les caractéristiques d'un ensemble de documents classés manuellement pour c_i ou $\bar{c}_i$ par un expert du domaine. De ces caractéristiques le processus inductif tire les caractéristiques que doit avoir le nouveau document pour être classé

3. Machine Learning

dans la catégorie *ci*. Les avantanges de l'approche de **ML** sont évidents. L'effort d'ingénierie va à la construction, non pas d'un classifieur, mais d'un constructeur automatique de classifieurs (l'apprenant).

Cela signifie que tout ce qui est nécessaire est que l'apprenant subisse une construction inductive et automatique d'un classifieur à partir d'une série de documents classifiés manuellement. Dans ce cas, nous n'avons plus besoin de traiter à nouveau des classifieurs qui existent déjà et la série initiale des catégories est mise à jour si le classifieur est porté à un tout autre domaine pour définir les règles manuellement.

Dans l'approche de ML, les documents pré-classifiés sont alors les ressources clés. Dans le cas le plus favorable, ils sont déjà disponibles, ce qui se passe généralement pour les organisations qui ont déjà effectué la catégorisation manuellement de même activité et ont décidé d'automatiser le processus. Le cas le moins favorable est le cas où les documents classés manuellement ne sont pas disponibles, ce qui se passe généralement pour les organisations qui commencent une activité de catégorisation et optent pour un mode automatique. L'approche ML est plus pratique également dans ce dernier cas. Il est en fait plus facile de classer manuellement un ensemble de documents que de construire et de modifier un ensemble de règles, car il est plus facile de caractériser des cas de "celui-ci" que de décrire ce concept en mots, ou de décrire une procédure de reconnaissance des cas.

1.4.3.1 Ensemble d'Apprentissage, de Test et de Validation

L'approche de **ML** repose sur la disponibilité d'un corpus initial

$$\Omega = d_1, \ldots, d_{|\Omega|} \subset D$$

de documents pré-classifiés sous

$$C = \{c_1, \ldots, c_{|C|}\}$$

En d'autres termes, les valeurs de la fonction $\breve{\Phi} : D \times C \to \{T, F\}$ sont connues pour chaque paire $\langle d_j, c_i \rangle$. Un document d_j est un exemple positif de c_i si

$$\breve{\Phi}(d_j, c_i) = T, \tag{1.1}$$

un exemple négatif de c_i si

$$\breve{\Phi}(d_j, c_i) = F. \tag{1.2}$$

Dans les paramètres de recherche, une fois qu'un classifieur Φ a été construit il est souhaitable d'évaluer son efficacité. Dans ce cas, avant la construction du classifieur, le corpus initial est divisé en deux séries, pas nécessairement de taille égale : un ensemble d'apprentissage et un ensemble de test. L'ensemble d'apprentissage est :

$$EA = \{d_1, \ldots, d_{|EA|}\} \tag{1.3}$$

Le classifieur Φ pour les catégories $C = \{c_1, \ldots, c_{|C|}\}$ est construit en observant les caractéristiques de ces documents. L'ensemble de test

$$ET = \{d_{|EA+1|}, \ldots, d_{|\Omega|}\} \tag{1.4}$$

est utilisé pour tester l'efficacité des classifieurs. Chaque $d_j \in ET$ est donné au classifieur, et les décisions du classifieur $\Phi(d_j, c_i)$ sont comparées avec les décisions d'expert $\breve{\Phi}(d_j, c_i)$. Une mesure d'efficacité de la classification est basée sur la fréquence des valeurs de $\Phi(d_j, c_i)$ correspondant aux valeurs de $\breve{\Phi}(d_j, c_i)$.

Les documents de ET ne peuvent pas participer d'une façon quelconque à la construction d'induction du classement. Si cette condition n'était satisfaite, les résultats expérimentaux obtenus seraient probablement trop bons, et l'évaluation n'aurait donc pas de caractère scientifique [Mitchell, 1999]. La validation est une phase indispensable à tout processus d'apprentissage. Elle consiste à vérifier que le modèle construit sur l'ensemble d'apprentissage permet de classer tout individu avec le minimum d'erreurs possible.

Nous citerons trois méthodes de validation généralement utilisées :
- validation par le test,
- validation croisée,
- validation « bootstrap ».

Dans le cas de la validation par le test, les résultats de l'évaluation précédente seraient une estimation pessimiste de la performance réelle, la dernière classification ayant été formée sur plus de données que le classifieur évalué. L'ensemble d'apprentissage permet de générer le modèle, l'ensemble de test permet d'évaluer l'erreur réelle du modèle sur un ensemble indépendant évitant ainsi un biais d'apprentissage. S'il s'agit de tester plusieurs modèles et de les comparer, nous pouvons sélectionner le meilleur modèle selon ses performances sur l'ensemble de validation et ensuite évaluer l'erreur réelle sur l'ensemble de test.

Une alternative est la validation croisée [Mitchell, 1999], dans laquelle k différents classifieurs $\Phi_1, \cdots, \Phi_k$ sont construits par le partitionnement initial du corpus en k ensembles disjoints $ET_1, \cdots, ET_k$ et la validation par test est ensuite appliquée de façon itérative sur les paires $\langle EA_i = \Omega - ET_i, ET_i \rangle$. L'efficacité finale est obtenue par le calcul individuel de l'efficacité de $\Phi_1, \cdots, \Phi_k$. La validation croisée ne construit pas de modèle utilisable, elle estime juste l'erreur réelle. En général le nombre k de parties est fixé à 10.

Une autre alternative est la « validation bootstrap ». Etant donné un échantillon S de taille n, nous tirons avec remise un ensemble d'apprentissage de taille n (un élément de S peut ne pas appartenir a l'ensemble d'apprentissage, ou y figurer plusieurs fois), l'ensemble de test est S

Ces deux dernières méthodes fournissent de bons estimateurs de l'erreur réelle mais sont très couteuses en temps de calcul. Elles sont utiles pour les petits échantillons. Dans ces approches, il arrive souvent que les paramètres internes des classifieurs doivent être réglés

Algorithme 1: Algorithme de la Validation croisée

Données : ET est un ensemble, k est entier;
Découper ET en k parties égales $ET_1, \cdots, ET_k$;
pour $i = 1$ **à** k **faire**
    Construire un modèle M avec l'ensemble $ET - ET_i$;
    Evaluer une mesure d'erreur e_i de M avec ET_i;
fin
retourner *l'espérance mathématique de la mesure des erreurs*

par des essais, pour obtenir les valeurs des paramètres rendant la meilleure efficacité. Afin de rendre possible cette optimisation, dans la validation par test, l'ensemble $\{d_1, \cdots, d_{|EA|}\}$ est en outre divisé en un sous ensemble d'apprentissage $EP = \{d_1, \cdots, d_{|EP|}\}$ à partir duquel le classifieur est construit, et un sous ensemble de validation $EV = \{d_{|EP|+1}, \cdots, d_{|EA|}\}$, sur lequel la répétition des tests du classifieur pour l'optimisation des paramètres est effectuée. La variante peut être utilisée dans la validation croisée. Evidement, pour la même raison que nous n'avons pas testé un classifieur sur les documents sur lesquels il a été formé, nous ne pouvons pas le tester sur les documents sur lesquels il a été optimisé. L'ensemble de test et de validation doivent être séparés.

1.4.4 Représentation des corpus documentaires

1.4.4.1 L'unité linguistique

Les textes en langage naturel ne peuvent pas être directement interprétés par un classifieur ou par les algorithmes de classification. Le sens d'un document peut être porté par un ensemble d'unités linguistiques particulières, aux caractéristiques plus ou moins élaborées issues de l'analyse du corpus documentaire. Les premières unités linguistiques qui représentent du sens sont les lemmes des mots. La reconnaissance de ces unités linguistiques nécessite d'effectuer un pré-traitement linguistique des mots du texte. L'unité linguistique peut être représentée par le mot ou la phrase. Dans le premier cas l'unité linguistique est le mot tel qu'il apparait dans le document. Chaque mot est extrait du texte en considérant des séparateurs comme l'espace, la tabulation, et la ponctuation. Le nombre de mots caractérisant un corpus de documents peut être très grand, il est donc nécessaire de conserver un sous ensemble de ces mots. Ce filtrage repose à la base sur les fréquences d'occurrences des mots dans le corpus.

D'autres approches utilisent non pas des mots, mais des groupes de mots, voir des phrases comme l'unité linguistique décrivant le sens. Ce type d'unité linguistique décrit le sens plus complètement qu'un simple mot [Lewis, 1992]. De plus, grâce à cette approche nous avons une relation d'ordre entre les mots, les co-occurrences de mots. L'inconvénient est que la fréquence d'apparition des groupes de mots ne permet pas d'offrir des statistiques fiables, car le grand nombre de combinaisons entre les mots engendre des fréquences trop faibles pour être exploitables.

Une autre approche pour représenter le corpus documentaire est l'utilisation de la technique des n-grammes [Shannon, 1948]. En général le n-gramme se définit comme étant une séquence de n caractères consécutifs. Le principe des n-gramme est que pour une chaine de k caractères entourée de blancs, nous générons $k+1$ n-grammes [Cavnar et Trenkle, 1994]. Un exemple de découpage pour le mot porte en bi-gramme ($n = 2$) est le suivant :
`_porte_ : _p, po , or, rt, te, e_`

Dès qu'on extrait tous les n-gramme d'un document on défini la liste des n-grammes triés par ordre décroissant de leur fréquence d'apparition. Ces méthodes sont indépendantes de la langue et ni la segmentation en unités linguistiques, ni des pré-traitements comme le filtrage et la lemmatisation ne sont nécessaires.

1.4.4.2 Prétraitement du texte

Si nous utilisons de mots comme unité linguistique nous remarquons que plusieurs mots ont des sens communs ou forment simplement une autre forme de conjugaison. Attribuer un sens différent à ces mots relèverait d'une redondance sans pertinence sémantique. Pour cette raison un traitement appelé stemmatisation (*ang : stemming*) est à effectuer. Elle consiste à retrouver la racine de chaque mot. C'est un traitement qui procède à une analyse morphologique du texte [Porter, 1980]. Ce traitement est basé sur un dictionnaire de suffixes qui permet d'extraire le radical du mot grâce à l'étude morphologique des mots.

Le traitement qui demande une analyse plus complexe que la stemmatisation est la lemmatisation qui est fondée sur un lexique. Un lexique est un ensemble de lemmes avec lequel nous pouvons faire référence au dictionnaire. L'objectif de la lemmatisation est d'associer à chaque mot une entrée dans le lexique. Comme de nombreux mots de même graphie peuvent provenir de différents lemmes, l'analyse morphologique est insuffisante. La lemmatisation nécessite donc de réaliser en plus une analyse syntaxique pour résoudre les ambigüités, elle effectue donc une analyse morpho-syntaxique [Schmid, 1994].

Avant d'effectuer un des pré-traitements précédents, il est usuel d'utiliser une « stop-list » pour éliminer tout les mots qui ne participent pas activement au sens du document. Elle contient les pronoms, les articles et les mots trop fréquents pour être discriminants, nous éliminons donc toutes les unités linguistiques non discriminantes. Le risque de la stop-list est que l'on peut éliminer des mots qui pourraient être utiles pour la classification.

1.4.4.3 L'indexation des documents et la réduction de dimension

Une procédure d'indexation du texte d_j en une représentation compacte de son contenu doit être appliquée uniformément aux documents d'apprentissage, de validation et de tests. Le choix d'une représentation du texte dépend de ce quel'on considère comme étant les unités linguistique du sens du texte (le problème de sémantique lexicale). Les approches de l'indexation sont partagées en :

– celles qui étudient les différents moyens de comprendre ce qu'est une unité linguistique,

– celles qui se basent sur différentes manières de calculer les poids des unités.

Le poids des unités varie le plus souvent entre 0 et 1. Dans un cas spécial le poids peut être binaire (1 indique la présence et 0 l'absence du terme dans le document). Dans le cas d'indexation non binaire, pour déterminer le poids de w_{kj} de l'unité t_k dans le document d_j, toutes les techniques d'indexation de IR qui représentent un document comme un vecteur de termes pondérés peuvent être utilisées. La plupart du temps, la fonction $tf * idf$ est utilisée [Salton et Buckley, 1988], définie comme suit :

$$tfidf\left(t_k, d_j\right) = \#\left(t_k, d_j\right) \cdot \log \frac{|T_r|}{\#_{T_r}\left(t_k\right)} \tag{1.5}$$

où $\#\left(t_k, d_j\right)$ désigne le nombre de fois que t_k se produit en d_j ,et $\#_{T_r}\left(t_k\right)$ indique la fréquence du document de terme t_k c'est à dire : le nombre de documents d'ensemble d'apprentissage (EP) dans lesquels t_k se produit. Cette fonction montre que, plus un terme apparait souvent dans un document, plus il est représentatif ; plus le nombre de documents contenant un terme est important moins ce terme est discriminatoire. Pour que le poids prenne des valeurs dans l'intervalle [0 , 1] le poids résultant de $tfidf$ est souvent normalisé :

$$w_{k,j} = \frac{tfidf\left(t_k, d_j\right)}{\sqrt{\sum_{s=1}^{|T_r|}\left(tfidf\left(t_k, d_j\right)\right)^2}} \tag{1.6}$$

Le rôle de la représentation textuelle est représenté mathématiquement de façon à ce que l'on puisse effectuer le traitement analytique, tout en conservant au maximum la sémantique. La représentation mathématique généralement utilisée est l'utilisation d'un espace vectoriel comme espace de représentation cible. La caractéristique principale de la représentation vectorielle est que chaque unité linguistique est associée à une dimension propre au sein de l'espace vectoriel. Deux textes utilisant les mêmes segments textuels seront donc projetés sur des vecteurs identiques. Le formalisme le plus utilisé pour représenter les textes est le formalisme vectoriel de Salton [Salton et Mcgill, 1983].

Le texte d_j est représenté comme vecteur des unités linguistique $\vec{d}_j = \left[w_{1,j}, \cdots, w_{|T|,j}\right]$, où T est l'ensemble des unités qui se trouvent au moins une fois dans au moins un document d'apprentissage (EP), et $0 \leq w_{i,j} \leq 1$ représente combien de fois une unité est présente dans le document d_j. Dans cette approche, chaque dimension de l'espace vectoriel correspond à un élément textuel, nommé terme d'indexation, préalablement extrait par calcul selon plusieurs méthodes proposées par Salton [Salton et Mcgill, 1983], [Salton et Buckley, 1988].

Cette action de sélection fait appel à un premier processus d'indexation. Salton a proposé plusieurs versions du processus de pré-traitement des unités linguistiques décrites précédemment. Dans un premier temps ce traitement était une stemmatisation. Ensuite, avec l'évolution des techniques informatiques, de nombreuses solutions sont apparues pour l'analyse morpho-syntaxique des textes. Le processus d'indexation lui-même consiste à effectuer un simple inventaire complet de tous les lemmes du corpus. Ensuite, vient le processus de sélection des lemmes qui vont constituer les unités linguistiques du domaine ou les dimensions

de l'espace vectoriel de représentation du corpus documentaire. Dans ce formalisme, chaque dimension de l'espace vectoriel correspond à un segment textuel, préalablement extrait du corpus documentaire. Une dimension est un mot, un paragraphe, une lettre ou un assemblage de lettres. Les unités linguistiques sont choisies dans l'ensemble des lemmes en fonction de leur pouvoir de discrimination. Elles constituent les termes d'indexation.

La sélection des termes d'indexation correspond donc à une réduction de la dimension de l'espace effectuée en fonction de critères de discrimination. Le critère de sélection des termes d'indexation le plus utilisé est la fréquence en documents IDF (*ang : Inverse Document Frequency*). Il consiste à calculer le nombre de documents dans lesquels apparaît un lemme puis de prendre l'inverse de ce nombre. Pour une collection de documents D, la sélection des unités linguistiques qui ont une fréquence en documents entre $D/100$ et $D/10$ génère le plus souvent un ensemble d'unités linguistiques ayant un pouvoir de discrimination satisfaisant pour la recherche documentaire [Salton *et al.*, 1975]. Un lemme l_i constitue une dimension de l'espace vectoriel de représentation du corpus de textes. Un texte ou document D est représenté par un vecteur $D = \left(ft_1, \cdots, ft_i, \cdots, ft_{|V|}\right)$ dans lequel V représente le vocabulaire ou l'ensemble des unités linguistiques sélectionnées et ft_i le nombre d'occurrences du lemme l_i dans le document D.

Une variante du modèle vectoriel standard est le modèle LSI (*ang : Latent Semantic Indexing*) qui prend en compte la structure sémantique des unités linguistiques [Deerwester *et al.*, 1990]. LSI utilise la matrice du modèle vectoriel standard, dans laquelle chaque élément $x_{i,j}$, où j est le document et i l'unité linguistique, est le nombre d'occurrences de l'unité linguistique u_i dans le document d_j.
Une décomposition en valeurs singulières SVD (*ang : Singular Value Decomposition*) de cette matrice est effectuée et seuls les premiers vecteurs propres sont pris en compte.
Nous appelons M la matrice où l'élément (i, j) décrit l'occurrence d'unités linguistiques i dans le document j. La ligne de cette matrice représente le vecteur correspondant à l'unité linguistique en précisant sa relation avec chaque document :

$$t_i^T = [x_{i,1}, \cdots, x_{i,n}]$$

La colonne de cette matrice représente le vecteur correspondent au document en précisant sa relation avec chaque unité linguistique :

$$d_j = \begin{vmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{vmatrix}$$

Le produit scalaire $t_i^T t_p$ entre deux vecteurs de l'unité linguistique montre la corrélation entre les unités dans les documents. Le produit scalaire des matrices MM^T contient tous ces produits scalaires des unités linguistiques. La décomposition de la matrice M est représentée par les matrices U et V -les matrices orthogonales et par la matrice Σ -la matrice diagonale

est de la forme :

$$M = U\Sigma V^T$$

Nous approximons la matrice Σ par la matrice réduite $\hat{\Sigma}$, pour représenter les documents dans un espace réduit de dimension. Chacune des dimensions de l'espace de représentation final correspond à une combinaison linéaire des unités linguistiques. L'espace de représentation n'a donc pas pour support un ensemble de termes d'indexation, ce qui rend les dimensions relativement difficiles à interpréter directement. Ce modèle permet de représenter les termes par des vecteurs qui sont une indication du profil d'occurrence du terme dans les documents. Cette propriété peut donc être utilisée pour établir une notion de similarité entre termes, ou représenter un document comme la moyenne des vecteurs représentant les termes qu'il contient.

1.4.5 Les techniques de classification

La construction des classifieurs du texte a été abordée de diverses manières. Ici, nous présentons seulement les méthodes qui ont été les plus populaires dans le domaine de la catégorisation du texte (TC). La classification, appelée également *induction supervisée*, consiste à analyser de nouveaux candidats et à les affecter, en fonction de leurs caractéristiques ou attributs, à telle ou telle classe prédéfinie. Les méthodes de classification ont pour but d'identifier les classes auxquelles appartiennent des objets à partir de certains traits descriptifs. La classification fournit de l'aide à la prise de décision comme par exemple pour établir un diagnostic médical à partir de la description clinique d'un patient, pour donner une réponse à la demande de prêt bancaire de la part d'un client sur la base de sa situation personnelle ou pour déclencher un processus d'alerte en fonction de signaux reçus par des capteurs.

La construction inductive d'un classifieur de classement pour une catégorie $c_i \in C$ consiste habituellement en la définition d'une fonction telle pour un document d_j, qu'elle retourne une valeur indiquant l'état de catégorisation qui représente le fait que $d_j \in c_i$. Cette fonction de l'état de catégorisation CSV (*ang : Categorization Status Value*) prend des valeurs entre 0 et 1. Les documents sont ensuite classés en fonction de leur valeur CSV_i. La fonction CSV_i prend différentes formes selon la méthode d'apprentissage utilisée : par exemple, dans l'approche de « naive bayes » $CSV_i(d_j)$ est définie en termes de probabilité.

La construction d'un classifieur peut consister en la définition d'une fonction $CSV_i : D \rightarrow \{T, F\}$ ou d'une fonction $CSV_i : D \rightarrow [0; 1]$. Un paramètre de seuil τ *(ang : threshold)* est défini tel que $CSV_i(d_j) \geq \tau$ est interprété comme $Vrai - T$ et $CSV_i(d_j) > \tau$ est interprété comme $Faux - F$.

Le paramètre de seuil peut être calculé *analytiquement* ou *expérimentalement*. La méthode analytique est possible uniquement en présence de résultat théorique qui indique comment calculer le seuil pour maximiser l'efficacité [Lewis, 1995]. C'est le cas pour les classifieurs de probabilité. Quand un tel résultat théorique n'est pas connu, il faut revenir

à la méthode expérimentale, qui consiste à tester différentes valeurs de τ_i sur un ensemble de documents de validation (EV) et de choisir la valeur qui maximise l'efficacité [Cohen et Singer, 1996], [Yang, 1999].

La procédure de classification est générée automatiquement à partir d'un ensemble d'exemples. Un exemple consiste en la description d'un cas avec la classification correspondante. Nous disposons, par exemple, d'une base des symptômes des patients avec la situation de leur état de santé respectif, et le diagnostic de maladie. Un système d'apprentissage doit alors, à partir de cet ensemble d'exemples, extraire une procédure de classification qui, au vu des symptômes d'un patient, devra établir un diagnostic médical. Il s'agit donc d'induire une procédure de classification générale à partir d'exemples. Le problème est donc un problème inductif, il s'agit d'extraire une règle générale à partir de données observées. La procédure générée devra classifier correctement les exemples de l'échantillon mais surtout avoir un bon pouvoir prédictif pour classifier correctement de nouvelles descriptions. Les méthodes statistiques supposent que les descriptions des objets d'une même classe se répartissent en respectant une structure spécifique à la classe. Les méthodes d'apprentissage à partir d'exemples sont très utilisées dans la recherche d'informations dans de grands ensembles de données.

1.4.5.1 Classifieur de Bayes

Les classifieurs probabilistes interprètent la fonction $CSV_i\left(d_j\right)$ en termes de $P\left(c_i|\vec{d_j}\right)$, ce qui représente la probabilité qu'un document représenté par un vecteur $\vec{d_j} = \left(w_{1,j}, \cdots, w_{|T|,j}\right)$ de termes qui appartient à c_i,et calcule cette probabilité en utilisant le théorème de Bayes, définie par :

$$P\left(c_i|\vec{d_j}\right) = \frac{P\left(c_i\right) P\left(\vec{d_j}|c_i\right)}{P\left(\vec{d_j}\right)} \tag{1.7}$$

où $P\left(\vec{d_j}\right)$ est la probabilité qu'un document choisi au hasard ait le vecteur $\vec{d_j}$ comme représentation,et $P\left(c_i\right)$ est la probabilité qu'un document choisi au hasard appartienne à c_i. L'estimation de probabilité $P\left(c_i|\vec{d_j}\right)$ est problématique, puisque le nombre de vecteurs $\vec{d_j}$ possibles est trop élevé. Pour cette raison il est courant de faire l'hypothèse que toutes les coordonnées du vecteur du document sont statistiquement indépendantes. Donc :

$$P\left(\vec{d_j}|c_i\right) = \prod_{k=1}^{|T|} P\left(w_k j|c_i\right) \tag{1.8}$$

Les classifieurs probabilistes qui utilisent cette hypothèse sont appelés les classifieurs de « Bayes naïf », et trouvent utilisation dans la plupart des approches probabilistes dans le domaine de catégorisation du texte [Wang *et al.*, 2003]. Le caractère naïve du classifieur est dû au fait qu'en général, cette hypothèse n'est pas vérifiée dans la pratique.

1.4.5.2 Classifieur par la méthode des SVM

Les méthodes des SVM (ang : Support Vector Machine) a été introduite par [Cortes et Vapnik, 1995]. La méthode des SVM géométriques peut être considérée comme la tentative de trouver, parmi toutes les surfaces $\sigma_1, \sigma_2, \cdots, \sigma_n$ d'un espace de dimensions $|T|$ ce qui sépare les exemples d'apprentissage positifs des négatifs. L'ensemble d'apprentissage est donné par un ensemble de vecteurs associés à leur classe d'appartenance : $(X_1, y_1), \cdots, (X_u, y_u), X_j \in R^n, y_j \in \{+1, -1\}$ avec

- y_j représente la classe d'appartenance. Dans un problème à deux classes la première classe correspond à une réponse positive ($y_j = +1$) et la deuxième classe correspond à une réponse négative ($y_j = -1$)
- X_j représente le vecteur du texte numéro j de l'ensemble d'apprentissage.

La méthode SVM sépare les vecteurs à classe positive des vecteurs à classe négative par un hyperplan 1.3 défini par l'équation suivante :

$$W \otimes X + b = 0, W \in R^n, b \in R \tag{1.9}$$

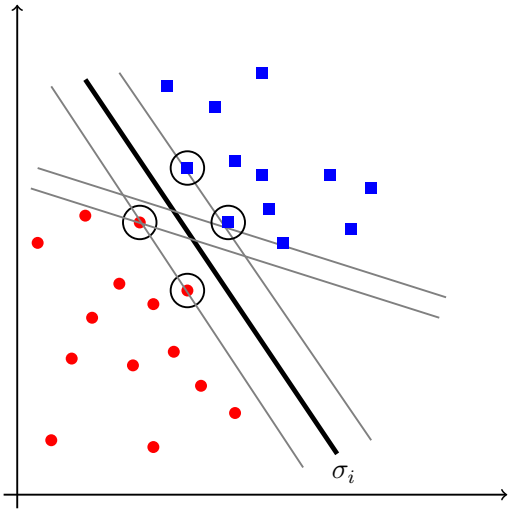

Figure 1.3 – Apprentissage du classifieur SVM. Les cercles et les carrés représentent respectivement les réponses positives et négatives, les lignes représentent les surfaces de décision. La surface de décision σ_i montre le meilleur cas.

En général, un tel hyperplan n'est pas unique. La méthode SVM détermine l'hyperplan optimal en maximisant la marge : la marge est la distance entre les vecteurs étiquetés positifs et les vecteurs étiquetés négatifs. L'ensemble d'apprentissage n'est pas nécessairement séparable linéairement, des variables d'écart ξ_j sont introduites pour tous les X_j. Ces ξ_j prennent en compte l'erreur de classification, et doivent satisfaire les inégalités suivantes :

$$W \otimes X + b \geq 1 - \xi_j \qquad \text{et} \qquad W \otimes X + b \leq 1 + \xi_j \tag{1.10}$$

En prenant en compte ces contraintes, nous devons minimiser la fonction d'objectif suivante :

$$\frac{1}{2}\|W\|^2 + C\sum_{j=1}^{u}\xi_j \tag{1.11}$$

Le premier terme de cette fonction correspond à la taille de la marge et le second terme représente l'erreur de classification, avec u représentant le nombre de vecteurs de l'ensemble d'apprentissage. Trouver la fonction objective précédente revient à résoudre le problème quadratique suivant : trouver la fonction de décision f telle que : $f(X) = Signe(g(X))$ dans laquelle la fonction $g(X)$ est :

$$g(X) = \sum_{i=1}^{m}\lambda_i y_i X_i \otimes X + b \tag{1.12}$$

avec :

- $Signe(x)$ représente la fonction suivante : $Signe(x) = \begin{cases} 1 & \text{si } x > 0 \\ -1 & \text{si } x > 0 \\ 0 & \text{si } x = 0 \end{cases}$
- y_i représente la classe d'appartenance,
- λ_i représente les paramètres à trouver,
- $X_i \otimes X$ représente le produit scalaire du vecteur X_i avec le vecteur X.

De plus, les SVM s'étendent très élégamment pour construire des modèles non linéaires en enrichissant l'espace de représentation. Pour cela, on remarque que, dans la formule précédente, seul intervient le produit scalaire entre deux points [wei Hsu *et al.*, 2003]. On peut utiliser comme produit scalaire, toute fonction noyau symétrique $K(X_i, X)$:

$$K(X_i, X) = \Phi^T(X_i)\Phi(X) \tag{1.13}$$

- linéaire : $K(X_i, X) = X_i^T X$
- polynomial :$K(X_i, X) = (\gamma X_i^T X + r)^d, \quad \gamma > 0$
- gaussien à base radiale (RBF) : $K(X_i, X) = exp(-\gamma(\|X_i - X\|)^2), \quad \gamma > 0$
- sigmoïdal : $K(X_i, X) = tanh(\gamma X_i^T X + r)$

où : γ , r et d sont des paramètres du noyau.

1.4.5.3 Classifieur par la méthode des arbres de décision

Un classifieur de texte basé sur la méthode d'arbre de décision est un arbre de nœuds internes qui sont marqués par des termes, les branches qui sortent des nœuds sont des tests sur les termes, et les feuilles sont marquées par catégories [Mitchell, 1999]. Ce classifieur classe un document du test d_j en testant récursivement les poids des nœuds internes de vecteur $\vec{d_j}$, jusqu'à ce qu'une feuille soit atteinte. L'étiquette de ce nœud est alors attribuée à d_j. La plupart de ces classifieurs utilise une représentation du document binaire, et sont donc crée par des arbres binaires 1.4.

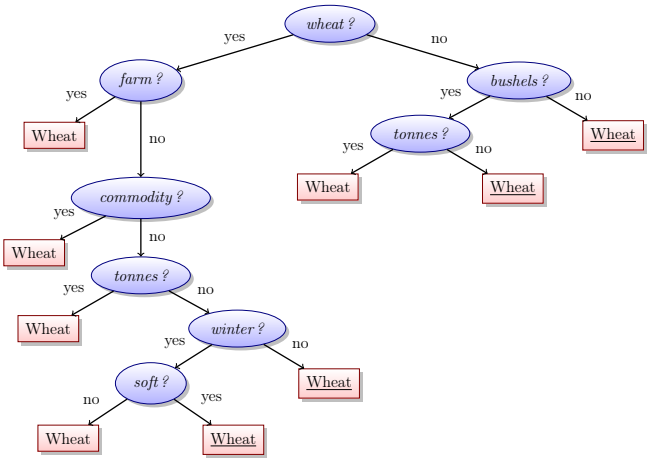

Figure 1.4 – Exemple d'arbre de décision appliquée sur le corpus Reuters

Il existe un certain nombre d'approches pour l'apprentissage de l'arbre de décision. Les plus populaires sont ID3 (utilisé par Fuhr [Fuhr *et al.*, 1991]), C4.5 (Cohen et Singer [Cohen et Singer, 1996]), et C5 (utilisé par Liet Jain [Li et Jain, 1998]).

Une méthode pour effectuer l'apprentissage d'un arbre de décision pour la catégorie c_i consiste à vérifier si tous les exemples d'apprentissage ont la même étiquette (c_i ou $\bar{c_i}$), dans le cas contraire nous sélectionnons un terme t_k, et nous partitionnons l'ensemble d'apprentissage en classes de documents qui ont la même valeur pour t_k, et à la fin l'on crée les sous-arbres pour chacune de ces classes. Ce processus est répété récursivement sur les sous-arbres jusqu'à ce que chaque feuille de l'arbre généré de cette façon contienne des exemples d'apprentissage attribués à la même catégorie c_i, qui est alors choisie comme l'étiquette de la feuille.
L'étape le plus important est le choix du terme de t_k pour effectuer la partition. Toutefois, une telle méthode de construction d'arbre peut faire l'objet de sur-apprentissage, comme certaines branches peuvent être trop spécifiques pour les données d'apprentissage. La plupart des méthodes d'apprentissage des arbres incluent une méthode pour la construction d'arbre et pour élaguer les branches trop spécifiques [Mitchell, 1999].

Algorithme 2: Classification d'une donnée dans un arbre de décision

Données : AD est arbre de décision, x est exemple;
$nc \leftarrow$ racine (AD);
tant que $nc \neq$ *feuille* **faire**
    en fonction de l'attribut testé dans nc et de sa valeur dans x, suivre l'une des branches de nc;
    $nc \leftarrow$ nœud atteint;
fin
retourner *étiquette (nc)*

1.4.5.4 Réseau de neurones

Un classifieur de texte basé sur les réseaux de neurones (*ang : neural networks NN*) est un réseau d'unités, où les unités d'entrée représentent les termes, l'unité (s) de sortie représentent la catégorie ou les catégories d'intérêts, et le poids sur les bords reliant les unités représentent les relations de dépendance. Pour classer un document de test d_j, ses poids w_{kj} sont chargés dans les unités d'entrée, l'activation de ces unités se propage à travers le réseau, et la valeur de l'unité de sortie (s) détermine la décision du classement. Une manière typique d'apprentissage de réseau de neurones est la rétro-propagation, qui consiste à rétro-propager l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés. Pour les réseaux de neurones, on utilise habituellement la rétro-propagation du gradient de l'erreur, qui consiste à corriger les erreurs selon l'importance des éléments qui ont justement participé à la réalisation de ces erreurs.
Le type le plus simple de classifieur de NN est le perceptron [Dagan *et al.*, 1997], qui est un classifieur linéaire. Dans cet algorithme, le classifieur de c_i est d'abord initialisé par la mise de tous les poids w_{kj} à la même valeur positive. Quand un exemple d'apprentissage d_j (représenté par un vecteur $\vec{d_j}$ de poids binaire) est examiné, et si le résultat de la classification est correct, rien n'est fait, alors que si le résultat est faux, les poids du classifieur sont modifié.

- si d_j est un exemple positif de c_i, le poids de w_{ki} des termes actifs (c'est-à-dire, les termes t_k tels que $w_{kj} = 1$) sont « promus » par augmentation d'une valeur $\alpha > 0$ (appelé taux d'apprentissage),
- si d_j est un exemple négatif de c_i les mêmes poids sont « rétrogradés » en diminuant leur valeur par α.

Lorsque le classifieur a atteint un niveau raisonnable d'efficacité, le fait qu'un poids w_{ki} est très faible signifie que t_k a contribué négativement à la procédure de classement, il peut donc être éliminé de la représentation. Le classifieur perceptron a montré une bonne efficacité [Dagan *et al.*, 1997]

Un réseau de neurones est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche est composée de n_i neurones, prenant leurs entrées sur les n_{i-1} neurones de la couche précédente. chaque synapse est associé un poids synaptique, de sorte que les n_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau i, ce qui est équivalent à multiplier le vecteur d'entrée par une matrice de transformation. Le réseau de neurones peut également contenir des boucles qui en changent radicalement les possibilités mais aussi la complexité. De la même façon que des boucles peuvent transformer une logique combinatoire en logique séquentielle, les boucles dans un réseau de neurones transforment un simple dispositif de reconnaissance d'entrées en une machine complexe capable de toutes sortes de comportements. La représentation schématique d'un neurone artificiel avec un index j est montre sur la figure 1.5.

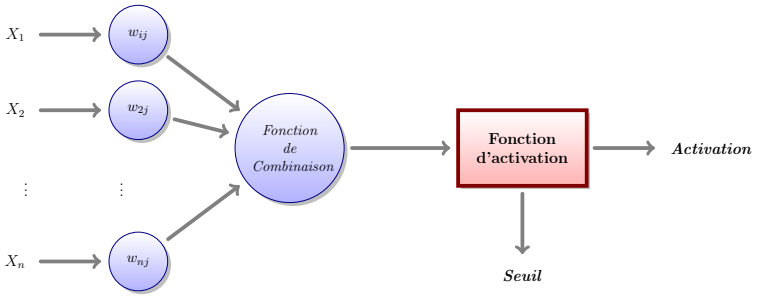

Figure 1.5 – Structure d'un neurone artificiel : Le neurone calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie.

1.4.6 Evaluation de classifieurs de textes

1.4.6.1 Mesures de d'éfficacité

L'évaluation du classifieur des documents est effectuée de façon expérimentale, plutôt que de façon analytique. Le traitement est expérimental car pour évaluer un système de façon analytique,donc pour prouver que le système est correct et complet, nous aurions besoin d'une spécification formelle du problème que le système tente de résoudre. La notion de systèmes de catégorisation de texte est, en raison de son caractère subjectif, par nature non-formalizable comme il s'agissait de déterminer l'appartenance d'un document à une catégorie. L'efficacité de classification se mesure généralement par les paramètres classiques du domaine de la recherche d'information :

- la précision,
- le rappel.

La *Précision* π_i est définie comme la probabilité conditionnelle $P\left(\breve{\Phi}(d_x, c_i) = T | \Phi(d_x, c_i) = T\right)$ qui signifie la probabilité que, si d_x est classé dans c_i, cette décision est correcte. La précision c'est un ratio entre le nombre de documents pertinents trouvés et le nombre total de documents trouvés. Elle mesure le bruit, et plus elle est proche de 100%, moins il y a de bruit, et donc meilleure est la réponse.

De même, le *rappel* ρ_i est défini comme $P\left(\Phi(d_x, c_i) = T | \breve{\Phi}(d_x, c_i) = T\right)$, qui signifie la probabilité que, si un document quelconque d_x devait être classé sous c_i, cette décision était prise. Le rappel est un ratio entre le nombre de documents pertinents trouvés et le nombre de documents pertinents présents dans la base. Plus il est proche de 100%,moins il y a de silence, et meilleure est la réponse.

La précision peut être considérée comme le « degré de solidité » du classifieur, tandis que le rappel peut être considéré comme son « degré d'exhaustivité ». L'évaluation de la pertinence dans les recherches documentaires essaie de quantifier le rappel et la précision, avec les indices de bruit et de silence (Figure 1.6).

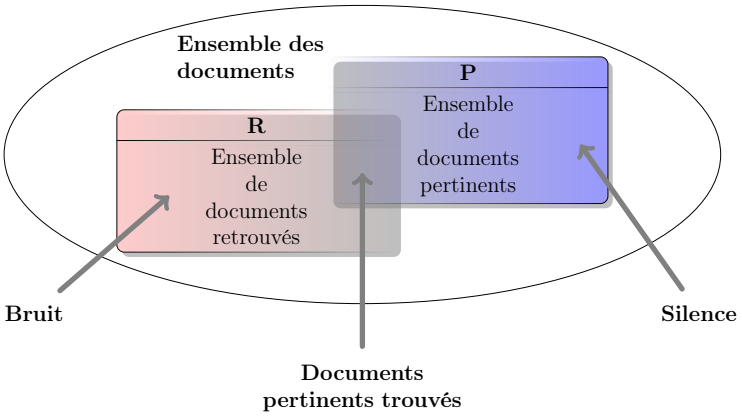

Figure 1.6 – Exemple de représentation du bruit et du silence :Le rôle de la pertinence et du rappel dans la définition du bruit et silence.

Catégorie c_i		Classement de l'Expert	
		Vrai	**Faux**
Jugement du Classifieur	**Positif**	TP_i	FP_i
	Négatif	FN_i	TN_i

Table 1.1 – Possibilité de résultat du classifieur pour une catégorie c_i

Le bruit est un ratio entre le nombre de documents ramenés en réponse, mais qui ne sont pas pertinents par rapport à la question posée, et le nombre total de documents. Donc :

$$Bruit = 1 - Prcision$$

Le silence est un ratio entre le nombre de documents pertinents qui n'apparaissent pas dans le résultat de la recherche et le nombre total de documents. Donc :

$$Silence = 1 - Rappel$$

Selon cette définition, la précision et le rappel peuvent être définis comme les probabilités subjectives, qui mesurent l'attente de l'utilisateur d'un système se comportant correctement lors de la classification d'un document inconnu pour une catégorie c_i. Ces probabilités peuvent être estimées par un tableau qui montre toutes les possibilités d'un classifieur [Tableau 1.1].

- FP_i (erreurs de commission) est le nombre de documents de test mal classés dans la catégorie c_i ,
- TN_i est l'ensemble de documents bien classés qui n'appartiennent pas à la catégorie c_i,
- TP_i est l'ensemble de documents bien classés qui appartiennent à la catégorie c_i,et
- FN_i (erreurs d'omission) est l'ensemble des documents de catégorie c_i non classés par le classifieur.

Catégories $C = \{c_i, \cdots, c_{\|C\|}\}$		Classement de l'Expert	
		Vrai	**Faux**
Jugement du Classifieur	**Positif**	$TP = \sum_{i=1}^{\|C\|} TP_i$	$FP = \sum_{i=1}^{\|C\|} FP_i$
	Négatif	$FN = \sum_{i=1}^{\|C\|} FN_i$	$TN = \sum_{i=1}^{\|C\|} TN_i$

Table 1.2 – Table de contingence globale

La précision et le rappel peuvent être exprimés localement de la façon suivante :

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \tag{1.14}$$

$$\rho_i = \frac{TP_i}{TP_i + FN_i} \tag{1.15}$$

Deux approches sont adoptées pour exprimer globalement la précision et le rappel :

- donner un poids égal à toutes les classes, on parle alors de « macro-moyenne »
- donner un poids proportionnel à la fréquence de chaque classe, il s'agit de la « micro-moyenne »

La micro-moyenne *(ang : micro-averaging)* calcule les mesures rappel et précision de façon globale : si l'on considère les tables de contingences associées à chaque catégorie, cela revient à sommer les cases VP et FP de chaque catégorie pour obtenir la table de contingence globale (voir la table 1.2). Les différentes mesures comme le π et ρ sont ensuite calculées à partir des valeurs cumulées. La micro-moyenne accorde donc des poids importants aux catégories ayant beaucoup d'exemples. La performance du classifieur dépend surtout de sa capacité à traiter les catégories les plus fréquentes [Sebastiani, 2002]. Ainsi,la précision micro-moyenne et le rappel micro-moyenne sont estimés comme suit :

$$\pi^{\mu} = \frac{TP}{TP + FP} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FP_i)} \tag{1.16}$$

$$\rho^{\mu} = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} (TP_i + FN_i)} \tag{1.17}$$

La macro-moyenne *(ang : macro-averaging)* évalue d'abord indépendamment chaque catégorie. Ensuite, la performance globale du classifieur est calculée en faisant la moyenne des mesures individuelles. Les différentes catégorie sont alors la même importance. La précision et le rappel macro-moyenne sont calculés comme suit :

$$\pi^{M} = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|} \tag{1.18}$$

$$\rho^{M} = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|} \tag{1.19}$$

Les mesures alternatives de précision et de rappel couramment utilisées dans la littérature, telles que taux de succès et le taux d'erreur. Le taux de succès et le taux d'erreur sont deux mesures souvent utilisés par la communauté de l'apprentissage automatique. Le taux de succès *(ang : accuracy rate)* désigne le pourcentage d'exemples bien classés par le classifieur, tandis que le taux d'erreur *(ang : error rate)* désigne le pourcentage d'exemples mal classés. Les deux taux sont estimés comme suit :

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.20}$$

$$E = \frac{FP + FN}{TP + TN + FP + FN} = 1 - A \tag{1.21}$$

Ni le rappel ni la précision ne donnent du sens séparément l'un de l'autre. En fait, le classifieur Φ tel que $\Phi(d_j, c_i) = T$ pour tous les d_j et c_i (l'accepteur trivial) a le rappel $\rho = 1$. Lorsque la fonction CSV_i a une valeur dans l'intervalle $[0, 1]$,il suffit de mettre chaque seuil à 0 pour obtenir l'accepteur trivial. Dans ce cas la précision est généralement très faible. En revanche il est bien connu que des niveaux plus élevés de la précision peuvent être obtenus au prix de faibles valeurs du rappel. Donc une classification doit être évalué en combinant des valeurs de précision et de rappel. Plusieurs mesures ont été proposées, parmi les quelles les plus fréquentes sont les suivants :

- seuil de rentabilité *(ang :breakeven)*,
- fonction F_β,
- fonction $Fscore$.

La première mesure -le seuil de rentabilité *(ang : breakeven)*, est la valeur à laquelle la précision est égale au rappel [Joachims, 1999] , [Yang, 1999]. Ce résultat est obtenu par le tracé de la précision en fonction du rappel pour divers seuils ; le seuil de rentabilité est la valeur de la précision ou de rappel pour lesquelles la courbe coupe la ligne $\rho = \pi$ Cette idée repose sur le fait que, en diminuant le seuil de 1 à 0, le rappel augmente toujours de 0 à 1et la précision diminue monotonement d'une valeur proche de 1.
La deuxième mesure est la fonction F_β [van Rijsbergen, 1979], pour certains $\beta \in (0, \infty)$ qui est telle que :

$$F_\beta = \frac{(\beta^2 + 1)\pi\rho}{\beta^2\pi + \rho}$$

F_β peut être considéré comme le degré relatif d'importance attribué à la précision et au rappel. Ce coefficient indique le poids que l'on souhaite affecter à la précision par rapport au rappel. Habituellement, le coefficient β prend la valeur 1. Dans ce cas la fonction F_β est appelée $Fscore$ et prend la forme :

$$F_1 = \frac{2\pi\rho}{\pi + \rho}$$

1.4.6.2 Quel est le meilleur classifieur ?

Beaucoup d'approches différentes ont été utilisées pour la catégorisation de textes. La question qui se pose est : quelle est la meilleure méthode pour la catégorisation de textes ?

Idéalement, pour comparer les performances de deux méthodes, on tente soit d'appliquer les deux méthodes sur les mêmes données en utilisant les mêmes mesures de performance, soit d'adopter une méthode d'évaluation contrôlée [Yang, 1999].
La première solution semble difficile à réaliser car :

1. les méthodes n'ont pas été appliquées sur les mêmes collections de données,
2. les évaluations diffèrent par les mesures utilisées pour évaluer la performance. Les auteurs n'utilisent pas les mêmes mesures de performance et peuvent calculer les moyennes de manières différentes : rappel et précision, taux de succès et taux d'erreur, le point moyen, le F_β,le micro- et macro-moyen, etc.
3. certaines conditions, non liées aux algorithmes d'apprentissage eux-mêmes, peuvent intervenir et influencer les résultats obtenus. Ceci inclut, entre autres, les unités utilisées pour représenter les textes (mot, stemme, lemme ou n-grammes), les techniques de réduction de dimension utilisées, les paramètres des algorithmes, les seuils, etc.

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

- 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 et XinLiu, 1999] comparent les machines à vecteurs supports, les plus proches voisins, les réseaux de neurones, une combinaison linéaire, et des réseaux bayesiens. Cette méthode de comparaison est la plus crédible de point de vue scientifique [Sebastiani, 2002].
- une **comparaison indirecte** : soient Φ' et Φ'' deux classifieurs. Ces deux classifieurs peuvent être comparés si deux conditions sont réunies :
  - les deux classifieurs sont testés par différents groupes de chercheurs (même avec de conditions d'expérimentation différentes) sur deux collections Ω' et Ω'' respectivement,
  - un ou plusieurs classifieurs de « références » $\bar{\Phi}_1, \bar{\Phi}_c, \cdots, \bar{\Phi}_m$ 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.

1.5 Conclusion

Dans ce chapitre nous avons présenté les techniques du domaine de Catégorisation du Texte. Ce domaine fait partie du domaine de la fouille de données textuelles. Nous avons présenté les techniques de classification, ainsi que les différents moyens de représentation de corpus documentaires et quelques mesures d'efficacité pour évaluer un classifieur.

L'une des raisons pour lesquelles depuis le début des années 90 l'efficacité des classifieurs de texte a été améliorée de façon spectaculaire est l'arrivée des méthodes d'Apprentissage Automatique, qui ont significativement augmenté l'utilisation de systèmes de ML.

Dans le chapitre qui suit nous monterons l'utilisation des techniques présentées dans ce chapitre pour la classification des documents semi-structurées (XML).

Chapitre 2

Classification des documents XML

2.1 Introduction

Le développement du document électronique et du Web a vu émerger puis s'imposer des formats de données semi-structurées, tels le XML. Ces nouveaux formats décrivent simultanément la structure logique des documents et le contenu de ceux-ci et permettent ainsi de représenter l'information sous une forme plus riche que le simple contenu. Celle-ci est adaptée à des besoins spécifiques qui permettent, par exemple, de faciliter l'accès à l'information ou d'optimiser le stockage et l'interrogation des documents. [Wisniewski *et al.*, 2005]
A coté de cela, les modèles classiques de recherche d'information et de classification de documents ont été principalement conçus pour traiter des documents plats sans prendre en compte d'aucune manière les informations de structure. En classification de documents, la structure du document joue un rôle important. D'une part les mots n'auront pas le même rôle ni la même importance suivant leur place dans le document (titre, mots clé, profondeur, méta-donnée, etc). D'autre part, des documents complexes peuvent appartenir à une classe même si une seule de leurs composantes est pertinente pour cette classe, or, cette information est souvent noyée dans les codages classiques [Vu *et al.*, 2003].

Dans ce chapitre, en premier temps, on parlera des documents semi-structurés, plus précisément les documents XML où on donnera un aperçu sur le langage XML et sa sémantique de balisage, ensuite, nous parlerons de la fouille dans ces documents XML c-à-d XML Mining ainsi qu'une comparaison avec les autres disciplines de fouilles (Text Mining, Web Mining), et enfin, nous aurons un aperçu sur quelques travaux concernant les techniques de catégorisation des documents XML qui utilisent non seulement le contenu mais aussi la structure des documents XML .

2.2 Les documents semi-structurés XML

2.2.1 Du document plat au document structuré

La ponctuation, ainsi que la présentation dans une moindre mesure (passages à la ligne, espacements divers, coupures de pages, énumérations, notes, ...etc), sont des marquages effectuées par l'auteur de tout document écrit, numérique ou non. Le lecteur humain en a besoin pour apporter du sens au texte qu'il lit. On appellera document plat tout texte ne comportant que ces deux types de marquage. On dira d'un tel document qu'il est non structuré, même si la ponctuation ou les passages à la ligne sont des éléments structurants

(séparation en phrases, en paragraphes,... etc.).
A l'opposé des documents plats, les documents dits structurés possèdent une structure régulière prépondérante à base de marquage descriptif. Par exemple, la structure d'une base de données relationnelle est représentée par des tables comportant plusieurs colonnes et plusieurs lignes, ainsi que par des relations entre les champs des différentes tables. L'ordre des éléments n'a généralement pas d'importance. On ne parle alors plus de texte mais de données ; ces données n'ont habituellement aucune signification intrinsèque, c'est-à-dire qu'il est impossible de les considérer sans examiner la structure dans laquelle elles sont inscrites.
Le document semi-structuré est un pont entre les données structurées et non structurées. Le langage XML, qui permet de produire des documents à la fois structurés et semi-structurés, est devenu un mode de représentation standard dans le domaine des documents électroniques. D'une part, par un marquage inséré dans le texte, il apporte des éléments sémantiques supplémentaires concernant la structure, le fond ou la forme du document. D'autre part, par le caractère flexible, irrégulier ou incomplet de sa structure, ainsi que par son format (textuel), il permet de stocker et d'échanger des données beaucoup plus aisément que les bases de données.

2.2.2 Le langage XML

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

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, est laissé à la libre volonté de l'auteur. C'est pourquoi on dit que le langage XML est *générique*.

2.2.2.1 Le document XML

Un document XML peut se représenter sous la forme d'une arborescence d'éléments. Cette arborescence comporte une *racine* (unique), qui englobe l'ensemble des autre éléments, ainsi que des branches et des feuilles, qui forment la structure même du document. Ils peuvent contenir du texte, ou bien d'autres éléments (ses "enfants").
Un exemple de document XML, représentant des informations sur des films présents dans une cinémathèque, 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 :

- "cinémathèque", "film", "titre",... etc. sont des *noms* ou des *types de balises*,
- `<film>` et `</film>` sont des balises (respectivement balises de début et de fin d'élément),

1. World Wide Web Consortium

- 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 (*"Woody Allen"*, *"1993"*, ...etc.) sont des éléments textuels.
- L'élément `acteurs` est le *parent* des éléments `acteur`, qui sont donc ses *enfants*. On dit également qu'un parent *contient* un enfant. Par extension, l'élément `film` est *l'ancêtre* des éléments `acteur` (et `titre`, ...etc.) qui sont ses *descendants*.
- L'élément `cinémathèque` 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 situés au plus bas de l'arborescence (`titre`, `acteur`, ...etc.) sont des *feuilles*.

```
<cinémathèque>
    <film id="1">
        <titre> Meurtre Mystérieux à Manhattan </titre>
        <réalisateur> Woody Allen </réalisateur>
        <année> 1993</année>
        <durée unité="min">108</durée>
        <acteurs>
              <acteur> Woody Allen </acteur>
              <acteur> Diane Keaton </acteur>
              <acteur> Anjelica Huston </acteur>
        </acteurs>
    </film>
    <film id="2">
    ...
    </film>}
    ...
</cinémathèque>
```

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

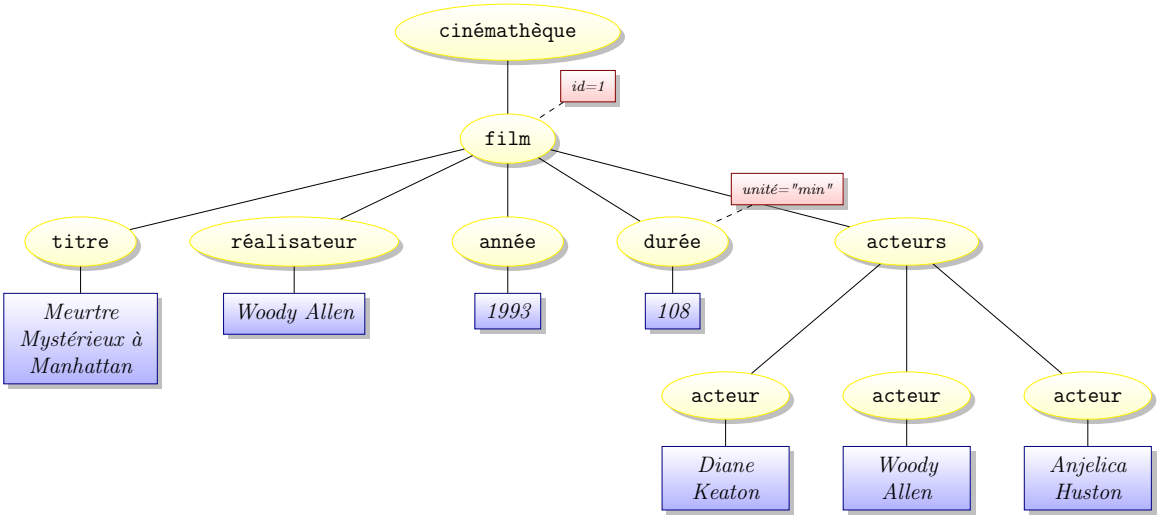

Figure 2.2 – Exemple de document **XML** représenté sous **forme arborescente**

Par ailleurs, les liens hiérarchiques (parenté, attributs nécessaires,...etc.) sont décrits par une *DTD* (*Document Type Definition*) ou par un *Schema*. La DTD ou le Schéma définissent

un “sous-langage” restreignant, imposant avec plus ou moins de contraintes d'utiliser une organisation prédéfinie. Les Schémas permettent de décrire beaucoup plus de caractéristiques que les DTD (notamment concernant les types).

2.2.2.2 La DTD

La DTD est une description générique d'une classe de documents XML. Elle impose des contraintes concernant le nom des balises à employer et les relations que celles-ci possèdent entre elles : inclusion, ordre, possibilité d'absence,... etc.
On peut dire par abus de langage que deux documents obéissant à la même DTD ont la même structure. Cependant il faut garder à l'esprit qu'ils peuvent présenter des différences sensibles dans leur structuration. Ainsi, pour un DTD représentant des articles scientifiques, comme celles de la collection INEX, un document peut contenir une bibliographie ou un résumé, tandis que l'autre non, l'un peut contenir trois sections dans le corps du texte, l'autre quatre,... etc.
On dit d'un document respectant une DTD donnée qu'il est *valide* par rapport à cette DTD. Un exemple de DTD correspondant au document XML montré à la figure 2.3.

Un document XML bien-formé est un document XML qui respecte certaines règles simples :

- Il ne doit exister qu'une seule balise racine,
- Toute balise ouverte doit être refermée (au `<balise>` doit être associé `<balise>`),
- Les noms des balises doivent commencer par une lettre ou "_", les autres caractères peuvent être des chiffres, des lettres, "_", "." ou "-".
- Les noms des balises ne doivent pas commencer par xml Par convention,
- les balises sont en minuscules.
- Quand un élément est vide, les balises peuvent être simplifiées `<balise><balise>` est identique à `<balise>`

```
<!ELEMENT cinémathèque (film*)>
<!ELEMENT film (titre, réalisateur, année?, durée?, acteurs?)>
<!ATTLIST film id CDATA #REQUIRED>
<!ELEMENT titre (#PCDATA)>
<!ELEMENT réalisateur (#PCDATA)>
<!ELEMENT année (#PCDATA)>
<!ELEMENT durée (#PCDATA)>
<!ATTLIST durée unité (min|h) #REQUIRED>
<!ELEMENT acteurs (acteur+)>
<!ELEMENT acteur (#PCDATA)>
```

Figure 2.3 – Exemple de **DTD**

2.2.2.3 Vues du XML

Il est possible de distinguer deux approches (ou vues) différentes du XML, correspondant à deux besoins différents, et à deux manières d'aborder la recherche ou l'extraction d'information [Fuhr et Großjohann, 2001] :

- l'approche *orientée document* considère le document sous sa forme traditionnelle, c'est-à-dire un texte dont la finalité principale reste la lecture par un être humain. Dans cette vue le marquage (ou balisage) sert essentiellement à fournir des informations concernant la structure (chapitres, sections ,...etc) et/ou la forme (caractères gras, italiques, ...etc) du texte. Des exemples de cette approche sont les manuels, les livres (voir figure 2.4), les articles ou les pages Web statiques.
- l'approche *orientée données*, plus proche des bases de données, est utilisée pour représenter et échanger des ensembles structurés de données, comme des horaires de vols, des catalogues, des bibliographies, ...etc. Cette vue est illustrée par un exemple à la figure 2.5.

```
<livre type = "roman">
    <titre> Le tour du monde en 80 jours </titre>
    <auteur> Jules Verne </auteur>
    <éditeur> Hatzel </éditeur>
    <chapitre n="I">
        <titre_chapitre>
          Dans le quel Phileas Fogg et Passepartout s'acceptent
          réciproquement, l'un comme maître, l'autre comme domestique
        </titre_chapitre>
        En l'année 1872, la maison portant le numéro 7 de Saville-row,
        Burlington Gardens –maison dans laquelle Sheridan mourut en
        18–   était habitée par Phileas Fogg, esq., l'un des membres
        les plus singuliers et les plus remarqués du Reform-Club de
        Londres, bien qu'il semblât prendre à tâche de ne rien faire
        qui pût attirer l'attention.
        ...
    </chapitre>
</livre>
```

Figure 2.4 – Représentation en XML d'un roman

En pratique il n'existe pas de frontière nette entre ces deux approches. Ainsi, les premiers éléments du document de la figure 2.4, qui informent sur le titre, l'auteur, l'éditeur peuvent être considérés comme étant orientés données, puisque le contenu sans la structure devient beaucoup moins informatif.

```
<bibitem type="article" key="SEB_02">
    <title> Machine learning in automated text categorization </title>
    <author> Fabrizio Sebastiani</author>
    <year>2002</year>
    <journal> ACM Computing Surveys</address>
    <pages> 1-47 </pages>
    <keywords>
        <item> Machine learning </item>
        <item> text categorization </item>
        <item> Fabrizio Sebastiani </item>
    </keywords>
</bibitem>
```

Figure 2.5 – Représentation d'un élément bibliographique sous un format XML

2.2.3 Sémantique du balisage

L'ajout d'une structure de type XML aux documents permet de prendre certaines libertés avec le texte lui-même. Notamment, l'ordre physique des éléments du fichier XML peut différer de l'ordre réel, logique, du texte lu par l'humain après mise en forme. C'est la sémantique particulière de certaines balises, attribuée par l'être humain qui a construit (ou structuré le document, qui autorise ces manipulations tout en conservant le sens du texte initial. A l'opposé, pour les processeurs automatiques de documents XML, les balises sont toutes équivalentes, et surtout toutes totalement vides de sens.
Nous allons nous attarder sur une division des balises proposée par [Lini *et al.*, 2001], dans le but d'identifier différentes catégories qu'il serait important de distinguer dans le cadre de la classification dans des documents XML. L'idée de départ était de permettre des traitements différents pendant une extraction de motifs (de séquence de caractères).

Les trois différentes classes proposées par [Lini *et al.*, 2001] sont les suivantes :

- ***Les balises "dures"*** (*"hard" tags*) sont les plus fréquentes. Elles interrompent la "linéarité" d'un texte et contribuent généralement à la structuration du document. Des exemples de balises dures sont les titres, les chapitres, les paragraphes.

```
...
<titre> Machine learning in automated text categorization </titre>
<auteur> Fabrizio Sebastiani </auteur>
...
```

- ***Les balises "transparentes"*** (*"soft" tag*) identifient des parties significatives du texte, comme les textes cités, les effets de forme ou les corrections, mais sont *"transparentes"* lorsque on lit le texte.

```
...
Les ministres de l'<gras> Union Africain </gras> se sont réunis
...
```

- ***Les balises de "saut"*** (*"jump" tags*) sont utilisées pour représenter des éléments particuliers comme les notes de marges, les références bibliographiques ou des définitions. Elles sont détachées du texte les entourant, comme la note suivante :

```
...
Le langage XML <note> eXtensible Markup Language <\note> est
devenu un mode de représentation standard dans le domaine des
documents électroniques.
...
```

2.3 XML Mining

XML Mining, nommé en premier temps dans [Lee *et al.*, 2001], est une application unique de Data Mining à des documents XML. Depuis son introduction, XML a beaucoup retenu l'attention dans les applications telles collaborations d'affaires (par exemple : ebXML [2] et

2. http://www.ebxml.org

Type de fouille	Data Mining	Text Mining	Web Mining	XML Mining
Type de données	données vectorielles numérique ou catégoriques	données non vectorielles		
		textuelles	textuelles ou multimédia	texte structurés
Type de Problème	Clustering, Classification,régression, prédiction	Vector space model (VSM), TAL (sens du mot)	Web Centent Mining, Web Strcuture Mining, Web Usage Mining	formalisme de connaissance, similarité structurelle
Application	bio-informatique, reconnaissance de forme, détection de fraude	catégorisation, Clustering et résumé automatique de texte, identification d'auteur	recherche, analyse de topologie, analyse de forme d'usage	Web personnalisé, alignement d'ontologies

Table 2.1 – comparaison entre Data mining, Tex mining, Web mining et XML mining

WebService [3]) et récemment la personnalisation Web (par exemple, Web 2.04 [4]).

XML Mining est une conséquence collective d'une variété d'efforts y compris non seulement le formalisme XML, mais aussi de Data Mining, Text Mining, et Web Mining récente. Plus important encore, XML Mining n'est pas une application simples des anciennes techniques du Text Mining (ou Data Mining) pour les documents XML. L'originalité de XML Mining nécessite alors nouvelles approches ou techniques pour ce nouveau formalisme XML [Jeong *et al.*, 2004].

2.3.1 Fouille : données, textes, Web et XML

La fouille est une activité pour découvrir des informations potentiellement utiles à partir de grands volumes de données à travers certains procédés d'extraction des données pertinentes à partir d'autres non pertinentes, la transformation en d'autres formes, et parfois association avec d'autres données et des modèles connus. Cette activité de fouille fait appel généralement aux divers applications, y compris la reconnaissance des formes, le regroupement et la classification, de prédiction , de réduction de dimension de données, de recherche documentaire et l'extraction, la découverte de connaissances, . . . etc.
Ici, nous concentrions davantage sur des applications qui traitent des données non vectorielles telles que les textes. En particulier pour cela, nous identifions trois applications de fouille - Text Mining, Web Mining, et XML Mining. Le tableau 2.1 résume leurs caractéristiques distinctives de l'extraction de données traditionnelles de traitement des données . Comme le montre le tableau ci-dessous, le type et la source de données sont le principal critère de différenciation. les nouvelles activités émergentes de fouille sont capables de traiter des données non vectorielles telles que des données textuelles, les données multimédias, et des textes structurés, alors que les activités de Data Mining traditionnelles traitent des données numériques (et catégorique). Chaque activité fouilles est détaillé dans les paragraphes suivants.

3. http://www.w3.org/2002/ws
4. http://en.wikipedia.org/wiki/Web_2.0

2.3.1.1 Data Mining

Comme nous l'avons vu au chapitre précédent, l'extraction de données est le processus de recherche d'informations implicite, inconnues, potentiellement utiles à partir de données volumineuses vectorielle. En général, ses tâches peuvent être catégorisées en analyse exploratoire des données, la modélisation descriptive, la modélisation prédictive, les modèles et les règles de la découverte, la recherche par le contenu. Pour accomplir ces tâches, les processus fouille de données se compose de plusieurs étapes : le prétraitement des données, la transformation et l'extraction des structures, le choix de l'algorithme et des paramètres, et l'interprétation et de validation.

2.3.1.2 Text Mining

Par rapport à fouille de données traditionnelles, le Text Mining est vaguement caractérisé comme le processus d'analyse des textes pour en extraire des informations utiles à un usage particulier, toutefois explicitement mentionné dans les textes. Comme les données de texture ne sont pas organisés, sans forme, et difficile à traiter algorithmiquement, les recherche en Text Mining sont focalisés sur la manière de transformer les données textuelles en numérique ou unité, tout en préservant le sens original des textes pour être exploité par les méthodes d'apprentissage automatique et statistiques.

Un autre problème, pour améliorer les performances de Text Mining, est d'interpréter correctement la signification d'un terme par le biais de la similarité sémantique entre les termes. La similarité sémantique est l'un des sujets les plus chauds en NLP (traitement du langage naturel). La plupart des mesures de similarité sémantique incorporer la relation de synonymie entre les mots / termes. Depuis que l'idée fondamentale derrière la plupart des activités de Text Mining est de quantifier la similarité entre les textes, surtout lorsque les mêmes concepts dans les documents sont expliqués dans des termes différents.

2.3.1.3 Web Mining

Le Web Mining est une application spécifique du « Text Mining » aux contenus, Structures, et Usage du Web.

- **Web Content Mining** analyse le contenu des ressources du Web, reconnue comme une forme de fouille de texte en général, même si des progrès récents dans la fouille de données sont capables de manipuler d'autres données multimédias, y compris l'image, son, vidéo,...etc. Pour les contenus web textuels, les techniques de fouille sont les même utilisés dans le Text Mining. En outre, le Web Content Mining peuvent tirer parti de la (semi-) nature structurée des pages Web textuelles. Avec l'apparition de XML, la structure logique dans une page a gagné plus d'attentions.

- **Structure Web Mining** exploite la structure (topologie) des hyperliens entre les pages Web, par conséquent, il se focalise sur des ensembles de pages Web, qui sont liés entre eux par des hyperliens. Grâce à la fouille de structure , on peut identifier

la pertinence relative des différentes pages qui semblent tout aussi pertinents quand on analyse leur contenu seulement. Un des algorithmes commercialement réussie pour le web structure d'exploration est *PageRank* utilisé dans *Google.com*, qui calcule la pertinence en comptant les hyperliens entrants provenant d'autres pages.

- **Web Usage Mining** prend soins des relevés de transactions pour un certain site web, souvent des fichiers *log* dans un serveur Web, au lieu des pages web. Par conséquent, il analyse les comportements des visiteurs et estime leurs préférences. Sur la base des tendances, Web Usage Mining permet de personnaliser le web. Les règles d'association permettent typiquement de guider les visiteurs vers des prochaines destinations (soit des produits associés ou d'autres pages Web) dont les visiteurs sont peut-être intéresser.

  Pour résumer, une bonne direction consiste à combiner systématiquement la fouille de contenus , de structure et d'usage du web ensemble. Par exemple, on peut,par exemple, explorer et rassembler des pages web en utilisant des mots clés en fonction du web content mining, le rang de leur pertinence en utilisant les scores des liens hypertextes grâce au Structure Web Mining et par conséquent recommander d'autres pages associés lorsque l'utilisateur choisit une page particulière. Aujourd'hui, cette stratégie est bien adaptée pour la commercialisation tels que *AdSense*[5] de Google.com, *AdCenter*[6] de Microsoft, et *SearchMarketing*[7] de Yahoo.com

2.3.1.4 XML Mining

XML Mining est un type spécial de la fouille de contenus web, mais aussi unique en ce que le contenu d'un document XML sont modulaires et bien structuré, et un contenu Web est plus probablement un texte simple. Le XML Mining doit être capable de manipuler la structure du documents ainsi que le contenus eux-mêmes. Les applications de XML Mining sont très divers, y compris Web personnalisé, Schéma matching, XML Message Mapping, l'alignement des ontologies, gestion d'entrepôt de données, découverte et composition des services Web,. . . etc.

2.3.2 Taxonomie de XML Mining

Cette sous-section décrit la taxonomie de XML Mining [Nayak *et al.*, 2002], plus précisément la XML Structure Mining et XML Content Mining. Cette taxonomie est représenté dans la figure 2.6. Pour les deux catégories, l'application des techniques du Data Mining, comme la classification et de clustering, sont discutés.

5. http://www.google.com/adsense
6. http://adcenter.microsoft.com
7. http://searchmarketing.yahoo.com

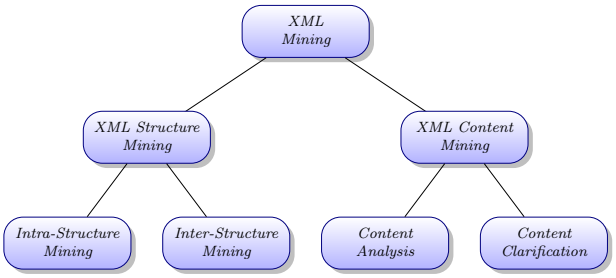

Figure 2.6 – Taxonomie de XML Mining

2.3.2.1 XML Structure Mining

Les balises (*Tags Element*) et leur imbrication, dictent la structure d'un document XML. La fouille de structures XML est essentiellement destinée aux schéma ou DTD. On peut distinguer deux divisions supplémentaires : XML intra-et inter-structure Mining.

XML intra-structure Mining : La fouille d'intra-structures des documents XML concerne la structure au sein des documents XML, pour découvrir des connaissances à partir de la structure interne des documents XML, qui est, leur définition type de document DTD. La *Classification* peut être appliqué pour associer un nouveau document XML à une classe prédéfinie de documents. Une DTD peut être interprété comme une description d'une classe de documents XML. La procédure de classification prend une collection de DTDs comme un ensemble d'apprentissage, et classe les nouveaux documents XML en fonction de cette collection de DTDs. Cette tâche est plus facile à réaliser sur des documents XML valides. Avec les DTDs déjà définies pour le nouveau document XML, la classification peut procéder en comparant les DTDs de classification avec de nouvelles DTD.

Le *Clustering* peut être appliqué pour identifier les similarités entre différents documents XML. Un algorithme de clustering prend une collection de DTDs de les regroupe sur la base de l'auto-similarité. Ces similarités sont ensuite utilisées pour générer de nouvelles DTDs.

Les *Règles d'association* peuvent être appliqués pour décrire la relation entre les balises qui ont tendance à se produire ensemble dans des documents XML et peuvent être utiles à l'avenir. En transformant la structure arborescente du document XML en un pseudo-transaction, il devient possible de générer des règles de la forme « si un document XML contient une balise <Article> jusqu'à 80% du temps, elle contiendra également une balise <Titre>.

XML inter-structure Mining : La fouille d'Inter-structures est concernée par la structure entre les documents xml. La connaissance est découverte au sujet du rapport entre les sujets des documents, les organismes et les noeuds sur le Web. La *classification* est appliquée avec des espaces de noms (*name spaces*) et URIs. Après avoir associé précédemment un ensemble de DTDs à un NameSpace ou à un URI particulier, cette information est employée pour classifier un nouveau de documents XML.

Le *Clustering* implique d'identifier les DTDs semblable. Les clusters sont employés en

définissant des hiérarchies de DTDs qui recouvre des exemples sur le Web, pour découvrir les autorités et les Hubs . Les créateurs de DTDs sont identifiés comme autorités, et les créateurs des instances sont des Hubs. Des techniques additionnelles de fouille sont exigées pour identifier tous les instances de DTD actuels sur le Web. L'application de la classification peut identifier les endroits le plus susceptibles pour extraire des instances.

2.3.2.2 XML Content Mining (Fouille du contenu des documents XML)

Le contenu désigne le texte situant entre chaque début et fin de balise dans un documents XML.Ainsi, La nature semi-structuré des documents XML lance un défi pour la fouille du contenu. Cependant, un certain nombre de langages d'interrogation conçus pour des données semi-structurées ont été mis en application. La fouille de contenu des documents XML peut être divisé en deux tâches : analyse du contenu et clarification structurale.

XML Content Analysis (analyse du contenu des documents XML) : Les tâches effectuées sur des documents de XML sont semblables à celles effectuées sur les documents textuelles. La *classification* est effectuée sur le contenu XML, en étiquetant le nouveau contenu de XML comme appartenant à une classe prédéfinie. Pour réduire le nombre de comparaisons, la nouvelle DTD du document est classifié par les DTDs pré-existantes. Puis, seulement les instances de lassifications des DTDs assorties doivent être considérées en classifiant un nouveau document.

Le *Clustering* identifie le potentiel pour de nouvelles classifications. encore une fois, la considération de DTDs mènera à un clustering plus vite : les DTDs semblable sont susceptibles d'avoir un certain nombre d'ensembles de valeur. Par exemple, tout les DTDs concernant des véhicules auront un ensemble de valeurs représentant des voitures, un autres ensembles représentant des bateaux,...etc. Cependant, les DTDs qui semblent différent peuvent avoir des contenus semblables. La fouille de contenu des documents XML hérite quelques problèmes considérés dans la fouille de textes, la synonymie et la polysémie peuvent poser des difficultés, mais les balises entourant le contenu peuvent habituellement aider à la résolution de l'ambiguïtés.

Clarification de structure : Le contenu peut fournir un appui pour un clustering alternatif de DTDs similaires. Deux DTDs distinctement structurées peuvent avoir des instances de documents avec un contenu identique. Vice versa, les DTDs fournissent un appui pour un clustering alternatif du contenu. Deux documents XML avec le contenu distinct peuvent être groupés ensemble étant donné que leur DTDs sont semblable.

Le *Contenu* peut également se révéler important en matière de Clustering des DTDs qui semblent différents, mais ont des instances avec un contenu similaire. En raison de l'hétérogénéité, l'incidence des synonymes est augmenté. Sont-elles distinctes les DTDs effectivement décrivant la même chose, qu'avec des termes différents ?. Ou, inversement, les DTD figurant similaires sont en réalité complètement différente, étant donné les homographes. Par exemple : <craft> la construction de bateaux </ craft> et <craft> bateau </craft> In-

terprétation de la première est l'occupation, et celle du navire plus tard. La similitude de leur contenu ne distingue pas l'intention sémantiques des étiquettes. La fouille, dans ce cas, peuvent fournir les probabilités d'une balise ayant une signification particulière, ou une relation entre le sens et une URI.

2.4 Catégorisation des documents XML

Nous présentons dans cet état de l'art sur la catégorisation des documents XML deux types d'approches : Celles qui prennent simultanément en compte le contenu et la structure, et celles qui sont basées sur l'analyse de la structure arborescente en utilisant les balises XML.

2.4.1 Utilisation de la structure et du contenu

2.4.1.1 Travaux de [Yang et Zhang, 2007]

L'article propose le modèle de vecteur structuré de liens - SLVM - (*eng : Structured Link Vector*) qui constitue une extention du modèle vectoriel (VSM : Vector Space Model) en incorporant les structures de documents XML. La classification est basée sur l'application de la méthode (SVM : Support Vector Machine) sur ces vecteurs. Ce modèle a été examiné avec le corpus *XML Wikipedia 2007*[8].

L'application directe du modèle VSM pour représenter les documents XML n'est pas souhaitable car la structure du document XML étiquetée par ses éléments (balises) sera ignorée, pour remédier à ce problème, SLVM représente un document XML comme un tableau des VSMs où chaque VSM est spécifique à un élément XML (élément XML correspond à la balise contenant un nœud feuille). Chaque document XML est représenté par un attribut matrice document Δ_x tel que :

$$\Delta_x = \begin{bmatrix} \Delta_{x(1,1)} & \Delta_{x(2,1)} & \cdots & \Delta_{x(m,1)} \\ \Delta_{x(1,2)} & \Delta_{x(2,2)} & \cdots & \Delta_{x(m,2)} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{x(1,n)} & \Delta_{x(2,n)} & \cdots & \Delta_{x(m,n)} \end{bmatrix}$$

où :

- m est le nombre des éléments XML et n le nombre de termes des documents XML.
- $\Delta_{x(i,j)} = TF(w_j, doc_x.e_i) \cdot IDF(w_j)$ où $TF(w_j, doc_x.e_i)$ est la fréquence du terme w_j dans l'élément e_i du document doc_x.

Cet attribut peut être normalisé comme suit :

$$\tilde{\Delta}_{x(i,j)} = \frac{\Delta_{x(i,j)}}{\sum_{j=1}^{n} \Delta_{x(i,j)}}$$

8. Pour des détails sur les corpus utilsés, voir la sous-section 2.4.3.1

La figure 2.7 représente un document XML et la figure 2.8 représente les différents attributs qui représente ce document.

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

Figure 2.7 – Exemple de document **XML**

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

(a) Attribut Document Vecteur

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

(b) Attribut Document Matrice

$$\tilde{\Delta}_x = \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \text{title} & \text{author} & \text{conference} \\ \end{matrix} \begin{pmatrix} 0.5 & 0 & 0 \\ 0.5 & 0 & 0 \\ 0.5 & 0 & \frac{\sqrt{2}}{2} \\ 0.5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \begin{matrix} \text{Ontology} \\ \text{Enabled} \\ \text{Web} \\ \text{Search} \\ \text{John} \\ \text{Intelligence} \end{matrix}$$

(c) Attribut Document Matrice Normalisé

Figure 2.8 – Differents attributs documents

Après la représentation des documents XML en attributs document-attribut, la classification est faite en appliquant la méthode SVM (Support Vector Machine) sur ces attributs, tout en choisissant comme fonction du noyau la mesure de similarité entre deux documents XML définit par :

$$K(X_i, X_j) = sim(doc_x, doc_y) = \sum_{i=1}^{n} {\Delta^n_{x_{(i)}}}^T M_e \Delta^n_{y_{(i)}}$$

où M_e est une matrice de dimension $m \times m$ appelée matrice de similarité des éléments XML.

2.4.1.2 Travaux de [Xing *et al.*, 2006]

[Xing *et al.*, 2006] proposent d'utiliser à la fois la distance d'édition des arbres et un critère de la longueur de description minimum (MDL) (*ang : Minimum Description Length*)

pour la classification des documents XML selon le contenus et la structure. La méthode modélise chaque catégorie, avec une grammaire normalisée régulière de couverture (NRHG)(*Ang : Normalized Regular Hedge Grammar*) qui est extraite d'une série de document en utilisant le principe de MDL. La distance entre un document XML et une catégorie est alors calculée par distance d'édition des arbres entre l'arbre de ce document et la grammaire calculé pour la catégorie. Le modèle est examiné avec le corpus *IEEE*.

Un document XML peut être représenté par un arbre ordonné et étiqueté, sa DTD est converti en grammaire normalisée de couverture régulière (NRHG) définie comme suit :

Définition : *une NRGH est un 5-uplet* (Σ, V_T, V_F, P, s)*, telque :*

1. Σ *est un ensemble fini de terminaux.*
2. V_T *est un ensemble fini des variables-arbres.*
3. V_F *est un ensemble fini des variables-forêts.*
4. P *est un ensemble de règles de production, dont chacune prend l'une des quatre formes :*
   (a) $vt \rightarrow x$ *,telque* v_t *est une variable-arbre* $\in V_T$ *,et* x *est un terminal* $\in \Sigma$
   (b) $vt \rightarrow a\langle v_f \rangle$*, telque* v_t *est une variable-arbre* $\in V_T$ *,* a *est un terminal* $\in \Sigma$ *et* v_f *est une variable-forêt* $\in V_F$*.*
   (c) $v_f \rightarrow v_t$*, telque* v_f *est une variable-forêt et* v_t *est une variable-arbre.*
   (d) $v_f \rightarrow v_t \acute{v}_f$*,telque* v_f *et* $\acute{v}_f$ *sont des variables-forêt et* v_t *est une variable-arbre*
5. $s \in V_T$ *est le symbole de départ, qui définit le schéma d'arbres qui peuvent être générés par cette grammaire.*

Dans la définition ci-dessus, les terminaux sont utilisées pour étiqueter les nœuds (les feuilles et les nœuds internes), les variables-arbre sont des symboles de grammaire à générer des arbres, et les variables-forêt sont utilisés pour générer des forêts (chaîne de variables-arbre). la règle (a) est utilisé pour générer un arbre avec un seul nœud, la règle (b) est utilisée pour mettre un nouveau nœud en tant que nouvelle racine de la forêt qui est généré par variable-forêt, la règle (c) est le cas de base pour générer des un arbre pour la forêt, et la règle (d) est utilisé pour concaténer un arbre avec une forêt pour former une nouvelle forêt.

On dit qu'un arbre étiqueté ordonné est conforme à un NRHG s'il peut être produit par la grammaire. Quand un arbre n'est pas conforme à un NRHG, il peut être transformé par une suite des opérations d' éditions (insertion, suppression, remplacement) tels que l'arbre résultant soit conforme au NRHG. Un coût est assigné à chacune de ces opérations. La distance d'édition entre un arbre et un NRHG est le coût minimum d'une suite des opérations d'éditions transformant l'arbre pour se conformer à la grammaire.

Le problème d'extraction des schémas XML revient,étant donné une collection de documents de XML $\{d_1, d_2, \cdots, D_n\}$, à trouver un schéma s, tel que $d_1, d_2, \cdots, D_n$ sont des instances de documents qui peuvent être générés par le schéma s. L'extraction de schéma

peut être simplifiée en tant que déduction des expressions régulières (automates finis non déterministes FNA) depuis une collection de séquences ou chaînes d'entrée (représentations des schémas XML), l'introduction du critère MLD (Minimum Length Description) permet de ranger les expressions candidates pour obtenir le meilleur schéma en minimisant $L = \lambda L_g + L_d$ tel que :

$$L_g = (S + T) \log S$$

où :

- S est le nombre des états et T est le nombre des transitions.
- L_d est le coût de simulation FNA.
- λ est employé pour ajuster la précision.

Pour mesurer la similarité entre les contenus textuels des documents XML, ils ont utilisé le modèle LSI (*ang : Latent Semantic Indexing*)qui est une variante du modèle vectoriel standard (VSM) pour réduire la dimension de l'espace des termes.

La conception du système de classification est illustré par la figure 2.9, Selon leur approche, il existe trois étapes pour obtenir un classifieur :

- La première étape est la sélection représentative et d'extraction de schéma.
- La deuxième étape consiste à calculer la distance entre les documents et les schémas (un par classe). Supposons qu'il existe n classes de documents dans le corpus d'apprentissage, un schéma unifié est généré pour chaque classe. Pour chaque document, un vecteur de distance $\langle d_1, d_2, \cdots, d_n \rangle$, qui représente la distance entre chaque document et le « centre » des points de la catégorie, est calculée et introduite dans la phase d'apprentissage du classifieur.
- La troisième étape est l'apprentissage du classifieur.

Pour classer un document, un vecteur de distance se compose des distances entre le document et le schéma de chaque classe est calculé. Le classificateur donne l'étiquette du document basé sur le vecteur de distance.

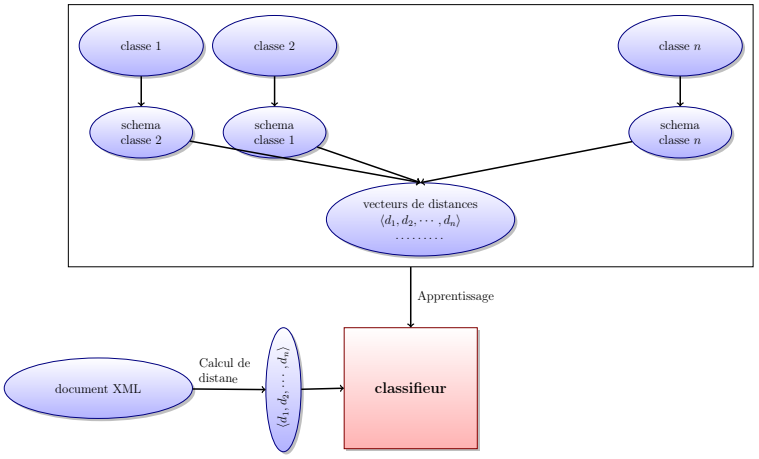

Figure 2.9 – Architecture du système proposé par [Xing *et al.*, 2006]

2.4.1.3 Trauvaux de [de Campos *et al.*, 2007]

Cet article propose d'employer des méthodes probabilistes pour la catégorisation des textes en considérant que quelques attributs structuraux de chaque document de XML traduit en texte plat. Différentes méthodes sont proposées pour ajouter l'information structurale, ainsi que deux modèles probabilistes : Bayesien naïf et OR Gate. Ces modèle ont été examiné avec le corpus *XML Wikipedia 2007*.

L'approche probabiliste classique pour la classification de textes peut être énoncée comme suit : soit un ensemble de classes $\{c_1, c_2, \ldots, c_n\}$ et, un document d_j à classifier, la probabilité a posteriori de chaque classe $p(c_i|d_j)$ est calculé selon la théorème de Bayes :

$$p(c_i|d_j) = \frac{p(c_i)p(d_j|c_i)}{p(d_j)} \quad \propto p(c_i)p(d_j|c_i)$$

et le document est assigné à la classe ayant la plus grande probabilité a posteriori, c.-à-d :

$$c^*(d_j) = \arg\max_{c_i} \{p(c_i)p(d_j|c_i)\}$$

Alors le problème est comment estimer les probabilités $p(ci)$ et $p(d_j|c_i)$.

Dans le premier modèle proposé, le classifieur de Bayes naïf (figure 2.10), un document est une séquence ordonnée des mots ou des termes extraits du même vocabulaire, en supposant que les occurrences et les positions des termes dans un document sont conditionnellement indépendants pour une classe donnée.

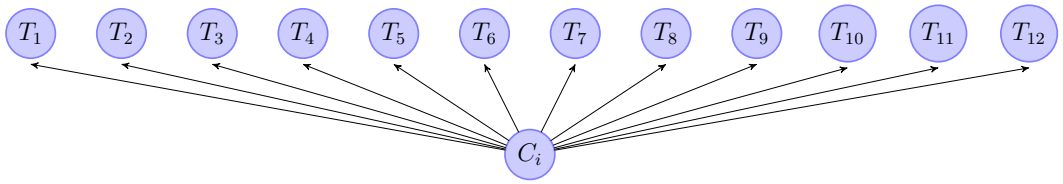

Figure 2.10 – Classifieur de Bayes naïf

Ainsi,$p(d_j|c_i)$ est calculée comme suit :

$$p(d_j|c_i) = p(|dj|)\frac{|dj|!}{\prod_{t_k \in d_j} n_{jk}!} \prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}}$$

Où t_k est un terme de d_j , n_{jk} est la fréquence d t_k dans les document d_j et $|d_j| = \sum_{t_k \in d_j} n_{jk}$ est le nombre de termes dans d_j .
Puisque $p(|dj|)\frac{|dj|!}{\prod_{t_k \in d_j} n_{jk}!}$ ne dépend pas de la classe c_i, il peut être omis :

$$p(d_j|c_i) \propto \prod_{t_k \in d_j} p(t_k|c_i)^{n_{jk}}$$

L'éstimation de la probabilité du $p(t_k|c_i)$ est effectué au moyen de l'éstimation de Laplace :

$$p(t_k|c_i) = \frac{N_{ik} + 1}{N_i + M}$$

Où N_{ik} est la fréquence du terme t_k dans les documents de c_i, N_i est le nombre de termes dans les documents de c_i et M est la taille du vocabulaire des termes.
L'estimation de la probabilité a priori de classes, $p(c_i)$ est donnée par :

$$p(c_i) = \frac{N_{i,doc}}{N_{doc}}$$

où $N_{i,doc}$ est le nombre de documents associés à la classe c_i et N_{doc} est le nombre total des documents.

Le deuxième modèle proposé est basé sur les réseaux bayesiens avec une topologie en deux couches (figure 2.11), où les nœuds de termes (chaque terme t_k est représenté par une variable T_k dont sa valeur $\in \{t_k, \bar{t}_k\}$) sont des « causes » et les nœuds de classes (chaque classe c_i est représentée par une variable binaire C_i dont sa valeur $\in \{c_i, \bar{c}_i\}$) sont des « effets ». S'il existe un arc du nœud T_k vers le nœud C_i cela signifie que le terme t_k se présente dans un documents de la classe c_i.

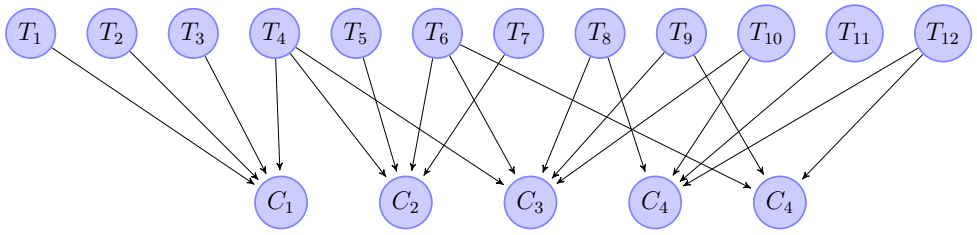

Figure 2.11 – Classifieur OR Gate

La probabilité conditionnelle dans ce modèle est définit comme suit :

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

où $R(pa(C_i))$ est un sous-ensemble de parents de C_i qui sont instanciés à sa valeur de t_k dans la configuration $pa(C_i)$, et $W(T_k, C_i)$ est un poids représentant la probabilité que l'occurrence de la « cause » T_k fait que l' « effet » vrai (c.-à-d. force la classe c_i pour se produire).

Une fois ces poids sont estimés (d'une façon simple, prendre $w(T_k, C_i)$ comme $p(c_i|t_k)$), la probabilité a posteriori $p \cdot p(c_i, d_j)$ peut être estimée comme suit :

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

```
<livre>
    <titre> Don Quichotte noble de La Mancha </titre>
    <auteur> Miguel de Cervantes Saavedra </auteur>
    <contenu >
        <chapitre> Un </chapitre>
        <texte> dans un endroit de La Mancha ... </texte>
    </contenu>
</livre>
```

Figure 2.12 – "Campos" : Fragement d'un document XML pour illuster les transformations

Enfin, pour représenter les documents XML, les auteurs proposent plusieurs méthodes pour transformer les propriétés structurelles balises) en texte plat, en plus de la méthode simple de prendre que du textes (ignorer les balises), ils proposent les méthodes suivantes :

1. **Méthode " Adding "** : considérer les balises d'une manière atomique, ou dans le cadre des autres balises où ils sont contenues (c-à-d considérant une partie du chemin de l'élément racine jusqu'à un certain niveau de profondeur).
2. **Méthode "Tagging"** : les termes (éléments textuels) sont préfixés par les balises (jusqu'à un certain niveau de profondeur).
3. **Méthode " No Text "** : considérer que les balises, cette méthodes est équivalente à la méthode "Adding" tout en éliminant les termes textuels.
4. **Méthode " Text Replication "** : assigne une valeur entière à chaque balise, proportionnelle à son contenu informatif pour la catégorisation (plus la valeur est haute, il est plus informatif). Cette valeur est employée pour replier des termes textuels.
   Par exemple : *(titre 1) (auteur 0) (chapitre 0) (texte 2)*

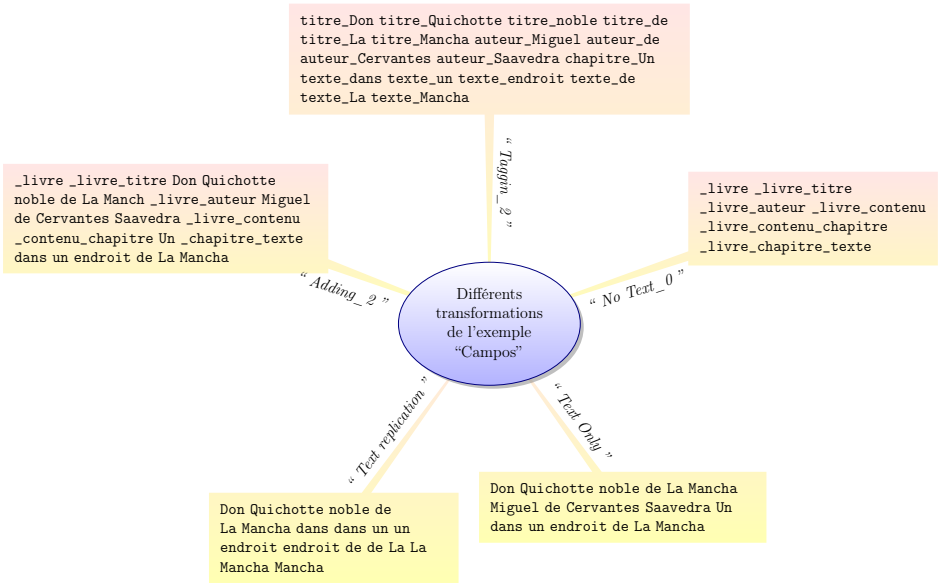

Figure 2.13 – Differents transformations proposé par Campos et al

2.4.1.4 Travaux de [Ghosh et Mitra, 2008]

Dans cet article, les auteurs proposent la combinaison d'information de structure et de contenu en utilisant les noyaux composés de SVM (*Support Vector Machine*) pour la catégorisation des documents XML. Le noyau composé est construit par une combinaison linéaire des noyaux de texte et de structure, les poids de combinaison sont déterminés par une méthode heuristique basée sur l'entropie de la distribution de fréquences des unités d'indexation. Ce modèle a été examiné avec le corpus *XML Wikipedia 2006*

La similarité de contenu est mesurée par le noyau textuel commun en utilisant la similarité *cosinus* entre les vecteurs de fréquence des termes. Tandis que les modèles booléens et cosinus sont utilisés pour mesurer la similarité structurelle entre les documents XML représentés par des chemins linéaires *TPaths* (un chemin s'étend de la racine de l'arbre XML à un nœud feuille contenant le contenu textuel).
Dans le modèle de similarité *cosinus*, la structure d'un document de XML est représentée par un vecteur de structures dont les éléments sont les scores de $tf * idf$ des TPaths dans le document. La similarité structurelle *'TPath cosinus'* entre deux documents de XML est calculée par la similarité *cosinus* entre les vecteurs de structures des deux documents. Dans le modèle de similarité booléen (*'TPath booléen'*), les éléments du vecteur de structures sont des valeurs binaires indiquant la présence ou l'absence de TPaths dans les documents.
Les noyaux composés sont obtenus en combinant le noyau 'TPath cosinus' et le noyau 'TPath booléen' avec le noyau textuel désigné respectivement par noyau 'TPath cosinus & Texte cosinus' et le noyau 'TPath booléen & Texte cosinus'. Pour déterminer les poids relatifs à la combinaison linéaire, une heuristique à base d'entropie a été employée. La méthode se résume comme suit : Soit $t_1, t_2, \ldots, t_k$ des unités indexes (des termes ou TPaths pour le noyau de textes et de structure(s) respectivement), $p_1, p_2, \ldots, p_k$ des valeurs normalisés des fréquences unités d'indexes respectivement. L'entropie de la distribution de fréquence de document dans le corpus est définie par :

$$H = -\sum_i p_i \log(p_i)$$

Soit H_s l'entropie de la distribution de la fréquence des TPaths du document dans le corpus d'apprentissage, et H_c l'entropie de la distribution de la fréquence des termes du texte. Alors le noyau composé est défini comme :

$$\mathcal{K}_{composite}(x_i, x_j) = \frac{Hs}{H_s + H_c}\mathcal{K}_{structure}(x_i, x_j) + \frac{H_c}{H_s + H_c}\mathcal{K}_{contenu}(x_i, x_j)$$

Où, $\mathcal{K}_{structure}$ et $\mathcal{K}_{contenu}$ correspondent au noyau structure et au noyau contenu respectivement.

2.4.2 Utilisation de la structure uniquement

2.4.2.1 Travaux de [Candillier *et al.*, 2005]

[Candillier *et al.*, 2005] a proposé de transformer chaque arbre représentant un document XML en un ensemble d'attribues-valeurs. Les ensembles d'attribues-valeurs sont établis en utilisant les différentes relations entre les nœuds de l'arbre d'entrée (relations de père-fils, relations de prochain-frère (enfant du même parent), ensemble de chemins distincts). La classification de ces ensembles est faite aussi par l'algorithme de la classification $C5$, Les expériences sont faites sur le corpus *Movies* et le corpus *IEEE*.

Il y a beaucoup de manières possibles de transformer un arbre en un ensemble d'attribues-valeurs. La première possibilité de base est de transformer l'arbre en un ensembles de balises présentes dans ses nœuds. Un document de XML serait ainsi transformé en sac-de-balises simples, mais cette possibilité ne tient pas compte de la structure de l'arbre, pour remédier à ce problème, il propose de construire les attributs suivants à partir des arbres disponibles :

- l'ensemble de relations de père-fils (dont le domaine est l'ensemble de paires de balises marquant les nœuds),
- l'ensemble de relations de prochain-frère (dont le domaine est l'ensemble de paires de balises marquant les nœuds),
- l'ensemble de chemins distincts (sous-chemins y compris), à partir de la racine (dont le domaine est l'ensemble d'une série finie des balises marquant les nœuds).

La valeur associée à chaque attribut,pour un document donné, est le nombre de ses occurrences dans ce document. Il définisse également de nouveaux attributs liés positions distinctes de nœud représentées dans les arbres. L'identification d'une telle position de nœud peut être codée, par exemple, par une série de nombres entiers : la racine est codée 0, son premier fils est 0 : 0 codé, son deuxième fils est 0 : 1,. . . etc.

L'algorithme proposé 3 *SSC (SubSpace Clustering)* est basé sur l'utilisation d'un modèle probabiliste , par assomption que les clusters suivent des distributions indépendantes sur chaque dimension, et il peut fournir en sortie une représentation interprétable des clusters a trouvé, comme ensemble de règles, et une manière de les visualiser effectivement. D'ailleurs, une nouvelle étape de sélection a été ajoutée pour garder seulement les meilleurs attributs pour chaque cluster, et soit ainsi moins sensible aux dimensions non pertinentes.
A noter que :
A est l'ensemble d'attributs possibles associés aux documents de XML,cet ensemble est divisé en groupes d'attributs :

- A_1 est l'ensembles des balises (Tag),
- A_2 est l'ensemble des relations père-fils,
- A_3 est l'ensemble des relations prochain-frère,
- A_4 est l'ensemble des positions des nœuds,
- A_5 est l'ensemble des chemins.

Ainsi A est composé de $SA = 5$ classes des attributs.
$Cut(C_k, A_i)$ dénote le partitionnement en deux parties,l'ensemble de données qui est formés

des documents inclus dans le cluster C_k, et transformé avec l'ensemble d'attributs A_i.

Algorithme 3: *SSC adapté à la classification -Etape 1-*

Données : D ensemble de documents XML;
Initialiser le cluster $C1$ avec tous les document de D;
Créer une hiérarchie vide H;
tant que $CUT = 1$ **faire**
  $CUT \leftarrow 1$;
  pour tous les $k \in [1..K]$ **faire**
    $CUT_k \leftarrow 0$;
    $i \leftarrow 1$;
    tant que $(CUT_k = 0)$ *et* $(i \leq SA)$ **faire**
      si *dans* $Cut(C_k, A_i)$, *pas de classe partagée par les differents partition* **alors**
        Executer le partitionnement;
        calculer les règles associées et mettre à jour la hiérarchie H;
        $CUT_k \leftarrow 1$;
        $CUT \leftarrow 1$;
      sinon
        $i \leftarrow i + 1$;
      fin
    fin
  fin
fin
Résultat : la hiérarchie H et la répartition courante

La deuxième étape décrite par l'algorithme 4,qui prend comme entrée le résultat de l'étape précédente et consiste à séparer les classes qui sont encore incluses dans les mêmes clusters. Elle-même se compose de deux étapes principales : la première essaye de distinguer les classes en utilisant des règles, afin d'être aussi compréhensible que possible, alors que la seconde emploie les modèles probabilistes qui sont des modèles plus riches, capable adapter une décision plus complexe.

- $S2$, $S3$, $S4$ et $S5$ représentent les probabilités des modèles basés sur des relations de prochain-frère, respectivement concernant le classes 2, 3, 4 et 5.
- $P6$ et $P11$ représentent les probabilités des modèles basés sur les chemins concernant classes 6 et 11.
- $NB(0.0.0)$ indique le nombre d'enfants du premier petit fils de la racine.

Ainsi, dans exemple 2.14, l'adhésion à la classe 1 dépend seulement de la présence de la balise $Movie$. Et de la même manière, l'adhésion à la classe 8 dépend seulement de l'absence des balises $Movie$, CL, BJ, AJ, et la présence de la relation père-fils entre les balises AT et AQ.

2.4.2.2 Travaux de [Knijf, 2006]

Le modèle proposé par [Knijf, 2006] permet une classification des documents XML en utilisant les arbres d'attributs fréquents. L'algorithme se compose de 4 étapes :

Algorithme 4: *SSC adapté à la classification -Etape 2-*

Données : La hiérarchie H et la répartition de l'étape précédente;
pour tous les $k \in [1..K]$ **faire**
  tant que C_k *contient des classes différentes* **faire**
    pour tous les $i \in [1..SA]$ **faire**
      pour tous les *classe* $\in C_k$ **faire**
        si *une règle peut distinguer la classe des autres* **alors**
          exécuter le partitionnement et mettre à jour la hiérarchie H;
        fin
      fin
    fin
    si *aucune séparation a été faite* **alors**
      pour tous les $i \in [1..SA]$ **faire**
        calculer le taux d'erreur de classification par la cross-validation du modèle probabiliste généré sur les attributs A_i;
      fin
    fin
    choisir le modèle dont taux est le plus bas et mettre à jour la hiérarchie H;
  fin
fin
Résultat : la hiérarchie H

- Pour chaque catégorie, découvrir un ensemble d'arbres d'attributs fréquents en utilisant l'algorithme de *FAT-Miner*.
- Sélectionner les arbres (modèles) émergents pour chaque catégorie.
- Transformer chaque document en vecteur où chaque composant indique si un arbre émergent particulier apparaisse dans le document.
- Finalement, appliquer un algorithme classique de classification sur ces vecteurs (l'arbre de décision binaire).

Le modèle utilise le corpus *XML Wikipedia 2006*.

Pour découvrir l'ensemble des arbres d'attributs fréquents, chaque documents XML est représenté par un arbre d'attributs enraciné ordonné :

$$T = \{V, E, \leq, L, v_0, M\}$$

où :

- V ensemble de nœuds,
- E ensemble d'arêtes,
- v_0 le nœud racine,
- L est une fonction d'étiquetage des nœuds de V,
- $\leq$ est une fonction binaire $\subset V^2$ qui définit un ordre entre les nøeuds frères,
- $M : V \rightarrow P(A)/\{\emptyset\}$ est une fonction qui assigne à chaque nœud un sous-ensemble de A où A est l'ensemble de tous les attributs.

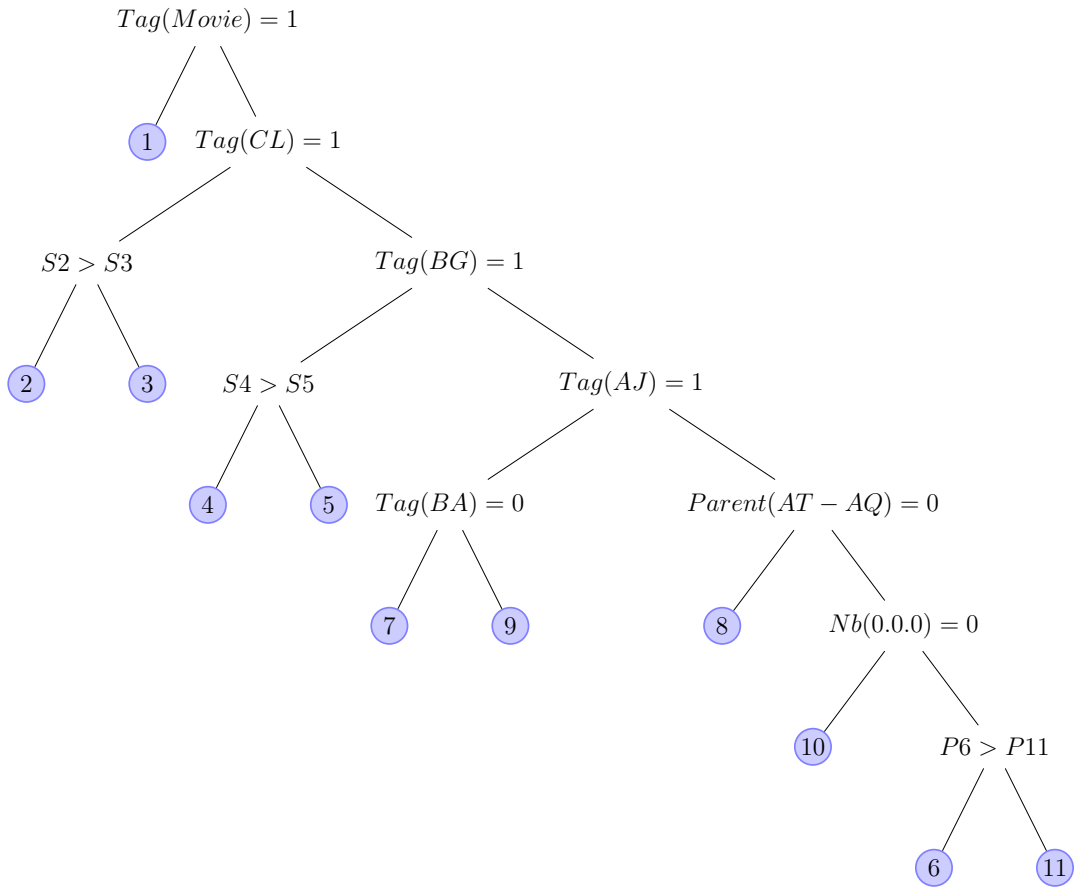

Figure 2.14 – La hiérarchie obtenue pour la classification d'une dataset

Etant donné deux arbres étiquetés T_1 et T_2, T_2 est dite sous-arbre induit de T_1 et T_1 est un super-arbre induit de T_2 s'il existe une fonction injective Φ qui satisfait les conditions suivantes :

1. Φ préserve l'étiquetage : $L_{T_2}(v) = L_{T_1}(\Phi(v))$
2. Φ préserve l'ordre entre les nœuds frères : si $v_1 \leq_{T_2} v_2$ alors $\Phi(v_1) \leq_{T_1} \Phi(v_2)$
3. Φ préserve la relation père-fils : $(v1, v2) \in E_{T_2}$ ssi $(\Phi(v_1), \Phi(v_2)) \in E_{T_1}$
4. Φ préserve les attributs : $\forall v \in V_{T_2} : M(v) \subseteq M(\Phi(v))$

La figure 2.15 montre un exemple d'arbre d_1 et un de ses sous-arbres induits T_1.

Le but de l'algorithme FAT-Miner est de trouver tous les sous-arbres d'attributs fréquents (selon leurs occurrences) dans une collection de documents XML. L'idée principale pour l'algorithme FAT-Miner est de diviser le processus de fouille en deux étapes : globale et locale.

La fouille globale énumère tous les sous-arbres fréquents en employant la technique extrême droite de prolongation, c'est-à-dire, $(k-1)$-arbre est augmenté à un k-arbre en ajoutant seulement un nouveau nœud à un nœud sur la branche extrême droite du $(k-1)$-arbre.

La fouille locale doit être exécutée pour chaque nœud des sous-arbres. Elle se réduit au calcul des ensembles d'attributs fréquents de tous les attributs auxquels le nœud du sous-arbre est mappé dans la base de données. Les méthodes locales et globales de fouille doivent

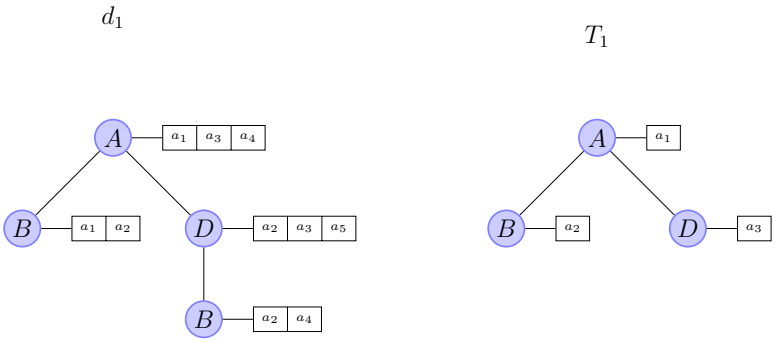

Figure 2.15 – T_1 est un sous-arbre induit de l'arbre d'attributs ordonné enraciné d_1

être combinées : l'idée est que toutefois qu'un arbre candidat T (de taille k) est produit, l'algorithme local d'exploitation détermine le premier ensemble d'attributs fréquents de v_k, dite A_1. S'il n'y en a aucun, alors T doit être taillé. Autrement tous les super-arbres de T sont calculés, où l'ensemble d'attribut de v_k égale A_1 Quand il n'y ont plus de prolongements gauche de T, le prochain itemset fréquent du v_k est calculé et ce modèle fréquent (*Pattern*) est alors davantage prolongé. Ce principe est illustré par un arbre d'énumération partiel de d_1 est montré sur la figure 2.16.

Pour sélectionner les modèles émergents pour chaque classe ou catégorie,la probabilité à priori $P(class|T)/P(class)$ ou bien le support de la règle $T \longrightarrow class$ est employée.

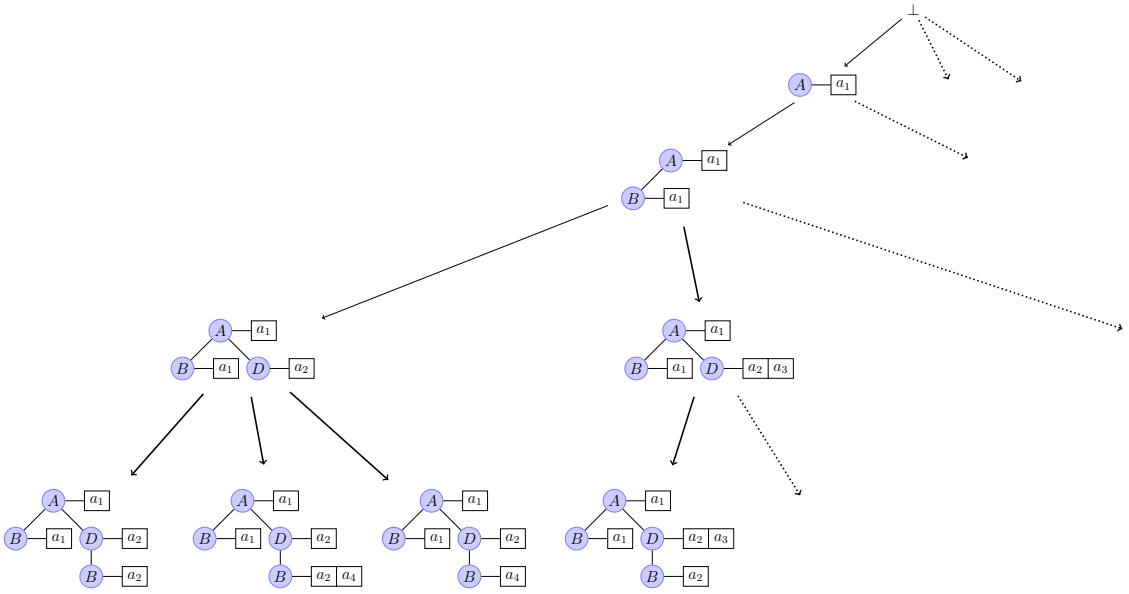

Figure 2.16 – Une partie de l'arbre d'énumération des modèles pour l'arbre d_1 (représenté sur la figure 2.15)

2.4.2.3 Travaux de [Garboni *et al.*, 2005]

Ce modèle [Garboni *et al.*, 2005] propose une mesure de similarité entre un document de XML et une catégorie de documents XML. Chaque document XML est transformé en séquences d'étiquettes de ses nœuds. Puis, chaque catégorie est transformé en un ensemble de séquences qui sont extraites à partir de l'ensemble de toutes les séquences de catégorie en utilisant une mesure séquentielle classique d'extraction de modèles. La mesure de similarité entre un document de XML et une catégorie des documents est ainsi calculée comme la plus longue sous-séquence commune entre la séquence du document et les séquences qui caractérisent la catégorie. Le modèle a été examiné avec le corpus *Movie*.

Afin d'exécuter une telle transformation, les nœuds de l'arbre XML doivent d'abord être étiquetés. Chaque étiquette est associée à sa profondeur dans l'arbre. Enfin une exploration première de l'arbre en profondeur donnera l'ordre correspondant. Cette dernière étape est appelée la « *réduction* ». La transformation est montrée par la figure 2.17.

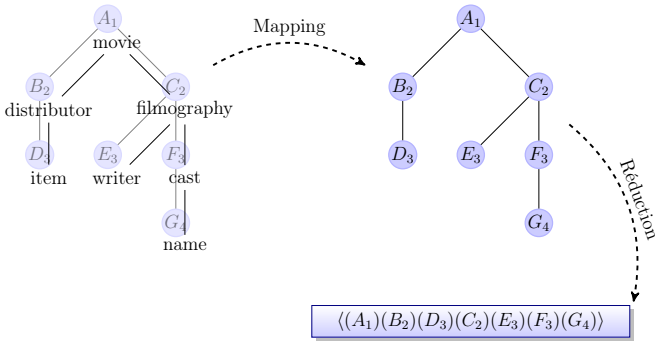

Figure 2.17 – Transformation d'un arbre XML vers une séquence

La structure originale de document XML (gauche supérieure) est d'abord alignée vers un nouvel arbre étiqueté. Par exemple, le nœud « filmographie » devient « C_2 » qui correspond à l'étiquette de la « filmographie » (C) liée à sa profondeur (2) dans l'arbre. L'étape de réduction vise à écrire l'ordre correspondant après une première navigation de l'arbre en profondeur. Une fois l'ensemble de la totalité des séquences est obtenu, un algorithme séquentiel classique d'extraction de modèles peut extraire les séquences fréquentes. Ces séquences, une fois alignées de nouveau dans des arbres, donneront les sous-arbres fréquents incorporés dans la collection.

Le principe général de ce modèle illustré par la figure 2.18 s'appuie sur les étapes suivantes :

1. **Etape de nettoyage** (*correspond à l'étape « 1 » sur la figure 2.18*) : L'idée principale est d'enlever les étiquettes (balises) non pertinentes pour des opérations de classification. Une étiquette qui est très fréquente dans la collection entière peut être considérée en tant que non pertinente puisqu'elle n'aidera pas en séparant un document des autres (l'étiquette n'est pas distinctive).

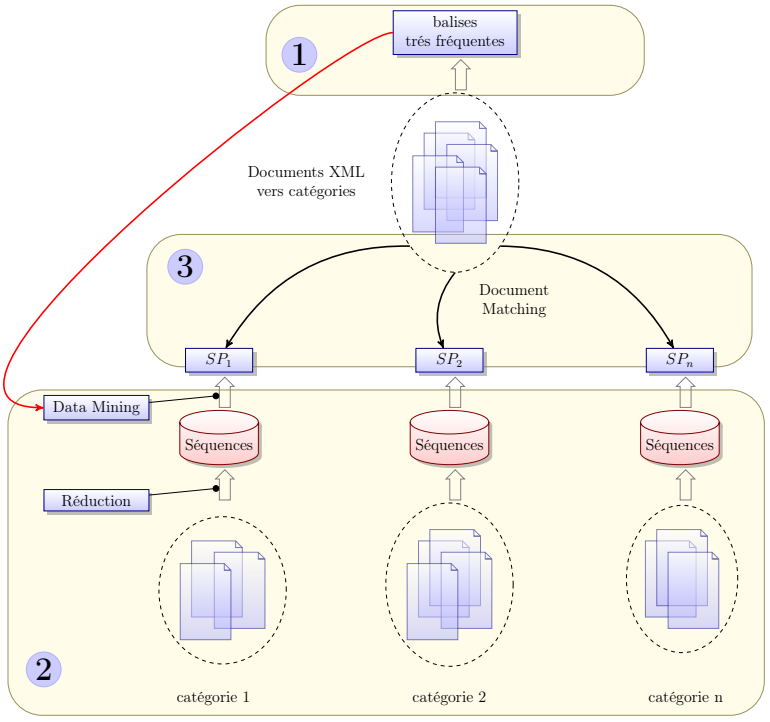

Figure 2.18 – Vue d'ensemble du modèle proposé par [Garboni *et al.*, 2005]

2. **Etape de fouille** (*correspond à l'étape « 2 » sur la figure 2.18*) : Pour chaque catégorie, le but est de transformer chaque document XML en séquence (selon la technique décrite précédemment). En outre, pendant l'opération d'alignement, les étiquettes fréquentes extraites à partir de l'étape 1 sont enlevées. Alors sur chaque ensemble de séquences correspondant aux catégories, une étape de fouille de données prévue est exécutée pour extraire les modèles séquentiels. Pour chaque catégorie C_i, un ensemble SP_i de séquences fréquentes qui caractérise C_i est associé.
3. **Etape d'attribution d'un document à une classe appropriée**(*correspond à l'étape « 2 » sur la figure 2.18*) : cette dernière étape consiste à calculer la similarité basée sur la plus longue sous-séquence commune entre un document D_i et les différentes catégories C_j donnée par :

$$Score(D_i, C_j) = \frac{\sum_{k=1}^{|C_j|} \frac{\left|LCS(D_i, sp_{C_j}(k))\right|}{\left|sp_{C_j}(k)\right|}}{|C_j|}$$

Où $\left|LCS(D_i, sp_{C_j}(k))\right|$ est la plus longue sous séquence commune entre le document $D_i$et le K^ime modèle séquentiel de C_j. Le document D_i est attribué à la classe qui possède le score le plus élevé.

2.4.2.4 Travaux de [Zaki et Aggarwal, 2003]

Cet article propose un classifieur structurel à base de règles "XRules", basé la découverte des règles structurales afin d'effectuer la tâche de classification en partant de l'dée que la

présence d'un genre particulier de modèle structural dans un document de XML est liée à sa probabilité de l'appartenance à une classe particulière.

Chaque document XML est représenté par un arbre enraciné étiqueté et ordonnés, noté par $T = (V, B)$ où V est l'ensemble des nœuds étiquetés et B est l'ensembles des branches (une branche $b = (x, y)$ est une paire de nœuds ordonnés où x est un parent de y). La tâche de classification contient deux phases : une phase d'apprentissage pour découvrir l'ensemble de règles structurelles (*Structural rule-set*), en identifiant un ensemble représentatif de modèles structuraux pour chaque catégorie. Une phase de test pour prévoir la catégorie du nouveau document, en employant le modèle de classification (l'ensemble de règles structurelles).

Phase d'apprentissage : Étant donné un ensemble de documents $\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{D}_i$ où $\mathcal{D}_i$ est l'ensemble de documents associé à la classe $c_j \in C = \{c_1, C_2, \ldots, c_n\}$. L'objectif est de découvrir un ensemble de règles structurelles $R = \{R^1, R^2, \ldots, R^m\}$ où chaque règle est de la forme $R^i : T^i \overset{\pi,\delta}{\Longrightarrow} c^i_j$ avec $\pi \geq \pi^{min}_j$ et $\delta \geq \delta^{min}$.
Cette phase se résume en trois étapes :

- la fouille des règles structurelles fréquentes spécifiques à chaque classe, avec le support et la confiance suffisants. Dans cette étape, on découvert les modèles structuraux fréquents pour chaque classe et puis on produit de ces règles qui satisfont un niveau de support minimum de l classe c_i π^{min}_i, et d'un seuil minimum global de confiance δ^{min}.
- Ordonner les règles selon une relation de priorité. Une fois un ensemble de règles de classification se produit, un procédé est exigé pour donner la priorité à l'ensemble de règle dans un niveau décroissant de la priorité pour éliminer les règles non-prédictives.
- Déterminer une classe spéciale appelée *"classe par défaut"*. Puisqu'un classifieur doit prévoir une classe pour tous les cas de test possibles, cette classe par défaut sera l'étiquette d'un document d'essai, si aucune des règles ne peut être employée pour prévoir la classe de ce document.

Phase de test : La phase de test prend comme entrée le modèle de classification issu de la phase d'apprentissage (collection de règles prédictives R, et d'une classe par défaut), et ensemble de documents XML $\acute{\mathcal{D}}$ dont les classes sont inconnues. Le but de cette phase est de prévoir la classe pour chaque document de test. Il y a deux étapes principales dans le test :

- Recherche de règles : pour chaque document de $\acute{\mathcal{D}}$, on cherche l'ensemble de toutes les règles correspondant à ce document appelé *matching rule-set*, $\mathcal{R}(S) = \left\{ R^i : T^i \overset{\delta^i}{\Longrightarrow} c^i | T^i \preceq S \right\}$
- Prédiction de classe : Combiner les statistiques de l'ensemble de règles du matching $\mathcal{R}(S)$ (*matching rule-set*) pour prévoir la classe la plus adéquate pour le document à classer. si $\mathcal{R}(S) = \emptyset$ alors la classe est prévue pour être la *classe par défaut*.

Corpus	Nb. Documents	Nb. Nœuds	Utilisé pour Structure	Utilisé pour Structure et contenu
Movies	9,463	≈ 190	Oui	Non
IEEE	12,108	≈ 150	Oui	Oui
XML Wikipedia 2006	≈ 150,000	≈ 10,000	Oui	Oui
XML Wikipedia 2007	96,611	≈ $9M$	Oui	Oui

Table 2.2 – Description des corpus utilisés

2.4.3 Analyse des résultats

Avant de présenter notre synthèse sur les différents articles discutés précédemment, nous donnerons une vue globale sur les corpus utilisés pour l'évaluation des classifieurs proposés ainsi que les différentes mesures employées.

2.4.3.1 Corpus utilisés

Les différent articles discutés pendant cet état de l'art, ont été évaluées sur quatre corpus XML différents dont les caractéristiques sont fournies dans le tableau 2.2. Le corpus **Movies** est un corpus artificiel basé sur le corpus d'IMDB [9] tandis que les corpus **IEEE** et **XML Wikipedia** sont des corpus réels de document [Denoyer et Gallinari, 2008], [Denoyer *et al.*, 2007]. Dans certains cas les classes correspondent à une partition normale du corpus, alors que dans d'autres cas, plusieurs classes ont été produits artificiellement d'un corpus homogène comme détaillé ci-dessous :

1. Le corpus **Movies** (*m-db-s-N*) se compose d'environ 9.500 documents de XML qui décrivent des films exprimés en 11 catégories différentes. Le nombre N mesure la difficulté de la tâche de classification : si N est plus élevé ,il y a plus de chevauchement parmi les 11 classes.
2. Le corpus **IEEE** se compose de 12.000 articles scientifiques des journaux d'IEEE dans le format de XML. La tâche de classification s'élève à identifier 18 catégories du corpus d'IEEE qui correspondent aux 18 journaux différents d'IEEE présents dans la collection.
3. Le corpus de **XML Wikipedia 2006** se compose de 150.000 documents XML en anglais partitionnés en 60 catégorie où chaque document appartient à exactement une catégorie et chaque catégorie correspond à un portail de Wikipedia.
4. Le corpus de **XML Wikipedia 2007** se compose de 96,611 documents XML organisés en 21 catégories qui correspondent aux portail de Wikipedia et essentiellement aux catégories thématiques.

9. http://www.imdb.org

Modèle \ Corpus	Movies (0 à 3)	IEEE
[Candillier *et al.*, 2005]	96.8% , 96.6% , 94.7% , 94.2%	94.1%
[Garboni *et al.*, 2005]	95% , 94% , 89% , 80%	

Table 2.3 – Résultats de classification en utilisant le *rappel*

2.4.3.2 Mesures d'évaluation

Différentes mesures ont été employées dans ces travaux, comme le rappel ,la précision , F_1 mesure , rappel micro et macro-moyenne.
Dans le domaine de la classification, pour une catégorie donnée, le *rappel* ρ est le rapport entre le nombre de documents correctement classifiés dans la catégorie et le nombre de tous les documents de cette catégorie. la *précision* π est le rapport entre le nombre de documents correctement classifiés dans la catégorie et le nombre de tous les documents assignés à cette catégorie. Le *rappel macro-moyenne (précisio, F_1 mesure)* ρ^M correspond au moyen du rappel (précision, F_1 mesure) de toutes les catégories. Le *rappel micro-moyenne* ρ^μ correspond au moyen du rappel (précision, F_1 mesure)) des catégories pondérés par la taille des catégories.

$$\pi = \frac{C}{C+F} \quad , \quad \rho = \frac{C}{C+M} \quad , \quad F_1 = \frac{2 * \pi * \rho}{\pi + \rho}$$

$$\pi^M = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|} \quad , \quad \rho^M = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|} \quad , \quad F_1^M = \frac{\sum_{i=1}^{|C|} F_{1i}}{|C|}$$

$$\pi^\mu = \frac{\sum_{i=1}^{|C|} |c_i| * \pi_i}{\sum_{i=1}^{|C|} |c_i|} \quad , \quad \rho^\mu = \frac{\sum_{i=1}^{|C|} |c_i| * \rho_i}{\sum_{i=1}^{|C|} |c_i|} \quad , \quad F_1^\mu = \frac{\sum_{i=1}^{|C|} |c_i| * F_{1i}}{\sum_{i=1}^{|C|} |c_i|}$$

Où : C est le nombre de documents bien classifiés, F est le nombre de documents mal classifiés et M est le nombre de documents non classifiés.

2.4.3.3 Synthèse

Nous constatons des différentes idées sous-jacentes présentées dans ces articles qui reflètent la façon de représenter les documents XML. Les modèles **basés-vecteur** qui transforment d'abord les documents XML à un vecteur ou à un ensemble d'attribues-valeurs et emploient ensuite les modèles vectoriels classiques pour la classification. Les modèles **basés-similarité** définissent une mesure de similarité entre des documents XML et puis emploient cette mesure pour la classification. Les modèles d'**arbres fréquents** concernent les modèles qui emploient une extension des ensembles d'attributs (items) fréquents aux arbres XML.

Les résultats de catégorisation prouvent d'abord que les modèles proposés sont tout à fait bons sur la tâche de classification. Ils ont des scores macro-moyennes qui sont inférieurs à leur micro-moyennes , ça montre que les petites catégories ne sont pas bonnes classifiées par rapport aux grandes. La prise en compte de éléments structuraux améliore ces scores de classification : la prise en compte des attributs, la réplication de textes ou la composition à

Article	Modèle	Micro-F_1	Macro-F_1
[Knijf, 2006]	Sans attribut	33.8%	33.8%
	Avec Attribut	47.9%	52.7%
[Xing *et al.*, 2006]		57.8%	60.0%

Table 2.4 – Résultats de classification en utilisant le *micro et macro-F_1*

Article	Modèle	Rappel Micro-moyenne	Rappel macro-moyenne
[de Campos *et al.*, 2007]	Naive Bayes *(Text only)*	77.6%	58.5%
	Naive Bayes *(Réplication)*	78.1%	63.7%
	OR Gate *(Text only)*	78.9%	76.0%
	OR Gate *(Réplication)*	75.3%	61.2%
[Yang et Zhang, 2007]	SLVM	87.2%	83.9%
[Ghosh et Mitra, 2008]	Noyau Structure *(Cosinus)*	58.4%	51.6%
	Noyau Contenu *(Cosinus)*	82.1%	75.2%
	Noyau Composé *(Cosinus)*	85.7%	84.2%

Table 2.5 – Résultats de classification en utilisant le *micro et macro-rappel*

la fois des contenus textuels et de structures.
Une chose commune à tous les modèles proposés, c'est que les éléments structuraux (balises ou Tags) sont considérés indépendants entre eux où ce n'est pas toujours le cas, car deux balises, par exemples, différentes en formes peuvent représenter le même concept .

2.5 Conclusion

XML est devenu un format standard pour la représentation, l'échange le stockage de données, en décrivant simultanément la structure logique et le contenu des documents et permettant ainsi de représenter l'information sous une forme plus riche que le simple contenu.

Nous avons présenté, dans ce chapitre, quelques techniques de classification des documents XML qui proposent d'utiliser simultanément de contenus et de structures ou de structures seulement, en se basant sur des différentes représentations des documents XML (vecteur, arbre,. . . ,etc). Les expériences menées par ces techniques prouvent que la prise en compte simultané de la structure et du contenu améliore les résultats de classification.

Dans le chapitre suivant, nous présenterons une autre vision pour la classification basée sur l'intégration de la sémantique dans le processus de classification, nous parlerons des ontologies, leur impact, et les manières de les intégrer.

Ontologie et Classification de textes

3.1 Introduction

L'ontologie est l'un des modèles de représentation de connaissances les plus avancés, constituée de concepts liés par des relations, et souvent structurés hiérarchiquement, elle permet d'organiser des connaissances en fonction du domaine considéré. Au cœur du Web sémantique, elle ajoute une couche sémantique au Web classique en décrivant les connaissances contenues dans les ressources. Considérée désormais dans ce domaine comme métadonnées de référence, l'ontologie, ainsi que leur création et leur développement, font l'objet de nombreux travaux de recherche.

Récemment, de nouvelles approches ont intégré l'utilisation de techniques de fouille de données dans le processus d'enrichissement d'ontologies. En effet, les deux domaines, fouille de données et ontologies sont extrêmement liés : d'une part les techniques de fouille de donnée aident à la construction du Web sémantique, d'autre part le Web sémantique aide à l'extraction de nouvelles connaissances. Ainsi, beaucoup de travaux utilisent les ontologies comme un guide de l'extraction de règles ou de motifs, permettant de discriminer les données par leur valeur sémantique et donc d'extraire des connaissances plus pertinentes. Il s'avère à l'inverse que peu de travaux visant à mettre à jour l'ontologie s'intéressent aux techniques de fouilles de données [Jorio *et al.*, 2007].

Dans ce chapitre, nous donnerons quelques définitions de l'ontologie, sa position par rapport aux autres ressources sémantiques. Ensuite, nous entamons les méthodologie, les langages et les outils de construction d'ontologies, et enfin nous détaillerons l'utilisation de ces ressources sémantiques dans le processus de classification de textes.

3.2 L'ontologie

3.2.1 Définitions

Dans le dictionnaire Reverso en ligne[1], le terme ontologie n'apparaît qu'au féminin singulier avec la définition suivante :
“ étude de l'être en tant qu'être, sans tenir compte de ses déterminations particulières ”.

1. http://dictionnaire.reverso.net/francais-definition/

En Informatique, la définition communément admise est celle énoncé par T.GRUBER qui définit *l'ontologie comme une spécification explicite d'une conceptualisation* [Gruber, 1993] . De ce point de vue, la construction d'une ontologie interviendrait après le travail de conceptualisation qui consiste à identifier, au sein d'un corpus, les connaissances spécifiques au domaine de connaissances à représenter.

Cette définition est encore précisée par B.STUDER qui voit *l'ontologie comme une spécification formelle et explicite d'une conceptualisation partagée* [Studer, 1998]. Ainsi, le terme *« spécification explicite »* indique q'une ontologie est un ensemble de concepts, de propriétés, d'axiomes, de fonctions et de contraintes explicitement définis. Le terme *« formelle »* précise que cette conceptualisation doit pouvoir être comprise et interprétée par une machine. *« partagée »* précise l'aspect consensuel du vocabulaire employé. Enfin le terme *« conceptualisation »* implique également l'aspect intentionnel, lié à un objectif à réaliser.

Une définition complémentaire proposée dans [J.CHARLET, 2002] : *" Une ontologie est une spécification normalisée représentant les classes des objets reconnus comme existant dans un domaine. Construire une ontologie, c'est aussi décider d'une manière d'être et d'exister des objets de ce domaine "*. Ainsi, une ontologie répond à des exigences complémentaires et symétriques :

- en tant que spécification, elle définit une représentation formelle des connaissances permettant son exploitation par un ordinateur ,
- en tant que reflet d'un point de vue - partiel- sur un domaine, que l'on cherche le plus consensuel possible, elle fournit une sémantique qui doit permettre de relier la forme exploitable par la machine à sa signification pour les humains.

3.2.2 Composantes d'une ontologie

Concrètement, une ontologie [Chaumier, 2007] modélise les connaissances d'un domaine sous forme d'un réseau de concepts normalisés et d'axiomes. Les concepts sont des classes génériques, définies par leurs relations sémantiques ou leurs propriétés (définition en intension par des conditions nécessaires et suffisantes) ou par la liste des instances relevant de cette classe (définition en extension). L'organisation des concepts est choisie de manière à favoriser leur classification : la structure d'une ontologie comporte donc systématiquement une hiérarchie de spécialisation des concepts et des relations définies par les concepts qu'elles relient.

3.2.2.1 Concepts

Les concepts, aussi appelés classes de l'ontologie, correspondent aux abstractions pertinentes d'un segment de la réalité (le domaine du problème), retenues en fonction des objectifs qu'on se donne et de l'application envisagée pour l'ontologie.

Un concept représente pour un objet matériel, une notion ou une idée [Uschold, 1996]. Il est composé de trois parties : un ou plusieurs termes, une notion et un ensemble d'objets.

- Notion : appelée intention du concept, correspond à la sémantique du concept, elle est définie à travers ses propriétés et ses attributs.
- ensemble d'objets : correspond aux objets définis par le concept, il est appelé extension ou instances du concept.
- Le ou les termes : permettent de désigner le concept. Ces termes sont aussi appelés labels de concept.

Par exemple, le terme « lapin » renvoie à un animal possédant de longues oreilles et une queue et à l'ensemble des objets ayant cette description.

Plusieurs propriétés pouvant être associées à un concept ou portent sur deux concepts dont le but d'être formalisées [Furst, 2002]. Parmi ceux portant sur un concept, nous citons :

- **Généricité** : un concept est générique s'il n'admet pas d'extension. Par exemple : la *vérité* est un concept générique.
- **Identité** : un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept. Cette propriété peut porter sur des attributs du concept ou sur d'autres concepts. Par exemple : le concept *étudiant* porte une propriété d'identité liée au *numéro d'étudiant*, deux étudiants étant identiques s'ils ont le même numéro.
- **Rigidité** : un concept est rigide si toute instance de ce concept en reste instance dans tout les mondes possibles. Par exemple : *humain* est un concept rigide, *étudiant* est un concept non rigide.
- **Anti-rigidité** : un concept est anti-rigide si toute instance de ce concept est essentiellement défini par son appartenance à l'extension d'un autre concept. Par exemple : *étudiant* est un concept anti-rigide car l'étudiant est avant tout un humain.
- **Unité** : un concept est un concept unité si, pour chacune de ses instances, les différentes parties de l'instance sont liées par une relation qui ne lie pas d'autres instances de concepts. Par exemple : les deux parties d'un couteau, manche et lame sont liées par une relation *" emmanché "* qui ne lie que cette lame et ce manche.

Les propriétés portant sur deux concepts sont :

- **Equivalence** : deux concepts sont équivalents s'ils ont même extension. e.g. : étoile du matin et étoile du soir.
- **Disjonction** : deux concepts sont disjoints si leurs extensions sont disjointes. e.g. : homme et femme.
- **Dépendance** : un concept c_1 est dépendant d'un concept c_2 si pour toute instance de c_1 il existe une instance de c_2 qui ne soit ni partie ni constituant de l'instance de c_2. Par exemple : parent est un concept dépendant de enfant (et vice-versa).

3.2.2.2 Relations

Les relations traduisent les associations existant entre les concepts présents dans le segment analysé de la réalité. Ces relations incluent les associations suivantes :

- Sous-classe-de (généralisation – spécialisation) ;
- Partie-de (agrégation ou composition) ;
- Associée-à ;
- Instance de,. . . etc.

Ces relations nous permettent d'apercevoir la structuration et l'interrelation des concepts, les uns par rapport aux autres. Nous pouvons distinguer les types de relations suivants :

- ***Relation taxonomique (ou subsomption)*** : La notion de subsomption (aussi appelée relation « *est un* », relation taxonomique ou relation de spécificité/généricité) est une relation binaire particulière qui implique l'engagement sémantique suivant [Guarino et Welty, 2001] : un concept c_1 subsume un concept c_2 si toute relation sémantique de c_1 est aussi relation sémantique de c_2, en d'autres termes si le concept c_2 est plus spécifique que le concept c_1. Les instances se rapportant au concept c_2 seront des instances de c_1, par contre une partie seulement des instances de c_1 seront des instances de c_2. La relation de subsomption permet d'organiser hiérarchiquement un ensemble de concepts.

- ***Relation associative*** : Les relations « *associatives* » sont des relations d'interaction entre deux concepts qui ne sont pas la relation de subsomption. Ces relations sont soit à des propriétés entre concepts soit à des propriétés d'attribut dans le cas où elles associent un concept à un type de données. La sémantique qui leur est associé est référencée par un label. Elle peut également être précisée à partir de propriétés logiques associées à la relation telles que la transitivité, la symétrie, la fonctionnalité. Par exemple : *" Père_de "*, *" Agé_de "*.

- ***Fonctions*** : constituent des cas particuliers de relations, dans laquelle un élément de la relation, le n^{ime} (extrant) est défini en fonction des $n-1$ éléments précédents (intrants). Par exemple : *" Mère_de "*.

Tout comme les concepts, les relations peuvent être spécifiées par des propriétés intrinsèques à une relation ou liant deux relations. Les propriétés intrinsèques à une relation sont :

- **Propriétés algébriques** : symétrie, réflexivité, transitivité.
- **Cardinalité** : nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept). Les relations portant une cardinalité représentent souvent des attributs. Par exemple : une pièce a au moins une porte, un humain a entre zéro et deux jambes.

Les propriétés liant deux relations sont :

- **Incompatibilité** : deux relations sont incompatibles si elles ne peuvent lier les mêmes instances de concepts. Par exemple : les relations *"être_rouge"* et *" être_vert"* sont

incompatibles.

- **Inverse** : deux relations binaires sont inverses l'une de l'autre si, quand l'une lie deux instances I_1 et I_2, l'autre lie I_2 et I_1. Par exemple : les relations *"Père_de"* et *" Enfant_de"* sont inverses l'une de l'autre.
- **Exclusivité** : deux relations sont exclusives si, quand l'une lie des instances de concepts, l'autre ne lient pas ces instances, et vice-versa. L'exclusivité entraine l'incompatibilité. Par exemple : l'appartenance et la non appartenance sont exclusives.

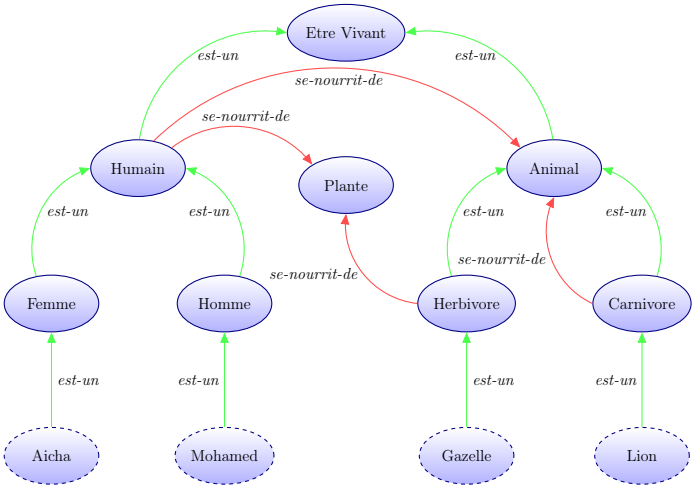

Figure 3.1 – Extrait d'une ontologie sur les êtres vivants

3.2.2.3 Axiomes

Les axiomes sont des expressions qui sont toujours vraies. Leur inclusion dans une ontologie peut avoir plusieurs objectifs : définir la signification des composants, définir des restrictions sur la valeur des attributs, définir les arguments d'une relation, vérifier la validité des informations spécifiées ou en déduire de nouvelles.

Les axiomes ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n'ayant pas un caractère strictement terminologique [Staab et Maedche, 2000]. Un exemple d'une axiome sur les personnes : *"Toute personne ayant un sexe masculin est un homme"*.

3.2.2.4 Instances

les instances (objets ou individus) constituent la définition extensionnelle de l'ontologie, ces instances véhiculent les connaissances à propos du domaine du problème. Elles sont utilisées pour représenter les concepts de l'ontologie. Par exemple : *" Makhlouf "* est une instance du concept *" Etudiant "*, *" Algérie"* est une instance du concept *"Pays"*.

3.2.3 Ontologies et ressources sémantiques

Plusieurs ressources sémantiques existent : vocabulaire contrôlé, thésaurus, taxonomie, ontologies. [Zargayouna, 2005].

Un *vocabulaire contrôlé* est une liste de termes définis par un groupe. Cette liste n'a pas l'ambiguïté du langage naturel. Concrètement, un vocabulaire contrôlé assure qu'un sujet sera décrit en utilisant le même terme préférentiel [2] chaque fois qu'il est indexé, facilitant ainsi la recherche d'informations sur un sujet spécifique. Ainsi, durant une recherche, on peut trouver plus facilement tous les renseignements relatifs à un sujets précis. Par contre, cette liste de termes ne contient pas de signification et n'est pas organisée.

Une *taxonomie* est un *vocabulaire contrôlé* ayant une structure hiérarchique. Elle est sémantiquement pauvre. Elle est souvent créée pour décrire des catégories et des sous-catégories de sujets qu'on trouve dans un domaine donné.

Un *thésaurus* est un vocabulaire contrôlé arrangé dans un ordre connu. Les termes qui le constituent sont organisés et régis par des relations sémantiques et hiérarchiques. Un thésaurus fournit accessoirement des définitions. L'accent est mis sur le choix des termes (on parle alors de descripteurs) et des relations entre termes. Les relations communément exprimées dans un thésaurus sont :

- les relations taxonomiques (de hiérarchie),
- les relations d'équivalence (synonymie),
- relations d'association (relations de proximité sémantique, proche-de, relié-à, etc.).

Le thésaurus peut être exploré par ordre alphabétique ou en parcourant une liste hiérarchique des termes. Un des meta-thésaurus les plus connus est UMLS (Unified Medical Language System) du National Library of Medicine qui définit plus de 40 relations.

Une *ontologie* est une structure hiérarchique des connaissances établie à l'aide de concepts précis pour créer un vocabulaire convenu permettant l'échange d'informations. une ontologie permet, d'une part de décrire les connaissances d'un domaine spécifique et d'autre part de représenter des relations complexes entre les concepts, ainsi que des axiomes et règles qui manquaient aux autres ressources sémantiques et qui permettent d'utiliser des moteurs de raisonnement pour inférer de nouvelles connaissances.

3.2.4 Type d'ontologies

Une classification des ontologies selon [Guarino, 2007] distingue quatre types d'ontologies :

- *Les ontologies générales (Ontologies de haut niveau)* : appelées aussi *ontologies génériques* ou *core ontology*, elle ne sont pas propres à un domaine. En effet, elles véhiculent des connaissances génériques réutilisables à travers différents domaines, ce qui rend leur précision très moyenne. ces ontologies ont une granularité large car elles portent sur

2. Un terme préférentiel est un mot qu'un grand nombre de personnes connaissent et utilisent.

des notions assez génériques telles que l'espace, le temps, les évènements, les états, etc. Les concepts des trois autres types d'ontologies peuvent y faire référence. (Exemples de ce type d'ontologies : SUMO (Suggested Upper Merged Ontology) [3] et BFO (Basic Formal Ontology) [4]).

- *Les ontologies de domaine* : Elles permettent de spécifier les connaissances d'un domaine. Ce type de'ontologies régit un ensemble de vocabulaires et de concepts qui décrivent un domaine d'application, il peut aussi être obtenu en spécialisant un ontologie de haut niveau. De plus, elles ne sont pas propres à une tâche particulière. (Exemples de ce type d'ontologies : MENELAS [5]dans le domaine médical et ENGMATH [6] pour les mathématiciens).

- *Les ontologies de tâches* : elles décrivent le vocabulaire concernant une tâche générique (par exemple : interpréter, classifier, etc), notamment en spécialisant les concepts d'une ontologie de haut niveau. Plus précisément, elle régissent un ensemble de vocabulaires et des concepts qui décrivent une structure de résolution de problèmes inhérente au tâches et indépendante du domaine. (Exemples de ce type d'ontologies : ONTOLINGUA [7]).

- *Les ontologies d'application* : elles sont la combinaison des ontologies du domaine et des ontologies orientés tâche, elles sont un domaine de validité restreint et correspondent à l'exécution d'une tâche. Concrètement elles contiennent des concepts dépendants d'un domaine et d'une tâche particulière, qui sont généralement subsumés par des concepts de ces deux ontologies. Plus précisément, il s'agit de mettre en relation les concepts d'un domaine et les concepts liés à une tâche particulière, de manière à en décrire l'exécution.

3.3 Construction d'ontologies

3.3.1 But de la construction d'une ontologie

Une ontologie définit un vocabulaire commun pour les chercheurs qui ont besoin de partager l'information dans un domaine. Elle inclue des définitions lisibles en machine des concepts de base de ce domaine et de leurs relations. Elle peut être développée pour différents raisons, parmi les quelles :

- Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels.
- Permettre la réutilisation du savoir sur un domaine.

3. http://www.ontologyportal.org/
4. http://www.ifomis.org/bfo
5. http://www-test.biomath.jussieu.fr/Menelas/
6. http://www-ksl.stanford.edu/knowledge-sharing/papers/engmath.html
7. http://ksl.stanford.edu/software/ontolingua/

– Expliciter ce qui est considéré comme implicite sur un domaine.
– Distinguer le savoir sur un domaine du savoir opérationnel.
– Analyser le savoir sur un domaine.

3.3.2 Le cycle de vie des ontologies

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. Un cycle de vie inspiré du génie logiciel comprend une étape initiale d'évaluation des besoins, une étape de construction, une étape de diffusion, et une étape d'utilisation. Après chaque utilisation significative, l'ontologie et les besoins sont réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite. Pour plus de détails, se référer à [Furst, 2002]

La phase de construction peut être décomposée en (03) étapes : conceptualisation, ontologisation, opérationalisation (figure figure 3.3). L'étape d'ontologisation peut être complétée d'une étape d'intégration au cours de laquelle une ou plusieurs ontologies vont être importées dans l'ontologie à construire. Les activités de documentation et d'évaluation sont nécessaires à chaque étape du processus de construction, l'évaluation précoce permettant de limiter la propagation d'erreurs. Le processus de construction peut être intégré au cycle de vie d'une ontologie comme indiqué en figure 3.2.

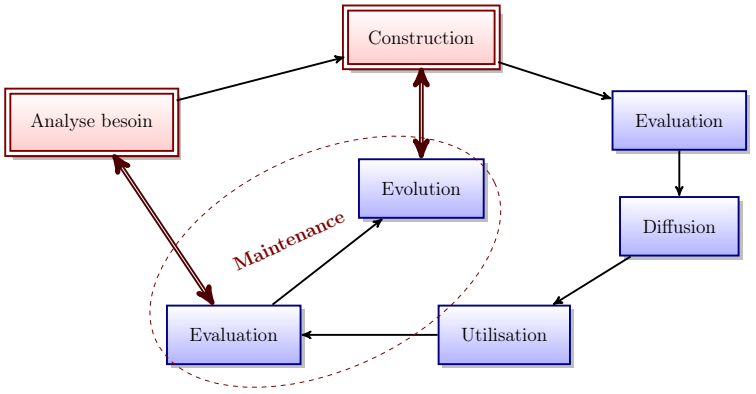

Figure 3.2 – Cycle de vie d'une ontologie

3.3.2.1 L'évaluation des besoins

Le but visé par la construction d'une ontologie se décline en 3 aspects :

1. *L'objectif opérationnel* : il est indispensable de bien préciser l'objectif opérationnel de l'ontologie, en particulier à travers des scénarios d'usage.
2. *Le domaine de connaissance* : il doit être délimité aussi précisément que possible, et découpé, si besoin est, en termes de connaissances du domaine, connaissances de raisonnement, connaissances de haut niveau (communes à plusieurs domaines) .

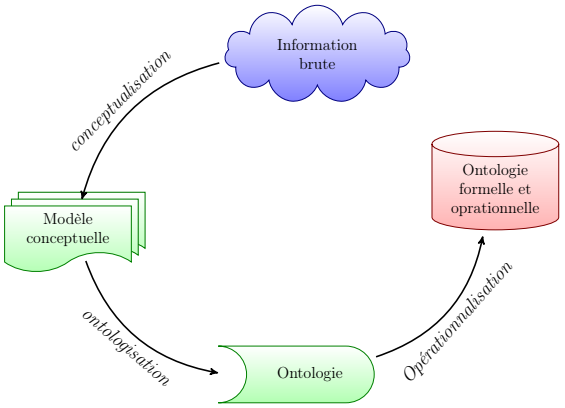

Figure 3.3 – Etapes de construction d'une ontologie

3. *Les utilisateurs* : ils doivent être identifiés autant que possible, ce qui permet de choisir, en accord avec l'objectif opérationnel, le degré de formalisme de l'ontologie et sa granularité.

Une fois le but défini, le processus de construction de l'ontologie peut démarrer, en commençant par la phase de conceptualisation.

3.3.2.2 La conceptualisation

La conceptualisation consiste à identifier, dans un corpus, les connaissances du domaine. Cette conceptualisation peut se décomposer en deux étapes. Tout d'abord, il faut faire le tri entre connaissances spécifiques au domaine et celles qui, bien que présentes dans le corpus, ne participent qu'à l'expression des connaissances du domaine. En outre, s'il est prévu d'intégrer d'autres ontologies, les connaissances spécifiées dans ces ontologies ne doivent pas être prises en compte. La nature conceptuelle (concepts, relations, propriétés des concepts et relations, règles, contraintes, etc) des connaissances ainsi extraites du corpus doit ensuite être précisée. Une fois les ressources cognitives passées au travers du tamis de la conceptualisation, il convient de formaliser, au cours de la phase d'ontologisation, le modèle conceptuel obtenu.

3.3.2.3 L'ontologisation

L'ontologisation est une traduction dans un certain formalisme de connaissances exprimées *a priori* en langage naturel. Le respect de la sémantique du domaine doit être assuré par des engagement ontologique [8] qui garantissent une cohérence entre l'ontologie et le domaine. Ces engagements doivent être garantis par une structuration sémantique des connaissances qui précise les liens sémantiques entre les différentes primitives conceptuelles, en particulier les liens de subsomption entre concepts et entre relations. L'ontologisation doit mener à la construction de hiérarchies de concepts, de relations, mais aussi d'attributs des concepts. Une fois le modèle conceptuel structuré, il faut le traduire dans un langage semi-formel de

8. notion proposée initialement par T. GRUBER comme un critère pour utiliser une spécification partagée d'un vocabulaire

représentation d'ontologies.

3.3.2.4 L'opérationalisation

L'opérationalisation consiste à outiller une ontologie pour permettre à une machine, via cette ontologie, de manipuler des connaissances du domaine. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie. Dans le cas où le langage d'ontologisation n'est pas opérationnel, il est nécessaire, soit d'outiller ce langage, dans la mesure du possible, soit de transcrire l'ontologie dans un langage opérationnel.

Finalement, l'ontologie opérationalisée est intégrée en machine au sein d'un système manipulant le modèle de connaissances utilisé via le langage opérationnel choisi. Mais avant d'être livrée aux utilisateurs, l'ontologie doit bien sur être testée par rapport au contexte d'usage pour lequel elle a été bâtie.

3.3.2.5 L'évaluation et l'évolution d'une ontologie

L'évaluation d'une ontologie se fait *a priori* par des tests correspondants à l'objectif opérationnel de l'ontologie en utilisant des questions de compétences. La validation de l'ontologie en amont de son opérationalisation est souhaitable. Elle évite de propager des erreurs qui, si les réponses fournies par le système aux questions de compétences se révèlent fausses, peuvent être difficilement repérables. La validité des hiérarchies doit donc être testée dès la phase d'ontologisation, aussi bien du point de vue formel que du point de vue sémantique. La validation formelle consiste à vérifier s'il n'y a pas de cycle (définition en boucle, redondance de concepts ou de relations, concept ou de relation isolé des autres). La validation sémantique permet de contrôler que la structure des hiérarchies est correcte vis-à-vis des principes différentiels utilisés.

Si l'évaluation de l'ontologie a échoué, il est nécessaire de la faire évoluer (modifier certaines parties de l'ontologie en s'assurant de ne pas provoquer de nouvelles erreurs). Un autre cas nécessitant l'évolution d'une ontologie est celui où les objectifs changent (contexte d'usage est modifié, ou que le domaine de connaissance est élargit),là on peut décider soit de construire une nouvelle ontologie avec les connaissances à ajouter et l'intégrer dans l'ontologie déjà constituée, soit d'agréger directement les nouvelles connaissances dans l'ontologie existante.

3.3.3 Méthodologies de construction d'ontologie

Nous présentons dans cette sous-section, quelques méthodologies pour le développement d'ontologies à savoir : *"Ontology Development 101"*, *"Unified Methodology"* et EXPLODE. Pour savoir plus au sujet des méthodologies pour le développement d'ontologies, se référer à [Fernandez-Lopez *et al.*, 2002].

3.3.3.1 *"Ontology Development 101"*

Développée par NOY et *McGuinness* en 2001 [Noy et McGuinness, 2001], propose une approche itérative pour le développement d'ontologie. Cette méthodologie se décompose en (07) étapes :

1. Déterminer le domaine et la portée de l'ontologie en répondant à un ensemble de questions fondamentales (*"Quel est le domaine que va couvrir l'ontologie ? "*, *" Dans quel but utiliserons nous l'ontologie ? "*, *" Qui va utiliser et maintenir l'ontologie ? "*, ...etc) et en identifiant les questions de compétence d'ontologie.
2. Envisager une éventuelle réutilisation des ontologies existantes.
3. Énumérer les termes importants dans l'ontologie.
4. Définir les classes et la hiérarchie des classes en suivant une des multiples approches existantes (de haut en bas, de bas en haut, combinée).
5. Définir les propriétés des classes (attributs).
6. Définir les facettes des attributs (Type de valeur, domaine, cardinalité).
7. Créer les instances des classes.

3.3.3.2 *"Unified Methodology"*

Proposée par USCHOLD en 1996 [Uschold, 1996] regroupe les méthodologies de TOVE (*TOronto Virtual Enterprise*) et d'ENTREPRISE dans un cadre unique. La méthodologie est établie en (05) étapes :

1. Identifier le but de l'ontologie, en particulier identifier et caractériser la gamme des utilisateurs prévus, identifier les utilisations de l'ontologie, identifier assez-généralement, les scénarios de motivation et les questions de compétence.
2. Décider à quel point l'ontologie doit être-formelle. Ceci est déterminé dans une grande partie par le but et les utilisateurs de l'ontologie.
3. Identifier la portée de l'ontologie. Deux manières principales de procéder à ce processus sont :
   - Créer des scénarios détaillés qui surgissent dans les applications.
   - Utilisation un *brainstorming* pour réaliser un travail plus complet et plus précis de portée.
4. Établir l'ontologie elle-même, à savoir produire les définitions et décider comment arranger ces définitions d'une manière particulière, par conséquent en structurant l'ontologie.
5. Évaluer l'ontologie et commencer un cycle de révision si nécessaire.

3.3.3.3 *" EXPLODE " (Extreme Programming for Lightweight Ontology DEvelopment)*

Présentée en 2002 par HRISTOZOVA et STERLING [Hristozova et Sterling, 2002], elle intègre des idées de la méthodologie *eXtreme Programming methodology* () et elle est parti-

culièrement appropriée aux environnements dynamiques et ouverts grâce à sa concentration sur la rétroaction et l'évaluation immédiates. La méthodologie ***" EXPLODE "*** se compose de (07) étapes, qui peuvent être réitérées jusqu'à ce que l'ontologie désirée soit créé :

1. Chercher les besoins du système. Les données recueillies seront employées pour créer une ontologie de base qui se compose d'un nombre restreint de concepts qui sont inévitables pour un domaine particulier.
2. définir les questions de compétence.
3. Réaliser un test de validation pour vérifier si les questions de compétence peuvent être répondues en utilisant l'ontologie existante.
4. En cas de résultat négatif à l'étape précédente, réaliser un test de redondance afin de contrôler l'existance de noms similaires pour les classes, les attributs ou les relations.
5. Estimer le coût et le temps de construction et de contrôle plus tard de l'ontologie développée.
6. Intégrer continûment l'ontologie avec d'autres modules de système (matériel/logiciel).
7. Exécuter des tests d'acceptation pour savoir d'avance ce que le système devrait faire et le comprendre.

3.3.4 Langages de représenation d'ontologies

3.3.4.1 RDF : *Resource Description Framework*

RDF (*Ressource Description Framework*) a été développé par le W3C[9]. La première ébauche de RDF a été publiée en octobre 1997, et les spécifications sont devenues une recommandation du W3C en février 1999. La dernière mise à jour comprenant des changements dans la terminologie, la syntaxe et les concepts manipulés, date de Février 2004 [Owl, 2004]. RDF est un langage de représentation des ressources du web, et en particulier les métadonnées (telles que le titre, l'auteur, la date de modification d'une page, etc.). Un document RDF est un ensemble de triplets de la forme ⟨ sujet, prédicat, objet ⟩. Où le sujet est la ressource à représenter, le prédicat est la propriété et l'objet est la valeur de cette propriété. Les éléments de ces triplets peuvent être des URIs, des littéraux ou des variables. Cet ensemble de triplets peut être représenté de façon naturelle par un multigraphe orienté étiqueté où les éléments apparaissant comme sujet ou objet sont des sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination est son objet. Dans l'exemple de la figure 3.4, on essaye d'expliquer que l'objet *TSS* a pour nom *Inertie totale* et a pour valeur 125.323 . Ce document peut être sérialisé en machine par un document RDF/XML (pour faciliter l'échange de données) (figure 3.5).

Même si RDF est une amélioration (« au niveau sémantique »), il reste insuffisant comme langage de représentation de connaissances. Il manque de primitives et possède peu de mécanismes pour gérer l'évolution des schémas.

9. World Wide Web Consortium

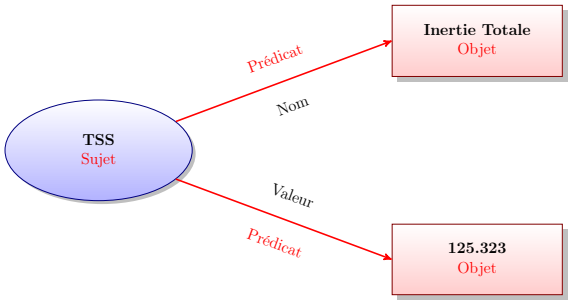

Figure 3.4 – Représenation du graphe RDF de la description de l'objet TSS

```
<?xml version="1.0">
    <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements1.1/">
        <rdf:Description rdf:about="TSS">
            <nom>Inertie Totale</nom>
            <valeur>125.323</valeur>
        </rdf:Description>
    </rdf:RDF>
```

Figure 3.5 – Exemple de la notation RDF/XML

3.3.4.2 RDF-S : *RDF Schema*

RDF Schema est un langage de spécifications de schéma associé à RDF. RDF-S offre une représentation du vocabulaire défini sur RDF pour permettre la modélisation des objets du modèle. Les primitives RDF-Schema les plus pertinentes sont : *rdf :ressource* qui est la classe la plus générale. Elle possède deux sous classes *rdfs :class* et *rdfs :property*. En spécifiant un schéma spécifique à un domaine en RDF-S, les classes et propriétés vont devenir des instances de ces deux classes. *rdfs :class* dénote l'ensemble des classes (dans un sens orienté objet), idem pour *rdfs :property* où chaque propriété sera une instance de cette classe. *rdfs :subClassOf* définit la relation de sous-classe entre classes. Cette relation est transitive donc si A est sous-classe de B et B est une sous-classe de C Alors A est une sous-classe de C. RDF-S interdit la réflexivité de cette relation ainsi que l'héritage cyclique, une classe ne peut pas être sous-classe d'elle-même ni d'une classe fille. *rdfs :subPropertyOf* définit à l'instar de *rdfs :subClassOf* une hiérarchie entre les propriétés. RDF-S permet aussi de définir le domaine et le type des propriétés par *rdfs :domain*, *rdfs :type*, *rdfs :range*.

Malgré que RDF-S permet de représenter des ontologies simples, il reste néanmoins pas assez expressif. Une mutualisation des efforts au sein du W3C a conduit à la définition d'un autre langage, OWL.

3.3.4.3 TopicMaps

Les TopicMaps (cartes Topiques) [DOE, 2001] sont un standard ISO. C'est une proposition concurrente à RDF(S). Elles comprennent quatre primitives :

– Les topics qui peuvent être n'importe quel sujet ou entité.

– Les noms donnés aux topics.
– Les occurrences qui sont des ressources, disposant d'une URI, qui peuvent être liées à des topics.
– Les portées permettent de spécifier le contexte dans lequel une relation est valide.

Les TopicMaps ne disposent pas d'une sémantique claire ce qui peut procurer une liberté d'utilisation mais aussi limiter ses applications. Une des particularités des TopicMaps, par rapport à RDF(S) du W3C, est la notion de *« scope »* qui permet de définir des contextes différents dans lesquels les éléments nommés identiquement peuvent avoir des significations différentes. Par contre, la notion de hiérarchies de classes n'existe pas dans les TopicMaps. Les TopicMaps permettent aussi bien l'expression de méta-données que l'exploitation des relations entre éléments pour différentes tâches, par exemple l'aide à la navigation sur le Web sémantique.

3.3.4.4 OWL : *Web Ontology Language*

Le langage OWL (*Web Ontology Language*) [Owl, 2004] a été conçu pour ajouter plus de sémantique à RDF et RDF-S ainsi que des mécanismes d'inférence. C'est un nouveau standard du Web sémantique. OWL possède des sous-langages avec des expressivités différentes et des pouvoirs d'inférence en conséquence (voir figure 3.6 : OWL LITE, OWL DL et OWL FULL.

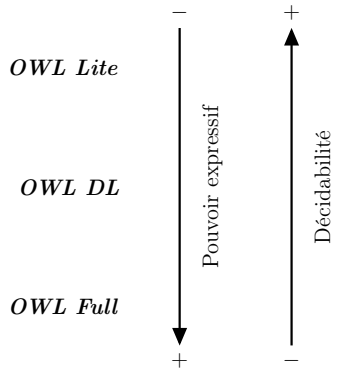

Figure 3.6 – Pouvoir expressif vs. décidabilité pour OWL

– *OWL LITE* : est utile aux applications qui ont besoin des hiérarchies de classifications et des caractéristiques de contraintes simples.
– *OWL DL* : est utile aux applications qui demandent un maximum d'expressivité tout en maintenant la complétude et la décidabilité. OWL DL contient tous les constructeurs du langage OWL mais avec des restrictions (par exemple, l'héritage multiple est interdit).
– *OWL FULL* : destiné aux applications qui demandent un maximum d'expressivité avec la liberté syntaxique de RDF sans aucune garantie de calculs. Il relaxe les contraintes de OWL DL et de ce fait viole certains raisonnements des logiques de description.

OWL LITE assure fondamentalement la classification ainsi que quelques contraintes simples. Par exemple, il prend en compte les cardinalités mais les restreint à 0 ou 1. *owl :minCardinality*, *owl :maxCardinality* spécifient les cardinalités minimales et maximales et *owl :cardinality* exprime une seule cardinalité si la valeur min et max coïncident. OWL reprend tous les constructeurs RDF ainsi que le vocabulaire de RDF-S à savoir *rdfs :class*, *rdfs :subClassOf*, *rdfs :property*, *rdfs :subPropertyOf*, *rdfs :domain* et *rdfs :range*. Les classes *owl :ObjectProperty* et *owl :DatatypeProperty* sont des sous-classes de la classe *rdfs :property*. OWL LITE permet d'exprimer des caractéristiques sur les propriétés :

- *owl :inverseOf* permet d'exprimer qu'une propriété est l'inverse d'une autre.
- *owl :TransitiveProperty* permet de spécifier qu'une propriété est transitive.
- *owl :SymetricProperty* exprime la symétrie d'une propriété.
- *owl :FunctionalProperty* permet d'exprimer qu'une propriété a au plus une valeur unique.
- *owl :InverseFunctionalProperty* exprime que l'inverse de cette propriété.

Pour exprimer des contraintes sur les propriétés OWL fournit *owl :allValuesFrom* qui exprime la quantification universelle et *owl :someValuesFrom* la quantification existentielle. OWL LITE permet d'exprimer des relations d'égalité ou de différence entre :

- Individus, par *owl :sameAs*, *owl :differentFrom* et *owl :allDifferent*.
- Classes, par *owl :equivalentClass*.
- Propriétés, par *owl :equivalentProperty*.

OWL permet aussi d'exprimer l'intersection par *owl :intersectionOf* mais en limite l'usage.

OWL DL inclut tous les constructeurs du langage OWL mais avec certaines contraintes. Une séparation de type est nécessaire, une classe par exemple ne peut pas être aussi un individu ou une propriété. Il lève certaines contraintes de OWL LITE comme la restriction sur les cardinalités.
Voici les constructeurs de OWL DL (mais aussi de OWL FULL) :

- *owl :oneOf* permet de décrire une classe par son extension (la liste de ses individus).
- *owl :disjointWith* permet d'affirmer que deux classes sont disjointes (n'ont aucune valeur commune).
- *owl :unionOf* permet d'exprimer qu'une classe est l'union de deux autres.
- *owl :ComplementOf* permet d'exprimer qu'une classe est le complément d'une autre.
- *owl :hasValue* que la valeur d'une propriété est un individu.

OWL FULL reprend tous les constructeurs d'OWL DL mais il les interprète de manière plus large. Il relâche toutes les contraintes, une classe par exemple peut être vue comme un individu (intension) ou une collection d'individus (extension). Il permet aussi de rajouter du sens au vocabulaire prédéfini de RDF ou OWL.
Ce gain en expressivité détériore la complexité. En effet OWL Full n'est pas décidable.

3.3.5 Outils de construction d'ontologie

De nombreux outils de construction d'ontologies utilisant des formalismes variés et offrant différentes fonctionnalités ont été développés. Tous ces outils offrent des supports pour le processus d'ontologisation, mais peu offrent une aide à la conceptualisation [Furst, 2002]. On peut cependant citer TERMINAE, qui, à travers l'outil d'ingénierie linguistique LEXTER, permet d'extraire d'un corpus textuel les candidats-termes d'un domaine [TERMINAE, 2002]. Ces concepts doivent ensuite être triés par un expert et organisés hiérarchiquement, puis la sémantique du domaine est précisée à travers des axiomes.

Les outils d'aide à la construction d'ontologie sont plus ou moins indépendants des formalismes de représentation. DOE (DIFFERNTIAL ONTOLOGY EDITOR ([DOE, 2002]) offre la possibilité de construire les hiérarchies de concepts et relations en utilisant les principes différentiels énoncés par *B. Bachimont*, puis en ajoutant les concepts référentiels. La sémantique des relations est ensuite précisée par des contraintes. Ce n'est qu'une fois l'ontologie ainsi structurée qu'elle est formalisée en utilisant la syntaxe XML (une extension permettant la transcription en RDF(S)/DAML+OIL est prévue).

De même, l'outil ODE (ONTOLOGY DESIGN ENVIRONMENT [ODE, 2002]) permet de construire des ontologies au niveau connaissance, comme le préconise la méthodologie METHONDTOLOGY proposée par A.GOMEZ-PEREZ. L'utilisateur construit son ontologie dans un modèle de type frame, en spécifiant les concepts du domaine, les termes associés, les attributs et leurs valeurs, les relations de subsomption (est-un).

ONTOEDIT (ONTOLOGY EDITOR ([ONTOEDIT, 2002]) est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. ONTOEDIT intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition.

Parmi les outils non liés à des formalismes de représentation, citons pour finir PROTÉGÉ [PROTEGE, 2002] interface modulaire permettant l'édition, la visualisation, le contrôle (vérification des contraintes) d'ontologies, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Le modèle de connaissances sous-jacent à PROTÉGÉ est issu du modèle des frames et contient des classes (concepts), des slots (propriétés) et des facets (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés. PROTÉGÉ autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

3.3.6 Critères d'évaluation d'ontologies

Nous pensons que les critères pertinents pour évaluer une ressource terminologique et ontologique sont ceux qui doivent être respectés lors de la construction de cette même ressource. Ainsi, il est important de respecter un certain nombre de contraintes pour obtenir une modélisation de qualité, à la fois adaptable et maintenable. Nous présentons ci-dessous une série de principes, issus de la littérature, qui nous semblent particulièrement pertinents [Borgo *et al.*, 1996], [Gruber, 1993] :

- **Clarté et Objectivité** : l'ontologie doit fournir la signification des termes définis en donnant des définitions objectives. Les ambiguïtés doivent être réduites, quand une définition peut être axiomatisée, elle doit l'être. Dans tous les cas, des définitions en langage naturel doivent être fournies.
- **Perfection** : une définition exprimée par des conditions nécessaires et suffisantes est préférable à une définition partielle.
- **Cohérence et Extensibilité** : une ontologie doit être cohérente pour permettre des inférences conformes aux définitions. Les axiomes doivent être consistants. La cohérence des définitions en langage naturelle doit être vérifiée autant que faire se peut. Ainsi, l'ajout de nouveaux concepts à l'ontologie ne devrait pas entraîner la révision des définitions existantes. Plus généralement, l'ontologie doit être construite de telle manière que l'on puisse l'étendre facilement, sans remettre en cause ce qui a déjà été fait.
- **Biais d'encodage minimal** : l'ontologie doit être conceptualisée indépendamment de tout langage d'implémentation. Le but étant de permettre le partage des connaissances, contenues dans l'ontologie, entre différentes applications utilisant des langages de représentation différents.
- **Engagements ontologiques minimaux** : une ontologie doit faire un minimum d'hypothèses sur le monde : elle doit contenir un vocabulaire partagé mais ne doit pas être une base de connaissances comportant des connaissances supplémentaires sur le monde à modéliser. Autrement dit, elle doit choisir de représenter un point de vue en particulier sur le domaine qui l'intéresse.
- **Principe de distinction ontologique** : à chaque fois que l'on peut identifier et isoler un noyau de propriétés considérées comme invariables pour une classe (critère d'identité), il faut créer une nouvelle classe de concepts.
- **Minimiser la distance sémantique entre les concepts enfants de mêmes parents** : il s'agit de la distance minimale entre les concepts enfants de mêmes parents. Les concepts similaires sont groupés et représentés comme des sous-classes d'une classe, et doivent être définis en utilisant les mêmes primitives, sachant que les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie.

3.4 Ontologies et Classification de textes

L'utilisation des ressources sémantiques (thésaurus, ontologies, etc.) en classification peut intervenir lors de la phase de classification ou lors de la phase de représentation des documents.

3.4.1 Phase de représentation

Les documents peuvent être représentés par des concepts au lieu des termes (mots enraciné ou non, lemmes,. . . , etc) ainsi les termes référant au même concept seront représentés par la même unité d'indexation. Cette représentation permet d'avoir un grand pouvoir d'expression mais peut par ce fait ralentir les traitements, et la construction des descriptions sémantiques associées à chaque document n'est pas une tâche facile car l'indexation automatique dans ce cas pose des problèmes notamment celui de l'ambiguïté des termes (homonymie et polysémie).

[Elberrichi *et al.*, 2008] propose d'utiliser le thésaurus WordNet [10] dans la catégorisation de textes, cette approche se compose de deux phases : apprentissage et classification. La phase d'apprentissage consiste à générer une nouvelle représentation de textes basée sur la fusion de termes avec leurs concepts associés (addition, remplacement, concept seulement), en cas d'ambiguïté d'association entre terme et concepts, le terme est représenté soit par tous les concepts associés, soit par le concept le plus approprié (le premier), ensuite pour réduire la dimension et sélectionner les concepts les plus représentatifs des documents , la méthode χ^2 multivariables est utilisée afin de créer un profil pour chaque catégorie. La phase de classification consiste à générer des vecteurs pondérés ($tf * idf$) pour toutes les catégories, ensuite une mesure de similarité (*Cosinus*) est utilisée pour calculer la distance entre le vecteur du document à classer et tous les vecteurs de catégories. En conséquence, le document sera assigné à la catégorie dont le vecteur est le plus proche avec le vecteur de document.

L'idée du modèle proposé par [Theobald *et al.*, 2003] est d'enrichir l'espace des représentation des documents XML , en plus des mots, par des annotations (paires de balise-mot), structure (des brindilles ou petites branches et des chemins), et l'information ontologique. En analysant un document XML, les mots sont extraits à partir du contenu textuel en éliminant les mots vides et en identifiant les noms, ensuite ces mots sont combinés avec les balises correspondantes dans des paires qui seront codées dans des termes simples de la forme *balise$mot* (par exemple « *car$Passat* »). Les brindilles sont une manière spécifique de diviser la structure graphique des documents XML en ensemble de petites unités tout en gardant l'ordre et les relations entre les éléments, les brindilles sont codées sous la forme « *balise-enfant gauche $ balise-père $ balise-enfant droit* » , par exemple : « *recherche $page d'accueil$enseignement* » , avec la balise « *page d'accueil* » étant le père des deux balises « *recherche* » et « *enseignant* » . les balises et les termes sont associés à des concepts d'une

10. WordNet : une large base de données lexicale, voir l'annexe

ontolgie basée sur WordNet, chaque balise ou terme est associé à un ou plusieurs sens-mots (*word senses*). En cas de polysémie, le concept aproprié est determiné par comparaison du contexte du terme ou balise avec tous les contextes des concepts associés.

[Taghva *et al.*, 2003] propose un modèle de classification des emails basée sur une ontologie, Les email ont été à l'origine contrôlés dans un système de LOTUS NOTES qui peut convertir des email en langage XML en utilisant un format appelé DXL [11]. Pour transformer ces documents DXL en format compréhensible aux CLIPS [12], un programme a été écrit pour prendre l'information de divers champs dans le document de DXL et leur écrire aux facettes d'instances des classes CLIPS. Une fois que les emails sont convertis en collection d'instances CLIPS, il peut être chargé dans un programme avec des définitions et des instances de classes produites par Protégé (figure 3.7). des règles prédéfinies permettent de relier les instances d'emails avec les instances de classes de l'ontologie. Ce programme CLIPS énumérera les descripteurs (unités d'indexation) découverts pour chaque email. Ces descripteurs sont alors passés à un classificateur bayésien pour classification.

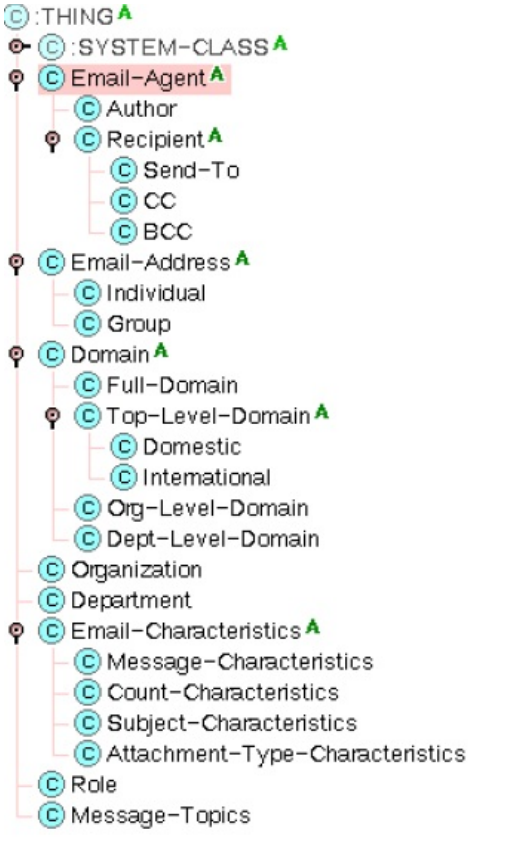

FIGURE 3.7 – Classes de l'ontologie définie par [Taghva *et al.*, 2003]

11. Domino eXtended markup Language
12. C Language Integrated Production System

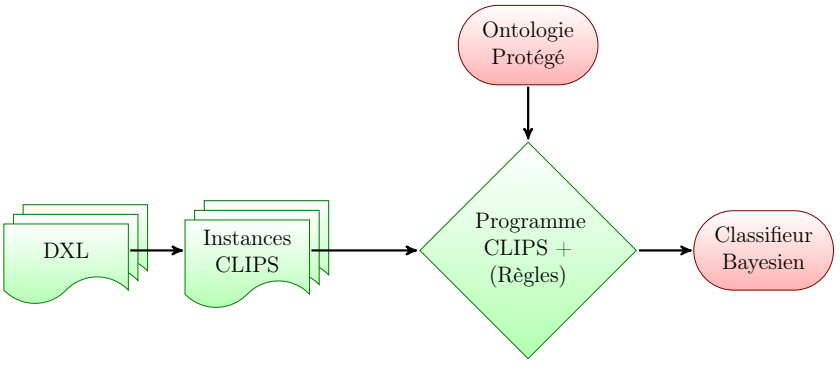

Figure 3.8 – Modèle proposé par [Taghva *et al.*, 2003]

3.4.2 Phase de classification

L'intérêt d'utiliser des ressources sémantiques en classification est de pouvoir classifier les documents qui partagent le maximum de concepts plutôt que le maximum de termes (mots ou phrases, . . . , etc).

[Uma *et al.*, 2007] propose une modèle de classification multi-niveaux basé sur une Machine à vecteurs de support (SVM) en utilisant une ontologie du domaine informatique construite à partir du DMOZ Directory [13] (figure 3.9). Après une étape de pré-traitement des documents (transformation, élimination des mots vides, stemmatisaion), les mot-clés qui sont requis pour la classification sont extraites parmi les termes jugés positifs par un classifieur SVM, ces mot-clés sont ensuite, associés aux concepts de l'ontologie pour classifier les documents.

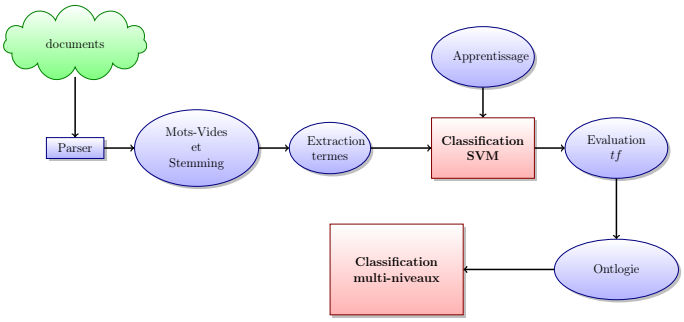

Figure 3.9 – Architecture du système proposé par [Uma *et al.*, 2007]

Une approche un peu similaire, proposée par [Noh *et al.*, 2003], présente un classifieur des pages Web basé sur une ontologie adaptative (figure 3.10). L'identification des termes représentants un ensemble de classes (Concepts de l'ontologie définie sur le domaine des pages web à classifier, voir la figure 3.10) est faite, dans un premier temps, par le calcul des produits de la fréquence des termes et la fréquence des documents (Pages Web), ensuite, le calcul du gain de l'information de chaque terme permet d'établir un ordre des priorités entre

13. http://www.dmoz.org

les termes. Ces termes sont compilés en un ensemble de règles de classification de termes en utilisant des algorithmes d'apprentissage automatique (C4.5, Bayesien Naîf). Les règles compilées permettent de classifier n'importe les page Web dans une catégorie définie dans une ontologie de domaine.

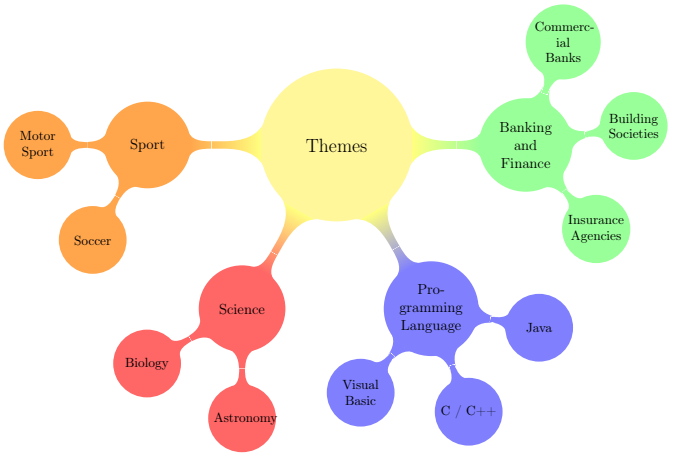

Figure 3.10 – Ontologie pour quatre thèmes proposée par [Noh *et al.*, 2003]

Dans le domaine de la bio-informatique, [Young-Sook *et al.*, 2003] propose un modèle basé sur l'utilisation de l'ontologie GENIA [14]dans un processus de classification des entités nommées (DNA, RNA, Protéine, ... etc). Le processus est divisé en deux tâches : l'identification et la classification des entités nommées (figure 3.11).

- La tâche de l'identification peut être formulée comme une classification binaire où chaque mot des textes est associé à T si ce mot fait partie d'une entité nommée, et à O dans le cas contraire, ce qui permet de créer un dictionnaire de mots-entité. Cette tâche est réalisée à l'aide d'un classifieur SVM.
- La tâche de classification a pour but d'associer les entités identifiés lors de la phase précédente aux classes sémantiques de l'ontologie GENA. Cette tâche est aussi réalisée à l'aide d'un classifieur SVM.

3.4.3 Synthèse

Les résultats obtenus par les expérimentions menées par les différents modèles proposés, montrent l'apport de l'intégration des ressources sémantiques dans le processus de classification que soit dans la phase de l'indexation ou la phase de classification.
Pour la collection Reuters-21578 [15], [Elberrichi *et al.*, 2008] a obtenu une performance de F-mesure atteint $\approx 72\%$ par rapport à une représentation traditionnelle (sac de mots) dont F-mesure $\approx 67\%$.

14. http://www-tsujii.is.s.u-tokyo.ac.jp/genia/
15. disponible à l'adresse http://www.daviddlewis.com/ressources/testcollections/reuters21578

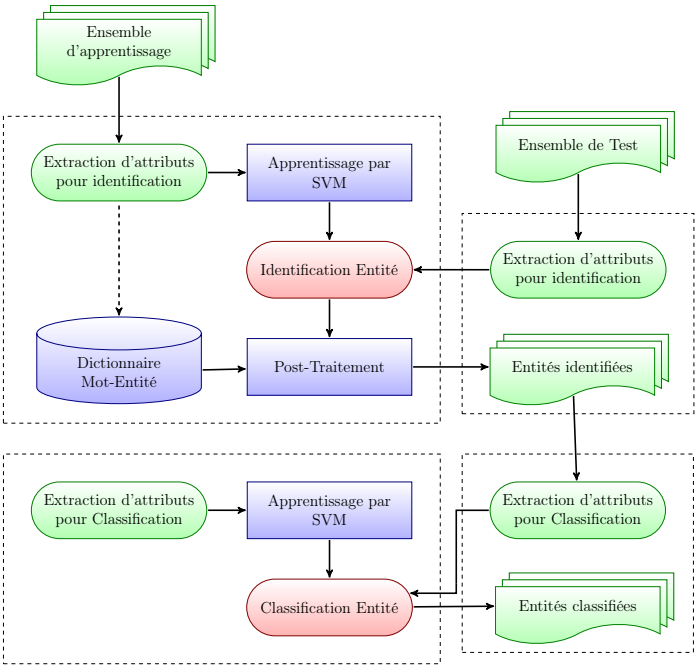

Figure 3.11 – Modèle proposé par [Young-Sook *et al.*, 2003]

Pour le corpus GENA [16], [Young-Sook *et al.*, 2003] a obtenu une performance de F-mesure atteint $\approx 78\%$ par rapport à une représentation de base dont F-mesure $\approx 74\%$.

3.5 Conclusion

Dans ce chapitre, nous avons vu un aperçu sur les ontologies comme des ressources sémantiques, leurs langages et outils de construction. Nous avons discuté l'intérêt que présente l'utilisation des ces ressources dans le processus de classification de textes.

Nous proposons dans le chapitre suivant, de décrire notre contribution , en présentant la représentation des documents XML que nous avons choisie, la définition de l'ontologie que nous intégrons dans le processus de la représentation, ainsi que la méthode de la classification pour laquelle nous avons optée.

16. GENIA Corpus 3.0 disponible à l'adresse http://www-tsujii.is.s.u-tokyo.ac.jp/genia/topics/Corpus/3.0/GENIA3.0

Contribution et Validation

4.1 Introduction

Ce chapitre présente notre contribution, nous donnerons une vue générale du corpus INEX utilisé, ensuite nous présenterons l'architecture globale de notre système en définissant l'approche utilisée pour représenter les documents XML, ainsi que la définition de notre ontologie et la façon de l'intégrer dans le processus de catégorisation. Enfin, nous présentons les résultats d'expérimentation du système sur le corpus de validation.

4.2 Description du corpus utilisé

Le corpus INEX *INitiative for the Evaluation of XML Retrieval* à été créé depuis 2002 dans le cadre d'une campagne d'évaluation des moteurs de recherche XML, il est composé de 96000 documents extraits du corpus *Wikipedia XML Corpus* [Denoyer et Gallinari, 2006], il est divisé en deux parties :

- Partie d'apprentissage composée de 50% des documents,
- Partie de test composée de 50% des documents restants.

Le corpus est téléchargeable du site XML Mining [1]. Les documents sont organisés dans 21 catégories qui correspondent à différents portails de *Wikipedia* et correspondent fondamentalement aux catégories thématiques. Les tableaux 4.1 et 4.2 donnent quelques statistiques du corpus complet.

A noter que les catégories ne sont pas bien équilibrées : quelques catégories sont grandes - *Portal : Law* se compose environ de 25% des documents - tandis que d'autre sont très petites : *Portal : Music* se compose environ de 0.5% des documents. C'est intéressant 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

1. http://xmlmining.lip6.fr

	Total	Apprentissage	Test
Nombre de documents	96,611	48,306	48,305
Nombre de nœuds	$\approx 9M$	4,505,141	4,487,819
Nombre de mots distincts	446,916		
Nombre de mots	33,944,462	17,261,996	16,682,466
Longueur moyenne des documents	351.4	357.3	345.5
Nombre de balises distinctes	$\approx 5,800$		
Taille du corpus	$\approx 720MO$	$\approx 360MO$	$\approx 360MO$

Table 4.1 – Statistiques du Corpus INEX

Id	Categorie	Taille
2112299	Portal : Law	24213
1597184	Portal : Literature	16929
1484914	Portal : Sports and games	14595
1620218	Portal : Politics	1355
1480358	Portal : Art	7624
1886386	Portal : Physics	5149
3091788	Portal : Christianity	4671
2773006	Portal : Chemistry	4567
1685758	Portal : History	3246
3091127	Portal : Spirituality	2704
2914908	Portal : Sexuality	2402
2879927	Portal : War	2217
1507239	Portal : Aviation	1217
2328885	Portal : Formula One	1188
1486363	Portal : Astronomy	1105
1895383	Portal : Trains	953
2314377	Portal : University	605
2635947	Portal : Comics	600
2257163	Portal : Pornography	458
2263642	Portal : Writing	412
474166	Portal : Music	401

Table 4.2 – Descrpition des différentes catégories

Portail : Chistianity et *Portail : Spirituality*.

Une autre chose à prise en compte, c'est lors de phase de construction du corpus de base *Wikipedia XML Corpus* c-à-d transformer le contenu de l'encyclopédie Wikipédia en documents XML, différentes balises sont introduites pour représenter les différentes parties d'un document, on distingue deux type de balises [Denoyer et Gallinari, 2006] :

- **Balises générales** (*article, section, paragraph, ...*) qui ne dépendent pas du langage de la collection. Ces balises correspondent aux informations structurelles contenues dans le format *wikitext*, par exemple :
  == *Main part* == est transformé en `<title>` *Main part* `<title>`.
- **Balises *Templates*** (*template_ weblink, template_ infobox, ...*) représentent l'information contenue dans les gabarits de Wikipédia. Ces gabarits sont employés pour représenter un type réitéré d'information. Par exemple, chaque pays décrit dans Wikipédia commence par une table contenant sa population, langue, taille, ...etc. Afin de rendre uniforme ce type d'information, Wikipédia emploie des gabarits ou style. Ces gabarits sont traduits en XML en utilisant des balises commençant par *template_...* (par exemple : `<template_country>`).

4.3 Vue globale du modèle proposé

Une vue globale de notre modèle proposé est donnée par la figure 4.1, notre modèle se décompose de (04) modules :

- **Module d'Analyse** : le rôle de ce module est d'analyser (*parser*) les documents XML du corpus et les transformer en textes plats (chemins, contenus textuels).
- **Module d'étiquetage** : Ce module prends en entrée les contenus textuels issus module de formatage, son rôle est d'associer chaque mot du contenu à sa catégorie syntaxique

à l'aide d'un étiqueteur *POS Tagger*.

- **Module Sémantique** : Ce module assure les fonctionnalités suivantes :
  - Relier chaque balise (élément d'un chemin) à un concept d'ontologie associé.
  - Relier chaque mot du contenu textuel à son voisinage sémantique suivant sa catégorie syntaxique.
  - Lancer une procédure de désambiguïsation basée sur un voisinage contextuel en cas d'ambiguïté,
- **Module d'indexation** : Ce module réalise le passage de la représentation textuelle des documents XML vers une représentation vectorielle.
- *Module de classification* : Ce module vise à établir un modèle de classification basé sur la méthode SVM (*Support Vector Machines*).

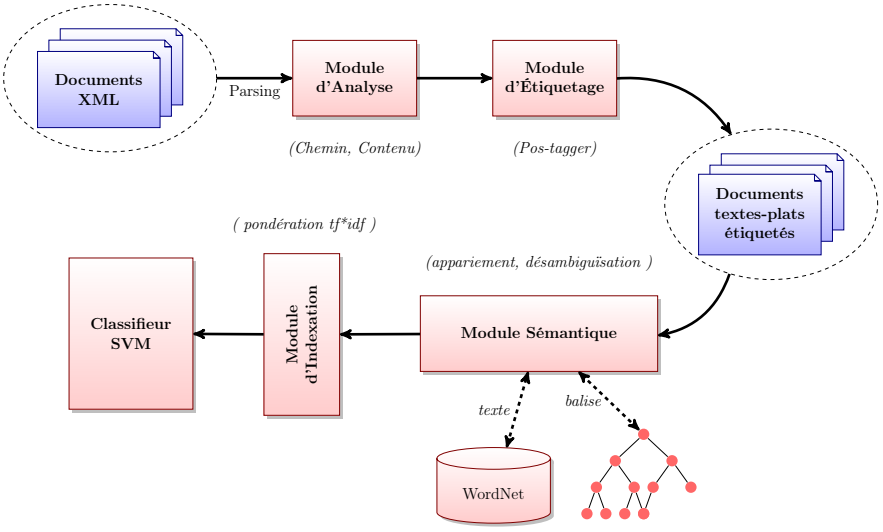

Figure 4.1 – Vue globale du modèle proposé

4.4 Analyse des documents XML

Le corpus documentaire visé par notre approche est constitué des documents XML présentant un contenu structurel et un contenu textuel.
Dans un document XML, on peut distinguer trois types d'éléments :

- *Élément de structuration* : ce sont les éléments qui ne portent pas de contenu textuel. Ils jouent essentiellement un rôle de structuration *(Balise* ou *Tag)*.
- *Élément feuille* : ce sont généralement les éléments porteurs de texte (*Texte*).
- *Élément mixte* : ce sont des éléments internes (non feuilles) qui comportent aussi bien un contenu textuel qu'un contenu structurel (*Attribut*).

Dans notre approche, une de nos préoccupations est d'intégrer l'information de structuration afin de rendre le texte plus compréhensif. Dans cette perspective, nous nous intéressons aux deux derniers types d'éléments, c'est à dire ceux qui comportent de textes. Les documents XML seront représentés par :

- un ensemble de leurs chemins (*Paths*). Un chemin (*Path*) est un succession de balises

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
    <name id="8495">Data set</name>
    <conversionwarning>0</conversionwarning>
    <body> In
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="simple" xlink:href="26685.xml">
          statistics
        </collectionlink>
        , a
        <emph3> data set </emph3>
            is a
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="simple" xlink:href="26691.xml">
            set
        </collectionlink>
        of
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="simple" xlink:href="7948.xml">
            data
        </collectionlink>
        consisting of:
        <numberlist>
            <item>
                a list of
                <unknownlink src="research␣subject"> research
                    subjects </unknownlink>
                and
            </item>
            <item>
                the
                <collectionlink xmlns:xlink="http://www.w3.org
                    /1999/xlink" xlink:type="simple" xlink:href="
                    501509.xml">
                data vector </collectionlink>
                associated with each.
            </item>
        </numberlist>
        See also:
        <collectionlink xmlns:xlink="http://www.w3.org/1999/xlink"
            xlink:type="simple" xlink:href="27579.xml">
            Statistical theory
        </collectionlink>
        <br>
        </br>
        <p>
            Another meaning:
            <collectionlink xmlns:xlink="http://www.w3.org/1999/
                xlink" xlink:type="simple" xlink:href="1042727.xml"
                >
                Data set (IBM mainframe)
            </collectionlink>
        </p>
    </body>
</article>
```

Figure 4.2 – Exemple de document **XML** du corpus INEX

partant du nœud racine et se terminant par un un nœud feuille. Cette représentation nous permet d'aplatir ces documents, tout en conservant leurs structures.

– un ensemble de paire *(balise, mot)* où mot apparaisse dans le contenu textuel correspond à cette balise. Cette combinaison de balises avec les mots des textes nous permet d'annoter structurellement les mots, et peut être interpréter comme si elles étaient (concept, valeur) ou (nom_attribut, valeur_attribut) dans 'un schéma de base

de données.

Enfin, pour permettre des différents cas de traitement sur les chemins extraits (étiquetage, intégration ressources sémantiques), une étapes intermédiaire est nécessaire pour éviter de ré-extraire ces chemins des documents XML directement, tout en tenant compte des considération suivantes :

- les balises vides (qui ne contiennent pas d'éléments textuels) sont ignorées.
- une série de balises identiques (même balise) est représentée par une seule balise.
- chaque *TPath* est représenté une série de balises représentant la structure, suivie par des mots du contenu textuel séparés par des point-virgules « ; ».

Ainsi, le document (figure 4.2) aura la représentation donnée sur la figure 4.3 :

```
/name;Data;set;
/body;In;a;is;a;of;consisting;of;See;also;
/body/collectionlink;statistics;
/body/emph3;data;set;
/body/collectionlink;set;
/body/collectionlink;data;
/body/item;a;list;of;and;
/body/item/unknownlink;research;subjects;
/body/item;the;associated;with;each;
/body/item/collectionlink;data;vector;
/body/collectionlink;Statistical;theory;
/body/p;Another;meaning;
/body/p/collectionlink;Data;set;IBM;mainframe;
```

Figure 4.3 – Représentation d'un document **XML** sous forme de *TPaths*

4.4.1 Analyseurs de documents XML

Il existe de différents types d'analyseurs (parser) pour analyser syntaxiquement les documents XML : **SAX** (*Simple API for Xml*) et **DOM** (*Document Object Model*). Ce sont des APIs [2] qui permettent d'analyser des documents XML et de naviguer dans l'arborescence pour avoir accès au contenu et attribut des éléments. SAX parcourt linéairement les documents et produit des événements à chaque rencontre de balise, attribut ou texte qui sont alors gérés ou non selon l'application. DOM construit une représentation objet de la structure et bien que beaucoup plus lourd en mémoire permet la navigation dans l'arborescence des documents XML.

Afin de parser les documents XML nous avons utilisé le parseur **JDOM**. JDOM [3] est un API-Java qui permet d'analyser, créer et manipuler des documents XML. **JDOM** a été inventé par *Brett McLaughlin* et *Jason Hunter* en 2000, il est conçu purement pour le XML, purement pour Java. Comme DOM, JDOM représente un document XML comme un arbre composé d'éléments, attributs, commentaires, nœuds des textes, sections de CDATA, ...etc. L'arbre entier est disponible à tout moment. A la différence du SAX, JDOM peut accéder à n'importe quelle partie de l'arbre à tout moment. A la différence de DOM, tous

2. API : Applications Programming Interface
3. http://www.jdom.org/

les différents genres de nœuds dans l'arbre sont représentés par les classes concrètes plutôt que des interfaces [JDOM,].

4.5 Etiquetage de contenus textuels

Aprés l'analyse des documents XML, nous procèdont à un étiquetage des éléments textuels suivant les catégories morpho-syntaxiques. A chaque mot est associé sa ou ses catégories grammaticales. La plupart des étiqueteurs réalisent conjointement l'analyse morphologique des mots et l'analyse flexionnelle en les ramenant aux lemmes. Nous citons parmi ces étiqueteurs TreeTagger [4] et le POS Tagger que nous présentons brièvement.

Part-Of-Speech Tagger (*POS Tagger*) [5] est un étiqueteur morpho-syntaxique un lemmatiseur développé par The Stanford Natural Language Processing Group de l'Université de Stanford. Il est écrit en JAVA et distribué librement à des fins d'évaluation, de recherche ou d'enseignement. Il offre plusieurs modèles spécifiques à un certaines langues (anglais, arabe, chinoise, allemande). Pour l'anglais, les étiquettes de *Penn Treebank* [Marcus *et al.*, 1993] sont utilisées. Par exemple, le texte suivant : « *He tooke a bag in his left hand and he left a town* » est étiqueté par le *POS Tagger* comme présenté dans le tableau 4.3.

He	PRP	he
took	VBP	take
a	DT	a
bag	NN	bag
in	IN	in
his	PRP$	his
left	JJ	left
hand	NN	hand
and	CC	and
he	PRP	he
left	VBD	leave
a	DT	a
town	NN	town

Table 4.3 – Exemple de sortie du POS tagger

Le document 4.3, aprés un étiquetage avec POS Tagger, aura le contenu donné suivant :

Une fois l'ensemble des mots extraits, on sélectionne généralement les lemmes des mots pleins (noms, verbes, adjectifs et adverbes). Nous avons utilisé d'abord un dictionnaire d'arrêt (*stop words*) pour supprimer les mots peu informatifs qui sont généralement des mots courts, communs qui n'ont pas de signification du point de vue de la classification, comme *"I"*, *"a"*, *"in"* et *"and"*.

Cette analyse ne donne aucune indication sur l'importance d'un terme dans le document. Des calculs de fréquence sont donc appliqués aux termes pour évaluer leur pertinence dans un document, c'est le rôle des mesures de pondération.

4. http://www.ims.uni-stuttgart.de/projekte/corplex/
5. http://nlp.stanford.edu/software/tagger.shtml

```
/name;Data_NNP;set_VBD;
/body;In_IN;a_DT;be_VBZ;a_DT;of_IN;consisting_VBG;of_IN;See_NNP;also_RB;
/body/collectionlink;statistics_NNS;
/body/emph3;data_NNS;set_VBD;
/body/collectionlink;set_NN;
/body/collectionlink;data_NNS;
/body/item;a_DT;list_NN;of_IN;and_CC;
/body/item/unknownlink;research_NN;subject_NNS;
/body/item;the_DT;associate_VBN;with_IN;each_DT;
/body/item/collectionlink;data_NNS;vector_NN;
/body/collectionlink;Statistical_NNP;theory_NN;
/body/p;Another_DT;meaning_NN;
/body/p/collectionlink;Data_NNP;set_VBD;IBM_NNP;mainframe_NN;
```

Figure 4.4 – Un document **XML** étiqueté par POS tagger

4.6 Intégration des ressources sémantiques

L'intégration des ressources sémantiques est assuré par le module sémantique, elle se fait de deux façons :

- **Sémantique textuelle** : en utilisant WordNet pour relier chaque mot étiqueté du contenu textuel à son voisinage sémantique.
- **Sémantique structurelle** : en utilisant une ontologie, que nous construisons, pour relier chaque balise d'un chemin à un concept associé.

4.6.1 Sémantique textuelle

4.6.1.1 WordNet

WordNet [Miller, 1995] est une ressource trés utilisée en catégorisation de textes. C'est un dictionnaire électronique de l'anglais qui a été créé et est maintenu au laboratoire de la Science cognitive de l'université de Princeton. Elle est composée d'ensembles de synonymes appelés *synsets*, où chaque terme est regroupé en classes d'équivalence sémantique. Chaque ensemble de synonymes représente un concept particulier. Chaque terme appartient de plus à une catégorie lexicale donnée (nom, verbe, adverbe, adjectif). Un terme peut appartenir à plusieurs synsets et à plusieurs catégories lexicales. Les ensembles de synonymes sont associés par des relations sémantiques : généricitéspécificité, antonymie (relation entre ensembles de mots qui, par leur sens, s'opposent). WordNet couvre le domaine de la langue générale en intégrant le sens des mots dans différents domaines (cf. Annexe A).

Par exemple, le mot " **left**" à (05) sens comme nom, (14) sens comme verbe, (04) sens comme adjectif et (01) sens comme adverbe.

1. **Nom** :
   - *Synset* : **left**
   - *Définition* : location near or direction toward the left side ; i.e. the side to the north when a person or object faces east
   - *Exemples* : "she stood on the left"
2. **Verbe** :

- *Synset* : **leave**, go forth, go away
- *Définition* : go away from a place
- *Exemples* : "At what time does your train leave ?" ; "She didn't leave until midnight" ; "The ship leaves at midnight"

3. **Adjectif** :
   - *Synset* : **left**, left-hand
   - *Définition* : intended for the left hand
   - *Exemples* : "I rarely lose a left-hand glove"
4. **Adverbe** :
   - *Synset* : **left**
   - *Définition* : toward or on the left ; also used figuratively
   - *Exemples* : "he looked right and left" ; "the political party has moved left"

Le module sémantique relie chaque mot du contenu textuel suivant sa catégorie grammaticale (nom, verbe, adjectif et adverbe) à son voisinage sémantique en utilisant des synsets dans WordNet. Dans le cas où ce mot est rattaché à un seul sysnet, ce mot est substitué par un mot de ses synonymes (nous considérons le premier mot du synset comme un représentant du synset). Dans le cas contraire (polysémie), une procédure de désambiguïsation est lancée.

4.6.1.2 Désambiguïsation sémantique

Un mot peut être rattachés à plusieurs *synsets* dans WordNet. Il faut alors décider du sens à fixer au mot. La désambiguïsation dans notre modèle est effectuée en calculant la corrélation d'un contexte d'un mot donné avec le contexte du *synset* qui peut être rattaché à ce mot. Le contexte d'un mot est l'ensemble de mots co-occurents localement (dans la balise associée). Le contexte d'un *synset* est la liste des termes reliés au synset dans WordNet (mots du synset, définition, exemples, hyperonymes, ...etc.). La corrélation est calculé par la similarité cosinus du vecteur des termes du contexte du mot à désambiguïser et le vecteur de l'un des synsets associées.

$$Sens(t) = \arg\max_{s} Cos(Cont(t), Cont(s)) \tag{4.1}$$

Où :

- $Cont(t)$ est le contexte du mot à désambiguïser.
- $Cont(s)$ est le contexte du synset contenant le mot t.
- Cos est la similarité Cosinus.

Afin d'utiliser Wordnet, nous avons JWNL (Java WordNet Library)[6] qui est un API JAVA développée par *Daniel C. Howe*, elle permet un accés au dictionnaire WordNet et au differentes relations existantes grâce à quelques fonctions utiles telle ques : *getIndexWord*, *getSenses*,*findRelationships* ...etc.

6. http://sourceforge.net/projects/jwordnet

4.6.2 Sémantique structurelle

Une autre fonctionnalité du module sémantique est d'identifier les relations sémantiques entre les éléments structurelles (balises), nous intéressons dans notre travail que'à la relation de synonymie entre balises. Puisque ces balises sont écrites par un être humain (auteur du document), il est parfois un peu difficile de déterminer le sens d'une balise par les ressources sémantiques comme WordNet, par exemple : les balises <*t*>, <*title*>, <*title_article*>, <*t_article*> peuvent désignés le même sens, le titre d'un article.

C'est dans ce but que nous proposons de construire une ontologie pour relever cette difficulté.

4.6.2.1 Construction de l'ontologie

L'ontologie d'un domaine est censée expliciter les concepts qui existent dans le monde à modéliser ainsi que leurs relations. En ce sens, une ontologie ne retiendra du domaine qu'un nombre fini de relations entre ses objets. Ce qui implique la définition de l'objectif pour laquelle l'ontologie est censée répondre. Dans le cadre de notre travail, notre ontologie à pour objectif d'aider à l'interprétation automatique des sens portés par les balises extraites des documents XML du corpus choisi. La figure 4.5 représente des classes et des instances (individus) de notre ontologie.

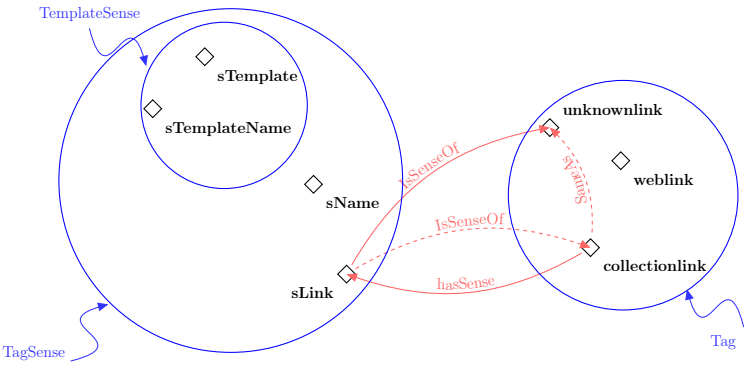

Figure 4.5 – Instance de la classe "TagSense"

Comme nous avons vu dans le chapitre précédent, il existe plusieurs méthodes pour le développement des ontologies. Pour concevoir une ontologie pour la classification des documents XML, nous suivons la méthode itérative pour le développement des ontologies proposé dans [Noy et McGuinness, 2001]. Nous décrivons ci-dessous comment nous avons suivi chacune des étapes de cette méthode.

Le domaine et la portée de l'ontologie

Nous commençons le développement de l'ontologie par définir son domaine et sa portée et ce en répondant aux questions suivantes :

1. Quel est le domaine que va couvrir l'ontologie ?
   Réponse : Le domaine de notre ontologie est la représentation des balises de documents XML.
2. Quels sont les buts de développement de l'ontologie ?
   Réponse : L'ontologie est conçue dans le but d'être utilisée pour la classification des documents XML.
3. Qui va utiliser l'ontologie ?
   Réponse : L'ontologie n'est pas destinée à être utiliser par un être humain, mais elle est conçue pour permettre son utilisation par les agents logiciels.
4. Pour quels types de questions, l'ontologie devrait apporter des réponses ?
   Réponse : Nous avons identifiés certaines questions de compétence, ainsi que les réponses que l'ontologie doit pouvoir fournir :
   - **Question de compétence** : Quel est le sens de la balise `<title>` ?
     Réponse : La balise `<title>` représente le titre d'un article (document XML)
   - **Question de compétence** : Quelles sont les représentations équivalentes de la balise `<title>` ?
     Réponse : La balise `<title>` peut avoir comme représentation équivalente les balises suivantes : `<Title>`, `<t>`,`<Title_Article>`, `<Article_Title>`,... etc.

Réutilisation des ontologies existantes

La réutilisation des ontologies existants peut être une condition si un système doit agir l'un sur l'autre avec d'autres applications qui ont déjà investie dans des ontologies particuliers. Dans notre cas, nous supposerons qu'une ontologie approprié n'existe pas déjà et nous commençons à développer notre ontologie à partir de de zéro (*scratch*).

Énumérer les termes importants dans l'ontologie

Une fois que les spécifications de l'ontologie ont été recueillies, la liste des limites les plus importantes qui composeront l'ontologie doit être identifiée. Par exemple les termes suivantes : *"Tag"*,*"TagSense"*, *"Meaning"*,*"SameAs"*, *"Sense"*,*"Title"*, *"Article"*, *"Section"*, *" Template "* doivent être représentés dans l'ontologie.
Cette liste de ces termes peut être obtenue avec l'aide des questions de compétence. Par exemple, la question *" Quel est le sens de la balise `<title>` ? "* introduit les concepts *"Tag"* pour représenter une balise et *"TagSense"* pour représenter le sens de la balise.

Définir les classes et la hiérarchie des classes

La prochaine étape est la définition de la hiérarchie de classe. Il y a plusieurs approches possibles en développant une hiérarchie de classe :

- e haut en bas (*Top-down*)
- de bas en haut (*Buttom-up*)
- combinée (*Middle-out*)

Classe	Propriété-objet	Propriété-donnée
Tag	hasSense, SameAs	hasForm
TagSense	IsSenseOf	hasMeaning

Table 4.4 – Differentes propriétés des classes

L'approche que nous avons suivi est l'approche de haut en bas. La hiérarchie de classe est très simple. La figure 4.6 montre une hiérarchie possible de classes.

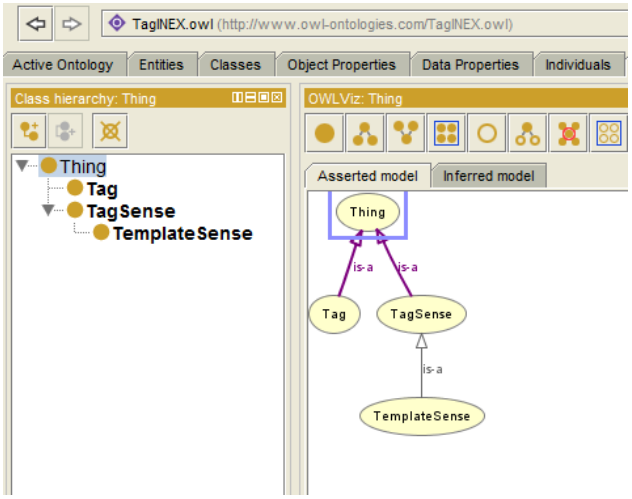

Figure 4.6 – Hiérarchie des classes

La classe *"Tag"* représente une balises, alors que la classe *"TagSense"* représente le sens porté par cette balise, la classe *" TemplateSense"* est une sous-classe de la classe *"TagSense"* qui représente le sens porté par les balises *Templates*.

Définir les propriétés des classes (attributs)

Puisque les classes seules ne fournissent pas des informations suffisantes pour représenter les balises, on doit alors décrire la structure interne de chaque classe (voir la table 4.4).

- *hasSense* : représente le sens porté par la balise « Tag ».
- *SameAs* : représente une liste des balise équivalentes à la balise « Tag ».
- *hasForm* : représente le nom (sa forme) de la balise « Tag ».
- *hasMeaning* : représente le sens porté par « TagSense ».

Définir les facettes des attributs

Une fois que nous avons accompli la hiérarchie des classes et la définition des attributs, nous devons définir les facettes des ces attributs. Les attributs peuvent avoir différentes facettes décrivant le type de valeur, domaine et la cardinalité . le tableau 4.5 présente les différents facettes de propriétés des classes de notre ontologie.

Propriété	type	Domaine	Cardinalité
hasForm	String	Tag	1..1
hasSense	TagSense	Tag	1..1
IsSenseOf	Tag	TagSense	1..n
hasMeaning	String	TagSense	1..1

Table 4.5 – Facettes des propriétés

Créer les instances des classes

La dernière étape consiste à créer les différents instances ou individus des classes dans la hiérarchie. Ainsi nous devons créer tous les instances de l'ontologie, en utilisant les balise des documents de XML extraites du corpus INEX cités précédemment. La figure 4.7 représente une instance de la classe "TagSense"

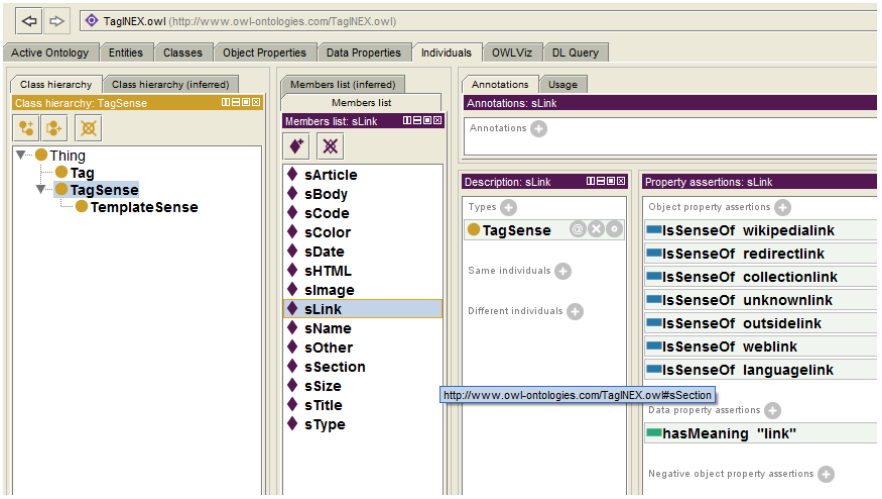

Figure 4.7 – Instance de la classe "TagSense"

4.6.2.2 Implémentation de l'ontologie

Protégé [PROTEGE, 2002] est un éditeur qui permet de construire une ontologie pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie. Protégé est également une librairie Java qui peut être étendue pour créer de véritables applications à bases de connaissances en utilisant un moteur d'inférence pour raisonner et déduire de nouveaux faits par application de règles d'inférence aux instances de l'ontologie et à l'ontologie elle même (méta-raisonnement).

Pour implémenter notre ontologie, nous avons utilisé l'éditeur **Protégé-OWL**. l'éditeur **Protégé-OWL** est une extension de Protégé qui supporte le langage OWL et permet aux utilisateurs de :

- Charger et sauvegarder des ontologies aux formats OWL et RDF
- Editer et visualiser les classes, les propriétés et les axiomes (SWRL rules).
- Définir les caractéristiques des classes comme des expressions OWL.

– Editer les individus.

4.6.2.3 Accès à l'ontologie

Pour pouvoir accéder à notre ontologie depuis le module sémantique, nous avons utilisé l'**API Jena**. **Jena** est une bibliothèque de classes Java qui facilite le développement d'applications pour le Web sémantique et fournit un environnement de programmation pour le RDF, RDF-S et OWL. Elle permet :

– La gestion des ontologies écrites en RDF.
– La gestion des ontologies écrites en OWL.
– Stockage en mémoire ou en disque.
– L'interrogation des ontologies.

Nous pouvons, à partir des méthodes des éléments *OntClass* et *ObjectProperty* obtenus à travers les méthodes correspondantes de l'API OWL : *createClass* et *createObjectProperty*, faire des traitements du type : *quels sont les concepts (classes) qui ont la propriété P ?*.

4.7 Classification des documents

4.7.1 Représentation vectorielle des documents

Puisque la plupart des algorithmes d'apprentissage utilisent la représentation *(attribut,valeur)*, nous devons transformer les documents textuels issus du module sémantique en vecteurs ou chaque terme (chemin, (balise, mot)) correspond à une dimension.
Pour transformer ces documents , nous avons utiliser un convertisseur *TextDirectoryLoader* de *Weka* qui permet de transformer une collections de documents textuels répertoriés suivant un schéma donné (figure 4.8) en un fichier au format ARFF [7] supportable par la plupart des classifieurs.

Ensuite, pour calculer le poids des termes, nous avons retenu deux dimensions :

– Une pondération locale qui détermine l'importance d'un terme dans un document. Elle est, généralement, représentée par sa fréquence *(tf)*.
– Une pondération globale qui détermine la distribution du terme dans la base documentaire. Elle est, généralement, représentée par l'inverse de la fréquence des documents qui contiennent le terme (*idf)*.

$$tfidf\left(t_k, d_j\right) = tf\left(t_k, d_j\right) \cdot \log \frac{|N|}{idf\left(t_k\right)} \tag{4.2}$$

Pour prendre en compte la longueur du documents et ne pas favoriser les documents longs par rapport aux documents courts, nous avons normalisé *tf*idf* comme suit :

$$tfidf'\left(t_k, d_j\right) = \frac{tfidf\left(t_k, d_j\right)}{\sqrt{\sum \left(tfidf\left(t_k, d_j\right)\right)^2}} \tag{4.3}$$

7. un fichier ARFF (Attribute-Relation File Format) est un fichier texte en ASCII qui décrit une liste d'instances partageant un ensemble d'attributs.

```
...
 |
 |-- texte_exemple
 |  |
    |-- classe_1
    |  |
    |  |-- fichier_1.txt
    |  |
    |  |-- fichier_2.txt
    |  |
    |  ...
    |
    |-- classe_2
    |  |
    |  |-- fichier_21.txt
    |  |
    |  |-- fichier_22.txt
    |  |
    |  ...
```

Figure 4.8 – Schéma d'organisation des documents

4.7.2 Classifieur SVM

Support Vector Machines (**SVM**) ont été prouvées en tant qu'un des algorithmes d'apprentissage les plus puissants pour la catégorisation des textes [Thorsten, 1998]. Joachims résume que SVMs sont trés adaptées pour la catégorisation de textes pour les raisons suivantes :

- Un espace d'entrée de haute dimension.
- Peu de termes non pertinents : Du fait que presque tout les termes contiennent des informations considérables. On présume qu'un bon classificateur devrait combiner beaucoup de termes et qu'une sélection agressive de termes peut avoir comme conséquence une perte d'information.
- Vecteurs de documents creux : En dépit de la dimensionnalité élevée de la représentation, les vecteurs de documents contient seulement quelques éléments non *nulls*.
- La plupart des problèmes de catégorisation des textes sont linéairement séparables.

4.8 Résultats et discussion

Dans un premier temps, nous avons pensé à utiliser le modèle proposé sans intégrer les ressources sémantiques à savoir WordNet et l'ontologie développée pour mesurer la performance du notre système vis-à-vis la représentation proposée des documents XML et la méthode de classification choisie. Les résultats sont présentés dans les figures 4.9 et 4.10.

```
Correctly Classified Instances       37350               77.3212 %
Incorrectly Classified Instances     10955               22.6788 %
Kappa statistic                          0.7308
Mean absolute error                      0.0216
Root mean squared error                  0.147
Relative absolute error                 26.1837 %
Root relative squared error             72.3531 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.478    0          0.99      0.478    0.644      0.739    1474166
                 0.572    0.017      0.751     0.572    0.649      0.778    1480358
                 0.901    0.012      0.932     0.901    0.916      0.945    1484914
                 0.822    0          0.981     0.822    0.894      0.911    1486363
                 0.697    0.002      0.857     0.697    0.769      0.848    1507239
                 0.882    0.061      0.754     0.882    0.813      0.911    1597184
                 0.6      0          0.952     0.6      0.736      0.8      1620218
                 0.254    0.001      0.924     0.254    0.399      0.627    1685758
                 0.572    0.007      0.832     0.572    0.678      0.782    1886386
                 0.696    0.001      0.885     0.696    0.779      0.848    1895383
                 0.891    0.165      0.644     0.891    0.747      0.863    2112299
                 0.842    0          0.985     0.842    0.908      0.921    2257163
                 0.652    0          0.955     0.652    0.775      0.826    2263642
                 0.153    0          0.857     0.153    0.259      0.576    2314377
                 0.912    0          0.993     0.912    0.951      0.956    2328885
                 0.289    0.001      0.693     0.289    0.408      0.644    2635947
                 0.796    0.004      0.915     0.796    0.852      0.896    2773006
                 0.708    0.003      0.857     0.708    0.776      0.853    2879927
                 0.392    0.001      0.89      0.392    0.544      0.695    2914908
                 0.613    0.005      0.79      0.613    0.69       0.804    3091127
                 0.795    0.005      0.893     0.795    0.841      0.895    3091788
Weighted Avg.    0.773    0.056      0.798     0.773    0.764      0.859
```

Figure 4.9 – Résultat de classification par stemmatisation seulement

```
=== 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
   96    5    1    0    0   15    0    0    0    0   84    0    0    0    0    0    0    0    0    0    0 |    a = 1474166
    0 2221  105    0    0  685    1    5   16    1  801    0    1    0    0    0    1   12    5   11   19 |    b = 1480358
    0   15 6546    0    1   76    2    3    3    0  604    0    1    0    0    0    0    6    1    2    7 |    c = 1484914
    0    1    0  456    4    2    0    0   19    0   72    0    0    0    0    0    0    1    0    0    0 |    d = 1486363
    0    0    0    0  430    3    0    0    5    0  171    0    0    0    0    0    0    8    0    0    0 |    e = 1507239
    0  265   22    0    5 7427    1    4   16    0  585    2    0    3    0    3    0   23    9   25   28 |    f = 1597184
    1    4   23    0    1   29  396    1    1    0  161    0    0    0    1    0    0    5    1   26   10 |    g = 1620218
    0   61   41    1    0  124    3  404   32    4  872    0    1    0    0    0    1   23    5    3   13 |    h = 1685758
    0   24   11    6    7  229    1    3 1520    1  693    0    1    0    0    4   88    0    0   63    8 |    i = 1886386
    0    0    0    0    0    1    0    0    0  337  146    0    0    0    0    0    0    0    0    0    0 |    j = 1895383
    0  237  195    2   20  494    3   14  106   38 10783    0    0    4    3   23   59   31   23   24   46 |    k = 2112299
    0    8    1    0    0   14    0    0    0    0   15  203    0    0    0    0    0    0    0    0    0 |    l = 2257163
    0    0    0    0    0   29    0    0    3    0   47    0  150    0    0    0    0    0    0    0    1 |    m = 2263642
    0    0    3    0    0   48    0    0    1    0  176    0    0   42    0    0    3    0    0    0    2 |    n = 2314377
    0    0    1    0    0    0    0    0    0    0   51    0    0    0  539    0    0    0    0    0    0 |    o = 2328885
    0    4    3    0    0  122    0    0    0    0   87    0    0    0    0   88    0    0    0    0    0 |    p = 2635947
    0    3    0    0    0   18    0    0   85    0  363    0    1    0    0    0 1843    0    0    0    1 |    q = 2773006
    0   10    2    0   32   62    0    2    0    0  212    0    0    0    0    0    0  800    4    1    5 |    r = 2879927
    0   57   64    0    0  216    3    0    4    0  342    1    0    0    0    7   16   11  478   11    9 |    s = 2914908
    0   13    3    0    2  160    3    0   14    0  243    0    1    0    0    2    3    2    5  815   63 |    t = 3091127
    0   29    6    0    0  100    3    1    3    0  247    0    1    0    0    0    0   11    6   51 1776 |    u = 3091788
```

Figure 4.10 – Matrice de confusion obtenue (stemmatisation seulement)

Ensuite, nous avons intégré les ressources sémantiques proposées dans le processus de représentation des documents XML. Les résultats obtenus (figures 4.11, 4.12 et 4.6) montrent que :

- L'espace de représentation des documents a été réduit ≈ 500 attributs (figure 4.15).
- Les performances de classification sont améliorées, nous avons eu un gain de ≈ 500 documents bien classifiés (figures 4.13 et 4.14).

```
Correctly Classified Instances       37861               78.379  %
Incorrectly Classified Instances     10444               21.621  %
Kappa statistic                          0.7442
Mean absolute error                      0.0206
Root mean squared error                  0.1435
Relative absolute error                 24.9623 %
Root relative squared error             70.6455 %
Total Number of Instances            48305

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.488     0          0.99      0.488     0.653      0.744    1474166
                 0.59      0.017      0.751     0.59      0.661      0.787    1480358
                 0.911     0.012      0.93      0.911     0.92       0.949    1484914
                 0.84      0          0.969     0.84      0.9        0.92     1486363
                 0.734     0.002      0.858     0.734     0.791      0.866    1507239
                 0.885     0.063      0.749     0.885     0.811      0.911    1597184
                 0.647     0.001      0.941     0.647     0.767      0.823    1620218
                 0.302     0.001      0.913     0.302     0.454      0.651    1685758
                 0.59      0.007      0.839     0.59      0.692      0.792    1886386
                 0.748     0.001      0.892     0.748     0.813      0.874    1895383
                 0.887     0.147      0.669     0.887     0.763      0.87     2112299
                 0.855     0          0.986     0.855     0.916      0.927    2257163
                 0.7       0          0.953     0.7       0.807      0.85     2263642
                 0.175     0          0.828     0.175     0.288      0.587    2314377
                 0.936     0          0.988     0.936     0.961      0.968    2328885
                 0.322     0.001      0.772     0.322     0.455      0.661    2635947
                 0.817     0.004      0.917     0.817     0.864      0.907    2773006
                 0.72      0.003      0.872     0.72      0.789      0.859    2879927
                 0.418     0.001      0.891     0.418     0.569      0.708    2914908
                 0.626     0.005      0.787     0.626     0.697      0.811    3091127
                 0.806     0.005      0.89      0.806     0.846      0.9      3091788
Weighted Avg.    0.784     0.052      0.803     0.784     0.776      0.866
```

Figure 4.11 – Résultat de classification par intégration des ressources sémantiques

```
=== 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
   98    5    1    0    0   16    0    0    0    0   81    0    0    0    0    0    0    0    0    0    0 |    a = 1474166
    0 2292  109    0    0  685    2   11   18    0  719    0    1    0    0    0    2   11    5   13   16 |    b = 1480358
    0   18 6619    0    2   86    3    3    3    1  517    0    1    0    0    0    0    5    1    1    7 |    c = 1484914
    0    1    0  466    4    8    0    2   19    0   54    0    0    0    0    0    0    1    0    0    0 |    d = 1486363
    0    1    0    0  453    1    0    0    4    0  153    0    0    0    0    0    0    5    0    0    0 |    e = 1507239
    0  263   29    0    6 7449    4    4   15    0  545    2    0    4    0    3    0   21    9   29   35 |    f = 1597184
    1    4   23    0    2   28  427    1    1    0  132    0    1    0    1    0    0    3    2   25    9 |    g = 1620218
    0   58   39    2    0  132    2  480   34    3  791    0    1    0    0    0    1   25    5    2   13 |    h = 1685758
    0   25   11    8    6  251    1    5 1568    1  622    0    1    0    0    4   86    0    0   63    7 |    i = 1886386
    0    0    0    0    0    1    0    0    1  362  120    0    0    0    0    0    0    0    0    0    0 |    j = 1895383
    0  262  202    2   21  501    6   18  103   39 10735    0    0    6    6   14   62   29   23   28   48 |    k = 2112299
    0    7    0    0    0   17    0    0    0    0   11  206    0    0    0    0    0    0    0    0    0 |    l = 2257163
    0    0    0    0    0   29    0    0    2    0   38    0  161    0    0    0    0    0    0    0    0 |    m = 2263642
    0    1    2    1    0   38    0    0    1    0  181    0    0   48    0    0    1    0    0    0    2 |    n = 2314377
    0    0    1    0    0    0    0    0    0    0   37    0    0    0  553    0    0    0    0    0    0 |    o = 2328885
    0    4    3    0    0  121    0    0    0    0   78    0    0    0    0   98    0    0    0    0    0 |    p = 2635947
    0    3    1    0    0   22    0    0   81    0  314    0    1    0    0    0 1891    0    0    0    1 |    q = 2773006
    0   10    3    0   32   62    1    2    0    0  194    0    0    0    0    0    0  814    4    1    7 |    r = 2879927
    0   57   68    1    0  227    3    0    3    0  300    1    0    0    0    6   16    9  509   11    8 |    s = 2914908
    0   16    3    0    2  173    2    0   13    0  203    0    1    0    0    2    3    3    7  832   69 |    t = 3091127
    0   24    7    1    0  102    3    0    4    0  227    0    1    0    0    0    0    7    6   52 1800 |    u = 3091788
```

Figure 4.12 – Matrice de confusion obtenue (intégration des ressources sémantiques)

4.9 Conclusion

Dans ce chapitre, nous avons présenté notre modèle de classification qui se base sur un classifieur SVM tout en intégrant des ressources sémantiques comme WordNet et une onto-

	Stemming seulement	Avec ressources sémantiques
Nombre d'attributs	8541	8032
Nombre de documents bien classifiés	37350	37861
Nombre de documents mal classifiés	10955	10444
Nombre de documents	48305	48305

Table 4.6 – Quelques résulats comparartifs

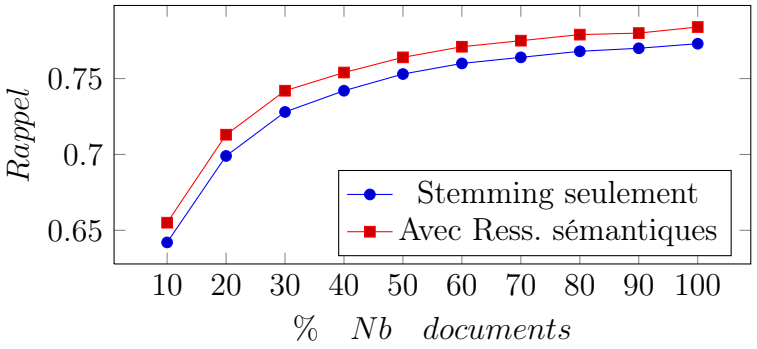

Figure 4.13 – Rappel Vs % Nb documents

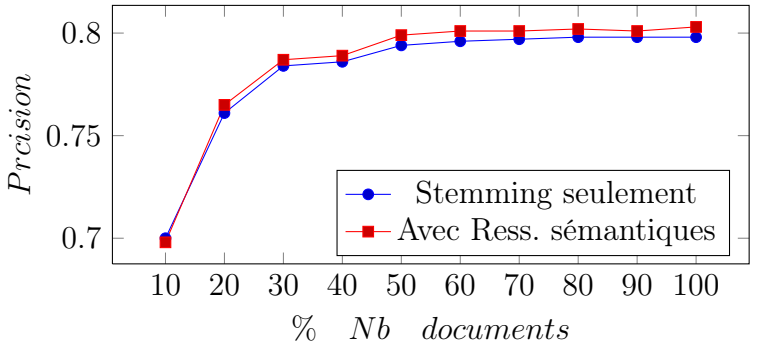

Figure 4.14 – Précision Vs % Nb documents

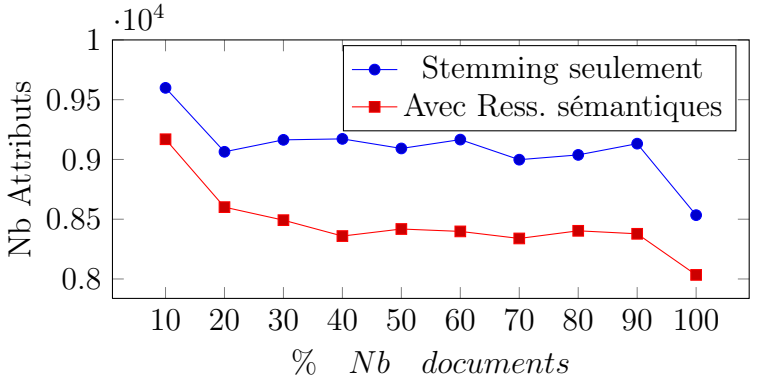

Figure 4.15 – Nb Attributs Vs % Nb documents

logie que nous avons développé, dans la phase d'indexation des documents XML.

Les résultats obtenus par notre modèle testé sur le corpus INEX sont des résultats encourageants 4.7. ces résultats montrent que :

– Le choix de la représentation des documents XML et la méthode de classification est acceptable.
– L'intégration des ressources sémantiques a amélioré les mesures d'évaluation (précision $\approx 80\%$).

Mesure	Stemming	Ressources sémantiques
Rappel	77.3%	78.4%
Précision	79.8%	80.3%
F-mesure	76.4%	77.6%

Table 4.7 – Mesures de performance de notre modèle

Conclusion et Persectives

Synthèse

Le travail présenté dans ce mémoire se situent dans le contexte de catégorisation de textes et de l'ingénierie des connaissances. Nous nous positionnons plus particulièrement dans le cadre d'une classification des documents semi-structurée profitant du cadre du web sémantique.
Les documents XML, par la définition explicite de leur structure, véhiculent des informations importantes. Les ontologies améliorent la "compréhension des textes" par les machines.

Nous nous sommes intéressés dans ce mémoire à proposer des méthodes qui mettent à profit la structure et le contenu sémantique des documents XML. Le modèle que nous proposons repose sur :

- L'intégration de la structure dans le processus de représentation des documents, ce qui permet l'annotation des termes suivant leurs positions dans les documents et d'aplatir les documents XML sans perdre l'information portée par leurs structures.
- L'intégration de la sémantique dans le processus d'indexation des documents, à savoir l'utilisation du WordNet pour modéliser la notion de voisinage sémantique entre les éléments textuels (mots), et le développement d'une ontologie pour les éléments structurels (balises).

Les résultats obtenus par nos expériences sur le corpus INEX montre l'apport de l'indexation sémantique dans la classification de documents XML.

Perspectives

Les perspectives de notre travail peuvent être vues selon les points suivants :

- Enrichissement de l'ontologie pour un domaine spécifié du fait que le corpus INEX est un corpus générique. Nous proposions d'enrichir l'ontologie créée afin de l'intégrer aussi dans le processus de classification.
- Pondération des éléments structurelles des documents XML pour prendre en compte la position du terme ou élément : un mot dans le titre n'a pas le même impact dans une autre partie du document.
- Prise en compte des autres éléments structurelles à savoir les attributs, ou d'autres éléments textuelles à savoire les nombres.
- Prise en compte des relations existantes entre les différentes synsets.

Bibliographie

[Androutsopoulos *et al.*, 2000] Androutsopoulos, I., Koutsias, J., Konstantinos, V. C. et Spyropoulos, C. D. (2000). An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. *In In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167.

[Belkin et Croft, 1992] Belkin, N. J. et Croft, W. B. (1992). Information filtering and information retrieval : two sides of the same coin ? *Commun. ACM*, 35(12):29–38.

[Borgo *et al.*, 1996] Borgo, S., Guarino, N. et Masolo, C. (1996). Stratified ontologies : the case of physical objects. *Workshop on Ontological Engineering (ECAI),Budapest.*

[Borko et Bernick, 1963] Borko, H. et Bernick, M. (1963). Automatic document classification. *J. ACM*, 10(2):151–162.

[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.

[Cavnar et Trenkle, 1994] Cavnar, W. B. et Trenkle, J. M. (1994). N-gram-based text categorization. *In In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175.

[Chaumier, 2007] Chaumier, J. (2007). Les ontologies. antécédents, aspects techniques et limites. *Documentaliste-Sciences de l'information*, pages 81–83.

[Cherfi *et al.*, 2005] Cherfi, H., Napoli, A. et Toussaint, Y. (2005). Méthodologies de sélection de règles d'association pour la fouille de textes. *In 12èmes Rencontres de la Société Francophone de Classification SFC'05*, pages 104–107, Montréal, Canada.

[Clech et Zighed, 2004] Clech, J. et Zighed, D. (2004). Une technique de réétiquetage dans un contexte de catégorisation de textes. *Documents Numérique*, 8:55–69.

[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.

[Cortes et Vapnik, 1995] Cortes, C. et Vapnik, V. (1995). Support-vector networks. *Machine Learning*, (20):273–297.

[Dagan *et al.*, 1997] Dagan, I., Karov, Y. et T, D. R. (1997). Mistake-driven learning in text categorization. *In In EMNLP-97, The Second Conference on Empirical Methods in Natural Language Processing*, pages 55–63.

[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. T., Furnas, G. W., Landauer, T. K. et Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

[Delorme, 2002] Delorme, J.-M. (2002). L'apport de la fouille de données dans l'analyse de texte.

[Denoyer et Gallinari, 2006] Denoyer, L. et Gallinari, P. (2006). The wikipedia XML corpus.

[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.

[DOE, 2001] DOE (2001). Making semantics addressable - topic maps unclothed. *Semantic Web Working Symposium.*

[DOE, 2002] DOE (2002). http://opales.ina.fr/public.

[Elberrichi *et al.*, 2008] Elberrichi, Z., Rahmoun, A. et Bentaalah, M. A. (2008). Using wordnet for text categorization. *The International Arab Journal of Information Technology.*

[Escudero *et al.*, 2000] Escudero, G., Màrquez, L. et Rigau, G. (2000). Boosting applied toe word sense disambiguation. *In ECML '00 : Proceedings of the 11th European Conference on Machine Learning*, pages 129–141, London, UK. Springer-Verlag.

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

[Feldman et Dagan, 1995] Feldman, R. et Dagan, I. (1995). Knowledge discovery in textual databases (kdt). *In In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95*, pages 112–117. AAAI Press.

[Fernandez-Lopez *et al.*, 2002] Fernandez-Lopez, M., Gomez-Perez, A., Euzenat, J., Gangemi, A., Kalfoglou, Y., Pisanelli, D., Schorlemmer, M., Steve, G., Stojanovic, L., Stumme, G. et Sure, Y. (2002). A survey on methodologies for developing, maintaining, integrating, evaluating and reengineering ontologies. OntoWeb deliverable 1.4, Universidad Politecnia de Madrid.

[Fuhr et Großjohann, 2001] Fuhr, N. et Grossjohann, K. (2001). Xirql : a query language for information retrieval in XML documents. *In* Kraft, D. H., Croft, W. B., Harper, D. J. et Zobel, J., éditeurs : *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–180. ACM 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.

[Fuhr et Knorz, 1984] Fuhr, N. et Knorz, G. (1984). Retrieval test evaluation of a rule based automatic indexing (air/phys). *In SIGIR '84 : Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–408, Swinton, UK, UK. British Computer Society.

[Furst, 2002] Furst, F. (2002). L'ingénierie ontologique.

[Gale *et al.*, 1992] Gale, W. A., Church, K. W. et Yarowsky, D. (1992). A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439.

[Garboni *et al.*, 2005] Garboni, C., Masseglia, F. et Trousse, B. (2005). Sequential pattern mining for structure-based XML document classification. *In INEX*, 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. pages 1–4.

[Gray et Harley, 1971] Gray, W. et Harley, A. (1971). Computer assisted indexing. *Information Storage and Retrieval*, 7(4):167–174.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *KNOWLEDGE ACQUISITION*, 5:199–220.

[Guarino, 2007] Guarino, N. (2007). Understanding, building, and using ontologies a commentary to " using explicit ontologies in kbs development". *International Journal of Human and Computer Studies*, pages 293–310.

[Guarino et Welty, 2001] Guarino, N. et Welty, C. (2001). Identity and subsumption. *In The Semantics of Relationships : an Interdisciplinary Perspective*. Kluwer.

[Hayes *et al.*, 1990] Hayes, P. J., Anderson, P. M., Nirenburg, I. B. et Schmandt, L. M. (1990). TCS : A shell for content-based text categorization. *In IEEE Conference on Artificial Intelligence Applications*.

[Hristozova et Sterling, 2002] Hristozova, M. et Sterling, L. (2002). An extreme method for developing lightweight ontologies. *the OAS'02 Workshop*.

[J.CHARLET, 2002] J.CHARLET (2002). L'ingénierie des connaissances : développements, résultats et perspectives pour la gestion des connaissances médicales.

[JDOM,] JDOM. Jdom : Java document object model. http://www.cafeconleche.org/books/xmljava/chapters/ch14.html#d0e22062.

[Jeong *et al.*, 2004] Jeong, B., Lee, D., Lee, J., et Cho, H. (2004). Towards XML mining : The role of kernel methods. 8.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. pages 200–209. Morgan Kaufmann.

[Jorio *et al.*, 2007] Jorio, L. D., Abrouk, L., Fiot, C., Hérin, D. et Teisseire, M. (2007). Enrichissement d'ontologie basé sur les motifs séquentiels. *OGHS*.

[Knijf, 2006] Knijf, J. D. (2006). Fat-cat : Frequent attributes tree based classification. *In INEX*, pages 485–496.

[Kodratoff, 1998] Kodratoff, Y. (1998). Techniques et outils de l'extraction de connaissances à partir des données. *Signaux*, (92):38–43.

[Larkey, 1999] Larkey, L. S. (1999). A patent search and classification system. *In* Fox, E. A. et Rowe, N., éditeurs : *Proceedings of DL-99, 4th ACM Conference on Digital Libraries*, pages 179–187, Berkeley, US. ACM Press, New York, US.

[Lee *et al.*, 2001] Lee, J., Lee, K. et Kim, W. (2001). Preparations for semantics-based xml mining. *In in Proceedings of IEEE International Conference on Data Mining*, pages 345–352.

[Lewis, 1995] Lewis, D. (1995). Evaluating and optmizing autonomous text classification systems. *In SIGIR '95 : Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254.

[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.

[Li et Jain, 1998] Li, Y. et Jain, A. K. (1998). Classification of text documents. *Pattern Recognition, International Conference on*, 2:1295.

[Liddy *et al.*, 1994] Liddy, E. D., Paik, W. et Yu, E. S. (1994). Text categorization for multiple users based on semantic features from a machine-readable dictionary. *ACM Trans. Inf. Syst.*, 12(3):278–295.

[Lini *et al.*, 2001] Lini, L., Lombardini, D., Paoli, M., Colazzo, D. et Sartiani, C. (2001). Xtresy : A text retrieval system for XML documents. pages 13–14.

[Marcus *et al.*, 1993] Marcus, M. P., Marcinkiewicz, M. A. et Santorini, B. (1993). Building a large annotated corpus of english : The penn treebank. *Association for Computational Linguistics*, (2):313–330.

[Miller, 1995] Miller, G. A. (1995). Wordnet : A lexical database for english. *Communications of the ACM*, 38:39–41.

[Mitchell, 1999] Mitchell, T. M. (1999). Machine learning and data mining. *Communications of the ACM*, 42:30–36.

[N. et B., 2004] N., L. et B., C. (2004). Fouille de textes hiérarchisée appliquée à la détection de fautes. volume 8, pages 107–133.

[Nayak *et al.*, 2002] Nayak, R., Witt, R. et Tonev, A. (2002). Data mining and XML documents.

[Noh *et al.*, 2003] Noh, S., Seo, H., Choi, J., Choi, K. et Jung, G. (2003). Classifying web pages using adaptive ontology.

[Noy et McGuinness, 2001] Noy, N. et McGuinness, D. (2001). Ontology development 101 : A guide to creating your first ontology. *Stanford Knowledge Systems Laboratory.*

[ODE, 2002] ODE (2002). Ontology design environment home page. http://www.swi.psy.uva.nl/wondertools/html/ODE.html.

[ONTOEDIT, 2002] ONTOEDIT (2002). Ontology editor home page. http://www.ontoprise.de/com.

[Owl, 2004] Owl (2004). Owl web ontology language.

[Porter, 1980] Porter, W. (1980). Synthesis of polynomic systems. *j-SIAM-J-MATH-ANAl*, 11:308–315.

[PROTEGE, 2002] PROTEGE (2002). Protégé ontology editor home page. http://protege.stanford.edu.

[Robertson et Harding, 1984] Robertson, S. et Harding, P. (1984). Probabilistic automatic indexing by learning from human indexers. *J.Document*, pages 264–270.

[Salton et Buckley, 1988] Salton, G. et Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *In Information Processing and Management*, pages 513–523.

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

[Salton *et al.*, 1975] Salton, G., Wong, A. et Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

[Schmid, 1994] Schmid, H. (1994). Part-of-speech tagging with neural networks. *In Proceedings of the 15th conference on Computational linguistics*, pages 172–176, Morristown, NJ, USA. Association for Computational Linguistics.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 625–56.

[Shapiro et Frawley, 1991] Shapiro, G. P. et Frawley, W. J., éditeurs (1991). *Knowledge-Discovery in Databases*. AAAI Press / MIT Press, Menlo Park (Ca) and Cambridge (Ma).

[Staab et Maedche, 2000] Staab, S. et Maedche, A. (2000). Axioms are objects, too – ontology engineering beyond the modeling of concepts and relations. Rapport technique, Research report, Institute AIFB, Karlsruhe.

[Studer, 1998] Studer, R. (1998). Knowledge engineering : Principles and methods.

[Taghva *et al.*, 2003] Taghva, K., Borsack, J., Coombs, J., Condit, A., Lumos, S. et Nartker, T. (2003). Ontology-based classification of email.

[TERMINAE, 2002] TERMINAE (2002). Terminae home page. http://www.lipn.univ-paris13.fr/szulman/TERMINAE.html.

[Theobald *et al.*, 2003] Theobald, M., Schenkel, R. et Weikum, G. (2003). Exploiting structure, annotation, and ontological knowledge for automatic classification of XML data. *International Workshop on the Web and Databases (WebDB)*.

[Thorsten, 1998] Thorsten, J. (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.

[Toussaint, 2004] Toussaint, Y. (2004). Extraction de connaissances à partir de textes structurés. volume 8, pages 11–34.

[Uma *et al.*, 2007] Uma, V., et Aghila, G. (2007). Svm based multilevel classifier using ontology.

[Uschold, 1996] Uschold, M. (1996). Building ontologies : Towards a unified methodology. *ES'96*.

[van Rijsbergen, 1979] van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, 2 édition.

[Vu *et al.*, 2003] Vu, H.-T., Denoyer, L. et Gallinari, P. (2003). Un modèle statistique pour la classification de documents structurés. *In Journées francophones d'Extraction et de Gestion des Connaissances (EGC 2003)*, Lyon, France.

[Wang *et al.*, 2003] Wang, Y., Hodges, J. et Tang, B. (2003). Classification of web documents using a naive bayes method. *In ICTAI '03 : Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 560, Washington, DC, USA. IEEE Computer Society.

[wei Hsu *et al.*, 2003] wei Hsu, C., chung Chang, C. et jen Lin, C. (2003). A practical guide to support vector classification. Rapport technique, Department Of Computer.

[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.

[Xing *et al.*, 2006] Xing, G., Guo, J. et Xia, Z. (2006). Classifying XML documents based on structure/content similarity. *In INEX*, pages 444–457.

[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. *Information Retrieval*, 1(1–2):69–90.

[Yang et XinLiu, 1999] Yang, Y. et XinLiu (1999). A re-examination of text categorization methods. *In SIGIR '99 : Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA. ACM.

[Young-Sook *et al.*, 2003] Young-Sook, K.-J. L., joong Lee, K., sook Hwang, Y. et chang Rim, H. (2003). Two-phase biomedical ne recognition based on svms. *In Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 33–40.

[Zaki et Aggarwal, 2003] Zaki, M. J. et Aggarwal, C. C. (2003). Xrules : an effective structural classifier for XML data. *In KDD '03 : Proceedings of the ninth ACM SIGKDD*

international conference on Knowledge discovery and data mining, pages 316–325, New York, NY, USA. ACM.

[Zargayouna, 2005] Zargayouna, H. (2005). *Indexation sémantique de documents XML*. Thèse de doctorat, Université Paris XI.

Annexe A

WordNet

WordNet est une base de données lexicale électronique développé depuis 1985 par des linguistes du *laboratoire des sciences cognitives de l'université de Princeton* et continue à être mis à jour. Elle contient des noms, des verbes, des adjectifs et des adverbes anglais, dans laquelle le sens des mots est représenté d'une façon différente de celle des dictionnaires traditionnels. La conception de WordNet est basé sur des théories de la représentation des connaissances mentales, qui semblaient montrer que les gens enregistrent, dans leur mémoire, les mots et leur concepts associés d'une façon hiérarchique.

Sa structure est celle d'un thésaurus, il est organisé autour de la structure des *synsets*, c'est-à-dire des ensembles de synonymes et de pointeurs décrivant des relations vers d'autres synsets. Chaque mot peut appartenir à un ou plusieurs synsets, et à une ou plusieurs catégorie du discours suivantes : nom, verbe, adjectif, adverbe.

Synset

Les unités dans WordNet ne sont pas des mots mais des ensembles de synonymes, ou des « Synsets », groupe de mots ou de phrases qui expriment le mêmes concept. Il n'y a pas beaucoup de vrais synonymes, comme « *shut / close* » (fermer) ou « *rug / carpet* » (tapis), mais la plupart des synonymes sont interchangeables dans beaucoup de contextes. Un synset est défini d'une façon différentielle par les relations qu'il entretient avec les sens voisins.

Les **noms** et **verbes** sont organisés en hiérarchies. Des relations d'hyperonymie (« *est-un* ») et d'hyponymie relient les « *ancêtres* » des noms et des verbes avec leurs « *spécialisations* ». Au niveau racine, ces hiérarchies sont organisées en types de base. Le réseau des noms est bien plus profond que celui des autres parties du discours.

A titre indicatif, les deux premiers niveaux de la hiérarchie des noms se constituent des concepts abstraits suivants :

- **Abstraction** : Attribute, MeasureQuantityAmount, Relation, Set, Space, Time, . . . etc.
- **Human Action** : Activity, Communication, Distribution, Inactivity, Judgment, Leaning, Legitimation, Motivation, Proclamation, Production, Speech Act, . . . etc.
- **Entity** : Anticipation, Causal Agent, Enclosure, Expanse, Location, Physical Object, Sky, Substance, Thing, . . . etc.
- **Event** : Group Action, Natural Event, Might-Have-Been, Migration, Miracle, Nonevent, Social Event, . . . etc.

- **Group, Grouping** : Association, Biological Group, People, Collection, Aggregation, Community, Ethnic Group, Kingdom, Multitude, Population, Race, Rare-Earth Element, ... etc.
- **Phenomenon** : EffectResult, Levitation, FortuneChance, Rebirth, Natural Phenomenon, Process, Pulsation, ... etc.
- **Possession** : Assets, Circumstances, PropertyMaterial Possession, Transferred Property, Treasure, ... etc.
- **Psychological Feature** : Cognition/Knowledge, Feeling, MotivationNeed, ... etc.
- **State** : Action/Activity, Existence, State Of Mind, Condition, Conflict, Damnation, Death, Degree, Dependency, Disorder, Employment, End, Freedom, Antagonism, Immaturity, Imminence, Imperfection, Integrity, Maturity, Omnipotence, Perfection, Physiological State, Relationship, State Of Affairs, Status, Temporary State, Natural State, ... etc.

L'organisation des **adjectifs** est différente. Un sens « *tête* » joue un rôle d'attracteur, des adjectifs « *satellites* » lui sont reliés par des relations de synonymie. On a donc une partition de l'ensemble des adjectifs en petits groupes. Les **adverbes** sont le plus souvent définis par les adjectifs dont ils dérivent. Ils héritent donc de la structure des adjectifs.

La version 3.0, la plus récente (janvier 2007) compte 117 597 synsets et 207 016 lemmes.

Relations

Relations sémantique (entre synsets)

Le tableau suivant présente un comptage des relations sémantiques de WordNet 2.1 par catégorie.

Relation	Entre	Et	Nombre	Exemple
Hypernym/Hyponym	Verbe	Verbe	13 124	Exhale / Breathe
Hypernym/Hyponym	Nom	Nom	75 134	Cat / Feline
Instance Hyponym	Nom	Nom	8 515	Eiffel Tower / Tower
Part	Nom	Nom	8 874	France / Europe
Member	Nom	Nom	12 262	France / European Union
Substance	Nom	Nom	793	Serum / Blood
Attribute	Adjectif	Nom	643	Inaccurate / Accuracy
Verb Entailment	Verbe	Verbe	409	Dream / Sleep
Verb Cause	Verbe	Verbe	219	Anesthetize / Sleep
Adjective Similar	Adjectif	Adjectif	22 622	Dying / Moribund
Topic Domain	Nom	Adjectif	1 108	Computer Science / Addressable
	Nom	Nom	4 146	Computer Science / Computer
Region Domain	Nom	Nom	1 246	French / France
Usage Domain	Nom	Nom	563	Neutralization / Euphemism
See Also	Adjectif	Adjectif	2 683	Black / Dark

Relations lexicales (entre lemmes)

Le tableau suivant présente un comptage des relations lexicales de WordNet 2.1 par catégorie.

Relation	Entre	Et	Nombre	Exemple
See Also	Verbe	Verbe	582	Sleep late / Sleep
Adjective Participle	Adjectif	Verbe	124	Applied / Apply
Antonym	Adjectif	Adjectif	4 080	Good / Bad
	Adverbe	Adverbe	718	Poorly / Well
	Nom	Nom	2 142	Winner / Looser
	Verbe	Verbe	1 089	Die / Be Born
Pertainym	Adjectif	Nom	4 814	Academic / Academia
	Adverbe	Adjectif	3 213	Boastfully / Boastful
Derivation	Nom	Verbe	21 579	Killing / Kill
	Adjectif	Nom	11 401	Dark / Darkness
	Nom	Nom	2 931	Automobile / Automobilist
	Verbe	Adjectif	1 508	Kill / Killable
Adjective Cluster	Adjectif	Adjectif	1 290	Strident / Noisy

Description en langage OWL de notre ontologie

Cette annexe présente une partie de la description de notre ontologie écrite en language OWL.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:swrla="http://swrl.stanford.edu/ontologies/3.3/swrla.owl
        #"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:owlapi="http://www.semanticweb.org/owlapi#"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege
        #"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:sqwrl="http://sqwrl.stanford.edu/ontologies/built-ins
        /3.4/sqwrl.owl#"
    xmlns="http://www.owl-ontologies.com/TagINEX.owl#"
  xml:base="http://www.owl-ontologies.com/TagINEX.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://sqwrl.stanford.edu/
        ontologies/built-ins/3.4/sqwrl.owl"/>
    <owl:imports rdf:resource="http://swrl.stanford.edu/ontologies
        /3.3/swrla.owl"/>
  </owl:Ontology>
  <owl:Class rdf:ID="Tag">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:allValuesFrom rdf:resource="#Tag"/>
        <owl:onProperty>
          <owl:TransitiveProperty rdf:ID="IsSameAs"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="hasSense"/>
        </owl:onProperty>
        <owl:allValuesFrom>
          <owl:Class rdf:ID="TagSense"/>
        </owl:allValuesFrom>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#
        Thing"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#TagSense"/>
```

```
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#TagSense">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:InverseFunctionalProperty rdf:ID="IsSenseOf"/>
        </owl:onProperty>
        <owl:allValuesFrom rdf:resource="#Tag"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#
        Thing"/>
  </owl:Class>
  <owl:Class rdf:ID="TemplateSense">
    <rdfs:subClassOf rdf:resource="#TagSense"/>
  </owl:Class>
  <owl:TransitiveProperty rdf:about="#IsSameAs">
    <rdfs:domain rdf:resource="#Tag"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        SymmetricProperty"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        ObjectProperty"/>
    <rdfs:range rdf:resource="#Tag"/>
    <owl:inverseOf rdf:resource="#IsSameAs"/>
  </owl:TransitiveProperty>
  <owl:FunctionalProperty rdf:about="#hasSense">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        ObjectProperty"/>
    <rdfs:domain rdf:resource="#Tag"/>
    <owl:inverseOf>
      <owl:InverseFunctionalProperty rdf:about="#IsSenseOf"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#TagSense"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasMeaning">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        DatatypeProperty"/>
    <rdfs:domain rdf:resource="#TagSense"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
        string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="hasForm">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#
        string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Tag"/>
  </owl:FunctionalProperty>
  <owl:InverseFunctionalProperty rdf:about="#IsSenseOf">
    <rdfs:range rdf:resource="#Tag"/>
    <rdfs:domain rdf:resource="#TagSense"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
        ObjectProperty"/>
  </owl:InverseFunctionalProperty>
  ...
  ...
  <owl:NamedIndividual rdf:ID="sSize">
    <hasMeaning rdf:datatype="http://www.w3.org/2001/XMLSchema#
        string"
    >size</hasMeaning>
    <IsSenseOf>
      <owl:NamedIndividual rdf:ID="font-size">
        <hasSense rdf:resource="#sSize"/>
```

```
            <rdf:type rdf:resource="#Tag"/>
            <hasForm rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"
            >font-size</hasForm>
          </owl:NamedIndividual>
        </IsSenseOf>
        <rdf:type rdf:resource="#TagSense"/>
      </owl:NamedIndividual>
    </rdf:RDF>

    <!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)
        http://protege.stanford.edu -->
```

RESUME:

La quantité d'information accessible aujourd'hui est telle que les outils, même sophistiqués, utilisés pour rechercher l'information dans les documents et pages Web ne suffisent plus: il faut maintenant pouvoir "découvrir" une information non explicitement contenue dans ces documents, afin de présenter une vue synthétique de grande quantité d'information. C'est précisément l'objectif de ce qu'on appelle la *fouille de documents*. Celle-ci utilise différentes techniques, extraire une information structurée dans du texte libre, regrouper les documents dans des classes existantes ou émergeantes, afin d'agréger ou de synthétiser l'information contenue dans une large collection. De plus, l'apparition des données semi structurées de type XML ou HTML a considérablement modifié le cadre habituel de la Recherche d'Information (RI). En effet, la notion même d'unité d'information est aujourd'hui complètement remise en cause et il est donc nécessaire d'une part d'adapter les modèles pour prendre en compte ce nouveau type de documents et d'autre part il faut s'intéresser aux nouvelles problématiques qui émergent.

Dans le cadre de ce sujet de Magister, nous nous intéressons plus particulièrement aux techniques de classification de documents XML. Plus précisément, le classement associe des documents à des catégories (ou classes) prédéfinies alors que le clustering a pour but d'identifier des classes non connue à l'avance. Pour cela on s'appuie traditionnellement sur des modèles statistiques qui manipulent des ensembles de mots. Certaines méthodes de classification réduisent donc les documents XML à leur partie purement textuelle, sans prendre avantage de la structure XML qui pourtant véhicule une information riche: le même mot n'a pas forcément même impact ou le même sens dans un titre et dans une autre partie du document.

Nous proposons de développer des modèles et des méthodes de classification prenant en compte à la fois le contenu des documents et/ou leurs structures. Par ailleurs, nous proposons d'utiliser une ontologie reliée aux termes du corpus pour modéliser la notion de voisinage sémantique à l'aide d'un calcul de similarité entre termes.

A fin de valider les résultats, nous proposons d'utiliser un corpus déjà établi comme INEX, par exemple.

Mots-clés : Fouille de données, Document semi-structurés, Classification automatique, ontologie

---

ملخـــص:

إن كمية المعلومات المتاحة اليوم مع تعدد الأدوات، وحتى المتطورة منها، التي تستخدم للبحث عن المعلومات الواردة في الوثائق وصفحات الويب لم تعد تكفي : يجب الآن السعي وراء "**اكتشاف**" معلومات غير واردة بشكل صريح في هذه الوثائق من أجل تقديم لمحة عامة عن كميات هائلة من المعلومات. هذا هو بالضبط الهدف مما يسمى **تنقيب الوثائق**. هذه الأخيرة تستخدم تقنيات مختلفة لاستخراج معلومات منظمة في نص حر ، تجميع الوثائق إلى فئات موجودة أو مستجدة. إضافة إلى ذلك ، أدى بروز معطيات شبه منظمة مثل HTML أو XML إلى تغيير الإطار المعتاد لاسترداد المعلومات . حيث أن فكرة وحدة المعلومات أصبحت اليوم موضع تساؤل ، لذلك من الضروري على حد سواء تكييف النماذج حتى تعكس هذا النوع الجديد من الوثائق و أخذ القضايا الجديدة التي تظهر بعين الاعتبار.

في إطار موضوع هذه الرسالة ، نهتم اهتماما خاصا بتقنيات تصنيف وثائق XML. وبشكل أكثر تحديدا ، فإن الترتيب يربط الوثائق إلى فئات أو أصناف معرفة سلفا، في حين أن هدف التجميع هو تحديد فئات غير معروفة مسبقا. لهذا فنحن نعتمد تقليديا على نماذج إحصائية تعالج مجموعة من الكلمات. بعض طرق التصنيف تختصر وثائق XML إلى أجزائها النصية البحتة ، من دون الاستفادة من بنية XML التي تحوي معلومات غنية، مثلا : ليس لنفس الكلمة بالضرورة نفس الأثر أو حتى نفس المعنى إن كانت موجودة بعنوان الوثيقة أو بأي جزء آخر منها.

نقترح تطوير نماذج وأساليب التصنيف مع الأخذ بعين الاعتبار كلا من محتوى الوثائق و بنيتها. علاوة على ذلك ، نقترح استخدام انطولوجيا مرتبطة بمصطلحات مجموعة الوثائق من أجل صياغة مفهوم الدلالية باستخدام حساب التشابه بين المصطلحات.

من أجل التحقق من صحة النتائج ، نقترح استخدام جسم معد مسبقا مثل INEX.

كلمات مفتاحية : تنقيب البيانات ، وثيقة شبه منظمة ، انطولوجيا، التصنيف التلقائي.