REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE AMAR TELIDJI DE LAGHOUAT

Faculté des Sciences
Département de Mathématiques et d'Informatique
Ecole Doctorale STIC
Scienecs et Technologies de l'Information et de la Communication

MEMOIRE
En vue de l'obtention du diplôme de
MAGISTER EN INFORMATIQUE
Option: Ingénierie des Systèmes d'Informations

Présenté par:
BESSINE Karima

Thème

Fouille de données dans les documents XML

Soutenu publiquement devant le jury composé de:

M. YAGOUBI Mohamed Bachir	Professeur	Université de Laghouat	Président
M. OUINTEN Youcef	Maître de conférences	Université de Laghouat	Examinateur
M. ZIADI Djelloul	Professeur	Université de Rouen	Examinateur
Mme. CHERROUN Hadda	Maître de conférences	Université de Laghouat	Rapporteur
M. MOUSSAOUI Abdelouahab	Professeur	Université de Sétif	Co-rapporteur

DÉDICACES

A la mémoire de mon père
A ma chère mère
A mon frère Yahia
A mes sœurs
A Hythem , Hiba, Tahar, Loudjine, Yacine et Hadjira
A tous les autres membres de ma famille
A mes amies
A mes professeurs,
Avec tous mes respects et mon éternelle reconnaissance.

KARIMA

REMERCIEMENTS

En tout premier lieu, louange à dieu le tout puissant qui ma donné la santé, la volonté et le pouvoir d'atteindre ce stade.

Je tiens à exprimer mes plus vifs remerciements et ma gratitude à mes directeurs de mémoire, Mme. Hadda CHERROUN et M. Abdelouaheb MOUSSAOUI, pour leurs encadrements, pour les remarques constructives qu'ils m'ont fournies ainsi que pour leurs précieux conseils durant toute la période de mon travail. Je les remercie également pour la patience et la confiance qu'ils m'ont accordées. Je n'oublierai pas aussi de les remercier pour leurs qualités humaines et leur soutien qui m'ont permis de mener à bien ce travail.

Ma gratitude, mon profond respect et mes remerciements à tous les membres du jury pour avoir accepté de rapporter et examiner ce travail.

Je tiens également a remercie ma chère amie Melle. Fatima OUEDNANI et sa famille de leur soutien et leur hospitalité.

Enfin, je remercie toute personne qui m'a aidé à réaliser ce travail de loin ou de près, plus particulièrement Mme. Meriem HADJ BRAHIM.

Résumé

XML a gagné beaucoup en popularité pour la représentation, le stockage et l'échange des informations. Les balises XML décrivent l'aspect structurel et sémantique de l'information dans les contenus des documents qui font donc les documents XML semi-structurés et auto-descriptifs. Comme la quantité des documents XML devient plus abondante, la capacité d'acquérir des connaissances à partir de sources XML diminue en raison de leur hétérogénéité et de l'irrégularité de structure. L'utilisation des techniques de la fouille de données devient donc essentielle pour extraire de l'information pertinente qui sera injectée dans un processus global d'extraction de connaissances.

Notre travail s'inscrit dans le domaine XML mining qui vise à utiliser les techniques de data mining pour découvrir les connaissances à partir les documents XML. Plus particulièrement nous nous intéressons au problème de classification non supervisée (clustering) des documents XML selon leur structure et/ou contenu. De nombreux algorithmes ont été développés pour le clustering de documents XML. Nous avons mis l'accent sur l'algorithme XCLS + et nous avons dirigé nos efforts sur l'amélioration des performances et qualité de clusters. Dans ce cadre, nous avons proposé deux approches. Dans la première approche, nous utilisons les résumés d'arbre comme représentation réduite de l'arbre correspondant au document XML où l'on a pris en compte seulement la structure. Dans la deuxième approche, nous avons enrichi le clustering des documents XML en prenant en compte à la fois la structure et le contenu.

Nous avons effectué une étude expérimentale utilisant un corpus réel INEX. Les résultats obtenus confirment que nos approches améliorent la qualité et les performances de clustering.

Mots clés : documents semi-structurés, XML, extraction de connaissance, fouille de données, XML mining, clustering.

Abstract

XML has gained popularity for representation, storage and exchange of information. XML tags describe the structural and semantic meaning of information in text documents thus make the XML documents semi-structured and self-describing. As the amount of XML documents becomes more abundant and its exploding trends, the ability to gain knowledge from XML sources decreases due to their heterogeneity and structural irregularity.

Our work is part of XML mining area that aims to use data mining techniques to discover knowledge from XML documents. Specifically, we address the problem of unsupervised classification (clustering) of XML documents. Many algorithms have been developed for the clustering of XML documents. We focused on the XCLS+ algorithm and we focused our efforts on improving its performance and quality of clusters. In this context, we propose two approaches. In the first approach, we use a reduced representation as a representation of the tree corresponding to the XML document where we take into count only the XML document structure. By mean the second approach, we have expanded the clustering of documents by considering both structure and content.

We have performed an experimental study using the real corpora INEX. Obtained results confirm that our approaches improve in a tangible way the quality and performances of clustering.

Keywords : semi-structured documents, XML, knowledge extraction, data mining, XML mining, clustering.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

INTRODUCTION GÉNÉRALE

Depuis longtemps et jusqu'à présent, nous distinguons généralement deux types de documents : les documents non structurés comme par exemple les documents textuels plats ou les images, et les documents très structurés comme les données relationnelles en base de données, mais actuellement nous assistons à l'émergence d'un autre type de documents dits semi-structurés qui représentent un compromis entre les deux types précédents.

Le type de documents semi-structurés est généralement représenté par le format XML (eXtensible Markup Langage). Ce format s'est imposé comme un standard très connu dans le domaine de la communication. Il constitue un format d'échange, par excellence, de données de différents types (texte, image, etc.) grâce à sa simplicité et son extensibilité.

Chaque jour, les données échangées sous forme de documents XML dans le Web deviennent de plus en plus volumineuses et riches en données où la nécessité de non seulement conserver ces gros volumes de données pour une utilisation ultérieure, mais aussi de les exploiter ainsi de découvrir des informations intéressantes est devenu évident. Mais avec la prolifération rapide de ces documents, la capacité d'acquérir des connaissances à partir de sources XML diminue en raison de leur hétérogénéité et de leurs structures irrégulières. Il faut donc faire appel à de nouvelles techniques ou d'adapter celles existantes au format de documents XML afin d'y extraire de l'information pertinente qui sera injectée dans un processus global d'extraction de connaissances. C'est exactement l'objectif du domaine de recherche ce qu'on appelle le XML mining ou la fouille de données dans les documents XML. Celle-ci utilise les techniques classiques de data mining afin d'extraire des connaissances cachées qui peuvent être utilisées de différentes manières dans différentes applications.

Contexte de travail

Notre travail s'inscrit dans le contexte de *la classification non supervisée (clustering) des documents semi-structrués XML.*

Le clustering des documents XML consiste à trouver des groupes (clusters) au sein d'un ensemble de documents en se basant sur la notion de similarité. Ces documents sont décrits par des attributs qui décrivent leurs propriétés.

Chaque cluster recherché regroupe un ensemble homogène de documents partageant des caractéristiques communes qui correspondent à des critères de proximité.

Pour leur traitement, les documents XML sont souvent mis dans une représentation intermédiaire afin de faciliter leur exploration. Souvent ces représentations sont : le vecteur, le graphe, l'arbre et les chemins de parcours de l'arbre. La représentation arbre est la plus courante et la plus naturelle étant donné que les documents XML possèdent une organisation hiérarchique semblable aux structures arborescentes [Francesca et al., 2003].

Parmi les travaux les plus récents basés sur la représentation arbre on trouve le travail de [Alishahi et al., 2010]. Bien que laquelle soit meilleure, elle pose problème dans la taille. En effet un document XML du Web peut atteindre jusqu'à 50 niveaux voir plus selon l'analyse du corpus étudié donc ce qui fait un nombre de nœuds exponentiel.

La méthode XCLS+ [Alishahi et al., 2010] utilise une représentation par niveau avec certaines restrictions. Les résultats de validation de leur méthode de clustering a donné une valeur d'entropie égale à 0 et une valeur de pureté et F-mesure égale à 1.

Dans notre travail, nous étudions une représentation optimisée qui est le résumé d'arbre [Dalamagas et al., 2005]. De plus, nous proposons une approche de clustering combinant la structure et le contenu.

Contribution

Notre contribution dans le cadre du clustering des documents XML se situe à deux niveaux :

1. Concernant les concepts du data mining, le XML mining et le clustering des documents XML. Nous proposons une synthèse détaillée des différents concepts de ces domaines en présentant les caractéristiques et méthodes principales de chacun d'eux.
2. Concernant notre apport direct au domaine, nous pouvons le résumer en :
   - La proposition d'une approche de clustering de documents XML basée sur la structure en utilisant des résumés d'arbres au lieu d'utiliser les arbres dans l'approche de XCLS+.
   - Dans le but d'améliorer la qualité sémantique des résultats de notre approche de clustering, nous proposons de prendre en compte le contenu des documents XML.

Organisation du mémoire

Hormis l'introduction et la conclusion, ce document est structuré autour de quatre chapitres : **Chapitre 1 : Introduction au Data mining.**
Ce chapitre est dédié à l'introduction des principaux concepts de la fouille de données sans se vouloir exhaustif. Il précise l'objectif de la fouille de données et les différentes étapes du processus de l'extraction des connaissances ainsi que les différentes tâches, applications et quelques techniques classique de data mining.

Chapitre 2 : XML Mining.
Ce chapitre s'intéresse, en premier lieu, à la présentation de quelques notions importantes comme les documents semi-structurés, la technologie XML et les documents XML, avant de décrire, le XML mining ou la fouille de données dans les documents XML. La définition du XML mining ainsi qu'une taxonomie de ce dernier sont présentées. Le processus et les techniques du XML mining font également partie de ce chapitre.

Chapitre 3 : Clusternig de documents XML : Etat de l'art.
Ce chapitre est centré sur la classification non supervisée de documents XML (clustering) qui est un domaine de recherche très actif. Un état de l'art des approches de clustering des documents XML proposées dans la littérature est dressé, en mettant l'accent sur celles qui prennent en compte la structure des documents XML et celles qui prennent à la fois le contenu des documents et leurs structures.

Chapitre 4 : Contribution et validation.
Ce chapitre est consacré à la présentation de notre contribution pour le clustering des documents XML. Nous présentons nos approches de clustering des documents XML. Ensuite nous montrons une vue générale du corpus INEX utilisé. Enfin, nous présentons les résultats d'expérimentation de nos approches sur le corpus de validation.

CHAPITRE 1

INTRODUCTION AU DATA MINING

« The best way to predict the future is to invent it ».
Alan Curtis Kay

Au cours des deux dernières décennies, nous avons assisté à une augmentation spectaculaire de la quantité de données disponibles et de différentes natures. Cette masse de données est facilement stockée grâce à l'évolution exponentielle des capacités des systèmes de stockage avec moindre coûts.

Paradoxalement, les données seules n'ont presque aucune valeur. Les gens sont en fait assoiffés de connaissances qui sont obtenues par la compréhension des données. Plus on a de données, plus il est difficile d'en tirer de la connaissance. Nous avons besoin alors d'outils performants nous permettant d'extraire des connaissances de ces masses de données.

Le data mining est dédiée à cette solution, en exploitant les différentes connaissances contenues dans les données, il concourt ainsi à prédire les tendances et le comprtements futurs offrant un meilleur support d'aide à la décision.

Dans le cadre de ce chapitre, nous introduisons ce vaste sujet, nous commençons par définir ce qu'est le data mining, ensuite nous présentons les différentes étapes du processus d'extraction de connaissances ECD (section 1.2), puis nous passerons à la présentation des différentes tâches du data mining (section 1.3), ainsi que ses applications (section 1.4). Enfin, nous concluons ce chapitre par quelques techniques misent en œuvre lors du processus du data mining (section 1.5).

1.1 Présentation du data mining

Le data mining, textuellement minage de données, mais souvent traduit en français par fouille de données, connu aussi sous les noms l'exploration de données, forage de données ou encore ECD (Extraction de Connaissances à partir de Données), il apparaît au milieu des années 1990 aux États-Unis comme une nouvelle discipline situe aux frontières communes des statistiques, des bases de données, de l'apprentissage automatique (machine learning), de l'intelligence artificielle et la visualisation des données.

Le data mining est un domaine qui consiste à comprendre les données, généralement par le moyen de méthodes statistiques. En d'autres termes, le data mining cherche à identifier des tendances parmi les données [Saitta, 2007]. Le data mining se réfère à l'extraction des connaissances à partir de grandes quantités de données, dont le but est de trouver des modèles et des relations qui étaient auparavant inconnues. Il est souvent comparé au minage de l'or dans les rivières : le gravier des alluvions représente l'énorme quantité de données et les pépites d'or représentent les connaissances cachées que l'on veut trouver.

Le data mining a été défini de manière presque aussi nombreux qu'il ya des auteurs qui ont écrit dans ce sujet. Parce qu'il se trouve à l'intersection de plusieurs disciplines que nous avons citées précédemment, la définition du data mining change avec la perspective de l'auteur :
[Fayyad et al., 1996a] ont défini le data mining comme « *un processus non trivial d'identification de modèles valides, nouveaux, potentiellement utiles et finalement compréhensibles dans les données* ». Selon [Hand, 1998], le data mining est un « processus d'analyse secondaire des grandes bases de données visant à trouver des relations qui sont d'intérêt ou de valeur pour les propriétaires de base de données ». Et d'après [Berry et Linoff, 2000], le data mining est « le processus d'exploration et d'analyse, par des moyens automatiques ou semi-automatiques, des grandes quantités de données afin de découvrir des modèles significatifs et des règles ».

Le terme data mining est souvent employé pour désigner l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de (souvent grandes) bases de données informatiques, de façon automatique ou semi automatiques, en vue de détecter dans ces données, des règles, des associations, des tendances inconnues ou cachées, des structures particulières tout en restituant l'essentiel de l'information utile en réduisant la quantité de données [Tufféry, 2007].

Le data mining diffère des autres méthodes de recherche en ce qu'il est destiné à travailler sur des données sans que l'on établisse une hypothèse de départ qu'il s'agira de vérifier. Ce sont des données elles-mêmes que sont déduites les corrélations intéressantes. Son principe est, généralement, à partir des **données** d'obtenir des **informations** pertinentes, et à partir de celle-ci de découvrir des **connaissances**. Il est important ici de différencier les trois termes en gras :

- Donnée : valeur d'une variable pour un objet (comme le montant d'un retrait d'argent par exemple).
- Information : résultat d'analyse sur les données (comme la répartition géographique de tous les retraits d'argent par exemple).
- Connaissance : information utile pour l'entreprise (comme la découverte du mauvais emplacement de certains distributeurs).

Le data mining est l'art d'extraire des connaissances à partir des données. Les données peuvent être stockées dans des entrepôts (Data Warehouse), dans des bases de données distribuées ou sur Internet : *Web Mining*. Le data mining ne se limite pas au traitement des données structurées sous forme de tables numériques ; il offre des moyens pour aborder les corpus en langage naturel (*Text Mining*), les images, le son ou la vidéo (*Multimedia Mining*).

Une confusion subsiste encore entre *data mining* et *KDD* (knowledge discovery in data bases), ou ECD en français (extraction des connaissances à partir des données). Le data mining est l'un des maillons de la chaîne de traitement pour la découverte des connaissances à partir des données. [Fayyad et al., 1996a] ont défini aussi le data mining comme une étape dans le processus KDD. Sous forme imagée, nous pourrions dire que l'ECD est un véhicule dont le data mining est le moteur.

1.2 Le processus d'extraction des connaissances KDD

Le terme KDD a été inventé lors du premier atelier KDD en 1989 pour souligner que la « connaissance » est le produit final d'une découverte guidée par les données. Le processus KDD est un processus interactif et itératif comporte un ensemble des étapes.

Le premier processus de base a été proposé par Fayyad et al. en 1996 [Cios et al., 2007] et plus tard est amélioré ou modifié par des autres. Ce processus est constitué de plusieurs étapes qui s'exécutent de manière séquentielle. Chaque étape commence après que les précédentes sont bien finalisées car elle utilise les sorties des étapes précédentes et dans le cas où les résultats ne sont pas satisfaisants, certaines étapes peuvent être refaites.

Les chercheurs ont proposé des différentes façons de diviser le processus KDD en phases tel que le modèle en neuf étapes par [Fayyad et al., 1996a] [Maimon et Rikach, 2005], le modèle en huit étapes par [Anand et al., 1998], le modèle en six étapes par [Cios et al., 2005]. La figure 1.1 représente les différentes étapes de processus KDD [Maimon et Rikach, 2005] :

1. **Développer une compréhension du domaine d'application**
   C'est l'étape initiale préparatoire de ce processus. Elle prépare la scène pour comprendre ce qu'il faut faire avec nombreuses décisions (les transformations, les algorithmes, les représentations, etc.). Les personnes qui sont en charge d'un projet KDD ont besoin de comprendre et de définir les objectifs de l'utilisateur final et l'environnement dans lequel le KDD aura lieu.
2. **La sélection et la création d'un ensemble de données sur laquelle sera effec-**

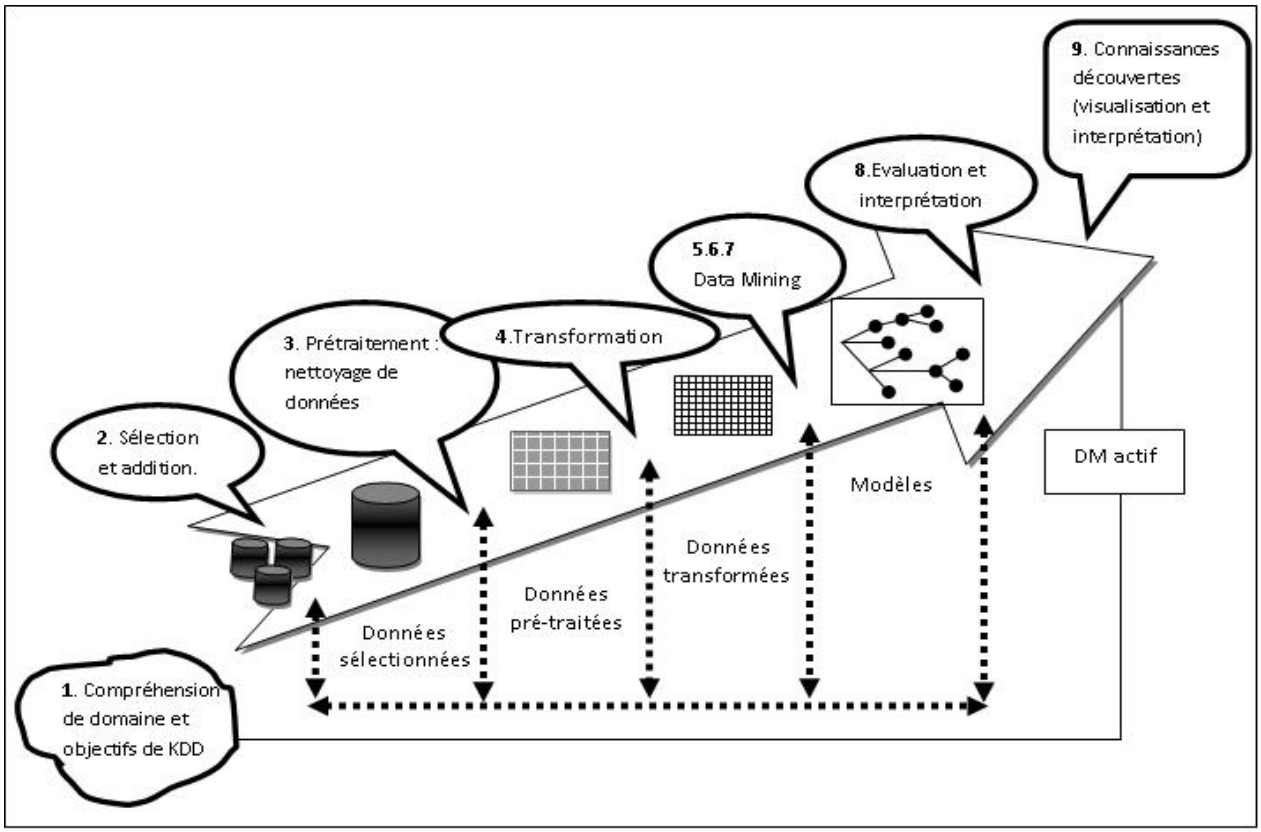


FIGURE 1.1 – Le processus d'extraction de connaissances [Maimon et Rikach, 2005].

tuée la découverte
Cette étape consiste à sélectionner l'ensemble des données qui sera utilisé pour effectuer les tâches de la découverte. La création de cet ensemble de données est cruciale pour la suite du processus car les connaissances à découvrir dépendent à lui. C'est pour cela qu'il s'effectue aussi de façon interactive et itérative.

3. **Le prétraitement et le nettoyage des données**
   Cette étape s'occupe de la fiabilité des données crées durant l'étape précédente. Elle comprend le nettoyage des données telles que la manipulation des valeurs manquantes et la suppression des bruits ou des valeurs aberrantes.
4. **La transformation des données**
   Cette phase inclut la réduction des dimensions et la transformation des attributs. Elle est essentielle pour la réussite de la totalité du projet KDD et généralement elle est très spécifiques au projet.
5. **Choix de la tâche du data mining**
   Nous devons choisir quel type de data mining sera utilisé en décidant le but du modèle (par exemple : classification, regroupement, etc.). Cela dépend surtout des objectifs de KDD.
6. **Choix de l'algorithme du data Mining**
   Dans cette étape, nous devons choisir la méthode spécifique à utiliser pour la recherche

des motifs en décidant quels modèles et paramètres sont appropriés. Par exemple si on considère la précision par rapport à l'intelligibilité, la première est mieux avec les réseaux de neurones tandis que la seconde est meilleure avec les arbres de décision.

7. **Mise en œuvre de l'algorithme du data mining**
   L'implémentation de l'algorithme de data mining choisi dans l'étape précédente est atteinte. Parfois il est nécessaire d'appliquer l'algorithme plusieurs fois pour avoir le résultat attendu.
8. **Evaluation**
   Ce stade inclut l'évaluation et l'interprétation des résultats obtenus à la lumière des objectifs définis dans la première étape. Cette étape donne la possibilité de retourner à une des étapes précédentes. Elle se concentre sur la compréhensibilité et l'utilité du modèle induit. Dans cette étape, les connaissances découvertes sont également documentées pour un usage ultérieur.
9. **Utilisation des connaissances découvertes**
   La dernière étape comprend l'incorporation des connaissances découvertes dans les ou d'autres systèmes pour d'autres actions. Cette étape permet de mesurer l'effet de ces connaissances sur le système, et elle peut également inclure la vérification et la résolution des conflits potentiels avec les connaissances antérieures.

1.3 Les tâches du data mining

Le data mining offre plusieurs tâches qui permettent de résoudre une multitude de problèmes d'ordre intellectuel, économique ou commercial. Chaque tâche revêt une fonction spécifique pour l'analyse et elle est réalisée grâce à des algorithmes différents. En général, les tâches du data mining peuvent être classées en deux catégories : *prédictive* et *descriptive*.

Les tâches *prédictives* visent à extrapoler de nouvelles informations à partir des informations observées. Cependant les tâches *descriptives* font ressortir (sous une autre forme) les informations qui existaient déjà. Nous pouvons lister cinq grandes fonctionnalités du data mining.

1.3.1 La classification

La classification est une fonction de la fouille de données qui examine les caractéristiques des éléments dans une collection et les attribue à des catégories ou des classes cibles. Il s'agit de construire un modèle représentatif (un classifieur) à partir d'un ensemble de données organisées en classes (ensemble que l'on appelle généralement le corpus d'apprentissage), et ensuite d'utiliser ce classifieur pour prédire la classe des nouveaux exemples (phase de classement).

La classification est une tâche de l'apprentissage supervisé, l'ensemble des exemples constituant le corpus d'apprentissage étant annotés, c'est à dire qu'ils portent le label de

leur classe donné a priori. Nombreux domaines d'application existent pour cette tâche, nous pouvons citér : l'attribution de crédit bancaire, la reconnaissance de gènes, la prédiction de sites archéologiques, le diagnostic médical, etc.

1.3.2 L'estimation

L'estimation consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Elle est similaire à la classification, sauf que la variable cible [1] est continue plutôt que catégorique. L'estimation permet par exemple d'estimer la pression artérielle d'un patient d'hôpital en fonction de son âge, son indice de masse corporelle, le taux de sodium, etc.

L'intérêt principal de cette tâche est de pouvoir ordonner les résultats afin d'avoir la possibilité de retenir seulement les meilleurs valeurs, technique qui sera surtout utilisée en marketing dans le but de pouvoir proposer des offres aux meilleurs clients potentiels de l'entreprise.

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. Par exemple, on peut estimer le revenu d'un ménage selon divers critères (type de véhicule et nombre, profession ou catégorie socioprofessionnelle, type d'habitation, etc.). Il sera ensuite possible de définir des tranches de revenus pour classifier les individus.

1.3.3 La prédiction

Consiste à prédire la valeur future d'un attribut en fonction des autres attributs. La prédiction ressemble à la classification et l'estimation mais dans une échelle temporelle différente. Elle s'appuie sur le passé et le présent mais son résultat se situe dans un futur généralement précisé. La seule méthode pour mesurer la qualité de la prédiction est d'attendre !

La prédiction permet par exemple de prévoir quels vont être les gagnants du championnat de foot-ball en tenant compte de la comparaison des résultats de chaque équipe ou encore de prévoir quel va être la taux de décroissance des décès sur la route l'année suivante en prenant compte de l'augmentation des limitations de vitesse et des mesures de réprimande plus strictes adoptées par la police en ce qui concerne l'utilisation de téléphone au volant.

1.3.4 Le clustering

Le clustering, en français analyse de cluster, segmentation, classification automatique ou regroupement, est une tâche dont l'objectif est de trouver des groupes (appelés des clusters) au sein d'un ensemble d'éléments (d'objets). Ces objets sont décrits par des attributs qui décrivent leurs propriétés. Chaque cluster recherché regroupe un ensemble homogène des éléments partageant des caractéristiques communes qui correspondent à des critères de

1. La variable cible peut être défini ici comme étant celle que l'on cherche, à l'aide des modèles de data mining, à expliquer ou à prédire sa valeur.

proximité.

Pour regrouper les éléments, Cette tâche fait appel à la notion de similarité. En effet, cette notion de similarité prend tout son sens en clustering car il s'agit d'évaluer à quel point deux éléments sont similaires (ou dissimilaires) pour les regrouper ou les séparer.

Contrairement aux tâches précédentes, il n'y a pas de variable cible pour le clustering, en plus on ne dispose d'aucune information a priori sur les objets à traités. C'est pour cette raison que nous dirons que le clustering est une tâche *d'apprentissage non supervisé* où les éléments sont regroupés suivant leur similitude de manière à vérifier les deux propriétés suivantes :

- Les objets d'un même cluster sont aussi similaires que possible : c'est l'homogénéité intra-classe (cohésion) qui traduit cette caractéristique.
- Les objets appartenant à des clusters différents sont aussi différents que possible : la plus grande hétérogénéité inter-classe (séparation) est recherchée.

La segmentation de marchés et des images, la localisation de tumeurs dans le cerveau sont des exemples d'application de cette tâche.

1.3.5 L'association

Le concept du règle d'association a été popularisé par [Agrawal et al., 1993]. La recherche des règles d'association est une méthode d'apprentissage non supervisé qui permet de découvrir à partir d'un ensemble de données (que l'on appelle après transactions), un ensemble de règles qui exprime une possibilité d'association entre différents items. Une transaction est une succession d'items exprimée selon un ordre donné. Une règle d'association est une expression de la forme $A \Rightarrow B$, où l'antécédent A et le conséquent B sont des sous-ensembles de l'ensemble d'items qui n'ont pas d'items communs. A chaque règle, une confiance est associée, si cette confiance dépasse un certain seuil alors la règle est induite.

Le domaine d'application classique des règles d'association concerne l'analyse du panier de la ménagère par les grandes surfaces de distribution. Ces règles leur permettent de réorganiser la disposition de leurs produits dans les rayons, d'offrir des promotions en fonctions d'habitudes d'achats découvertes, etc.

1.4 Les applications du Data Mining

Le data mining est une spécialité transverse : elle regroupe un ensemble de théories et d'algorithmes ouverts à tout domaine métier susceptible de drainer une masse de données. Dans la suite, nous présentons quelques applications courantes du data mining que nous avons inspiré de l'article « Applications and trends in data mining » de [Gupta et al., 2007].

- Finance. Les données financières recueillies dans le secteur bancaire et financier sont souvent de qualité relativement complète, fiable et élevée, ce qui facilite l'analyse systéma-

tique et l'exploration des données. Le data mining peut aider à la prédiction de paiement du prêts et l'analyse des politiques de crédit des clients, la classification des clients pour le marketing ciblé, la détection du blanchiment d'argent et autres délits financiers, etc.

- L'industrie de détail. L'industrie de détail fournit une riche source pour la fouille de données. Le data mining peut aider à identifier le comportement des clients, découvrir les habitudes d'achat et les tendances des clients, améliorer la qualité de service clients, améliorer les ratios de consommation des marchandises, conception plus efficace des transports des marchandises et des politiques de distribution.etc.

- Télécommunications. Le marché des télécommunications est en pleine expansion et très concurrentiel. Cela crée une grande demande de data mining afin d'aider à comprendre les activités concernées, identifier les modes de télécommunication, détecter les activités frauduleuses, et améliorer la qualité de service.

- Bioinformatique. La bioinformatique est un domaine de recherche en développement rapide, qui a des racines aussi bien dans la biologie que dans la technologie d'informations. L'application du data mining en bioinformatique permet de prédire les structures des protéines, identifier les modèles de séquence de gènes qui jouent un rôle dans diverses maladies, analyser les réseaux génétiques et les voies de protéines., aide au diagnostic, etc.

- Text mining. Le text mining ou la fouille de texte en français a été mentionné pour la première fois mentionné dans [Feldman et Dagan, 1995]. Le text mining est le processus d'analyse de texte pour en extraire des informations utiles à des fins particulières [Witten et al., 2003]. L'utilisation de text mining permet d'explorer le contenu des documents, extraire les relations entre entités textuelles (termes) de documents et découvrir des tendances et des concepts, la catégorisation de texte, le traitement automatique des messages, courriels, etc.

- Web mining. Le terme Web mining est utilisé pour la première fois par Oren ETZIONI en 1996 dans son article [Etzioni, 1996], il a dit que le Web mining est l'utilisation de techniques du data mining pour découvrir et extraire automatiquement des informations à partir de documents et des services Web. En générale, le Web mining peut être classé en trois catégories selon le type des données analysées : *Web Content Mining*, *Web Structure Mining* et *Web Usage Mining* [Devi et Sreevani, 2010]. Cependant ils existent ceratains approches qui divisent le Web mining en Web Content Mining et Web Usage Mining où le Web Structure est traité comme une partie de Web Content.

- ***Web Content Mining***
  Le Web content mining analyse le contenu des ressources Web [Jeong et al., 2006] [Stumme et al., 2006]. Aujourd'hui, il est considéré comme une forme de text mining.

Le Web Content Mining vise à découvrir des connaissances dans lequel les objets principaux sont les collections traditionnelles de documents textuels et (plus récemment avec l'apparition du Multimédia Data Mining) les collections de documents multimédias tels que les images, les vidéos et les audio qui sont incorporées ou liées aux pages Web. En plus des techniques classiques de text mining, le Web content mining peut profiter de la nature semi-structurée des textes des pages Web. Avec l'apparition de XML, la structure logique dans une page a gagné plus d'attentions.

- ***Web Structure Mining***
  Web structure Mining [Stumme et al., 2006] opère sur la structure des hyperliens des pages Web. Il exploite l'information supplémentaire qui est (souvent implicitement) contenue dans la structure de l'hypertexte. Par conséquent, un domaine d'application important est 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 mining est *PageRank* utilisé dans le moteur de recherche Google [2], qui calcule la pertinence d'une page en comptant les hyperliens entrants à partir d'autres pages.

- ***Web usage mining (fouille des comportements des utilisateurs du Web)***
  Web usage mining [Devi et Sreevani, 2010] analyse les résultats de l'interaction utilisateur avec un serveur web, y compris les logs web, les flux de clics (Click stream), et les transactions de base de données sur un site web ou un groupe des sites connexes. Par l'analyse des fichiers log, le Web usage mining peut découvrir des connaissances à propos des caractéristiques d'utilisation des systèmes et les intérêts des utilisateurs. Une telle connaissance a diverses applications telles que la collaboration et la personnalisation de Web, le marketing, la conception et l'évaluation des sites web, etc.

1.5 Aperçu général des techniques classiques du data mining

Le data mining repose sur un ensemble d'outils et techniques qui sont issus de plusieurs disciplines (base de données, statistiques, l'apprentissage automatique, ...) afin de construire ses modèles. Selon le type de données disponibles et le type de connaissances recherchées, une technique est choisée dans cet ensemble. Dans cette section, nous introuduisons quelques techniques classiques du data mining, nous notons que cette liste est loin d'être exhaustive.

1.5.1 Les arbres de décision

Les arbres de décision constituent une technique préliminaire puissante de data mining, et l'une des techniques de classification supervisée de données. Cette technique se base sur

2. www.google.com

un découpage selon les attributs. Les arbres de décision sont des structures en forme d'arbre qui représentent un ensemble de décisions (représentés par les nœuds internes). Ces décisions génèrent des règles (qui sont sur les chemins menant de la racine aux feuilles), qui sont ensuite utilisées pour classer les données. Les arbres de décision sont la technique privilégiée pour la construction de modèles compréhensibles.

1.5.2 Les réseaux bayésiens

Les réseaux bayésiens constituent un corps de techniques puissantes pour la mise en forme et la mise en œuvre de modèles d'aide à la décision. Un réseau bayésien est un modèle probabiliste graphique permettant d'acquérir, de capitaliser et d'exploiter des connaissances. Il constitue à la fois un formalisme de représentation des connaissances, ainsi qu'un outil permettant d'expliquer et de prédire l'état de variables d'intérêt d'un domaine de connaissances en fonction de l'état de variables observées. Dans le cadre de l'extraction de connaissance, cela signifie que ces applications sont capables d'inférer des connaissances à partir d'enregistrements incomplets.

1.5.3 Les réseaux de neurones

Les réseaux de neurones font partie d'une classe d'outils utilisés pour la prédiction, la classification et l'analyse des clusters. Un réseau de neurones est une structure artificielle basée sur une modélisation formelle du neurone.

Multiples données d'entrées, modélisation, classification, généralisation, apprentissage sont les principales caractéristiques des réseaux de neurones et surtout la dernière qui a été considérée comme le grand avantage des réseaux de neurones, ce qui permet de résoudre des problèmes sans nécessiter l'écriture de règles complexes, tout en étant tolérant aux erreurs. En revanche, le manque de lisibilité des modèles crées est un frein à l'utilisation des réseaux de neurones. En cas d'erreurs, il est impossible d'en déterminer la cause.

1.5.4 Les algorithmes génétiques

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes d'évolution de la nature : croisement, mutation, sélection. Cette technique d'optimisation permet d'éviter l'explosion combinatoire du nombre de solutions à un problème. Leur application à l'extraction de connaissance permet l'exploration de l'ensemble de toutes les règles possibles entre les données, afin de converger vers les plus intéressantes. Ils sont utilisés par exemple pour prévoir des événements rares.

Les algorithmes génétiques constituent parfois une alternative intéressante aux réseaux de neurones mais sont le plus souvent complémentaires.

1.6 Conclusion

Ce chapitre est consacré à l'étude du champ de data mining, qui vise à extraire des connaissances à partir de grands volumes de données, et qui s'inscrit dans un contexte d'aide

à la décision.

Dans cette étude qui est certainement loin d'être complète, nous avons présenté en premier lieu, les différents concepts de data mining, ainsi que les différentes étapes du processus d'extraction de connaissances (ECD) et sa relation avec le data mining, où nous trouvons que le data mining est son cœur.

Nous avons également présenté les tâches principales effectuées par le data mining qui sont divisées en deux catégories prédectives et descriptives dont le but de résoudre les différents problèmes. Nous avons aussi jeté un œil sur certaines applications du data mining dans des secteurs varies.

Nous avons ainsi vu quelques techniques classiques utilisées par le data mining pour effectuer ces tâches. Le choix d'une technique par rapport à une tâche donnée est conditionné par le type de résultat que l'expert veut obtenir.

Le data mining est un bon outil d'aide à la décision, son utilisation aujourd'hui devient une nécessité. Le data mining ne se limite pas au traitement des données structurées, mais il passe au traitement des données semi structurées dans ce qu'on appelle le XML Mining qui est l'objectif du deuxième chapitre.

CHAPITRE 2

XML MINING

« L'information n'est souvent qu'un empêchement à la vraie connaissance. ».
Louis Gauthier

De jour en jour, le Web devient plus répandu pour l'échange et la découverte des informations, il existe une augmentation de la richesse de la connaissance avec la possibilité de trouver des informations sur tout.

En outre, la technologie XML est devenue très populaire pour la représentation, le stockage et l'échange des données grace à sa propriété qui intègre l'aspect sémantique et structurel aux contenus des documents. XML permet de décrire des documents à la fois structurés et semi-structurés. La croissance explosive des sources XML présente une énorme opportunité ou un défi pour l'extraction des informations utiles, et l'utilisation des techniques de la fouille de données devient essentielle pour améliorer la gestion des documents XML.

Le XML mining ou la fouille de documents XML est un sujet de recherche émergent qui vise à développer et à utiliser des techniques du data mining pour découvrir des connaissances à partir de documents XML.

Ce chapitre est scindé en deux parties logiques. Une première partie présente une description sur XML et les documents XML. Où nous allons définir en premier les documents semi-structurés et le langage XML (section 2.1), ensuite, nous allons présenter les documents XML (section 2.2). La deuxième partie présente un aperçu sur la fouille de données dans les documents XML (XML mining), où nous donnerons la définition de XML mining (section 2.3). Ensuite, nous présenterons une taxonomie de XML mining (section 2.4), puis, le processus d'extraction de connaissances à partir de documents XML (section 2.5). En dernier lieu, nous parlerons sur les techniques de XML mining (section 2.6), où nous intéréssons plus au clusteing des documents XML.

2.1 Documents semi-structurés : le langage XML

La notion de document a évolué pour passer du concept de document plat à un concept où le document est devenu un objet contenant plusieurs sources d'informatons (vidéo, texte, image...). Pour exploiter ces sources d'une manière rigoureuse, le format de document est défini par une structure logique [Ben Aouicha, 2009].

Pour décrire cette structure une série de formalismes de structuration documentaire a vu le jour, dont certains ont été normalisés et standardisés. Ces formalismes, répondent à des besoins évolutifs de manipulation de document. XML est le formalisme le plus utilisé aujourd'hui. Il a été principalement défini pour faciliter l'échange des documents. XML permet de produire des documents à la fois structurés et semi-structurés.

2.1.1 La structure de document

La structure est *"la manière dont les parties d'un tout sont arrangées entre elles"* [1]. Décrire la structure d'un document consiste à identifier et décrire les différents éléments textuels - ou non textuels - qui le constituent. Elle peut parfois rajouter des informations (métadonnées) qui ne sont pas explicitées dans le contenu. Cette description peut prendre plusieurs formes, mais il n'existe pas une topologie exhaustive et normative de tous les types de structures documentaires, nous pouvons généralement classifier les structures suivant l'objectif pour lesquelles elles ont été définies. Parmi les types de structure qui répondent aux usages les plus courants des documents nous pouvons distinguer trois types de structures [Chatti, 2006] :

- **La structure physique** : désigne toute structure visant à décrire les caractéristiques d'affichage d'un contenu (les caractéristiques topographiques : police, taille, couleur, etc.).
- **La structure logique** : regroupe généralement les structures dont le but est de définir une organisation de contenu suivant les entités logiques qui le composent.
  C'est à ce second type que se rapporte l'utilisation du terme structure dans les documents XML.
- **La structure sémantique** : est une désignation fréquemment employée pour indiquer les structures explicitant des informations relatives au sens véhiculé par différentes parties d'un contenu.

2.1.2 Les documents semi-structurés

Parallèlement aux documents de type HTML, de nouveaux documents appelés documents semi-structurés sont apparus conjointement dans les domaines de la recherche d'information (RI) ou du document numérique et dans le domaine des bases de données (BD). Ce type de document représente un compromis entre les données fortement structurées issues de la communauté BD (données relationnelles par exemple) et les données faiblement structurées issues des communautés document numérique et RI (documents plats, images, etc).

1. Selon le dictionnaire Larousse 2010 http ://www.larousse.fr/dictionnaires

Un document semi-structuré peut être défini par : *"Par semi-structuré, nous signifions que même si les données possèdent une structure, celle-ci n'est pas aussi rigide, aussi régulière ou complète que la structure requise par les systèmes de gestion de bases de données traditionnels"*.

Les documents semi-structurés sont caractérisés par une structure :

- Irrégulière : les documents semi-structurés peuvent comporter des éléments hétérogènes qui représentent la même information. Un même attribut peut être monovalué dans certaines instances et multivalué dans d'autres. Des informations supplémentaires (annotations, détails) peuvent apparaître à certains endroits.
- Partielle : coexistence de données structurées et non structurées (texte brut).
- Implicite : même si une certaine structuration peut sembler présente, il peut ne pas exister de description explicite de la structure de données (données et structure sont mélangées).

La différence entre les documents structurés et les documents semi-structurés est la suivante : contrairement aux documents plats, dans les documents entièrement structurés, nous ne parlons plus de "texte", mais plutôt de données. Ces dernières n'ont habituellement aucune signification intrinsèque, c'est à dire il est impossible de les considérer sans examiner la structure dans laquelle elles sont inscrites. Les documents structurés possèdent une structure régulière, ne contiennent pas d'éléments mixtes (contenant du texte et d'autres éléments), et l'ordre des différents éléments qu'ils contiennent est généralement non significatif.

Les documents semi-structurés sont un pont entre les données structurées et non structurées. Ce sont des documents qui possèdent une structure flexible et des contenus hétérogènes, et qui n'ont pas de schéma a priori mais plutôt dont le schéma peut être extrait à partir de la donnée. La modification, l'ajout ou la suppression d'une donnée entraîne une modification de la structure de l'ensemble.

2.1.3 Le langage XML

XML (e**X**tensible **M**arkup **L**anguage) est un standard mis en place par W3C[2] (World Wide Web Consortium) depuis 10 Février 1998 (*XML 1.0*), mais les premiers travaux autour de XML débutent en Juin 1996. Il s'agit d'un langage de format de document. Il dérive de SGML (Standard Generalized Markup Language) et HTML (HyperText Markup Language) [Carton, 2010]. XML définit une syntaxe générique utilisée pour marquer les données avec un balisage simple et lisible par les humains.

A l'origine XML a été défini dans le but de faciliter l'échange des documents complexes dans le Web que HTML ne permettait pas. Le langage HTML ne pouvait pas répondre aux nouveaux objectifs voulus de Web en raison d'un jeu de balises prédéfini, de l'entremêlement des descriptions physiques et structurelles, et d'un mécanisme de lien hypertexte simplifié. Quant au formalisme SGML, il s'est révélé beaucoup trop complexe pour pouvoir

2. http ://www.w3.org

implémenter des navigateurs Web le supportant. De plus, il présente l'inconvénient d'imposer toujours aux documents d'être conforment à une DTD (Document Type Definition) existante [Chatti, 2006].

XML n'est pas réellement un langage, c'est un méta langage utilisé pour définir des langages de description. Ce méta langage est particulièrement performant pour l'échange et le stockage de tous les types de données et de leurs structures. Il est formé de balises qui permettent une structuration hiérarchique du document et offrent la possibilité d'identifier de façon logique la structure et l'organisation de l'information échangée.

2.2 Le document XML

Le langage XML est un format orienté texte [Carton, 2010]. Un document XML est simplement une suite de caractères respectant quelques règles. Il peut être stocké dans un fichier et/ou manipulé par des logiciels en utilisant un codage des caractères. Un exemple de document XML est donné dans la figure 2.1. Ce document représente une bibliographie de livres sur XML. Il a été tronqué ci-dessous pour réduire l'espace occupé. Ce document contient une liste de livres avec pour chaque livre, le titre, l'auteur, l'éditeur, l'année de parution, le numéro ISBN et éventuellement une URL.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!--Time-stamp: "bibliographie.xml 3 Mar 2010 16 :24 :04"-->
<!DOCTYPE bibliographie SYSTEM "bibliographie.dtd" :04">

<bibliographie>
    <livre clé="Michard01" lang="fr">
        <titre> XML langage et applications </titre>
        <auteur> Alain Michard </auteur>
        <année> 2001 </année>
        <édition>  Eyrolles  </édition>
        <isbn> 2-212-09206-7 </isbn>
        <url> http://www.edition-eyrolles/livres/michard/ </url>
    </livre>


    <livre clé="Zeldman03" lang="en">
        <titre> Designing with web standards </titre>
        <auteur> Jeffrey Zeldman </auteur>
        <année> 2003 </année>
        <édition>  New Riders  </édition>
        <isbn> 0-7357-1201-8 </isbn>
    </livre>
    ...
</bibliograhie>
```

FIGURE 2.1 – Exemple de document XML (*bibliographie.xml*) représenté sous forme textuelle (ou sérialisée).

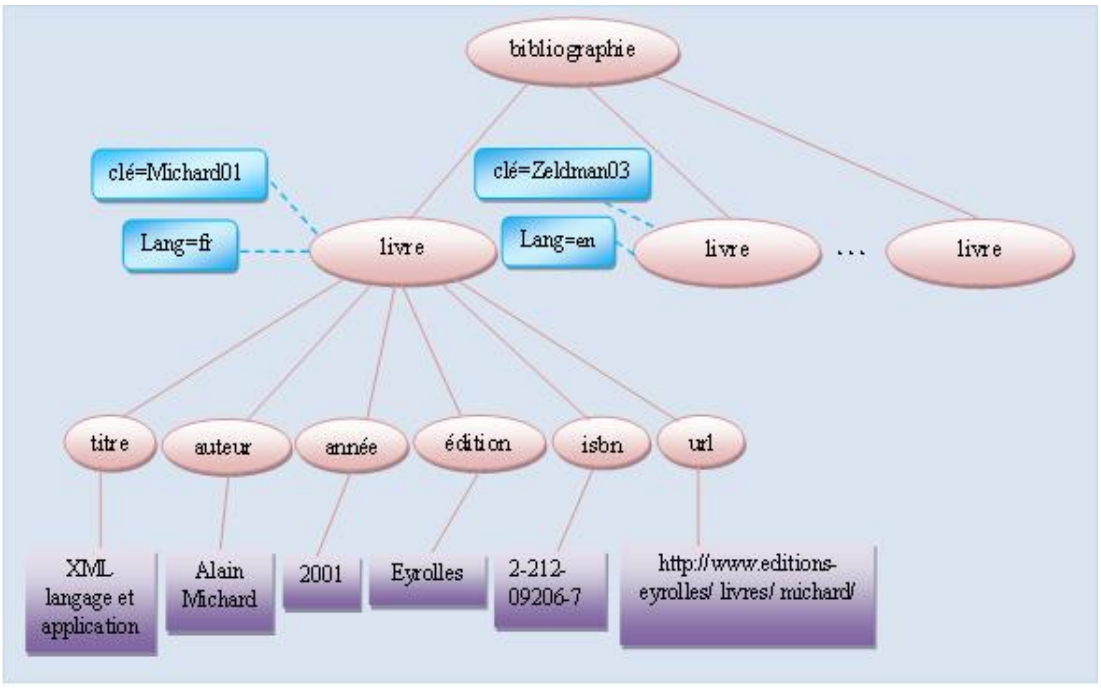


FIGURE 2.2 – Exemple de document XML (*bibliographie.xml*) représenté sous forme arborescente.

Un document XML se découpe en deux parties consécutives qui sont ***le prologue*** et ***le corps***.

Le prologue contient deux déclarations facultatives mais fortement conseillées ainsi que des commentaires et des instructions de traitement. La première déclaration est l'entête XML qui précise entre autre la version de XML et le codage du fichier. La seconde déclaration est la déclaration du DTD qui définit la structure du document. Cette déclaration est omise lorsque on utilse des schémas XML ou d'autres types de modèles qui remplacent les DTD.

Le corps du document est constitué de son contenu qui est organisé de façon hiérarchique un peu à la manière des répertoires qui servent à l'organisation des fichiers. L'unité de cette organisation est l'élément qui est la base du document XML. Chaque élément peut contenir des attributs, du texte ou d'autres éléments. Les éléments ne peuvent pas se chevaucher. Le choix du noms 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*. A la place du terme élément, on peut utiliser les termes balise, tag ou encore nœud.

Comme montre l'exemple de la figure 2.2, un document XML peut se représenter sous forme d'une arborescente, et cela est dû à son organisation particulière (l'imbrication des éléments sans possibilité de chevauchement). Chaque nœud de l'arborescence est un élément. La racine de l'arborescence est *l'élément racine* du document XML qui contient tous les autres éléments. Les feuilles de l'arbre sont généralement les éléments textuels. Nous pouvons illustrer par cet exemple les bases de la terminologie XML :

– "bibliographie", "livre","titre", etc. sont des noms ou des types de balises ;

- <livre> et </livre> sont des balises (de début et de fin d'élément respectivement) ;
- 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 ;
- clé ="Michard01" est un attribut (clé) ayant la valeur " Michard01".
- Les parties purement textuelles ("Alain Michard","Eyrolles", etc.) sont des éléments textuels.
- L'élément " livre" est le parent des éléments "titre", "auteur", etc., qui sont donc ses enfants. On dit également qu'un parent contient un enfant. Par extension, l'élément "biblioraphie" est l'encêtre des éléments "titre" (et isbn,etc.) qui sont ses descendants.
- L'élément "bibliographie" est l'élément racine, il est l'encêtre de tous les autres éléments. Les éléments situés au plus bas de l'arborescence sont des feuilles.

2.2.1 Classification des documents XML

Les documents XML *corrects* peuvent avoir deux qualifications : les documents dits *bien formés* et les documents dits *valides*.

2.2.1.1 Document bien formé

Les page Web contenant du XML ou les données XML ne sont pas classées automatiquement comme des documents XML corrects, elles sont considérés comme des documents XML si elles sont bien formées *(Well-formed)*. Cette contrainte est de nature syntaxique. Un document bien formé doit respecter les règles syntaxiques propres au langage XML, concernant l'encodage, la déclaration des attributs, le pairage des balises de début et de fin, etc. Il s'agit en quelque sorte de l'orthographe d'XML. Parmi ces règles nous pouvons citer :

- l'entête doit se trouver au tout début du document ;
- il existe un seul élément *(racine)* qui contient tous les autres éléments du document ;
- les balises correctement imbriqués : chaque balise ouvrante a une balise fermante associée et il n'y a pas de chevauchement entre les éléments ;
- les balises ne doivent pas commencer par "xml" et doivent commencer par une lettre ou un tiret souligné "_", les autres carctères peuvent être des chiffres, des lettres ou les trois symboles de ponctuation suivant : le tiret suoligné "_", le trait d'union "-" et le point"." ;
- dans la balise ouvrante, le caractère "<" doit être immédiatement suivi du nom de l'élément. En revanche, il peut y avoir des espaces entre le nom et le caractère ">". La balise fermante ne peut pas contenir d'espace ;
- les attributs lorsqu'ils existent, doivent comporter obligatoirement une valeur qui doit être toujours entornée par des guillemets (" "). Dans un élément, un attribut doit être unique.

Le document XML représenté dans la figure 2.1 est un exemple d'un document XML bien formé.

2.2.1.2 Document valide

Les documents XML valides sont un sous-ensemble de documents XML bien formés. Un document XML valide est un document bien formé, qui se conforme aussi aux règles d'un

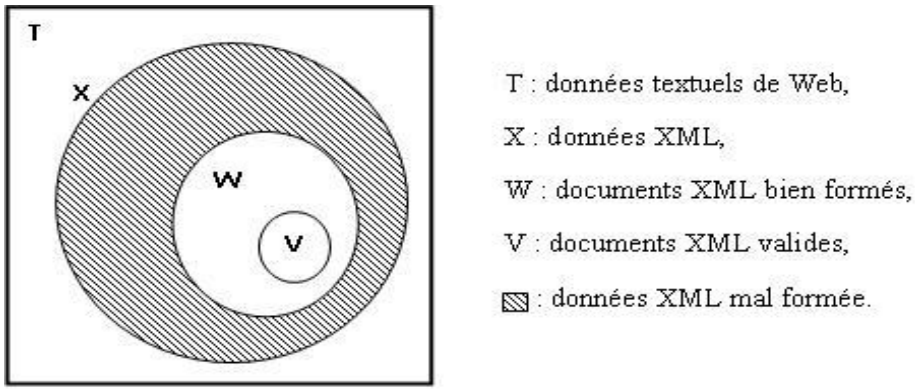


FIGURE 2.3 – La relation entre les différents données XML [Nayak et al., 2002].

modèle de documents donné par une DTD ou un schéma XML. La figure 2.3 illustre les différents types de données XML et comment elles sont liées.

2.2.2 Les langages de définitions de structure des documents XML

Cette sous-section décrit les langages qui permettent de définir la structure des données de documents XML, plus précisément les DTD et les Schémas XML, qui sont deux langages les plus utilisés dans ce domaine.

2.2.2.1 Le DTD

Une DTD (Document Type Definition) est une forme de grammaire relativement ancienne car elle est issue de l'univers SGML [Brillant, 2007]. Le rôle d'une DTD est de définir précisément la structure d'un document. Il s'agit d'un certain nombre de contraintes que doit respecter un document pour être valide. Ces contraintes spécifient quels sont les éléments qui peuvent apparaître dans le contenu d'un élément, l'ordre éventuel de ces éléments et la présence de texte brut. Elles définissent aussi, pour chaque élément, les attributs autorisés et les attributs obligatoires. Un exemple de DTD est représenté dans la figure 2.4.

La syntaxe des DTD est différente de la syntaxe des documents XML. Il n'y a pas des balises ouvrantes et fermantes. La DTD contient des déclarations d'éléments et d'attributs délimités par les chaînes de caractères "<!" et ">". Un mot clé juste après la chaîne "<!" indique le type de la déclaration (ELEMENT (ou ATTLIST) pour déclarer un élément (ou attribut)). Une DTD peut être interne, externe ou mixte [Carton, 2010] :

- Elle est ***interne*** si elle est directement incluse dans le document. Sa déclaration prend la forme suivante : `<!DOCTYPE élément-racine [déclarations]>`, où son contenu est encadré par des caractères crochets "[" et "]".
- Elle est ***externe*** si le document contient seulement une référence (adresse) vers un autre document contenant la DTD dont l'extension est généralement ".dtd". L'adresse de la DTD peut être donnée explicitement par une URL ou par un FPI (Formal Public Indentifier). Les FPI sont des noms symboliques donnés aux documents. Ils sont utilisés

```
<!ELEMENT bibliographie (livre)+ >
<!ELEMENT livre (titre, auteur, année, édition, isbn, url?) >
<!ATTLIST  livre clé  NMTOKEN #REQUIRED >
<!ATTLIST  livre lang (fr | en) #REQUIRED >
<!ELEMENT titre   (#PCDATA) >
<!ELEMENT auteur  (#PCDATA) >
<!ELEMENT année   (#PCDATA) >
<!ELEMENT édition (#PCDATA) >
<!ELEMENT isbn    (#PCDATA) >
<!ELEMENT url     (#PCDATA) >
```

FIGURE 2.4 – Exemple de DTD externe correspondant au document *bibliographie.xml*.

avec des catalogues qui établissent les correspondances entre ces noms symboliques et les adresses réelles des documents. La déclaration de la DTD externe prend les formes suivants :

- URL : le mot clé SYSTEM est utilisé suivi de l'URL délimitée par des guillemets `<!DOCTYPE élément-racine SYSTEM "url">`. L'url peut être soit une URL complète soit plus simplement le nom d'un fichier local comme dans l'exemple de la figure 2.4 ("bibloigraphy.dtd").
- FPI : on utilise le mot clé PUBLIC suivi du FPI et d'une URL délimitée par des guillemets `<!DOCTYPE élément-racine PUBLIC "fpi" "url">`. L'URL est utilisée dans le cas où le FPI ne permet pas à l'application de retrouver la DTD.

– Elle est finalement ***mixte*** si elle est constituée d'une partie interne et d'une partie externe. La déclaration prend alors une des deux formes suivantes :
- `<!DOCTYPE élément-racine SYSTEM "url" [déclarations]>`,
- `<!DOCTYPE élément-racine PUBLIC "fpi" "url" [déclarations]>`.

Les DTD ont l'avantage d'être relativement simples à utiliser mais elles sont parfois aussi un peu limitées, parmi ses limitations nous pouvons citer :

- Les DTD ne sont pas au format XML. Cela signifie qu'il est nécessaire d'utiliser un outil spécial pour manipuler un tel fichier, différent de celui utilisé pour l'édition du fichier XML.
- Pas de gestion des "espaces de noms" (Les espaces de noms permettent d'utiliser simultanément des éléments de même nom mais définis dans des modèles différents).
- Le "typage" des données (la possibilité de spécifier par exemple qu'un attribut ne doit être qu'un nombre entier) est extrêmement limité.

2.2.2.2 Schéma XML

Schéma XML porte également les acronymes de **XSD** (**X**ML **S**chema **D**ocument) ou **WXS** (**W**3C **X**ML **S**chema). Est une recommandation du W3C [3] publié le 2 mai 2001. Il fourni un moyen pour définir la structure et le contenu des documents XML. Les schémas XML ont été introduits pour combler certaines lacunes des DTD. La figure 2.5 montre un exemple de schéma XML équivalent à la DTD de la figure 2.4.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="fr">
      Schéma XML pour bibliographie.xml
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="bibliography" type="Bibliography" />

  <xsd:complexType name="Bibliography">
    <xsd:sequence>
      <xsd:element name="book" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="titre"    type="xsd:string" />
            <xsd:element name="auteur"   type="xsd:string" />
            <xsd:element name="année"    type="xsd:string" />
            <xsd:element name="édition"  type="xsd:string" />
            <xsd:element name="isbn"     type="xsd:string" />
            <xsd:element name="url"  type="xsd:string" minOccurs="0" />
          </xsd:sequence>
          <xsd:attribute name="clé"  type="xsd:NMTOKEN" use="required" />
          <xsd:attribute name="lang" type="xsd:NMTOKEN" use="required" />
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

FIGURE 2.5 – Exemple de Shema XML correspondant au document XML *bibliographie.xml*.

Malgré qu'ils sont jugés trop compliqués, les schémas XML apportent plusieurs avantages pour les documents XML dont lesquels nous pouvons trouver :

– **Syntaxe XML**

La syntaxe des DTD est une syntaxe héritée de SGML qui est différente de la syntaxe XML. En revanche, la syntaxe des schémas est une syntaxe purement XML. Un schéma est, en effet, un document XML à part entière avec un élément racine `xsd:schema` et un espace de noms.

3. http ://www.w3.org/XML/Schema

– **Nombreux types de données prédéfinis**

Les DTD manquent cruellement de précision dans la description des contenus des éléments. Cette lacune se manifeste surtout au niveau des contenus textuels et des contenus mixtes. Le seul type possible pour les contenus textuels est `#PCDATA` qui autorise toutes les chaînes de caractères. Pour les attributs, les types sont un peu nombreux mais ils restent encore très limités. A l'inverse, les schémas possèdent une multitude de types prédéfinis pour les contenus textuels : les chaînes de caractères, les nombres entiers et flottants, ainsi que les heures et les dates.

– **Possibilité de définir de nouveaux types**

Les schémas permettent de définir une hiérarchie de types qui sont obtenus par extension ou restriction de types déjà définis. L'extension de type est similaire à l'héritage des langages de programmation orienté objet comme Java ou C++. Elle permet de définir un nouveau type en ajoutant des éléments et/ou des attributs à un type existant. La restriction permet au contraire d'imposer des contraintes supplémentaires au contenu et aux attributs.

– **Substitution des éléments**

Les schémas XML prévoient plusieurs mécanismes de substitution au niveau des types et des éléments. Dans la substitution de type, un document peut changer explicitement le type associé à un élément afin d'y placer un contenu différent de celui prévu par le schéma. La substitution d'élément va encore plus loin.

– **Prise en compte des espaces de noms**

Les DTD proviennent de SGML et sont antérieures aux espaces de noms. Pour cette raison, elles ne les prennent pas en compte. Les schémas, au contraire, prennent en compte les espaces de noms. Un schéma déclare d'abord un espace de noms cible. Les éléments et les attributs sont ensuite déclarés, dans le schéma, avec leur nom local. Un document qui mélange des éléments et des attributs provenant d'espaces de noms différents peut encore être validé à l'aide des différents schémas pour les espaces de noms.

2.3 Définition du XML Mining

L'extraction des connaissances à partir des documents XML (XML Mining) diffère sensiblement de celle de données numériques, symboliques et textuelles. Le XML Mining n'est pas un résultat du hasard mais c'est une conséquence collective d'une variété d'efforts, y compris non seulement le formalisme XML mais aussi le data mining, le text mining et le Web mining.

Le XML mining est nommé pour la première fois dans IEEE conférence internationale sur le data mining en 2001 [Lee et al., 2001]. Le XML mining est l'utilisation des techniques de data mining pour découvrir et extraire automatiquement des informations à partir de sources de documents XML [Nayak, 2008]. La figure 2.6 illustre ce principe. Depuis son introduction, XML a gagné beaucoup d'attention dans les applications telles que la collaboration d'affaires (par exemple : ebXML et Web Service) et récemment la personnalisation Web (par exemple, Web 2.0).

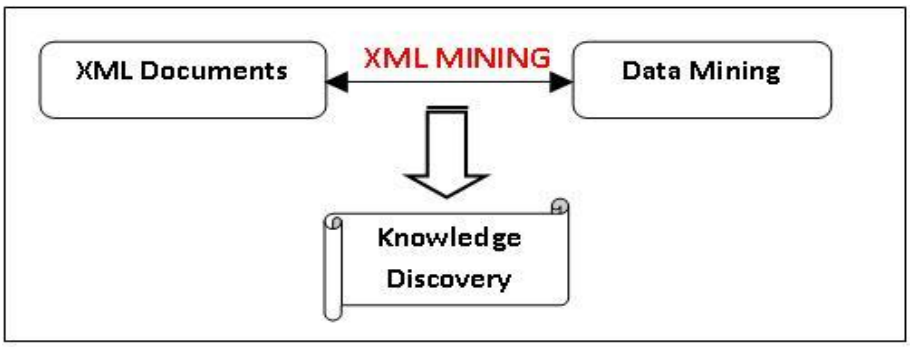


FIGURE 2.6 – Schéma du XML Mining [Nayak, 2005].

Joeng et al. [Jeong et al., 2006] définissent le XML mining comme un type spécial de *Web content mining*. Mais aussi unique en ce que le contenu d'un document XML est modulaire et bien structuré tandis que le contenu Web est plus probablement un texte simple. XML mining doit être capable de manipuler la structure du contenu ainsi que les contenus eux-mêmes.

Le XML mining diffère évidemment de data mining, text mining et Web mining. Le type et la source des données sont les principaux critères de différenciation.

2.4 Catégories du XML Mining

La fouille de documents XML diffère sensiblement des autres données structurées. XML mining comprend la fouille du structure (XML Structure Mining) ainsi que du contenu (XML Content Mining) des documents XML [Nayak et al., 2002], comme le schématise la figure 2.7. La fouille du contenu des documents XML est généralement effectuée dans le cadre d'une structure XML connue qui peut-être déterminée par la fouille de la structure des documents XML. La fouille de contenu XML peut, cependant, jouer également un rôle dans la clarification de la structure XML. Par conséquent, pour éviter toute perte d'information, les données structurelles et contextuelles sont souvent combinées pour la meilleure utilisation des documents XML [Nayak, 2008].

2.4.1 Fouille du Structure des documents XML (XML Structure Mining)

La structure d'un document XML est dictée par les balises et leur imbrication. La fouille de structure porte sur l'information que véhicule l'organisation hiérarchique du système de balises. La prise en compte de cette structure hiérarchique fonde l'originalité de ce type de fouille. XML Structure Mining est essentiellement la fouille des schémas. Elle est scindée en *Intra-Structure Mining* et *Inter-Structure Mining.*

- ***Fouille d'Intra-Structure (Intra-Structure Mining)***
  Intra-Structure Mining intéresse à la structure au sein d'un document XML. La connaissance est découverte sur la structure interne des documents XML, qui est, leur Définition de Type de Document (DTD).

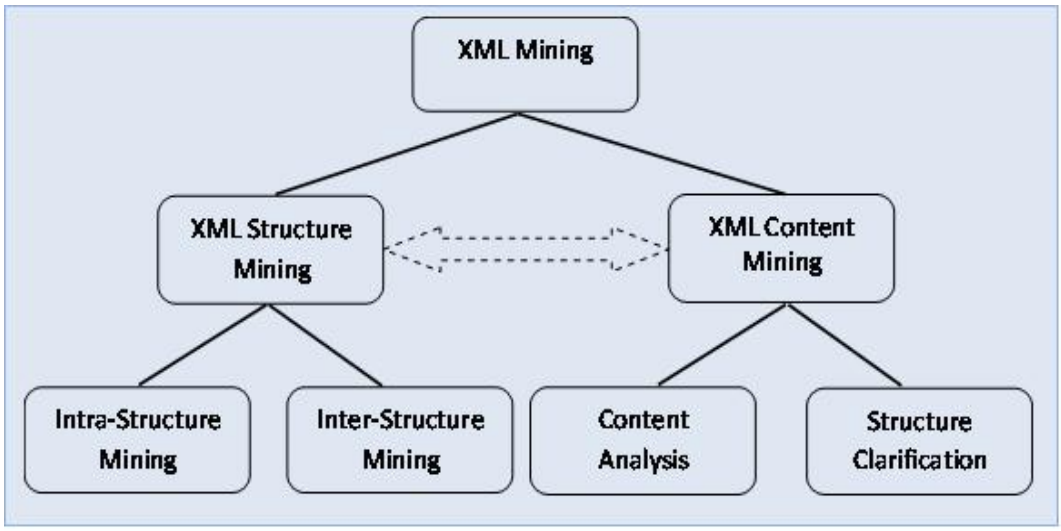


FIGURE 2.7 – Taxonomie du XML Mining [Nayak, 2008].

– ***Fouille d'Inter-Structure (Inter-Structure Mining)***
Inter-Structure Mining intéresse à la structure entre les documents XML. Les connaissances sur la relation entre les sujets, les organisations et les nœuds sur le web sont découvertes.

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

Le contenu est le texte qui se trouve entre chaque balise de début et de fin dans un document XML. La fouille de contenu utilise habituellement les techniques du text mining pour extraire l'information contenue dans le document. La nature semi-structurée des documents XML représente un défi pour la fouille de contenu. XML Content Mining peut être divisé en deux tâches : *Analyse du contenu* et *Clarification de structure*.

– ***Analyse du contenu des documents XML (Content Analysis)***
L'analyse du contenu porte sur l'analyse des textes au sein des documents XML. Les tâches similaires à celles effectuées sur les documents textuels peuvent être effectuées sur des documents XML. Cependant, XML représente ses données dans un format de structure hiérarchique qui rend l'analyse du contenu plus difficile que pour le texte brut [Nayak, 2008].

– ***Clarification de structure (Structure Clarification)***
La clarification de structure s'occupe de la distinction des documents similaires basée sur leurs contenu [Nayak, 2005]. Par exemple, le contenu peut fournir un appui pour le clustering alternatif des schémas similaires. Deux schémas distinctement structurées peuvent avoir des instances de documents avec un contenu identique. Vice versa, les schémas fournissent un appui pour le clustering alternatif de contenu. Deux documents XML ayant un contenu distinct peuvent être regroupés ensemble étant donné que leurs schémas sont similaires.

2.4.3 Fouille de la structure et le contenu des documents XML (XML Structure and Content Mining)

La nécessité de découvrir des connaissances à partir des documents XML à la fois selon la structure et le contenu est devenue un défi due à l'augmentation dans des contextes d'application pour laquelle la manipulation à la fois de la structure et de contenu dans les documents XML est essentiel. L'association de la fouille de structure et de contenu devrait permettre d'élargir et d'améliorer les techniques de XML Content Mining pour améliorer la qualité sémantique des résultats obtenus. Elle représente également un point de convergence pour les travaux de recherche dans les données semi-structurées et le text mining.

2.5 Processus de XML Mining

Le processus d'extraction de connaissances à partir des documents XML (ou des schémas) peut se décomposer en plusieurs étapes, selon [Nayak, 2008], le processus de XML mining combine le prétraitement, la découverte des modèles et le post-traitement.

- **Le prétraitement des documents XML**
  L'objectif principal de cette étape est de préparer la structure et/ou le contenu afin de permettre aux techniques du data mining d'identifier des tendances intéressantes. La sortie de cette étape est l'une des représentations de document XML qui représente sa structure et/ou son contenu (arbre, vecteur, etc.).
- **La découverte des modèles**
  Cette étape inclut le choix et l'application d'une technique du XML mining telle que le clustering.
- **Le post-traitement**
  Le post-traitement contient l'interprétation et l'évaluation des modèles découverts dans l'étape précédente, et éventuellement la reconfiguration de processus.

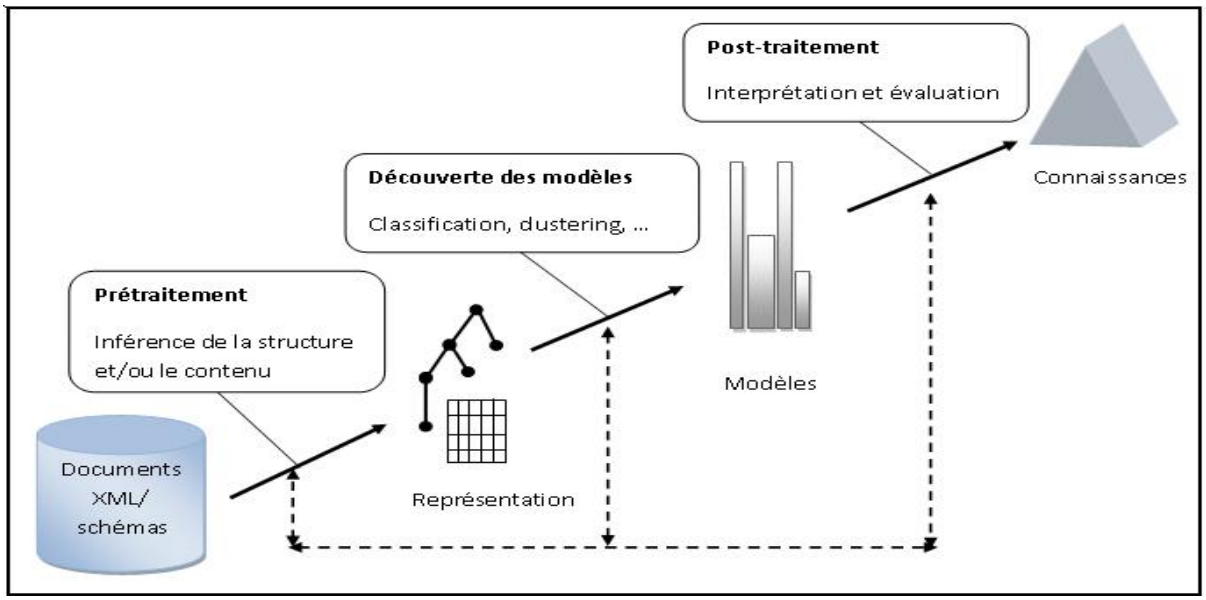


FIGURE 2.8 – Le processus de XML Mining.

2.6 Les techniques de XML Mining

La fouille de documents XML s'appuie sur les techniques traditionnelles de la fouille de données et en particulier *l'association*, *la classification* et *le clustering*.

2.6.1 XML Association

Le domaine de l'extraction des règles d'association à partir des documents XML est augmenté au cours des dernières années bien qu'ils soit encore à son stade naissant.

La découverte des règles d'association à partir des documents XML est différente de celles extraites de données relationnelles car l'information peut être structurée différemment dans le format XML. En règle générale, la découverte des règles d'association de documents XML consiste à trouver des relations entre les sous-structures d'un ou de plusieurs documents XML qui sont généralement représentés sous formes des arbres étiquetés ordonnés ou non. Alors cette tâche consiste à détecter des associations entre les arbres (y compris les sous arbres, les sous graphes et les chemins) au lieu des items dans le cas classique.

La recherche des sous-structures fréquentes consiste à extraire des sous-structures (sous-arbres, sous-graphes ou chemins) qui se produisent fréquemment au sein d'un document XML ou dans un ensemble de documents XML. Ces sous-structures fréquentes permet de générer des règles d'association. Cette technique est utilisée par un grand nombre des travaux pour découvrir les règles d'association à partir des documents XML comme [Mazuran et al., 2009]. Cependant, ils existent d'autres travaux [Braga et al., 2003] [Wan et Dobbie, 2003] qui utilisent le langage de requête XQuery pour extraire les règles d'association à partir des documents XML.

2.6.2 XML Classification

La classification supervisée de documents est un problème important, à l'intersection de deux domaines de recherche majeurs : la recherche d'information et l'apprentissage automatique. Elle est applicable à une grande variété des problèmes en XML telles que l'indexation des documents, l'organisation de corpus, etc. Cette tâche consiste à attribuer à un document une ou plusieurs catégories (ou classes) parmi un ensemble prédéfini.

La classification des documents XML a principalement trois approches qui sont basées sur le contenu, la structure ou le contenu et la structure. Les travaux actuels pour la classification des documents XML utilisent les méthodes classiques de la recherche d'informations. Le modèle *Naïf Bayes* et le classifieur *SVM (Support Vector Machines)* sont actuellement largement utilisés pour cette tâche. Mais nous pouvons trouver aussi d'autres classifieurs spécifiques comme XRules [Zaki et Aggarwal, 2003] qui est basé sur l'utilisation d'un ensemble des règles structurelles pour classer les documents XML. Ces règles sont extraites à l'aide des sous-arbres fréquents, elles prennent la forme $T \Rightarrow c_i$, où T est une structure (sous arbre), et c_i est l'une des classes prédéfinies.

2.6.3 XML Clustering

Le clustering de documents XML est la tâche qui peut être appliquée pour organiser une grande masse de documents XML en groupes mais sans aucune information a priori sur la taxonomie en se basant sur leur similitude ; chaque cluster contient les documents qui partagent des caractéristiques similaires. Les clusters peuvent être dérivés en se basant sur le contenu, la structure ou les deux.

Cette problématique trouve de très nombreuses applications en différents champs comme la recherche d'information, l'intégration des données, l'analyse et l'organisation des corpus, etc.

Le clustering de documents est appliqué en recherche d'information depuis longtemps sous l'hypothèse que des documents similaires sont pertinents pour les mêmes requêtes. Pour une requête donnée, le système de recherche d'information va tout d'abord chercher le cluster le plus pertinent pour la requête puis trier les documents par ordre de pertinence dans ce cluster. L'utilisation de clustering permet alors de limiter la complexité de la recherche et de renvoyer un sous-ensemble de documents tous pertinents pour la requête car le choix du cluster est évalué à partir des caractéristiques communes des documents qui le composent. Elle permet aussi de réduire le temps de réponse et d'augmenter la précision des moteurs de recherche ainsi que d'assister l'utilisateur dans sa recherche.

L'intégration des données est une problématique importante à cause du nombre croissant de sources de données XML disponibles via le Web. Le clustering peut être utilisé dans le but de faciliter cette tâche. L'idée est de concilier en premier entre les sources similaires appartenant au même cluster. Cette idée est appliquée dans le travail de [Lee et al., 2002] pour intégrer des sources de données XML à partir de leurs DTD. Ils ont trouvé d'abord des groupes de DTD qui sont semblables car cela permet aux intégrateurs de systèmes de se concentrer sur les DTD au sein de chaque cluster pour obtenir une DTD intégrée pour le cluster. Concilier des DTD similaires est plus facile que de concilier les DTD qui sont différents dans la structure et la sémantique car celle-ci implique une restructuration en plus. Le processus de regroupement est appliqué de manière récursive pour les DTD des clusters jusqu'à obtention d'un nombre gérable de DTD.

Le clustering peut également servir à l'analyse et l'organisation des larges collections de documents XML (corpus). Par exemple, le clustering permet la segmentation du corpus en groupes de documents thématiquement homogènes.

Comme toutes les méthodes de fouilles, les documents semi-structurés XML ne sont pas directement analysables par les techniques de clustering : elle passent par des représentations intermédiaires. Ces représentations sont décrites dans la sous-section suivante.

2.6.3.1 Les représentations d'un document XML

Un aspect important et caractérisant des travaux sur le clustering de documents XML est la manière de représenter les documents. La représentation des documents est une étape importante qui précède le processus de clustering. Celle-ci permet de surcroît d'améliorer les

performances et la qualité de regroupement. Plusieurs formes de modèles ont été proposé pour représenter les documents XML. Dans la suite, nous proposons notre propore synthèse de ces représentations (figure 2.9).

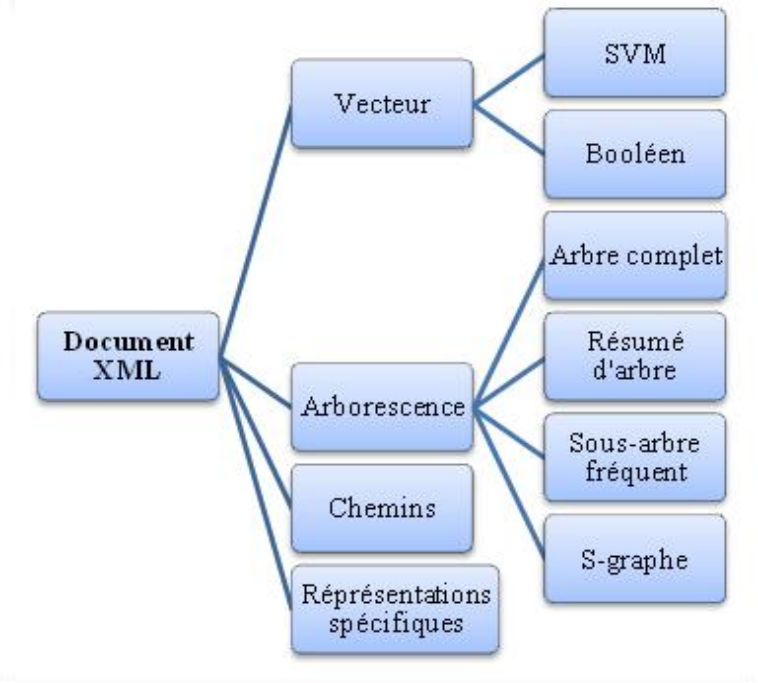


FIGURE 2.9 – Les représentations d'un document XML.

A. Le modèle vectoriel

Le modèle le plus simple utilisé pour représenter les documents XML est le « ***Modèle d'espace vectoriel*** ». Dans cette représentation, chaque document XML d_j est représenté par un vecteur des termes à n dimensions $(w_{1,j}, \cdots, w_{n,j})$, où $w_{i,j}$ est le poids du terme t_i dans le document d_j. Un terme peut être un mot, un lemme ou un composant (plusieurs mots, lemmes ou stems). Le poids d'un terme représente le degré de son importance dans le document. Il y a principalement trois facteurs permettant la pondération d'un terme :

- sa fréquence dans le document,
- sa fréquence dans le corpus,
- sa normalisation.

Parmi les méthodes de pondération la plus utilisé on trouve le codage *tf-idf (Term Frequency-Inverse Document Frequency)*. Dans ce codage l'importance d'un terme est proportionnelle à la fréquence d'apparition de ce terme dans le document et inversement proportionnelle à la fréquence en documents (nombre de documents où le terme apparaît). Soit $tf_{i,j}$ la fréquence d'apparition du terme t_i dans le document d_j et df_i la fréquence en documents du terme t_i. Le poids $w_{i,j}$ du terme t_i dans le document d_j est calculé comme suit :

$$w_{i,j} = tf_{i,j}.\log\left(\frac{N}{df_i}\right). \tag{2.1}$$

Où N est le nombre de documents dans le corpus.

Un autre type de vecteur a été utilisé pour représenter les documents XML, ***le vecteur de booléens*** ou vecteur de bits selon [Kim, 2010], où les poids sont réduits aux simples valeurs $d(e_1, e_2, \cdots, e_n)$ où $e_i = 1$ si $e_i \in d$, sinon $e_i = 0$.

Les vecteurs décrit précédemment sont généralement utilisés dans des approches de clustering pour les documents XML basées sur ***une seule granularité***. Il existe d'autres approches de clustering qui prennent plusieurs granularités ainsi que leurs relations en compte. Dans ce cas, les documents XML sont représentés par ***une matrice*** M à $C - dimensions$ (C est le nombre de granularités) dans un espace euclidien basée sur l'un des deux modèles : *booléen* [Yoon et al., 2001] et *pondéré* [Yang et al., 2005]. Dans *le modèle booléen*, $M(g_1, \cdots, g_C) = 1$ si la caractéristique correspondant à l'intersection des granularités $g_1, \cdots, g_C$ existe, sinon $M(g_1, \cdots, g_C) = 0$. Pour *le modèle pondéré*, chaque élément de M représente le poids de la caractéristique correspondant à l'intersection des granularités. La figure 2.10 représente une matrice de booléens à *3 dimensions* avec les granularités « document, chemin, terme » indiquant la présence (ou l'absence) d'un terme w_j dans un chemin P_i dans un document D_m.

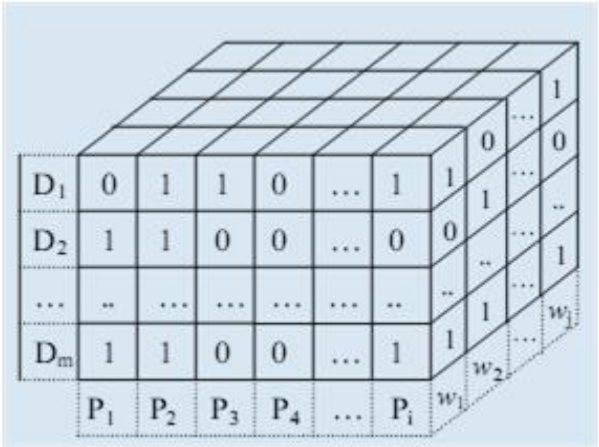


FIGURE 2.10 – Matrice de booléens à 3-dimensions.

B. Graphe ou structure arborescente

Comme nous avons déjà vu précédemment, les documents XML peuvent être représentés par des « *arbres étiquetés* » dont les étiquettes correspondant à des tags XML. Cet arbre peut être ordonné ou non. Un arbre est dit ordonné si les fils de chacun de ses nœuds (non feuilles), sont liés par une relation d'ordre total de gauche à droite. Puisque les documents XML sont représentés comme des arbres, le problème du clustering des documents XML peut être vu comme un problème de clustering des arbres.

Plusieurs travaux portent sur l'utilisation des arbres étiquetés mais en utilisant d'autres techniques, comme les résumés d'arbres [Dalamagas et al., 2005] et la détection des sous-arbres fréquents.

Résumé d'arbre. Un résumé d'arbre est obtenu par deux transformations :

1. ***La réduction des nœuds imbriqués répétés (nested-repeated nodes)***
   Un nœud imbriqué répétée est un nœud non-feuille dont l'étiquette est la même avec celui de son ancêtre. Cette étape permet de réduire la profondeur de l'arbre de sorte que tout nœud (non feuille) ayant la même étiquette avec son ancêtre, ses fils devient des descendants directs (fils) de cet ancêtre.
2. ***L'élimination des nœuds répétés***
   L'objectif de cette phase est de réduire les nœuds répétés dans l'arbre. Un nœud est un nœud répété dont le chemin (à partir de la racine jusqu'au nœud lui-même) a déjà été parcouru avant.

La figure 2.11 illustre un exemple d'extraction d'un résumé pour l'arbre T_1.

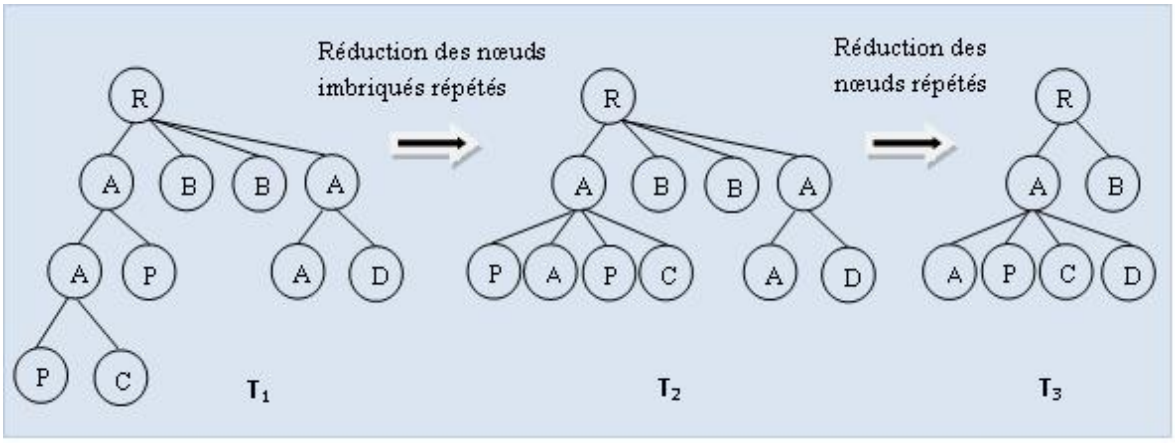


FIGURE 2.11 – Exemple d'extraction d'un résumé d'arbre.

Sous-arbres fréquents. Un document XML peut être représenté par les sous-arbres fréquents qu'il contient [Kutty et al., 2009]. Les sous-arbres fréquents (figure 2.12) sont des sous-arbres qui apparaissent souvent dans des collections d'arbres étiquetés. Plus formellement, étant donné une collection de documents XML $D = \{D_1, D_2, \cdots, D_n\}$ modélisés comme des arbres $DT = \{DT_1, DT_2, \cdots, DT_n\}$ où n représente le nombre de documents XML dans le corpus.
Soit DT' un sous-arbre ($DT' \subseteq DT_K$) qui préserve la relation parent-enfant entre les nœuds que celle de l'arbre DT_K. Ce type de sous-arbre est appelé un sous-arbre induit. Le support de DT' est défini par :

$$Support(DT') = P(DT')/n \tag{2.2}$$

où $P(DT')$ est le nombre d'arbres en DT dans laquelle DT' est un sous-arbre induit et n est le nombre d'arbres dans DT.
Un sous-arbre induit DT' est *fréquent* si son support n'est pas inférieur à un seuil minimum du support min_supp défini par l'utilisateur.

$$Support(DT') \geq min\_supp. \tag{2.3}$$

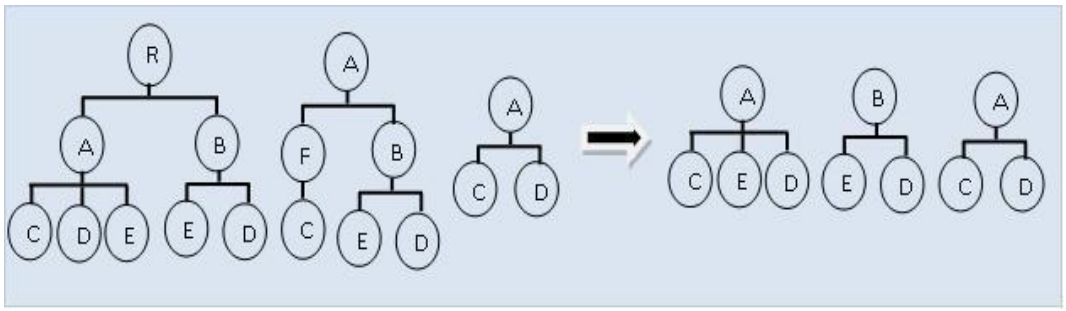


FIGURE 2.12 – Exemple d'extraction des sous arbres fréquents.

En raison de nombre élevé des sous-arbres fréquents générés à des seuils inférieurs de support, des chercheurs récents ont porté sur des représentations condensées sans aucune perte d'informations [Kutty et al., 2007].

La représentation s-graph *(structure graph).* Elle est proposée par [Lian et al., 2004]. La s-graph $SG(d) = (N, E)$ d'un document XML d est un graphe orienté de telle sorte que N est l'ensemble de tous les éléments et les attributs dans le document d et $(a, b) \in E$ si et seulement si a est un élément parent de l'élément b ou b est un attribut de l'élément a dans d. La figure 2.13(b) montre un s-graph correspondant au document de la figure 2.13(a).

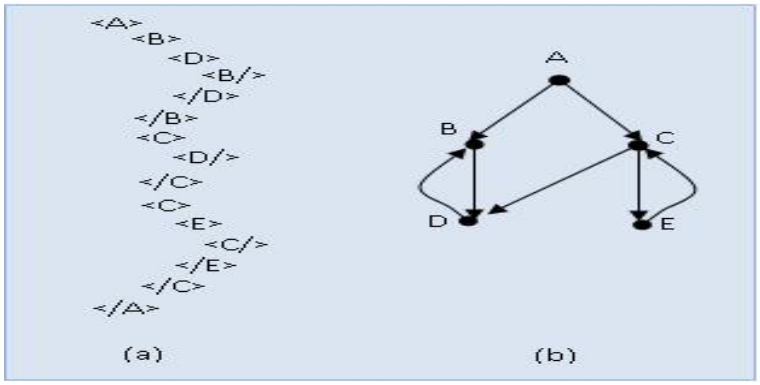


FIGURE 2.13 – (a) Structure d'un document, (b) Son s-graph.

C. Chemins

Un autre type de travaux sur le clustering des documents XML représentent les documents XML par la linéarisation de leurs arbres XML en un ensemble de chemins. Un chemin est une séquence des nœuds qui commence par la racine et se termine par un nœud feuille, sa longueur est égale au nombre des nœuds qu'il contient. Les chemins extraits peuvent ne pas être tous utilisés pour représenter le document XML à cause de coûts en temps de calcul lorsque les documents manipulés sont volumineux. Par exemple, certains travaux fixent la longueur des chemins utilisés [Vercoustre et al., 2006b], d'autres utilisent un seul chemin qui est le plus long chemin [Kim, 2009].

D. Représentations spécifiques

Il existe d'autres représentations que nous avons recensé dans le cadre des représentations spécifiques. Ces représentations décrivent les documents XML de manière différente de celles présenté précédemment. La représentation série chronologique (time series) proposée par [Flesca et al., 2002] est un exemple de ces représentation, elle permet de représenter le squelette d'un document XML sous forme d'une série chronologique des balises.

2.6.3.2 Mesures de similarités dans les documents XML

Le langage XML a enrichi les données par des informations supplémentaires relatives à son structure qu'il est nécessaire d'exploiter. Dans ce contexte, plusieurs travaux ont été effectués pour mesurer la similarité dans les documents XML dans la littérature. Certaines mesures sont à l'origine des adaptations à des mesures existantes pour d'autres applications telles que la classification des données textuelles.

Dans la large gamme de méthodes proposées pour comparer les documents XML, certaines mesures de similarité sont basées sur la sémantique, d'autres sur la structure des documents et certaines combinant les deux en exploitant effectivement ces information structurelles.

Les mesures de similarité dites "sémantiques" sont initialement basées sur le contenu des documents ne tiennent pas en compte de la structure des documents.

Les mesures de similarité basées sur la structure des documents utilisent les informations relatives à l'organisation hiérarchique des données apportées par l'arbre du document XML :

- l'ensemble de nœuds,
- leur position dans l'arbre,
- les relations de descendance et de fraternité,
- les chemins complets de la racine aux feuilles de l'arbre,
- etc.

Nous pouvons aussi classer les différentes approches proposées pour comparer les documents XML en fonction de la représentation utilisée, c'est-à-dire à base d'arbre, graphe, vecteur, etc. Une classification graphique des principales approches existantes est donnée dans la figure 2.14.

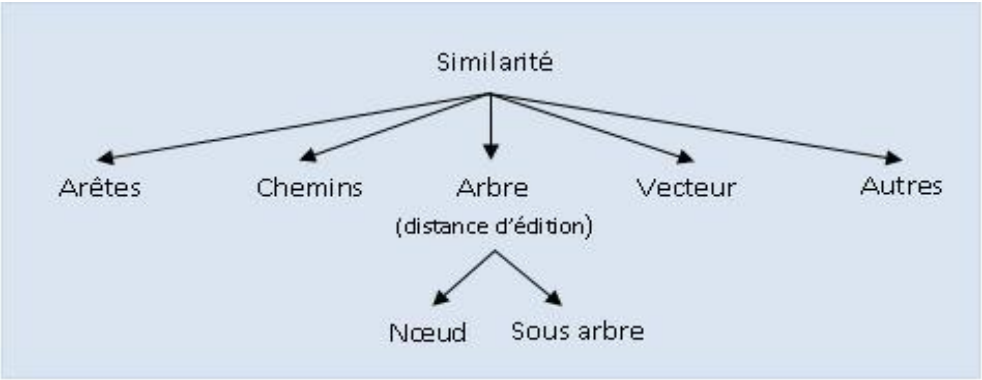


FIGURE 2.14 – Les principales approches de similarité dans les documents XML.

Les mesures de similarité structurelle les plus connus pour comparer les documents XML sont les mesures de similarité basées sur ***les arêtes*** [Lian et al., 2004] [Antonellis et al., 2008] et ***les chemins*** qui sont appelées les métriques ensemblistes (set metric method) selon [Zhang et al., 2002]. Deux documents XML sont similaires s'ils partagent plus des arêtes (respectivement des chemins). Soit T_1 et T_2 deux arbres de documents XML, e_{T_1} et e_{T_2} sont ensemble des arêtes de T_1 et T_2 respectivement, p_{T_1} et p_{T_1} sont l'ensemble des chemins de T_1 et T_2 respectivement.

- La similarité basée sur les arêtes entre deux documents XML T_1 et T_2 est définie comme suit :

$$sim_e(T_1, T_2) = \frac{|e_{T_1} \cap e_{T_2}|}{|e_{T_1} \cup e_{T_1}|}. \tag{2.4}$$

  Où $|e_{T_1} \cup e_{T_1}|$ est le nombre des arêtes communes entre les deux documents T_1 et T_2 ; $|e_{T_1} \cup e_{T_1}|$ désigne l'union des arêtes.

- La similarité basée sur les chemins entre deux documents XML T_1 et T_2 est définie comme suit :

$$sim_p(T_1, T_2) = \frac{|p_{T_1} \cap p_{T_2}|}{|p_{T_1} \cup p_{T_1}|}. \tag{2.5}$$

  Où $|p_{T_1} \cap p_{T_2}|$ est le nombre des chemins communs entre les deux documents T_1 et T_2 ; $|p_{T_1} \cup p_{T_2}|$ désigne l'union des chemins.

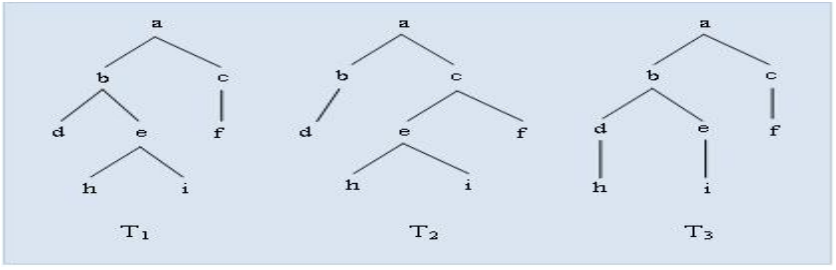


FIGURE 2.15 – Exemple d'arbres.

Contrairement à la similarité basée sur les chemins, la similarité basée sur les arêtes ne tient pas en compte les caractéristiques structurelles des arbres de documents XML, c'est-à-dire le niveau de la relation entre le nœud et l'arête. En effet si nous calculons la similarité basée sur les arêtes et sur les chemins entre les trois arbres présentés dans l'exemple de la figure 2.15, nous obtenons :

- $e_{T_1} = \{a \to b, a \to c, b \to d, b \to e, c \to f, e \to h, e \to i\}$;
- $e_{T_2} = \{a \to b, a \to c, b \to d, c \to e, c \to f, e \to h, e \to i\}$;
- $e_{T_3} = \{a \to b, a \to c, b \to d, b \to e, c \to f, d \to h, e \to i\}$.

$sim_e(T_1, T_2) = \frac{6}{8}$, $sim_e(T_1, T_3) = \frac{6}{8}$ et $sim_e(T_2, T_3) = \frac{5}{9}$.

- $p_{T_1} = \{a \to b \to d, a \to b \to e \to h, a \to b \to e \to i, a \to c \to f\}$;
- $p_{T_2} = \{a \to b \to d, a \to c \to e \to h, a \to b \to e \to i, a \to c \to f\}$;
- $p_{T_3} = \{a \to b \to d \to h, a \to b \to e \to i, a \to c \to f\}$.

$sim_p(T_1, T_2) = \frac{2}{6}$, $sim_p(T_1, T_3) = \frac{2}{5}$ et $sim_p(T_2, T_3) = \frac{1}{6}$.

Il est claire que $sim_e(T_1, T_2)$ et $sim_e(T_2, T_3)$ sont les mêmes, donc nous ne pouvons pas déterminer qui est le plus semblable à T_1, par contre la similarité basée sur les chemins montre que $sim_p(T_1, T_2) \prec sim_p(T_2, T_3)$ et T_3 est le plus proche de T_1 que T_2.

Une grande partie de la recherche sur les mesures de similarité entre deux arbres des documents XML basée sur **la distance d'édition**. La distance d'édition est utilisée pour la première fois pour comparer deux chaînes de caractères. Elle a été définie par *Levenshtein* en 1965. Il s'agit de compter le nombre minimal d'opérations nécessaires pour transformer une chaîne à une autre. Les opérations de base sont l'insertion, la substitution et la suppression d'un caractère. Par exemple, si nous considérons les deux mots *abaa* et *aab*, la distance d'édition entre ces deux mots $d_{edit}(abaa, aab) = 2$. *abaa* se récrit en *aab* via (par exemple) la suppression du b et la substitution du dernier a par un b. Cette distance a été calculée par un algorithme de programmation dynamique présenté dans [Wanger et Fischer, 1974].

La distance d'édition des arbres est une généralisation de la distance d'édition utilisée dans les chaînes de caractères aux arbres étiquetés. Elle est définie par le coût minimal pour transformer un arbre en un autre par les opérations d'édition. Le problème donc est de trouver la séquence d'opérations qui vérifie cette caractéristique (coût minimal). Plus cette distance est faible, plus la similarité est forte. Une opération d'édition entre deux arbres T_1 et T_2 est l'une des opérations suivantes [Ben Hassena et Miclet, 2008] :

- ***Substitution (modification)***
  Cette opération consiste simplement à modifier l'étiquette d'un nœud sans changer la structure de l'arbre. La figure 2.16 illustre ce principe. L'étiquette i dans l'arbre T_1 est substituée par une étiquette b pour former l'arbre T_2.
- ***Suppression***
  Cette opération modifie la structure de l'arbre. Lors de la suppression d'un nœud v, les nœuds fils de v s'insère à la place que v occupait, devenant ainsi des fils du nœud père de v. La figure 2.16 présente un exemple de suppression (la suppression de nœud a dans l'arbre T_2 donne l'arbre T_3).
- ***Insertion***
  Insérer un nœud w en tant que fils d'un nœud v fait de w le père d'une séquence de nœuds consécutifs précédemment fils de v. Un exemple d'insertion est fourni par la figure 2.16 (l'insertion de nœud a dans T_3 comme un fils de b donne T_4). Notons que contrairement aux opérations de substitution et de suppression, il n'existe pas une seule et unique façon d'insérer un nœud à un endroit donné en suivant cette définition.

Il existe différentes approches pour déterminer les séquences d'opérations possibles et la distance d'édition des arbres. Tous utilisent des opérations d'édition semblables avec des variantes mineures. Certaines approches posent des conditions ou des restrictions sur les opérations d'édition utilisées par exemple *la suppression d'un nœud est valide seulement lorsque le nœud est un nœud feuille, chaque nouveau nœud est inséré comme un nœud feuille.* [Dalamagas et al., 2006] résume dans le tableau 2.1 les principales approches proposées pour calculer la distance d'édition des arbres.

Nous avons trouvé aussi que certaines approches donnent des coûts similaires à chaque

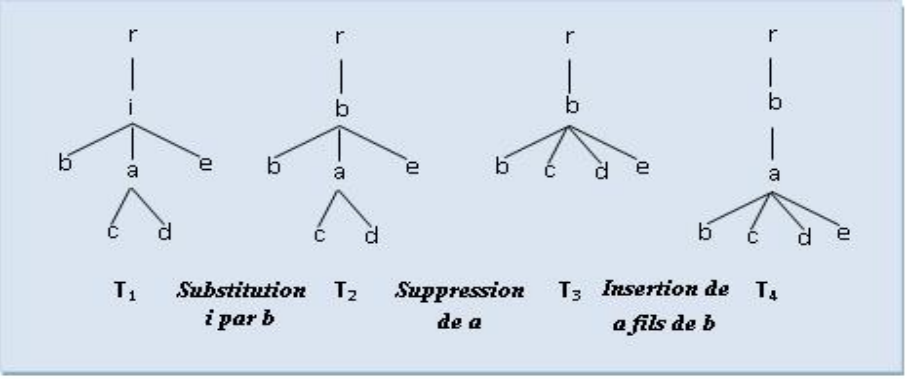


FIGURE 2.16 – Les opérations d'édition.

Algorithme	Opérations	Réservé aux feuilles	Complexité
[Selkow, 1977]	Insérer un nœud, Supprimer un nœud, Substituer un nœud.	Insérer un nœud, Supprimer un nœud.	$4^{min(NM)}$, M et N sont le nombre de nœuds pour chaque arbre.
[Zhang et Shasha, 1989]	Insérer un nœud, Supprimer un nœud, Substituer un nœud.		$O(M.N.b.d)$, M et N est le nombre de nœuds pour chaque arbre, b et d sont les profondeurs des deux arbres, respectivement.
[Chawathe et al., 1996]	Insérer un nœud, Supprimer un nœud, Substituer un nœud, Déplacer sous arbre.	Insérer un nœud, Supprimer un nœud.	$O(ND)$, N le nombre de nœuds dans les arbres et D le nombre de nœuds alignés.
[Chawathe, 1999]	Insérer un nœud, Supprimer un nœud, Substituer un nœud.	Insérer un nœud, Supprimer un nœud.	$O(M.N)$, M et N sont les dimensions de la matrice qui représente le graphe d'édition.

TABLE 2.1 – *Les algorithmes de la distance d'édition des arbres.*

opération comme [Zhang et Shasha, 1989], d'autres donnent des coûts variables à chaque type d'opération, ils considèrent que le coût d'une opération d'insertion ou de suppression n'est pas équivalent à celui d'une substitution. Ainsi qu'il a un impact d'insérer ou supprimer un nœud prés du nœud racine ou de nœud feuille. Les opérations près du nœud racine sont plus importantes que celles près des nœuds feuilles [Zhang et al., 2002]. [Zhang et al., 2002] présentent plusieurs facteurs qui sont pris en considération pour attribuer un coût à une opération d'édition : le type d'opération, le niveau de nœud, nombre de fils et l'interprétation sémantique des nœuds (un même nœud peut avoir différentes interprétations avec un nœud père différent).

Les approches de la distance d'édition qui utilisent des opérations d'édition sur les nœuds ne distinguent pas entre les documents XML produit à partir de la même DTD (qui peuvent avoir des tailles très différentes sur le compte des éléments optionnels et répétés). Puisque ces approches permettent le changement d'un seul nœud à la fois, ce qui implique une grande distance et par conséquent ne peuvent pas reconnaître que ses documents doivent être re-

groupés comme étant issus de la même DTD. Les approches qui appliquent des opérations d'édition sur des sous-arbres plutôt que sur des nœuds comme [Nierman et Jagadish, 2002] sont introduits pour surmonter cette limitation.

Dans le cas, où les documents XML sont représentés sous formes **de vecteurs**, nombreuses mesures de similarité ont été utilisées dans ce modèle, parmi eux nous pouvons citer : la distance Euclidienne, le Produit interne, la mesure de Cosinus, le coefficient de Dice et le coefficient de Jaccard. Ces mesures sont définies successivement selon les équations suivantes :

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{2.6}$$

$$d(x,y) = \sum_{i}^{m}(x_i * y_i) \tag{2.7}$$

$$d(x,y) = \frac{\sum_{i}^{m}(x_i * y_i)}{\sqrt{\sum_{i}^{m}(x_i)^2 + \sum_{i}^{m}(y_i)^2}} \tag{2.8}$$

$$d(x,y) = 2 * \frac{\sum_{i}^{m}(x_i * y_i)}{\sqrt{\sum_{i}^{m}(x_i)^2 + \sum_{i}^{m}(y_i)^2}} \tag{2.9}$$

$$d(x,y) = \frac{\sum_{i}^{m}(x_i * y_i)}{\sqrt{\sum_{i}^{m}(x_i)^2 + \sum_{i}^{m}(y_i)^2 - \sum_{i}^{m}(x_i * y_i)}}. \tag{2.10}$$

2.6.3.3 Les mesures d'évaluation

Les critères ou les mesures d'évaluation de la qualité du clustering des documents XML peuvent être divisées en deux familles. La première famille contient les mesures *supervisées* qui calculent le degré de correspondance entre le clustering produit par l'algorithme et un partitionnement connu des documents. Ces mesures sont aussi connues sous le nom de *mesures de qualité externes*. Nous présentons dans la suite les mesures d'évaluation externes les plus connues. Mais avant, nous définissons les différents variables utilisés dans ces mesures :

- n_i^r : Nombre de documents de la classe Z_r affectés au cluster C_i,
- n_i : nombre de documents XML dans C_i,
- n_r : nombre de documents XML dans Z_r,
- k : nombre de classes,
- q : nombre de clusters,

- n : nombre de documents dans la collection.

– ***Rappel*** *(recall) et* ***Précision*** *(precision)*
Le rappel et la précision sont deux mesures très populaires dans la recherche d'information. En classification, *le rappel* R est le rapport entre le nombre de documents correctement classés dans le cluster et le nombre total de documents de la classe. *La précision* P est le rapport entre le nombre de documents correctement classés dans le cluster et le nombre de tous les documents de ce cluster. Une haute valeur de précision signifie une haute précision de la tâche de clustering pour chaque cluster, tandis qu'une faible valeur de rappel signifie qu'il y a beaucoup de documents XML qui ne sont pas dans le cluster approprié bien qu'ils le devraient. Une haute valeur de rappel et de précision indique une excellente qualité de clustering.

$$R(Z_r, C_i) = \frac{n_i^r}{n_r}, \tag{2.11}$$

$$P(Z_r, C_i) = \frac{n_i^r}{n_i}. \tag{2.12}$$

– ***F_1-mesure*** *(F_1-measure)*
Elle est aussi connu sous le nom F-mesure ou F-score. Elle combine les mesures de précision et de rappel pour essayer d'assigner chaque classe calculée à une classe prédéfinie de telle sorte que deux classes calculées ne peuvent pas être assignés à la même classe prédéfinie. La F_1-measure mesure un équilibre entre la précision et le rappel. C'est la moyenne harmonique des deux valeurs précédentes.

$$F_1 = \frac{2 * P * R}{P + R}. \tag{2.13}$$

Les valeurs F-mesure sont dans l'intervalle $[0, 1]$. Les grandes valeurs de F-mesure correspondent à la qualité de clustering supérieure. La F_1-mesure globale de tout le clustering peut être calculée par deux types différents de la moyenne, *macro-et micro-moyenne* [**?**].
Macro-moyenne F_1. En macro-moyenne, la F_1-mesure est calculée localement pour chaque cluster, puis la moyenne sur tous les clusters est prise :

$$F_{1(marco-moyenne)} = \frac{\sum_{i=1}^{q} F_{1i}}{|q|}. \tag{2.14}$$

La micro-moyenne F_1 correspond à la moyenne des mesures F_{1i} pondérée par la taille de chaque cluster .

$$F_{1(mirco-moyenne)} = \frac{\sum_{i=1}^{q} n_i F_{1i}}{\sum_{i=1}^{q} n_i} \tag{2.15}$$

– ***Entropie*** *(entropy)*
Elle mesure la façon dont les classes prédéfinies sont distribuées ou réparties dans chaque cluster. Une valeur faible de l'entropie correspond à un meilleur clustering. Etant donné un cluster C_i, l'entropie d'un cluster est définie comme suite :

$$E(C_i) = \frac{1}{\log k} \sum_{r=1}^{k} \frac{n_i^r}{n_i} \log \frac{n_i^r}{n_i}. \tag{2.16}$$

L'entropie totale est obtenue par la somme pondérée des entropies de clusters :

$$Entropie = \sum_{i=1}^{q} \frac{n_i}{n} E(C_i). \tag{2.17}$$

– ***Pureté*** *(purity)*
Il mesure la proportion de documents de la classe calculée appartenant à la classe à priori la plus fréquente, c'est-à-dire le pourcentage de documents de la classe calculée appartenant à la classe majoritaire. La pureté d'une classe calculée est égale à 1 si tous les documents sont issus de la même classe a priori.

$$P(C_i) = \frac{1}{n_i} \max(n_i^r). \tag{2.18}$$

La pureté totale est obtenue par la somme pondérée des puretés de clusters, sa formule est la suivante :

$$Pureté = \sum_{i=1}^{q} \frac{n_i}{n} P(C_i). \tag{2.19}$$

– ***Rand corrigé*** *(Corrected Rand)*
Cette métrique est peut être utiliser pour comparer le résultat de la classification automatique avec une classification existante, ou bien de comparer deux partitions obtenues à partir de l'algorithme de classification automatique.

$$CR = \frac{\sum_{i=1}^{q}\sum_{r=1}^{k} \binom{n_i^r}{2} - \binom{n}{2}^{-1} \sum_{r=1}^{k} \binom{n_r}{2} \sum_{i=1}^{q} \binom{n_i}{2}}{\frac{1}{2}\left[\sum_{r=1}^{k} \binom{n_r}{2} + \sum_{i=1}^{q} \binom{n_i}{2}\right] - \binom{n}{2}^{-1} \sum_{r=1}^{k} \binom{n_r}{2} \sum_{i=1}^{q} \binom{n_i}{2}}. \tag{2.20}$$

Où $\binom{n}{2} = \frac{n(n-1)}{2}$.
La valeur de $CR \in [0, 1]$. Une valeur proche de 0 correspond à une partition aléatoire.

La seconde famille contient les mesures *non supervisées* qui se basent sur des informations internes au clustering comme par exemple la distance entre les documents d'un cluster et le centroïde de celui-ci. Ces mesures se basent souvent sur la définition la plus simple du clustering qui définit que les documents d'un même cluster doivent être les plus proches possible entre eux et que les documents de deux clusters distincts doivent être les plus

éloignés possible. Ces mesures non supervisées permettent d'évaluer la compacité ainsi que la séparabilité des clusters. Ces mesures sont également appelées *mesures de qualité internes*. Un exemple de ces mesures est **le coefficient silhouette** (CS). Il peut être calculé pour chaque document, pour chaque cluster et pour le clustering entier. Pour un document$x$ il est défini comme :

$$CS(x) = \frac{b_x - a_x}{\max(a_x, b_x)}, \tag{2.21}$$

avec a_x la distance moyenne entre le document x et tous les autres documents appartenant au même cluster que x, et b_x la distance moyenne entre x et tous les documents n'appartenant pas à ce même cluster. Le coefficient $CS(x)$ varie entre -1 et 1. Une valeur positive ($a_x < b_x$) signifie que les documents appartenant au même cluster que x sont plus proches de x que des documents des autres groupes.
Pour un cluster, le coefficient silhouette est la moyenne des coefficients des documents appartenant à ce cluster :

$$CS(C_i) = \frac{1}{|C_i|} \sum_{x \in C_i} (CS(x)). \tag{2.22}$$

Enfin, pour un clustering, le coefficient silhouette est égal à la moyenne des coefficients de ses clusters :

$$CS(C) = \frac{1}{K} \sum_{i=1}^{K} (CS(C_i)). \tag{2.23}$$

Le coefficient pour le clustering varie également entre -1 et 1, une valeur positive indiquant que les clusters sont très compacts et bien séparés.

2.7 Conclusion

XML est un standard qui a prouvé son existence et son efficacité dans le processus de transmission et de partage des données sur le Web. Le XML mining est apparu comme une solution pour acquérir des nouvelles connaissances à partir des sources XML qui sont devenus plus abondantes, hétérogènes et de structure irrégulière.

Dans ce chapitre, nous avons présenté la fouille de données dans les documents XML, là où, nous avons parlé en premier sur le métalangage XML qui permet de représenter les documents semi-structurés grâce à son système de balisage qui offre une structure hiérarchique et logique aux documents.

Ensuite, nous avons présenté les documents XML qui sont classés en documents bien formés respectant des règles syntaxiques propre au langage XML, et les documents valides ou bien formés conforment à une DTD ou un schéma XML. Ces derniers sont aussi discutés dans ce chapitre comme des langages de définition de structure des documents XML. Nous avons montré que les schémas XML apportent plusieurs avantages par apport aux DTD en dépit de leur complexité.

Dans un deuxième temps, nous avons présenté le XML mining qui est un domaine de recherche très actif basé sur les techniques de data mining. Le XML mining est un domaine à part entière vu les sources de données sur lesqelles s'opère. Nous avons vu également les trois catégories de XML mining qui sont : XML content mining, XML structure mining et Combined XML mining. De plus, nous avons présenté le processus de découverte des connaissances à partir de documents XML. Enfin, Nous avons évoqué sur les techniques de data mining utilisées par le XML mining notamment l'association, la classification et le clustering où nous nous sommes focalisé.

Dans le clustering, nous avons présentés les différents modèles proposés pour représenter les documents XML pour lesquelles nous avons proposé une taxonomie. Les plus utilisés sont : le modèle vectoriel, l'arbre et les chemins. Nous avons présenté aussi différentes mesures de similarité entre les documents XML dont certains sont des adaptations de mesures existantes pour d'autres applications telles que la classification des données textuelles. Nous avons parlé sur ceux basés sur les vecteurs, les arbres, les arêtes et les chemins. Nous avons présenté différents critères permettant l'évaluation de résultats de clustering de manière supervisée ou non. Les critères sont dit externes car ils utilisent des informations externes au clustering pour son évaluation contrairement aux critères interne qui ne les utilisent pas. Plusieurs critères existent car plusieurs approches différentes peuvent être envisagées pour juger de la qualité d'un partitionnement. Pour prendre une idée sur les méthodes de clustering des documents XML, un état d'art des travaux de clustering de ce type de documents est présentée dans le chapitre suivant.

CHAPITRE 3

CLUSTERING DE DOCUMENTS XML : ETAT DE L'ART

« Le collectionneur est un créateur essayant de bâtir un tout cohérent et par là capable de faire œuvre originale ! ».

Michel Melot, "Extrait de la Revue de l'Art"

La nécessité de regrouper les documents XML a conduit à une augmentation des recherches sur des algorithmes de clustering pour ce type de documents.

Dans le clustering des documents XML, nous distinguons trois approches de regroupement, la première basée sur le contenu, la deuxième basée sur la structure et la dernière combine le contenu et la structure des documents XML.

Dans ce chapitre, nous présentons un aperçu des méthodes utilisées pour le clustering des documents XML. Les travaux existants sur le clustering de documents XML se distinguent par la manière de représenter les documents, les mesures de similarité utilisées et les méthodes de clustering adoptées.

3.1 Travaux sur le clustering des documents XML

Dans cette section, nous présentons quelques travaux sur la problématique de clustering des documents XML en mettant l'accent sur les modèles de représentation, les mesures de similarité et les méthodes de clustering utilisées. Ces travaux sont divisés selon deux approches : celles qui prennent en compte la structure du document uniquement et celles qui prennent en considération la structure et le contenu.

3.1.1 Selon la Structure

Plusieurs approches de clustering ont été récemment mises au point, mettant principalement l'accent sur le regroupement des documents XML par la structure. Ceci est principalement motivé par plusieurs applications dans la gestion de données semi-structurées, en particulier dans les environnements Web, comme la recherche et l'intégration des données semi-structurées.

L'objectif de l'utilisation de la structure uniquement est généralement d'organiser une grande collection de documents hétérogènes en plus petites collections (clusters) qui peuvent être stockées et recherchées plus efficacement. Une partie de l'objectif est d'identifier les sous-structures qui caractérisent les documents dans un cluster et de construire un représentant du cluster, qui peut être un schéma ou une DTD.

Dans les travaux de clustering structurel proposés, nous distinguons deux principaux courants ou deux niveaux de regroupement :

1. *niveau intentionnel :* dans ce niveau, les DTD ou les schémas XML sont pris en compte. L'idée des auteurs est au lieu de classer directement les documents XML, ils classent leurs DTD ou leurs schémas XML.
2. *niveau extensionnel :* les corps des documents XML sont classés.

Dans cette sous section, nous nous intéressons aux approches de clustering extensionnelles, où de nombreux travaux ont été proposés.

3.1.1.1 Travaux de [Nierman et Jagadish, 2002]

L'idée de regrouper les documents XML générés par la même DTD en utilisant la structure a été proposée la première fois par [Nierman et Jagadish, 2002], les auteurs ont suggéré d'utiliser la distance d'édition entre les structures d'arbres (ordonnées et étiquetées). En plus des opérations d'insertion, suppression et renommage d'un nœud de [Zhang et Shasha, 1989], ils ont introduit également deux autres opérations, l'insertion et la suppression de sous-arbres. Plus précisément, le fonctionnement $InsertTreeT(A, i)$ ajoute A comme un enfant de T à une position i et $DeleteTreeT(T_i)$ supprime T_i comme l'i$^{\text{ème}}$ enfant de T.

Mais, ils imposent que l'utilisation de ces deux opérations est limitée au sous-arbre partagé entre l'arbre de source et de destination. Sans cette restriction, on pourrait supprimer l'arbre complet de source en une seule étape et d'insérer l'arbre de destination complet dans une deuxième étape, ce qui rend totalement inutiles les opérations d'insertion et de suppression. Une deuxième restriction impose qu'un arbre qui a été insérée par l'opération $InsertTree$ ne peut pas ensuite avoir des nœuds supplémentaires insérés, par analogie, un arbre qui a été supprimés via l'opération $DeleteTree$ ne peut pas préalablement eu des nœuds (fils) supprimés. Mais le calcul des distances d'édition est assez coûteux et l'algorithme nécessite le calcul de cette distance entre chaque paire de documents (A, B), sa complexité en

temps est de l'ordre de $O(|A||B|)$.

La méthode hiérarchique ascendante est suivie pour regrouper les documents XML. Plus précisément la méthode UPGMA (Unweighted Pair Group Averaging Method). La distance entre deux clusters C_i et C_j est calculée comme suit :

$$Distance(C_i, C_j) = \frac{\sum_{k=1}^{|C_i|}\sum_{l=1}^{|C_j|} d(doc_k^{C^i}, doc_l^{C^j})}{|C_i|\,|C_j|}. \tag{3.1}$$

Où $|C_i|$ est le nombre de documents XML dans le cluster C_i et et $doc_k^{C_i}$ est le k$^{\text{ème}}$ document XML dans le cluster C_i.

L'expérience a été effectuée sur deux ensembles de données : réelles ACMSIGMOD et synthétiques. Afin d'évaluer les résultats du regroupement, ils ont introduit la notion de "mis-clustering" ou "mal-clustering". Étant donné un dendrogramme [1], le nombre de mis-clusterings est égal au nombre minimum de documents dans le dendrogramme qui doivent être déplacés afin que tous les documents de la même DTD soient regroupés. Les auteurs affirment que leur approche est plus performante que celle de [Chawathe, 1999] sur les mêmes collections de documents utilisées, et elle atteint une valeur de mis-clustering nulle.

3.1.1.2 Travaux de [Lian et al., 2004]

Lian et al. proposent de représenter la structure d'un document XML par ***un s-graph ou structure graph***. La distance entre deux documents C_1 et C_2 est définie par la formule suivante :

$$dist(C_1, C_2) = 1 - \frac{|sg(C_1) \cap sg(C_2)|}{\max\{|sg(C_1)|, |sg(C_2)|\}}. \tag{3.2}$$

Où $|sg(C_i)|$ est le nombre des arrêtes de $sg(C_i)$ et $sg(C_1) \cap sg(C_2)$ est l'ensemble des arêtes communes entre $sg(C_1)$ et $sg(C_2)$.

Pour regrouper les documents XML, les auteurs ont proposé l'algorithme S-GRACE, un algorithme de clustering hiérarchique qui s'applique sur les s-graphs.

L'expérience a été réalisée sur des données synthétiques et réelles *DBLP*, avec quatre indicateurs qui mesurent la qualité des clusters découverts par S-GRACE qui sont :

- CS : la mesure de la proximité entre les clusters découverts par S-GRACE et les clusters dans les données.
- IS : la similarité moyenne de toutes les paires de clusters trouvés par S-GRACE.
- SD : l'écart type du nombre de documents dans les clusters trouvés par S-GRACE.
- R : le ratio de documents aberrants trouvés par S-GRACE.

Un meilleur clustering se traduit par une grande valeur de CS (près à 1) et une petite valeur de IS et SD proche de 0. La valeur de R doit être proche de la valeur R définie dans la collection des documents utilisés.

1. Un dendrogramme est le diagramme illustrant en arbre les groupes définis par une classification hiérarchique.

Les résultats de l'évaluation montrent l'efficacité de cette approche, cependant, elle présente un problème majeur souligné par [Costa et al., 2004] sur la similarité en grain vrac (loose-grained) qui se produit. En effet, deux documents peuvent partager le même s-graph, et ont toujours des différences structurelles significatives. En outre, la similarité entre les deux s-graphs de la figure 3.1 est 1 en fonction de leur définition. Ainsi, la mesure ne tient pas compte des documents similaires qui ne partagent pas des arêtes communes, même s'ils ont certains éléments avec les mêmes étiquettes.

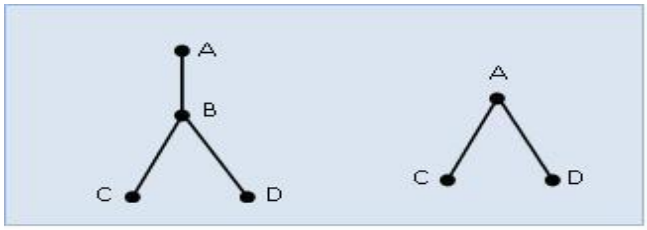


FIGURE 3.1 – Exemple de deux s-graphes simples.

3.1.1.3 Travaux de [Dalamagas et al., 2005]

Cette approche est basée sur l'extraction des résumés structurels des arbres ordonnés et étiquetés correspondants aux documents XML par la réduction des éléments répétés et les éléments imbriqués répétés. L'idée de motivation est que l'imbrication et la répétition des éléments sont la raison principale pour les documents XML diffèrent dans la structure même s'ils proviennent de la même DTD.

Pour mesurer la similarité entre les résumés structurels des documents XML, les auteurs ont défini une distance structurelle S basée sur la distance d'édition :

$$S(T_1, T_2) = \frac{D(T_1, T_2)}{D'(T_1, T_2)}. \tag{3.3}$$

Où :

- $D(T_1, T_2)$ est la distance d'édition entre les résumés d'arbres T_1 et T_2.
- $D'(T_1, T_2)$ est le coût de supprimer tous les nœuds de T_1 et d'insérer tous les nœuds du T_2.

Dalamagas et al. ont utilisé un algorithme de programmation dynamique pour calculer la distance d'édition qui est proche de l'algorithme de Chawathe [Chawathe, 1999] en termes des opérations d'édition utilisés. Les opérations d'éditions autorisées ici sont l'insertion, la suppression et la substitution (l'insertion et la suppression sont limitées aux nœuds feuilles) avec un coût égal à 1 pour chaque opération.

Les auteurs ont choisi l'algorithme Single-link de clustering hiérarchique pour regrouper les documents XML.

L'expérience a été réalisée sur des données synthétiques (1000 documents) et réelles (150 documents provenant du ACMSIGMOD et ADC/NASA). Le rappel et la précision sont utilisés pour évaluer la qualité de clustering. Les résultats de l'évaluation indiquent que l'utilisation des résumés d'arbres permet d'améliorer la procédure de regroupement et de préserver la qualité de clustering (précision = rappel = 1.0).

3.1.1.4 Travaux de [Iyer et Simovici, 2008]

Les auteurs proposent de modéliser un document XML comme un arbre enraciné et étiqueté puis de représenter les chemins marqués enracinée de l'arbre comme un multi-ensemble (multiset) des chemins, qui est une fonction de mappage de chaque chemin à sa multiplicité, c'est-à-dire, le nombre d'occurrences du chemin dans l'arbre.
Soit l'arbre étiqueté enraciné (T, v_0, l, L), où (T, v_0) est un arbre dont la racine v_0, $l : V \to L$ est une fonction, L est l'ensemble des étiquettes des éléments ; $l(v)$ est l'étiquette du nœud v. Le multi-ensemble des chemins marqués enracinés d'un arbre étiqueté enraciné (T, v_0, l, L), désigné par $RLP(T, v_0, l, L)$, est un multi-ensemble des séquences d'étiquettes. Cet ensemble peut être défini récursivement comme suit :

1. Si $T = (\{v_0\}, \phi)$ et $l(v_0) = a$, alors $RLP(T, v_0, l, L) = 1(a)$.
2. Supposons que les descendants directs de v_0 en T sont $v_1, \cdots, v_m$, et les sous-arbres de T ayant des racines dans $v_1, \cdots, v_m$ sont $T_1, \cdots, T_m$, respectivement. Alors,
   $RLP(T, v_0, l, L) = 1(a) + \sum_{i=1}^{m} aRLP(T_i, v_i, l, L).$

La figure 3.2 représente un exemple des Multi-ensembles de chemins correspondant aux arbres T_1 et T_2.

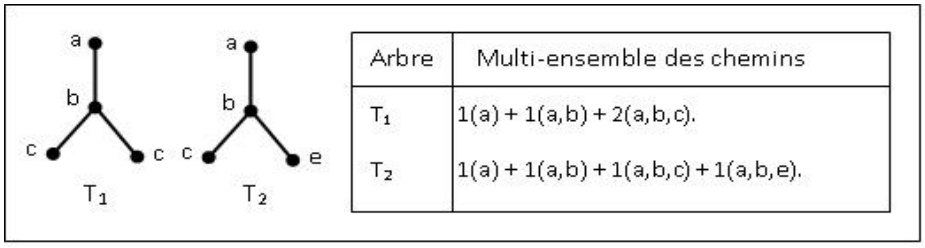


FIGURE 3.2 – Exemple des multi-ensembles des chemins.

Les auteurs ont introduit deux mesures de dissimilarité, faible et forte, entre les arbres enracinés et étiquetés basés sur la différence symétrique de leurs multi-ensembles, et la méthode hiérarchique ascendante moyenne est appliquée pour regrouper les documents similaires. La dissimilarité *faible (weak dissimilarity)* entre deux arbres R et R' est défini comme suit :

$$d_{\oplus}(R, R') = \sum_{k \in N} 2^{-(k+1)} |L|^{-k} \cdot \sum \frac{(RLP(R) \oplus RLP(R'))(P)}{(RLP(R) \cup RLP(R'))(P)} \tag{3.4}$$

et la dissimilarité forte (strong dissimilait) entre R et R' est défini par :

$$d_{\boxplus}(R, R') = \sum_{k \in N} 2^{-(k+1)} |L|^{-k} \cdot \sum \Psi_{RLP(R), RLP(R')}(P) \tag{3.5}$$

où :

- $p \in seq(L)$(seq : séquence), $l(p) = k(l : longeur)$,
- $\oplus$: c'est l'opérateur de la différence symétrique faible

$$(RLP(R) \oplus RLP(R'))(P) = \left|(RLP(R) - RLP(R'))(P)\right|. \tag{3.6}$$

- Ψ est une fonction définie par :

$$\Psi_{RLP(R), RLP(R')}(P) = \begin{cases} 0 & si RLP(R)(p) = RLP(R')(p) \quad (3.7) \\ \dfrac{RLP(R) \boxplus RLP(R')}{RLP(R) \cup RLP(R')} & \text{autre} \quad (3.8) \end{cases}$$

- $\boxplus$: c'est l'opérateur de la différence symétrique forte
  $(RLP(R) \boxplus RLP(R'))(p) = \phi(RLP(R), RLP(R'))$ où ϕ est une fonction défini comme suit :

$$\phi(RLP(R), RLP(R'))(P) = \begin{cases} 0 & si RLP(R)(x) = RLP(R')(x) \quad (3.9) \\ \max\left\{RLP(R)(x), RLP(R')(x)\right\} & \text{Si exactement une de RLP(R)(x),RLP(R') positive} \quad (3.10) \end{cases}$$

Cette méthode a été effectuée sur trois ensembles de données *Niagara* (45 documents), *Sigmod* (48 documents) et données synthétisés (45 documents). La qualité des clusters produits a été évaluée par l'utilisation du coefficient de Silhouette. Les résultats trouvés montrent que leur approche séparent bien les documents qui sont structurellement différents où ils ont atteint une valeur de Silhouette égale à 1.

3.1.1.5 Travaux de [Kim, 2010]

Dans cet article, l'auteur propose de regrouper les documents XML par l'utilisation d'un vecteur de bits dans la représentation des documents. Un vecteur de bits est construit par les chemins de l'arbre correspondant au document XML. Etant donné P l'ensemble des chemins correspondant à l'ensemble des documents XML D, si $p_j \in d_i$ alors $(d_i, p_j) = 1$, sinon $(d_i, p_j) = 0$.

La similarité entre deux documents XML est mesurée par l'opération *ET bit à bit (bit-wise and)* entre les vecteurs de bits correspondants aux documents. L'auteur propose deux méthodes différentes de l'utilisation des vecteurs de bits pour regrouper les documents XML :

- **Les informations de DTD sont disponibles**
  Les opérations *ET bit à bit* sont effectuées entre tous les vecteurs de bits des DTD et les vecteurs de bits des documents à regrouper. Un document est affecté dans le groupe de DTD où son opération ET bit à bit a la plus grande valeur.
- **Les informations de DTD ne sont pas disponibles**
  Dans ce cas, l'opération *ET bit à bit* est utilisée entre les vecteurs de bits correspondants aux documents XML. Les deux algorithmes Single-linkage et Complete-linkage de clustering hiérarchique agglomérative sont appliqués pour regrouper les documents XML dans ce cas.

L'expérience qui a été réalisé sur 80 documents XML fournit par l'université du Wisconsin, et qui sont dérivés de huit DTD montre que les clusters obtenus dans les deux cas sont bien formés (nombre de documents incorrectement regroupés NIC=0).

3.1.1.6 Travaux de [Alishahi et al., 2010]

Alishahi et al. proposent une nouvelle structure de niveau basée sur la structure proposée dans [Nayak et Xu, 2006], l'idée de cette structure est outre le nom des éléments, il faut examiner les informations sur leurs parents, aussi. Chaque niveau de la nouvelle structure va stocker les noms d'éléments et de leurs parents pour trouver les éléments communs entre deux objets.

Les auteurs proposent aussi, un algorithme appelé XCLS+ qui est une amélioration de l'algorithme XCLS (XML Clustering by Level Structure) proposé dans [Nayak et Xu, 2006]. Dans l'algorithme XCLS+, les auteurs essayent d'ordonner les éléments par les noms de balises (Tag names) et ils suivent les étapes de l'algorithme 1 pour trouver tous les éléments communs.

La similarité entre deux documents en XCLS+ est calculée par la formule :

$$LevelSim+_{1,2} = \frac{0.5 \times \sum_{i=0}^{L-1} (CN_1^i + CP_1^i) \times (r)^{L-i-1} + 0.5 \times \sum_{j=0}^{L-1} \left(CN_2^j + CP_2^j\right) \times (r)^{L-j-1}}{\left(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}\right) + 0.5 \times \left(\sum_{i=0}^{L-1} CP_1^i \times r^{L-i-1} + \sum_{j=0}^{L-1} CP_2^j \times r^{L-j-1}\right)} \quad (3.11)$$

Où :

- CN_1^i et CN_2^j : désignent le nombre d'éléments communs dans le niveau i de l'objet 1 et le niveau j de l'objet 2 respectivement.
- CP_1^i et CP_2^j : représent le nombre d'occurrences de tous les éléments communs dans le niveau i de l'objet 1 qui ont le même parent et le niveau j de l'objet 2 qui ont le même parent.
- N^k : Nombre d'éléments dans le niveau k de l'arbre.
- r (base de Poids) : facteur de l'augmentation de poids. Cela est généralement supérieur à 1 pour indiquer que les éléments de niveau supérieur ont plus d'importance que les éléments de niveau inférieur.
- L : Nombre de niveaux dans l'arbre.

Algorithme 1 : Les étapes de l'appariement des éléments dans XCLS+

Entrées :
$O1$, $O2$: deux nouvelles structures de niveau représentées comme un tableau ordonné par TagName, Level;
$OL1$: Le nombre de lignes de $O1$ dans $\{OL1_1, OL1_2, ..., OL1_w\}$;
$OL2$: Le nombre de lignes de $O2$ dans $\{OL2_1, OL2_2, ..., OL2_z\}$;
Sorties :
$CN1\,[1..w]$ nombre;
$CN2\,[1..z]$ nombre;

```
début
    répéter
        Comparer chaque ligne i de O1 à chaque ligne j de O2;
        si O1.TagName = O2.TagName alors
            CN1[OL1_i] ++;
            CN2[OL2_j] ++;
            Aller aux lignes suivantes des deux objets O1 et O2;
        fin
        si O1.TagName ≻ O2.TagName alors
            Aller à la ligne suivante de O2 seulement;
        sinon
            Aller à la ligne suivante de O1 seulement;
        fin
    jusqu'à toutes les lignes des deux objets sont vérifiées;
fin
```

Deux ensembles de documents XML ont été utilisé avec les mesures : entropie, pureté et F-mesure pour évaluer les performances de cet algorithme par rapport à l'algorithme XCLS. Le premier ensemble contient des documents XML hétérogènes de l'université du Wisconsin (460 documents) avec un nombre d'éléments varie entre 10 et 100 éléments et un niveau d'imbrication varie entre 2 et 15 niveaux. Le deuxième ensemble contient des documents XML homogènes de l'université de Washington (164 documents) dérivé de quatre sous DTD de dblp DTD.

Les résultats ont montrés que les performances de cet algorithme sont améliorées par rapport aux celles fourni par XCLS, et il atteint une valeur d'entropie égale à 0 et une valeur de pureté et F-mesure égale à 1 dans le cas des document hétérogènes. XCLS+ possède une complexité de temporelle linéaire.

3.1.2 Selon le contenu et la Structure

La combinaison de l'information de structure et de contenu dans le clustering des documents XML est une tendance récente qui vise à améliorer la précision du processus de regroupement et la qualité sémantique des résultats obtenus. A notre connaissance peu de travaux dans la littérature utilisent cette approche.

3.1.2.1 Travaux de [Doucet et A-Myka, 2002]

Doucet et Ahonen-Myka représentent un document XML en utilisant le modèle vectoriel dans lequel les modalités sont soit les balises XML, soit les mots du texte (contenu), soit une combinaison des deux (balises et texte). La technique *Tf-Idf* est utilisée pour normaliser les éléments du vecteur et l'algorithme K-means avec la similarité cosinus sont appliqués.

Le clustering a été évalué sur 12232 documents de la collection INEX, qui ont été divisés en 19 classes. Ils ont exécuté l'algorithme *k-means* avec $k \in \{5, 10, 15, 20, 25, 35\}$ pour le texte seul, les balises seules et texte & balises. Ils ont ensuite calculé l'entropie et la pureté.

Les résultats trouvés montrent que pour un nombre k quelconque de clusters, la meilleure qualité est obtenue avec les caractéristiques du texte. Pour le cas des balises uniquement la qualité est moindre mais quand ils l'ont combinés aux caractéristiques du texte, la détérioration est juste moyenne. De manière générale, les résultats sont loin d'être concluants.

3.1.2.2 Travaux de [Vercoustre et al., 2006b]

Cet article propose un modèle générique pour le clustering de documents XML basé sur la structure seul ou la structure et le contenu. Ce modèle consiste à représenter un document XML par l'ensemble de chemins (ou sous chemins) de l'arbre XML de longueur comprise entre n et m, deux valeurs données a priori, et qui sont définis en fonction de certains critères (longueur, commençant par la racine, ...). La motivation des auteurs de prendre en compte les sous-chemins, et pas seulement les chemins, est que la ressemblance entre documents peut apparaître à différents niveaux. Ces chemins sont ensuite considérés comme de simples mots sur lesquels on peut les mettre dans un modèle d'espace vectoriel après une pondération par *tf-idf*.

Lorsqu'ils considèrent le contenu et la structure du document, les chemins textuels (ou sous chemins textuels) sont utilisés. Un chemin textuel est un chemin prolongé par un mot du contenu textuel associé à un nœud. Lorsque le chemin est terminal, on considère le contenu textuel de la feuille. Si le chemin n'est pas terminal, on considère l'ensemble du contenu des feuilles qui sont des descendants du nœud considéré.

Cette approche utilise un algorithme de clustering par partitionnement assez proche de l'algorithme k-means appelé SClust.

Dans de cette approche, les chemins sont individuellement considérés comme des mots. Ce choix pose le problème d'une éventuelle dépendance entre eux. Par exemple *bdy.sec* et *bdy.sec.st* ne sont pas indépendants puisque le second peut exister seulement si le premier existe, le premier chemin est intégré dans le second. Pour traiter cette dépendance, les auteurs divisent les chemins en utilisant une variable par longueur et traitent chaque ensemble de chemins comme des variables différentes. La similarité donc est mesurée par :

$$d(x,y) = \sqrt{\sum_{k=1}^{p}\sum_{j=1}^{m_k}(x_j^k - y_j^k)^2} \tag{3.12}$$

où p est le nombre de variables, et m_k est le nombre de modalités pour le variable k.

L'entropie, pureté, F-mesures et l'indice de rand corrigé sont utilisés pour évaluer cette approche qui a été réalisée sur deux collections d'INEX : la collection *d'IEEE* qui est divisé en *INEX-s* (structure uniquement), et *INEX-cs* (structure et contenu) et la collection *Movie DB* (9643 documents XML, 11 catégories). Les résultats obtenus montrent que cette approche a réussi dans le cas de la structure seule. Mais dans le cas de la structure et le contenu, ils n'ont pas réussi à regrouper la totalité des collections, en raison de la grande taille du vocabulaire généré.

3.1.2.3 Travaux de [Tran et al., 2008]

Dans ce travail, les auteurs proposent de représenter le contenu et la structure du document XML à l'aide du modèle d'espace vectoriel. La similarité est donc calculée par la combinaison de la valeur de similarité de contenu et la valeur de similarité de structure. Soit deux documents d_x et d_y, la similarité entre ces deux documents est défini comme suit :

$$docSim(d_x,d_y) = (contSim(d_x,d_y) \times \lambda) + (structSim(d_x,d_y) \times (1-\lambda)). \tag{3.13}$$

Le λ, allant de 0 à 1, est défini par l'utilisateur pour ajuster l'importance de la similarité de contenu (contSim) et la similarité de structure (structSim).

- **Similarité de structure (*structSim*)**
  La structure d'un document XML est représentée par l'ensemble de chemins de l'arbre XML. Etant donné un ensemble des documents XML $\{d_1, d_2, \cdots, d_n\}$, désigné par D, un ensemble de chemins distincts $\{p_1, p_2, \cdots, p_f\}$, notée P, sont extraites de D. La structure d'un document, d_i, est modélisé par un vecteur $\{p_{i,1}, p_{i,2}, \cdots, p_{i,f}\}$, où chaque élément du vecteur représente la fréquence d'un chemin dans P qui apparaît dans le document. La similarité (*structSim*) entre deux documents XML, d_x et d_y, est calculée par l'utilisation de la distance Euclidienne.

- **Similarité de contenu (*contSim*)**
  Le contenu d'un document, d_i, est modélisé comme un vecteur $\{t_{i,1}, t_{i,2}, \cdots, t_{i,m}\}$, où

chaque élément du vecteur représente la fréquence d'un terme dans T (l'ensemble de termes) qui apparaît dans le document (les termes sont choisis après l'élimination des stop-word et lemmatisation). Le *noyau sémantique latent (latent semantic Kernel)* [Cristianini et al.2002] est utilisé pour mesurer les associations sémantiques du contenu textuel des documents XML.

L'évaluation a été réalisée sur trois collection d'INEX 2006 : *IEEE* (3000 documents, 60 catégories), *Wikipédia* (6054 documents, 18 catégories) et une collection hétérogène (3900 documents, 78 catégories) composée d'un mélange des documents Wikipedia et IEEE. Mais ils ont utilisé seulement de 1000 à 1300 documents de chaque collection.

La micro-moyenne et la macro-moyenne de F_1 sont utilisées pour cette évaluation. L'expérience montre que la performance de l'approche proposée est meilleure lorsque le jeu de données est dans un environnement hétérogène, plutôt que dans un environnement homogène. Cela est dû à la combinaison à la fois de la similarité de la structure et de contenu.

3.1.2.4 Travaux de [Kim, 2009]

Dans cet article, une nouvelle représentation appelée chemin virtuel qui peut représenter la structure et le contenu d'un document XML est proposée. Un chemin virtuel est un chemin composé par des nœuds importants où chaque nœud représente un niveau de l'arbre XML à l'exception du dernier niveau. Un nœud important est un nœud qui a le plus grand nombre d'enfants parmi les nœuds de son niveau. Si plus d'un nœud satisfont la condition dans le même niveau, alors un nœud quelconque est sélectionné. La complexité en temps pour extraire un chemin virtuel est $O(2n)$, où n est le nombre des nœuds dans l'arbre XML.

Pour mesurer la similarité entre deux chemins, l'auteur propose d'utiliser le nom du nœud, et sa position dans le chemin. La similarité entre les noms des deux nœuds est obtenue par l'ensemble des synonymes. Le système donc attribue une valeur de 0 à 1 au poids de nom selon le niveau de conformité des deux noms d'après les synonymes qui sont extraits à partir d'une base de données des synonymes comme *WordNet*. Cependant, même si les noms des nœuds comparés sont les mêmes, le poids peut être différent selon les positions des nœuds dans les chemins. Plus le chemin vers le nœud devient court, plus le poids attribué à ce nœud est grand. Soit le poids de niveau d'un nœud x_i, $(1 \leq i \leq n)$ dans un chemin représentant est Lev_{x_i}, les poids des niveaux satisfont les conditions suivantes :

$$Lev_{x_1} \geq Lev_{x_2} \geq ... \geq Lev_{x_n} \text{ et } \sum_{k=1}^{n} Lev_{x_i} = 1.$$

La similarité entre deux documents X et Y est défini par la formule suivante :

$$Sim(X,Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} Name(x_i, y_j) \times min(Lev_{x_i}, Lev_{y_j}),$$

où : $Name(x_i, y_j)$ est le poids de nom des deux nœuds x_i et y_j.
La raison de choisir la valeur minimale des deux poids de niveaux par l'auteur est pour réduire l'influence de poids quand la différence entre les niveaux des nœuds comparés est grand.

Un clustering hiérarchique ascendante est appliquée avec les deux algorithmes : *Single-link* et *Complete-link*.

Dans le but de trouver un critère d'arrêt de regroupement quand une meilleure hiérarchie des clusters est réalisée, l'auteur définit une fonction $h(C)$ qui permet de mesurer la dissimilarité entre les documents de même groupe : $h(C) = \min_{X,Y \in C} Sim(X, Y)$, où $0 \leq h(C) \leq 1$. l'algorithme de clustering se termine dans le regroupement $\Re_t$ si $\exists C_j \in \Re_{t+1} : h(C_j) \leq \theta$.
Où θ est un seuil ($0 \leq \theta \leq 1$).

L'expérience réalisée sur les documents XML fournies par l'université du Wisconsin (80 documents provenant de huit DTD) montre que quand l'algorithme *Complete-link* est utilisé avec un seuil $\theta = 0.5$, une meilleure hiérarchie des clusters est atteinte.

Le choix de nœud représentant dans le cas de deux nœuds qui ont le même nombre d'enfants reste une lacune dans cette approche.

3.1.2.5 Travaux de [Kutty et al., 2009]

Ce travail propose une nouvelle approche **h**ybride pour le **c**lustering des documents **X**ML appelée **HCX**. Cette approche détermine d'abord la similarité structurelle sous la forme des sous arbres fréquents, ensuite elle utilise ces sous arbres fréquents pour représenter le contenu des documents XML afin de déterminer la similarité de contenu. Les auteurs ont utilisé les sous arbres fréquents pour réduire le contenu des documents XML au lieu d'utiliser tout le contenu.

HCX comporte deux phases : la première consiste à modéliser les documents XML par des arbres enracinés, ordonnés et étiquetés (les attributs sont pris en compte), puis de trouver les relations structurelles communes entre les documents XML sous la forme des sous arbres fréquents induits fermés (CFI).

La deuxième phase consiste dans un premier temps à représenter le contenu de chaque document sous la forme d'un vecteur c'est-à-dire représenter le contenu des nœuds des CFI qu'il couvert. Un document est donc représenté par un vecteur contenant la somme des occurrences de tous les termes qui sont obtenu après prétraitement. Où la somme des occurences d'un terme t_i dans tous les CFI d'un document est calculée par la formule suivante :

$$\xi(t_i) = \sum_{i=1}^{k} t_i(CFI_k). \tag{3.14}$$

Les vecteurs sont ensuite combinés dans une représentation matricielle $D \times T$, où D représente les documents, T représente les termes et la valeur de la cellule de la matrice représente la somme des occurrences d'un terme ξ.

Les auteurs ont suivi la méthode dichotomie répétée *(repeted bisection)* pour regrouper les documents XML. Cette méthode divise la matrice en deux groupes, puis sélectionne l'un des groupes selon un critère de regroupement. Ce processus est répété jusqu'à ce que le nombre désiré de clusters est atteint. La similarité entre deux vecteurs est calculée par la mesure de cosinus.

La micro-moyenne et la macro-moyenne de F1 ont été utilisé pour évaluer le clustering qui a été réalisé sur deux corpus réels ACM SIGMOD (140 documents) et INEX 2007 Wikipedia (48305 documents). L'évaluation montre l'efficacité de cette méthode pour les collections de documents volumineux tout en utilisant un nombre considérablement réduit de caractéristiques de contenu.

3.1.3 Discussion

Nous synthétisons dans la table 3.1 et la table 3.2 les différentes caractéristiques des travaux décrits auparavant (suivant la structure seule et la structure & le contenu respectivement) selon cinq critères : le modèle de représentation, la mesure de similarité, la méthode de clustering suivi, le corpus sur lequel le clustering est appliqué et le meilleur résultat trouvé.

De cette étude nous pouvons souligner les observations suivantes :

1. Dans les travaux de clustering basés sur la structure, différents modèles de représentation de la structure des documents XML ont été utilisés dont la plupart sont basés sur l'arbre et ses dérivés (résumé d'arbre, s-graph), ou par la linéarisation de l'arbre correspondant au document XML (chemins).
2. Bien que le modèle vectoriel possède des inconvénients (par exemple considère que tous les termes sont indépendants), il est le plus utilisé dans la plus part des travaux proposés basés sur la structure et le contenu pour représenter les documents XML, en transformant ces derniers pour qu'ils s'adaptent à ce modèle afin de prendre en compte la structure des documents.
3. La plupart des modèles de représentation de la structure proposés dans ces articles prennent en considération les informations structurelles (parent/enfant ou ancêtre /descendant) entre les éléments des documents XML. De plus tous les travaux prennent en compte les attributs dans la construction de son modèle de représentation (l'attribut d'un nœud est considéré comme un nœud fils) sauf dans [Dalamagas et al., 2005] [Kim, 2010], nous ne trouvons aucune indication à l'utilisation des attributs dans les modèles proposés.
4. Certains travaux ont porté sur le regroupement de grandes collections de documents à l'aide des représentations de documents qui impliquent à la fois la structure et le contenu des documents, ou la structure seule [Vercoustre et al., 2006b].
5. Les résultats de clustering indiquent que les travaux proposés n'ont pas tous réussi à bien regrouper les documents XML surtout dans le cas des travaux basés sur la structure et le contenu à part certains travaux qui ont réussi à regrouper les documents

avec une qualité variant entre moyenne et bonne comme [Kutty et al., 2009], tandis que *Vercoustre et al.* n'ont pas réussi à regrouper la collection des documents XML.

6. Parmi les problèmes qui empêchent le bon regroupement des documents XML réside dans la taille de documents où nous constatons que les travaux récents tendent vers la réduction de la taille des documents XML utilisés.

Auteurs	Modèle de représentation	Mesure de similarité	Méthode de clustering	Corpus	Principaux résultats atteints	Observation
[Nierman et Jagadish, 2002]	Arbre ordonné et étiqueté	Distance d'édition	Hiérarchique ascendante (UPGMA)	ACMSIGMOD et données synthétiques	Mis-clustering = 0.	
[Lian et al., 2004]	S-graph	Basé sur l'intersection des arêtes	Hiérarchique (S-GRACE)	DBLP et données synthétiques	CS = 0.90, IS = 0.01, SD = 1, R = 0.01.	la mesure ne tient pas en compte les docs qui ont les mêmes éléments et différentes arêtes.
[Dalamagas et al., 2005]	Résumé d'arbre	Distance d'édition	Hiérarchique (Single link)	ACMSIGMOD + ADC /NASA (150 docs), données synthétiques (1000 docs)	Précision = 1.0 Rappel = 1.0	
[Iyer et Simovici, 2008]	Multi-ensemble des chemins	La différence symétrique (faible et forte)	Hiérarchique ascendante moyenne	Niagara (45 docs) Sigmod (48 docs) synthétique (45 docs)	Silhouette = 1.0	
[Kim, 2010]	Vecteur de bits	Et bit à bit	Hiérarchique agglomérative (DTD non disponoble)	Documents XML de l'université du Wisconsin (80 docs)	nbr de documents incorrectement regroupés NIC = 0.	le nbr des docs utilisés est petit.
[Alishahi et al., 2010]	Structure de niveau	LevelSim+	XCLS+	Univ-Wisconsin (460 docs) + Univ-Wishington (164 docs)	Entropie = 0 Pureté = 1 F-mesure = 1	

TABLE 3.1: *Un tableau récapitulatif du synthèse des traveaux de recherche basés sur la stucture.*

Auteurs	Modèle de représentation	Mesure de similarité	Méthode de clustering	Corpus	Principaux résultats atteints	Observation
[Doucet et A-Myka, 2002]	VSM	Cosinus	Par partionnement (k-means)	INEX (12232 docs)	Text & tags (k =35) : entropie = 0.612, pureté = 0.385	Ils n'ont pas réussi (qualité < moyenne)
[Vercoustre et al., 2006b]	VSM (chemins, chemins textuels)	Distance Euclidienne	Par partitionnement (Sclust)	INEX : IEEE (INEX-s et INEX-cs) et MovieDB (9643 docs)	Structure seul : entropie = 0.185, pureté = 0.963, rand corrigé = 0.423, F-mesure= 0.661	Ils sont échoués à regrouper les docs dans le cas "structure et contenu"
[Tran et al., 2008]	VSM	structsim : Euclidienne + contsim : LSK	Hiérarchique	IEEE (6054 docs), Wikipidia (3000 docs) et docs hétérogènes (3900 docs)	Micro-moy F_1 = 0.346, Macro-moy F_1 = 0.29	L'approche est meilleure lorsque le jeu de données est dans un environnement hétérogène.
[Kim, 2009]	Chemin virtuel	utilisée le nom de noeud et sa position	Hiérarchique (SL et CL)	Université du Wisconsin (80 docs)	θ = 0.5, une meilleure hiérarchie des clusters est atteinte.	Le nbr des docs utilisés est petit. Le choix de nœud représentant dans le cas de deux nœuds qui ont le même nombre d'enfants : nécessite une stratégie spécifique.
[Kutty et al., 2009]	VSM	Cosinus	Dichotomie répétée (repeted bisection)	ACM SIGMOD (140 docs), INEX 2007 Wikipedia (48305 docs)	Micro-moy F1= 0.89, Macro-moy F1= 0.64	

TABLE 3.2: *Un tableau récapitulatif du synthèse des traveaux de recherche basés sur le contenu et la stucture.*

- VSM : modèle d'espace vectoriel (Vector Space Model VSM en anglais).
- SL : l'algorithme Single-link.
- CL : l'algorithme Complete-link.
- LSK : Latent Semantic Kernel.

3.2 Conclusion

Le clustering des documents XML est un problème de recherche très important avec des applications pertinentes dans de nombreux domaines pratiques comme la recherche d'information, l'intégration de données, le Web mining et le traitement de requête XML.

Dans ce chapitre, nous avons présenté une synthèse de quelques travaux développés pour le clustering des documents XML, ces travaux sont divisés selon deux approches, la première vise à exploiter la structure du document XML, et la deuxième approche a pour objectif de bénéficier de l'information issue de la structure et le contenu du document XML. Lors de cette synthèse, nous avons trouvé que la manière de représenter les documents XML joue un rôle très important dans le bon regroupement des documents XML, ainsi que le choix de la mesure de similarité adéquate à cette représentation.

Nous avons constaté aussi que le modèle vectoriel se trouve toujours sur le trône des représentations les plus utilisés dans les approches qui utilisent la structure et le contenu. Ainsi que la réduction de la taille des documents XML et l'utilisation de la structure et le contenu fait l'objectif des travaux récents pour améliorer la qualité de clustering.

CHAPITRE 4

CONTRIBUTION ET VALIDATION

« La connaissance ne repose pas sur la vérité seule, mais sur une erreur aussi. ».

Carl Gustav

Ce chapitre présente notre contribution. D'abord nous commençons par rappeler notre motivation, ensuite nous présentons nos approches de clustering de documents XML (section 4.2) ; où dans un premier temps nous présentons l'approche sans la prise en considération des caractéristiques de contenu et dans un deuxième temps nous les prenons en compte, puis nous montrons une vue générale du corpus INEX utilisé (section 4.4). Enfin, nous présentons les résultats d'expérimentation de nos approches sur le corpus de validation (section 4.5).

4.1 Motivation

Le bon fonctionnement du processus d'extraction de connaissances à savoir dans notre cas le clustering des documents XML nécessite l'association de plusieurs facteurs comme le choix de l'algorithme, la mesure de similarité et surtout le modèle de représentation utilisé. Ce dernier joue un rôle très important dans l'efficacité globale du processus de clustering et sur la qualité de regroupement.

Mais il existe un autre facteur qui empêche le bon regroupement des documents XML qui est lié aux documents XML lui-même, nous parlons alors sur la taille de document XML car ce dernier peut atteindre des profondeurs élevées et par conséquent un nombre de nœuds exponentiel. Ce qui nécessite de réduire la taille de documents mais sans perte d'informations.

Notre approche de clustering documents XML se situe dans ce contexte. Elle est inspirée de l'approche de XCLS+ présentée dans le chapitre précédent.

Notre approche qui est basée dans un premier temps sur la prise en compte de la structure seulement, fournit deux représentations pour chaque document :

- Le résumé d'arbre [Dalamagas et al., 2005] qui est une représentation optimale de l'arbre permet de diminuer la taille de document XML en réduisant la profondeur et la largeur de l'arbre correspondant au document XML.
- La représentation structure de niveau qui consiste pour chaque niveau dans le document XML, à regrouper les éléments et leurs parents dans un vecteur.

Cette approche essaye donc d'exploiter les avantages de ces deux représentations pour pouvoir réaliser un clustering efficace.

D'autre part, nous proposons d'ajouter le contenu des documents XML à l'approche précédente dans le but d'améliorer la qualité sémantique du clustering.

Dans la suite de ce chapitre, nous proposons d'évaluer notre approche de clustering sur une collection de documents XML de corpus INEX.

4.2 Approches de clustering proposées

Nous avons proposé deux approches de clustering des documents XML. La première approche consiste à proposer une solution tenons en compte la structure et le résumé d'arbre puis l'augmenter par des informations sur le contenu dans la deuxième approche.

4.2.1 Approche basée sur la structure

Comme nous avons mentionné dans le début de ce chapitre, nous avons adopté dans notre approche de clustering la représentation combinant le résumé d'arbre afin de réduire la taille des documents XML utilisés, et la représentation structure de niveau proposée par [Alishahi et al., 2010] pour représenter les éléments structurels d'un document XML. Cette représentation permet de regrouper les nœuds de chaque niveau dans le document XML (l'arbre) ainsi que leurs parents dans un vecteur. La représentation générale est vue comme un vecteur à plusieurs niveaux, où chaque niveau contient une liste des nœuds. Le cas de plusieurs valeurs pour un élément est ignoré, il est considéré comme une information redondante. Un exemple de cette représentation est donné dans la figure 4.1.

Nous avons cherché à étudier l'effet de la représentation résumé d'arbre sur l'approche de XCLS+ et donc proposer une approche qui pallie aux inconvénients des approches existantes. L'approche de clustering proposée comporte deux phases comme le montre la figure 4.2.

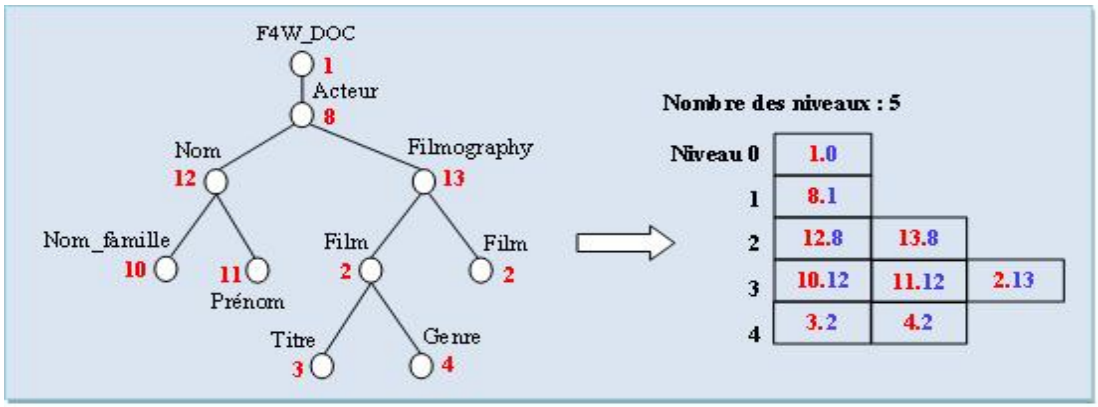


FIGURE 4.1 – Exemple d'une structure de niveau d'un document XML [Alishahi et al., 2010].

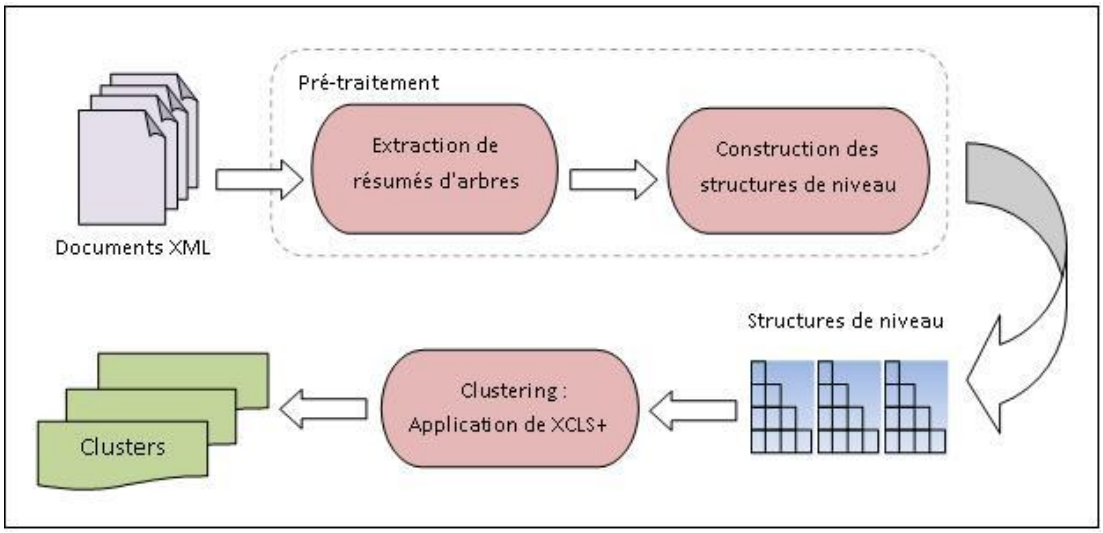


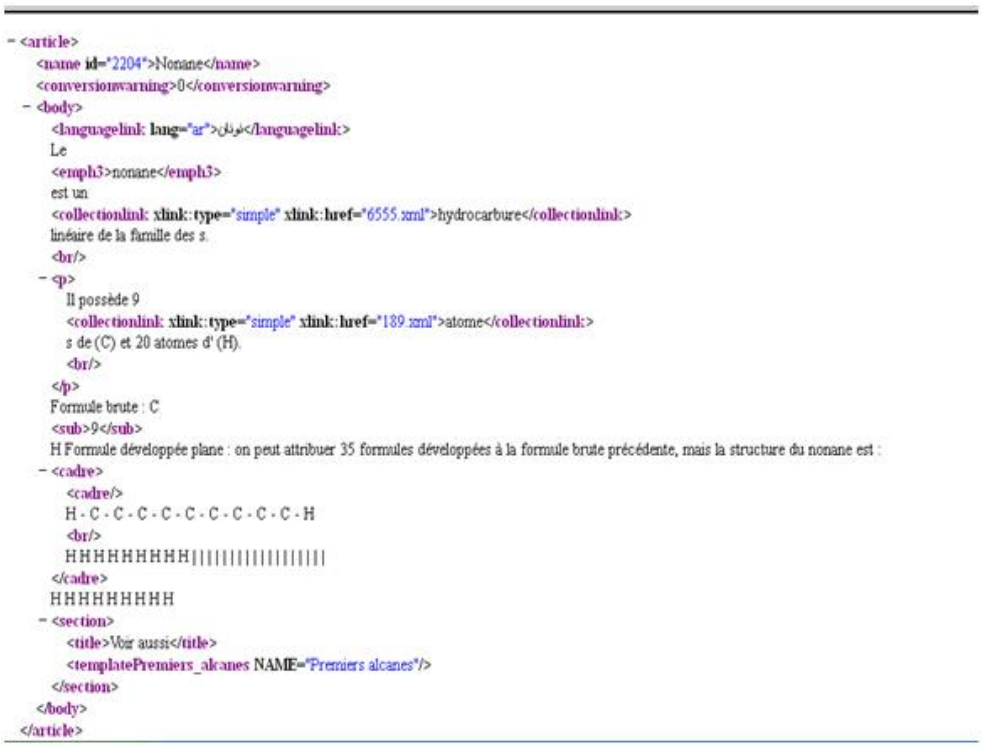
FIGURE 4.2 – L'approche de clustering basée sur la structure.

4.2.1.1 Phase I : Le pré-traitement

Le but de cette phase est d'analyser les documents XML et les transformer en structures de niveau. Dans cette étape, on commence par représenter les structures des documents XML sous forme d'arbres XML puis d'extraire leurs résumés en réduisant les nœuds imbriqués répétés (réduire la profondeur) et éliminant les nœuds répétés.Ces résumés d'arbres sont ensuite utilisés pour construire les structures de niveau. Pour l'implémentation, les documents XML sont représentés sous forme de tables où chaque ligne représente un élément (nom, niveau, parent). Ces tables sont ensuite triées par nom d'élément (tagname) afin de faciliter la recherche des éléments communs. Dans ce niveau les documents sont prêts pour passer à l'étape de regroupement. Les figures 4.3, 4.4 et 4.5 représente respectivement un exemple des prét-raitements d'un document XML.

```
- <article>
    <name id="2204">Nonane</name>
    <conversionwarning>0</conversionwarning>
  - <body>
      <languagelink lang="ar">نونان</languagelink>
      <languagelink lang="de">Nonan</languagelink>
      <languagelink lang="en">Nonane</languagelink>
      <languagelink lang="ja">ノナン</languagelink>
      <languagelink lang="sv">Nonan</languagelink>
      Le
      <emph3>nonane</emph3>
      est un
      <collectionlink xlink:type="simple" xlink:href="6555.xml"> hydrocarbure</collectionlink>
      linéaire de la famille des
      <collectionlink xlink:type="simple" xlink:href="207.xml">alcane</collectionlink>
      s.
      <br/>
    - <p>
        Il possède 9
        <collectionlink xlink:type="simple" xlink:href="189.xml">atome</collectionlink>
        s de
        <collectionlink xlink:type="simple" xlink:href="767.xml">carbone</collectionlink>
        (C) et 20 atomes d'
        <collectionlink xlink:type="simple" xlink:href="1400.xml">hydrogène</collectionlink>
        (H).
        <br/>
      </p>
      Formule brute : C
      <sub>9</sub>
      H
      <sub>20</sub>
      <br/>
      Formule développée plane : on peut attribuer 35 formules développées à la formule brute précédente, mais la structure du nonane est :
    - <cadre>
      - <cadre>
        - <cadre>
          - <cadre>
            - <cadre>
                H H H H H H H H H
                <br/>
                | | | | | | | | |
                <br/>
              </cadre>
            </cadre>
          </cadre>
        </cadre>
        H - C - C - C - C - C - C - C - C - C - H
        <br/>
      - <cadre>
        - <cadre>
          - <cadre>
            - <cadre>
                | | | | | | | | |
                <br/>
              </cadre>
            </cadre>
          </cadre>
        </cadre>
      </cadre>
      H H H H H H H H H
      <br/>
    - <section>
        <title>Voir aussi</title>
        <templatePremiers_alcanes NAME="Premiers alcanes"> </templatePremiers_alcanes>
      </section>
    </body>
</article>
```

FIGURE 4.3 – Un exemple de document XML *"2204"*.



```
- <article>
    <name id="2204">Nonane</name>
    <conversionwarning>0</conversionwarning>
  - <body>
      <languagelink lang="ar">نونان</languagelink>
      Le
      <emph3>nonane</emph3>
      est un
      <collectionlink xlink:type="simple" xlink:href="6555.xml">hydrocarbure</collectionlink>
      linéaire de la famille des s.
      <br/>
    - <p>
        Il possède 9
        <collectionlink xlink:type="simple" xlink:href="189.xml">atome</collectionlink>
        s de (C) et 20 atomes d' (H).
        <br/>
      </p>
      Formule brute : C
      <sub>9</sub>
      H Formule développée plane : on peut attribuer 35 formules développées à la formule brute précédente, mais la structure du nonane est :
    - <cadre>
        <cadre/>
        H - C - C - C - C - C - C - C - C - C - H
        <br/>
        HHHHHHHHH||||||||||||||||||
      </cadre>
      HHHHHHHHH
    - <section>
        <title>Voir aussi</title>
        <templatePremiers_alcanes NAME="Premiers alcanes"/>
      </section>
    </body>
  </article>
```

FIGURE 4.4 – Le document *"2204"* après l'application des règles de résumé d'arbre.

TagName	Level	Parenet
article	0	
body	1	article
br	2	body
br	3	p
br	3	cadre
cadre	2	body
cadre	3	cadre
collectionlink	2	body
collectionlink	3	p
conversionwarning	1	article
emph3	2	body
languagelink	2	body
name	1	article
p	2	body
section	2	body
sub	2	body
templatePremiers_alcanes	3	section
title	3	section

FIGURE 4.5 – Le document *"2204"* sous forme de tableau.

4.2.1.2 Phase II : Le clustering

Afin de regrouper les documents XML, nous avons utilisé l'algorithme XCLS+ [Alishahi et al., 2010]. Ce dernier est une amélioration de l'algorithme XCLS [Nayak et Xu, 2006]. XCLS+ et XCLS ont la même procédure de regroupement mais la différence réside dans la manière de rechercher des éléments communs et la mesure de similarité. Nous rappelons que la formule de similarité et la méthode d'appariement des éléments pour l'algorithme XCLS+ sont présentées dans le chapitre précédent (3.1.1.6).

La procédure générale de regroupement est :
- Le premier document introduit est celui qui forme le premier cluster.
- Le document suivant est rajouté et la similarité est calculée entre ces documents.
- Si la similarité calculée est supérieure au seuil donné par l'utilisateur, le document est placé dans le cluster existant et un représentant du cluster est formé en fusionnant les structures des documents. Cette fusion consiste à rassembler les éléments des mêmes niveaux dans une nouvelle structure par niveau appelée structure par niveau de cluster.
- Si la similarité calculée est inférieure au seuil, le document entré forme un nouveau cluster.

Cette procédure est répétée pour chaque nouveau document entré. Il est comparé avec les clusters existants si la similarité maximale calculée est plus grande que le seuil donné, alors le document est placé dans le cluster le plus proche, sinon il forme un nouveau cluster.

Outre, la spécification du seuil par l'utilisateur, cet algorithme possède aussi un autre inconvénient qui est la séquence d'injection des documents qui peut engendrer des résultats différents. Pour éviter ce problème, une deuxième phase appelée "réaffectation". Dans cette phase, certains de ces documents sont choisis aléatoirement et sont affectés aux clusters, encore une fois. Cette tâche est répétée jusqu'à ce qu'il n'y ait pas d'amélioration dans le placement des documents dans deux itérations consécutives. L'algorithme 2 décrit les différentes étapes de XCLS+.

Selon le principe des algorithmes incrémentaux nous savons qu'après que le cluster de chaque document a été reconnu, nous devons combiner la représentation du document et son cluster correspondant. Cela permettra de mettre à jour le représentant du cluster utilisé pour le regroupement de nouveaux cas.

Dans XCLS+, la structure de niveau d'un cluster est l'union des éléments de deux objets pour chaque niveau. Rappelons que dans chaque case de la structure de niveau, il y a deux éléments d'information le nom de l'élément et le nom de son parent. Dans le cas de deux éléments similaires avec des parents différents, les deux cases doivent être incluses dans la structure de niveau combiné. La figure 4.8 représente une combinaison de deux structures de niveau dans l'algorithme XCLS+ où nous constatons qu'au niveau 2 des deux structures de niveau, l'élément 14 est apparu avec des parents différents donc dans la structure au niveau du cluster, l'élément 14 est utilisé deux fois, chaque fois avec un autre parent.

Algorithme 2 : L'algorithme XCLS+

Entrées : Documents XML, LevelSim_Threshold `// seuil`
Sorties : Ensemble de clusters

début
  `// Phase 1: Allocation`
  pour *tous les documents XML à regrouper* **faire**
    Lire le document (représenté par la structure de niveau);
    Calculer le *levelSim* entre le document et chaque cluster existant;
    Assigner le document à un cluster existant si un maximum de $LevelSim(s)$ se trouve entre deux objets et $LevelSim \succ LevelSim\_Threshold$;
    Autrement, former un nouveau cluster contenant le document;
  fin
  `// Phase 2: réaffectation`
  pour *tous les documents XML* **faire**
    Lire le document sélectionné aleatoirement;
    Calculer le *levelSim* entre le document et chaque cluster existant;
    Assigner le document à un cluster existant si un maximum de $LevelSim(s)$ se trouve entre deux objets et $LevelSim \succ LevelSim\_Threshold$;
    Autrement, former un nouveau cluster contenant le document;
  fin
  `// Stop si aucune amélioration entre deux itérations.`
fin

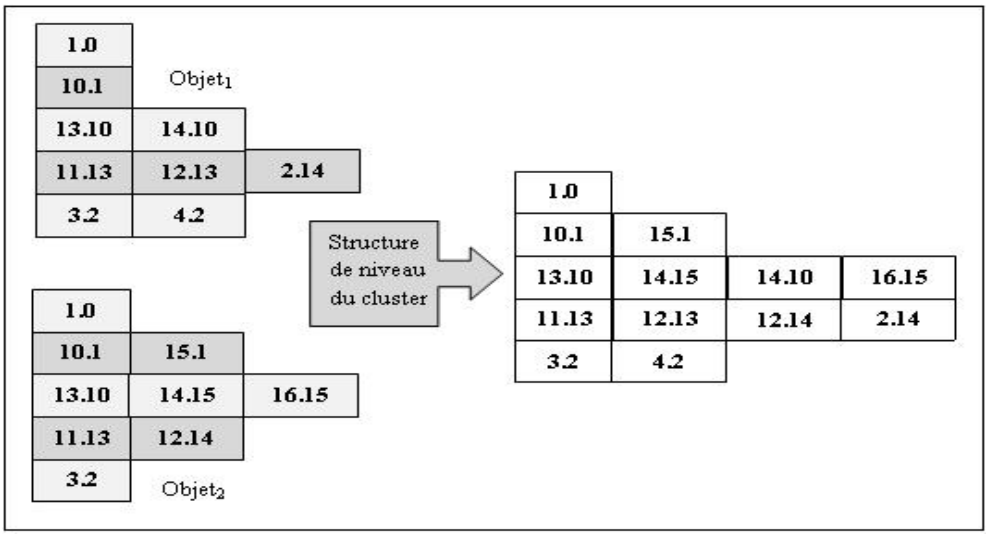


FIGURE 4.6 – Exemple d'une structure de niveau d'un cluster en XCLS+.

4.2.2 Approche basée sur la structure et le contenu

Dans le but d'améliorer la qualité sémantique de clustering, nous avons essayé d'ajouter le contenu des balises (les éléments textuels) aux propriétés structurelles dans l'opération de regroupement. Dans ce cas, un document XML peut être vu comme un tableau où chaque ligne représente un élément (nom, niveau, parent, texte).

Une vue globale de notre approche est illustrée par la figure 4.7. Dans cette approche, nous traitons la structure d'une façon similaire à la première approche. Nous décrivons dans cette approche l'ajout du contenu dans le processus de regroupement et les modifications faites par cet ajout. Il s'agit donc d'exprimer comment est faites l'extraction du contenu et la représentation utilisée, la nouvelle formule de similarité proposée et le représentant de cluster.

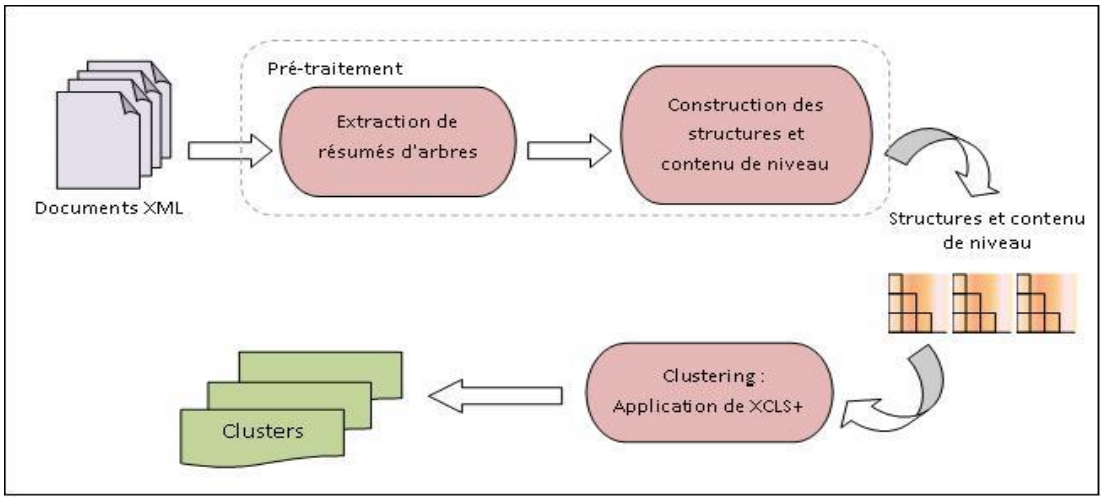


FIGURE 4.7 – L'approche de clustering basée sur la structure et le contenu.

4.2.2.1 Représentation structure et contenu de niveau

Afin de représenter la structure et le contenu des documents XML, nous avons gardé la même idée de la représentation structure de niveau avec l'ajout de contenu texte des éléments. Nous avons appelé cette représentation *"structure et contenu de niveau"*. Cette représentation permet de regrouper les nœuds de chaque niveau dans le document XML (l'arbre) dans un vecteur où chaque nœud est représenté par son nom (tagname), son parent et son contenu textuel (un ensemble de mots).

Le contenu des documents XML passent par cette représentation après une phase de nettoyage où nous avons effectué quelques opérations de préparation des textes :

- La suppression des ponctuations.
- La suppression des nombres.
- La suppression des mots vides (ou stop words, en anglais) qui sont des mots peu informatifs, généralement courts, communs qui n'ont pas de signification du point de vue du clustering, comme "la", "le", "de", "à", etc.

4.2.2.2 Similarité

Dû à l'information de contenu que nous avons ajouté, il est nécessaire de prendre en compte cette information dans la formule de similarité. La similarité en termes de contenu, que nous avons adopté, entre deux objets est calculée sur la base des mots communs qu'ils partagent.

Pour trouver les mots communs entre deux objets (un objet peut être soit un document ou un cluster), nous essayons d'ordonner les mots par ordre alphabétique et selon l'algorithme 3, nous pouvons trouver tous les mots communs. Nous notons que nous enregistrons les mots de chaque document ainsi leurs niveaux dans un tableau afin de faciliter la recherche des mots communs.

Algorithme 3 : Les étapes de l'appariement de contenu

Entrées :
$O1$, $O2$: deux nouvelles contenu de niveau représentées comme un tableau ordonné par Mot, niveau;
$OL1$: Le nombre de lignes de $O1$ dans $\{OL1_1, OL1_2, ..., OL1_n\}$;
$OL2$: Le nombre de lignes de $O2$ dans $\{OL2_1, OL2_2, ..., OL2_m\}$;
Sorties :
$CT1\,[1..n]$ nombre;
$CT2\,[1..m]$ nombre;

```
début
    répéter
        Comparer chaque ligne i de O1 à chaque ligne j de O2;
        si O1.Mot = O2.Mot alors
            CT1[OL1_i] + +;
            CT2[OL2_j] + +;
            Aller aux lignes suivantes des deux objets O1 et O2;
        fin
        si O1.Mot ≻ O2.Mot alors
            Aller à la ligne suivante de O2 seulement;
        sinon
            Aller à la ligne suivante de O1 seulement;
        fin
    jusqu'à toutes les lignes des deux objets sont vérifiées;
fin
```

La description de l'algorithme est la suivante : nous ordonnons les tableaux par nom, ensuite nous essayons de trouver les mots communs. Pour chaque paire de mots appariés trouvée, et en utilisant l'attribut de niveau des mots, les paramètres CT_1^i et CT_2^j sont calculés. CT_1^i et CT_2^j sont le nombre des mots communs dans le niveau i dans l'objet 1 et le niveau j dans l'objet 2 respectivement.

La prise en compte du contenu nous a poussé à changer la formule du similarité de niveau ($LevelSim+$) présentée par l'équation 4.1

$$LevelSim+_{1,2} = \frac{0.5 \times \sum_{i=0}^{L-1} \left(CN_1^i + CP_1^i\right) \times (r)^{L-i-1} + 0.5 \times \sum_{j=0}^{L-1} \left(CN_2^j + CP_2^j\right) \times (r)^{L-j-1}}{\left(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}\right) + 0.5 \times \left(\sum_{i=0}^{L-1} CP_1^i \times r^{L-i-1} + \sum_{j=0}^{L-1} CP_2^j \times r^{L-j-1}\right)} \tag{4.1}$$

par la formule de similarité de niveau $LevelSimSC_{1,2}$ présentée par l'équation 4.2.

$$LevelSimSC_{1,2} = \frac{0.5 \times \sum_{i=0}^{L-1} \left(CN_1^i + CP_1^i + CT_1^i\right) \times (r)^{L-i-1} + 0.5 \times \sum_{j=0}^{L-1} \left(CN_2^j + CP_2^j + CT_2^j\right) \times (r)^{L-j-1}}{\left(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}\right) + \left(\sum_{k=0}^{L-1} T^k \times (r)^{L-k-1}\right) + 0.5 \times \left(\sum_{i=0}^{L-1} CP_1^i \times r^{L-i-1} + \sum_{j=0}^{L-1} CP_2^j \times r^{L-j-1}\right)} \quad (4.2)$$

La plupart des termes de la formule 4.2 sont exactement les mêmes utilisés dans la formule 4.1. Cependant, il ya quelques nouveaux termes qui sont décrits dans ce qui suit :

- CT_1^i et CT_2^j : le nombre des mots communs dans le niveau i dans l'objet 1 et le niveau j dans l'objet 2 respectivement.
- T^k : Nombre de mots dans le niveau k de l'arbre.

4.2.2.3 Représentant de cluster

Dans notre cas, la représentation structure de niveau a été augmentée par l'ajout de texte donc chaque case de la structure contient trois éléments d'information le nom de l'élément le nom de son parent et le texte qu'il contient. Le représentant de cluster dans ce cas a le même principe que celui présenté précédemment sauf que dans le cas de deux éléments similaires avec des parents similaires, nous gardons l'un des deux éléments mais avec un nouveau texte qui est l'union de texte des deux éléments afin d'éviter la perte d'informations. Un exemple d'un représentant de cluster est donné par la figure 4.8.

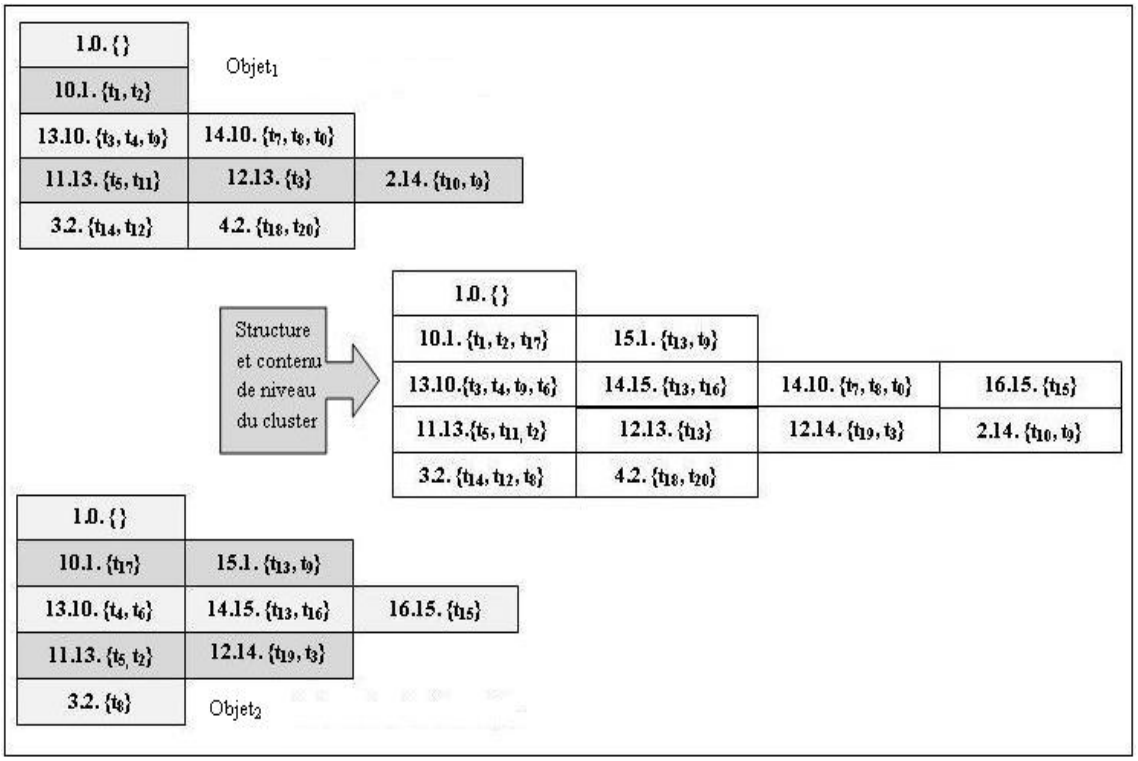


FIGURE 4.8 – Exemple d'une représentation structure et contenu de niveau d'un cluster.

4.3 Implémentation

Afin de valider nos approches, nous avons effectué une expérimentale donc le but principale est de mesurer :

- l'effet de l'utilisation d'une représentation résumé d'arbre par rapport à une représentation arbre.
- l'effet de la prise en compte du contenu sur la qualité obtenu.

Avant la description de ces résultats, nous commençons par décrire l'environnement et outils de développement puis la description du corpus test.

Outil traitant XML

Différents types d'analyseurs syntaxiques (parser) existent. Les plus importants sont : SAX et DOM. **SAX** [1] *(Simple API for XML)* et **DOM** [2] *(Document Object Model)* sont des APIs [3] qui offrent un accès aux données du document XML. Ils permettent la navigation dans le document XML et de transmettre à l'application les informations utiles au traitement du document. SAX est basé sur un modèle événementiel, il permet de parcourir linéairement le document XML et de déclencher des événements à chaque rencontre d'une balise, attribut ou texte spécifique.

DOM permet de charger un document XML en mémoire sous forme d'une hiérarchie d'objets. A l'opposé de SAX, DOM est un consommateur de ressources, et à mesure que le document augmente de taille, cette consommation augmente aussi et parfois DOM devient insuffisant. De plus, cela rend l'utilisation de DOM lente.

Afin d'analyser les documents XML, nous avons choisi le parseur JDOM. JDOM [4] est une API pure Java pour l'analyse, la création, la manipulation et la sérialisation des documents XML. JDOM a été inventé par Brett McLaughlin et Jason Hunter en printemps 2000 [Rusty, 2003]. JDOM, il utilise des collections SAX pour analyser les documents XML. Il s'inspire largement de DOM pour manipuler les éléments d'un Model Object Document spécifique crée grâce à un constructeur basé sur SAX. On peut donc construire des documents, naviguer dans leur structure, ajouter, modifier, ou supprimer soit des éléments soit du contenu. Contrairement à SAX, JDOM peut accéder à n'importe quelle partie de l'arbre à tout moment et contrairement à DOM, tous les différents types de nœuds de l'arbre sont représentés par des classes concrètes plutôt que des interfaces.

Notons que nous avons implémenté l'algorithme XCLS+, la structure de résumé d'arbre ainsi que les critères d'évaluation en langage Java en utilisant la plateforme Eclipse-SDK version 3.7.1. L'implémentation et toutes les expériences sont faites sur une machine Intel Pentium CPU 2.16 GHZ avec 2 Go de RAM.

1. http ://www.saxproject.org/
2. http ://www.w3.org/DOM/
3. API : Application Programming Interface
4. http ://www.jdom.org/

4.4 Description du corpus

Le corpus INEX 2006 a été rassemblé dans le cadre d'une campagne d'évaluation des moteurs de recherche XML crée en 2002 est appelée *INEX* (*Initiative for the Evaluation of XML Retrieval*). Les documents XML de ce corpus sont extraits à partir du corpus Wikipédia XML [Denoyer et Gallinari, 2007]. Il est utilisé dans les deux tâches de classification et clustering.

Le corpus est composé de huit principales collections correspondant à huit langues différentes : Anglais, Français, Allemand, Néerlandais, Espagnol, Chinois, Arabe et Japonais. Chaque collection est un ensemble de documents XML créés à l'aide des auteurs de Wikipedia et encodé en UTF-8. Le tabeau 4.1 donne une description de chaque collection. Les collections sont téléchargeables sur le site Web : *http ://www-connex.lip6.fr/~denoyer/wikipediaXML/*

Nom de collection	Langue	Nombre de documents	Taille de collection(MB)
main-english	Anglais	659388	4600
20060130_french	Français	110838	730
20060123_german	Allemand	305099	2079
20060227_dutch	Néerlandais	125004	607
20060130_spanish	Espagnol	79236	504
20060303_chinese	Chinois	56661	360
20060326_arabian	Arabe	11637	53
20060303_japanese	Japonais	187492	1425

TABLE 4.1 – *Caractéristiques du corpus INEX.*

Chaque collection contient un ensemble de documents XML où chaque nom de fichier (document) est un nombre correspondant à un ID de fichier (par exemple : 1943.xml). Chaque fichier correspond à un article Wikipédia. Lors de l'étape de construction du corpus de base Wikipedia XML c'est-à-dire 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, 2007] :

- **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 : == Mainpart == est transformé en <title> Mainpart </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>).

Des statistiques sur la structure des documents des principales collections sont données dans la table 4.2.

Langue	Moy. du taille de doc (octets)	Moy. du profondeur de doc	Moy. du nbr de nœuds/doc
Anglais	7261	6.72	161.35
Français	6902	7.07	175.54
Allemand	7106	6.76	151.99
Néerlandais	5092	6.41	122.8
Espagnol	6669	6.65	165.74
Chinois	6664	6.91	179.23
Arabe	4826	5.85	182.1
Japonais	7973	7.1	94.96

TABLE 4.2 – *Les statistiques sur la structure de documents des principales collections dans le corpus INEX.*

Les documents XML du corpus sont organisés en une hiérarchie de catégories définies par les auteurs des articles. La table 4.3 représente des statistiques sur les catégories des principales collections.

Langue	Nombres de catégories	Moyenne du nombre de catégories/document
Anglais	113483	2.2849
Français	28600	1.9570
Allemand	27981	2.5840
Néerlandais	13847	1.6628
Espagnol	12462	1.6180
Chinois	27147	2.0797
Japonais	26730	2.0039

TABLE 4.3 – *Les statistiques sur les catégories des principales collections dans le corpus INEX.*

Concernant nos expérimentations, nous avons choisi d'effectuer nos test sur un échantillon de 500 documents provenus de 16 catégories comme le montre la table 4.4 avec un nombre d'éléments variant entre 19 et 1384 éléments et un niveau d'imbrication variant entre 4 et 47 niveaux. Ces catégories appartiennent à la collection "langue française".

Classe	nombre de documents
Elément chimique	51
Constellation	50
Langage de programmation	26
Mathématique	33
Organisation internationale	23
Arbre fruitier	31
Solfège	21
Protocole réseau	21
Liste de jeux vidéo	27
Fromage français	70
Mammifère	46
Alphabet latin	25
Logiciel libre	36
Coupe de monde de football	16
Festival d'Angoulême	13
Langue slave	11

TABLE 4.4 – *Les différentes catégories de l'ensemble des 500 documents XML utilisé avec 500 docs.*

4.5 Résultats et discussion

Dans cette section seront présentés les résultats des différentes expériences menées afin d'évaluer les performances des approches proposées. Dans un premier temps, il s'agira de présenter l'ensemble des expériences sur l'approche proposée basée sur la structure (sous-section 4.5.1). Enfin, les expériences sur l'approche proposée basée sur la structure et le contenu et les résultats obtenus (sous-section 4.5.2) seront détaillées. Les mesures choisies pour évaluer la qualité du clustering obtenues sont l'entropie, la pureté et la F-mesure.

4.5.1 Les expériences sur l'approche proposée basée sur la structure

Pour évaluer l'intérêt d'utiliser le résumé d'arbre dans l'approche de clustering proposée, nous tentons de comparer l'algorithme XCLS+ avec et sans le résumé d'arbre pour les aspects de la similarité, la qualité de regroupement et le temps d'exécution.

4.5.1.1 Similarité

Nous avons effectué deux expériences pour évaluer la similarité en utilisant la structure de résumé d'arbre et les résultats obtenus sont montrés dans les deux tableaux 4.5 et 4.6.

Les tables 4.5 et 4.6 reportent des échantillons de calcul de similarité. Pour chaque paire de documents, nous avons calculé deux valeurs de similarité : en utilisant le résumé d'arbre et sans le résumé, et l'écart a été calculé entre ces deux valeurs de similarité.

Dans la table 4.5, nous avons prémédité le choix de cet échantillon qui repose sur les cas les plus défavorables, c'est-à-dire là où l'un des deux documents contient plusieurs niveaux imbriqués du même nœud. Le $Document_1$ dans la table vérifie cette caractéristique où nous avons donné son niveau avant et après l'utilisation du résumé d'arbre (niv_s et niv_r respectivement). Le deuxième document n'a aucun nœud imbriqué répété. Par contre dans la table 4.6, les deux documents contiennent des nœuds imbriqués répétés et donc la réduction de l'arbre touche les deux documents.

Les résultats trouvés dans les deux expériences montrent que les valeurs de similarité obtenues dans les deux cas (sans et avec le résumé) sont très proches, et parfois l'écart entre les deux valeurs de similarité atteint la valeur zéro. Nous pouvons dire que les résultats obtenus sont des résultats satisfaisants et les figures 4.9 et 4.10 prouvent notre proposition.

Paire de documents				Similarité		Ecart
$Document_1$			$Document_2$	Avec résumé	Sans résumé	
niv_s	niv_r	id	id			
8	7	28	478	0.917	0.911	0.006
8	6	43	58	0.928	0.921	0.007
9	5	2272	65447	0.903	0.904	0.001
10	7	1018	1015	0.908	0.908	0
10	6	273	274	0.839	0.848	0.009
10	3	77463	2511	0.893	0.981	0.088
10	4	144266	76762	0.961	0.961	0
11	8	4022	4004	0.870	0.869	0.001
11	7	4004	4002	0.943	0.943	0
12	7	2508	2906	0.738	0.741	0.003
12	6	628	766	0.887	0.882	0.005
12	6	691	105588	0.732	0.745	0.013
12	6	90022	101920	0.707	0.707	0
12	6	1801	1926	0.788	0.805	0.017
13	6	77724	79738	0.926	0.933	0.007
13	4	77118	187045	0.971	0.961	0.01
13	8	2551	2515	0.8	0.782	0.018
14	6	76591	472114	0.930	0.930	0
14	6	74747	77801	0.708	0.742	0.034
14	7	11367	458705	0.960	0.948	0.012
14	7	74893	475943	0.965	0.941	0.024
15	8	2050	100481	0.923	0.925	0.002
15	8	2357	101115	0.940	0.930	0.01
15	9	105726	19165	0.914	0.906	0.008
16	9	76357	170138	0.888	0.897	0.009
17	7	2632	101146	0.802	0.789	0.013
17	6	74920	153006	0.784	0.794	0.01
18	8	511693	76788	0.907	0.905	0.002
18	7	47590	64761	0.772	0.792	0.02
18	6	77375	100527	0.902	0.903	0.001
18	4	150185	179006	0.917	0.902	0.015
19	5	870	9082	0.964	0.959	0.005
20	6	107507	40910	0.823	0.841	0.018
22	9	76309	77407	0.889	0.914	0.025
23	6	14083	105108	0.745	0.747	0.002
23	13	76371	186607	0.751	0.767	0.016
25	6	1890	498377	0.834	0.833	0.001
25	7	10382	188481	0.853	0.821	0.032
28	7	133381	2626	0.815	0.815	0
29	6	82063	498370	0.873	0.882	0.009
30	6	81840	97068	0.948	0.944	0.004
33	6	1035	300	0.695	0.699	0.004
34	7	1660	883	0.885	0.884	0.001
35	6	1943	7619	0.877	0.890	0.013
39	6	64419	7149	0.929	0.929	0
39	17	101130	170134	0.932	0.924	0.008
41	7	164351	531610	0.831	0.831	0
43	7	80553	85144	0.885	0.878	0.007
47	7	7445	35967	0.826	0.826	0
48	6	6590	79330	0.860	0.882	0.022

TABLE 4.5 – *Résultats de la similarité entre documents XML avec et sans le résumé où $document_1$ contenant des nœuds imbriqués répétés.*

Paire de documents		Similarité		Ecart
$Document_1$	$Document_2$	Avec résumé	Sans résumé	
id	id			
400	399	0.938	0.937	0.001
628	623	0.915	0.9	0.015
691	699	0.966	0.965	0.001
699	698	0.971	0.969	0.002
870	878	0.967	0.966	0.001
64419	272	0.801	0.785	0.016
2210	8713	0.952	0.959	0.007
1445	1660	0.868	0.867	0.001
76565	39057	0.925	0.929	0.004
40909	90021	0.719	0.744	0.025
74747	74920	0.854	0.853	0.001
90022	81429	0.812	0.807	0.005
511693	100001	0.875	0.908	0.033
75284	76371	0.955	0.957	0.002
2605	2050	0.883	0.950	0.067
76591	1466	0.897	0.901	0.004
2283	47590	0.853	0.875	0.022
2127	11367	0.975	0.976	0.001
177563	76783	0.965	0.976	0.011
107573	1890	0.887	0.886	0.001
81840	105726	0.972	0.993	0.021
2551	2538	0.728	0.745	0.017
2508	105155	0.751	0.784	0.033
105119	76357	0.906	0.904	0.002
2032	77118	0.942	0.932	0.01
133381	180051	0.877	0.859	0.018
150027	81835	0.887	0.873	0.014
82063	107507	0.877	0.881	0.004
2357	542	0.9	0.921	0.021
79865	2316	0.932	0.922	0.01
77375	5291	0.916	0.909	0.007
1943	5288	0.832	0.832	0
2593	1723	0.805	0.795	0.01
5287	112046	0.959	0.959	0
77724	13966	0.934	0.952	0.018
164351	76309	0.940	0.938	0.002
8721	5290	0.860	0.878	0.018
101130	74893	0.985	0.995	0.01
1035	7445	0.880	0.899	0.019
150130	179836	0.951	0.950	0.001
2014	14086	0.857	0.886	0.029
14077	2091	0.924	0.915	0.009
76781	78088	0.983	0.972	0.011
34655	7576	0.861	0.862	0.001
3073	19834	0.990	0.979	0.011
144266	29365	0.957	0.964	0.007
179011	40881	0.965	0.946	0.019
165666	82426	0.900	0.907	0.007
77463	177615	0.914	0.941	0.027
6590	154512	0.924	0.919	0.005

TABLE 4.6 – *Résultats de la similarité entre documents XML avec et sans le résumé -les deux documents contient des nœuds imbriqués répétés-*

.

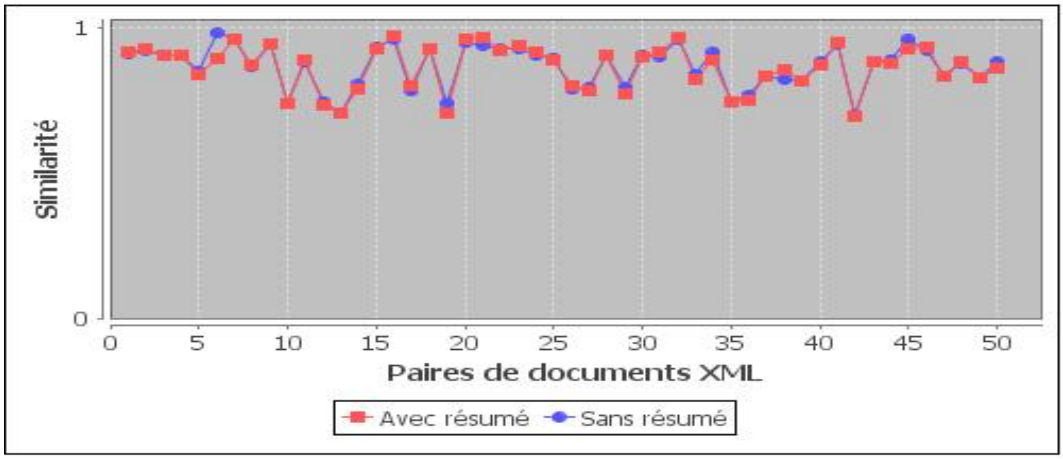


FIGURE 4.9 – Comparaison de similarité trouvée sans et avec le résumé correspondant aux données de tableau 4.5.

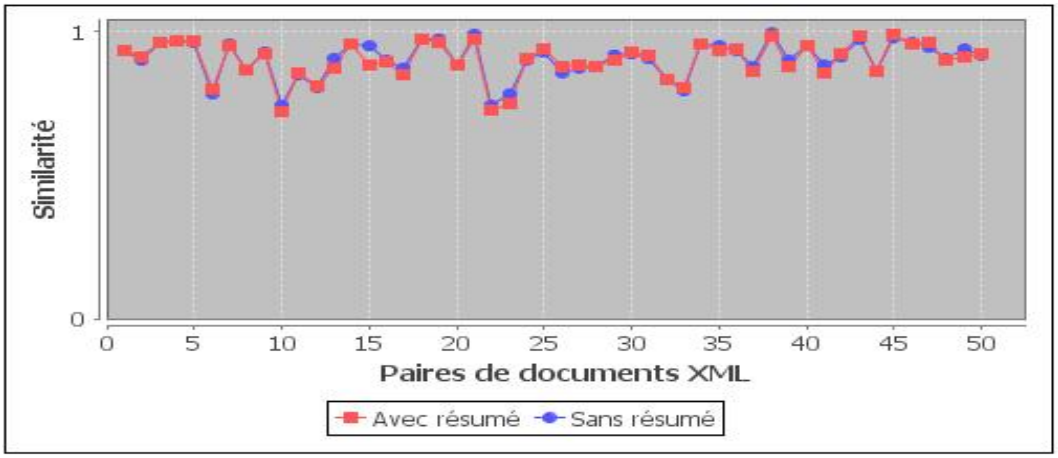
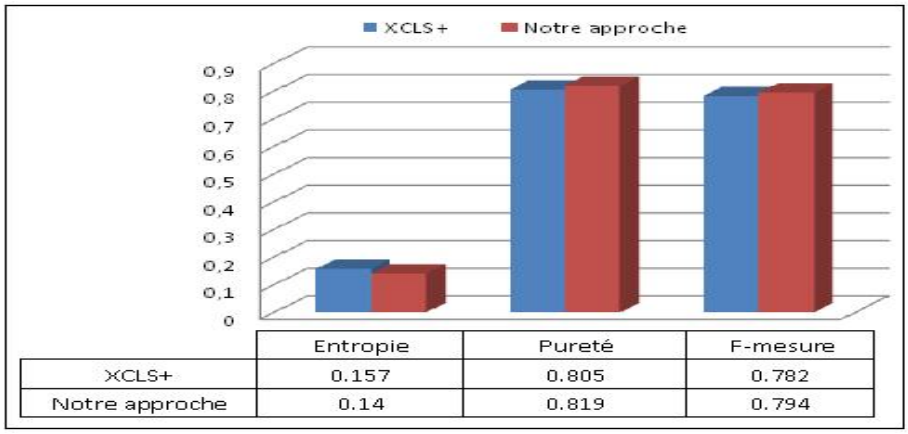


FIGURE 4.10 – Comparaison de similarité trouvée sans et avec le résumé correspondant aux données de tableau 4.6.

4.5.1.2 Evaluation de la qualité du clustering

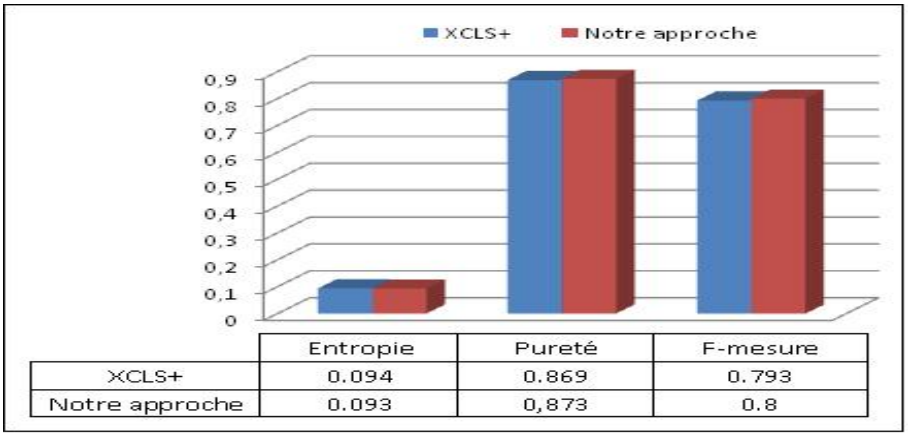
Pour évaluer l'approche proposée en terme de qualité de clusters obtenus, nous présentons dans ce qui suit les expérimentations réalisées sur l'ensemble de documents XML utilisée, où nous avons essayé d'exécuter l'algorithme XCLS+ avec le résumé d'arbre (notre approche) avec des valeurs différentes de paramètres (seuil et r). Les résultats obtenus après le calcul des différents critères d'évaluation à savoir l'entropie, la pureté et la F-mesure sont comparés avec ceux de l'approche XCLS+ en utilisant les mêmes données. Ces résultats sont présentés sur les figures 4.11, 4.12, 4.13, 4.14, 4.15 et 4.16.

Les graphiques comparatifs des résultats en figures 4.11, 4.12, 4.13, 4.14, 4.15 et 4.16 montrent que les valeurs de l'entropie, la pureté et la F-mesure obtenus par nos approches sont très proches et à celles trouvées par l'approche XCLS+. Alors nous pouvons dire que les clusters obtenus dans les deux cas ont presque la même qualité, et en général le résumé d'arbre a maintenu la qualité de clustering.



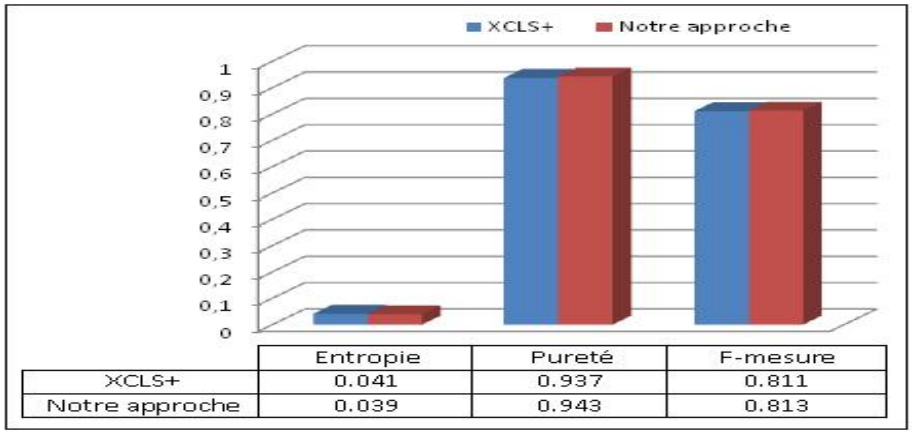
	Entropie	Pureté	F-mesure
XCLS+	0.157	0.805	0.782
Notre approche	0.14	0.819	0.794

FIGURE 4.11 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.75 et r =1.5).



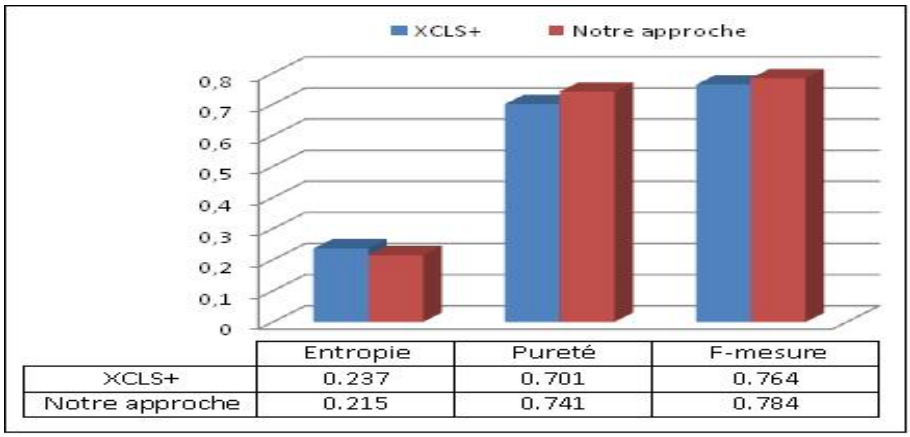
	Entropie	Pureté	F-mesure
XCLS+	0.094	0.869	0.793
Notre approche	0.093	0,873	0.8

FIGURE 4.12 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.80 et r =1.5).



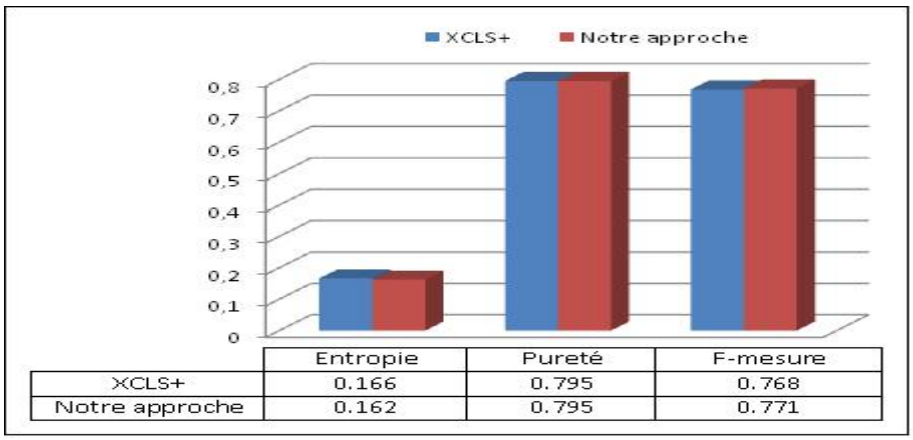
	Entropie	Pureté	F-mesure
XCLS+	0.041	0.937	0.811
Notre approche	0.039	0.943	0.813

FIGURE 4.13 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.85 et r =1.5).



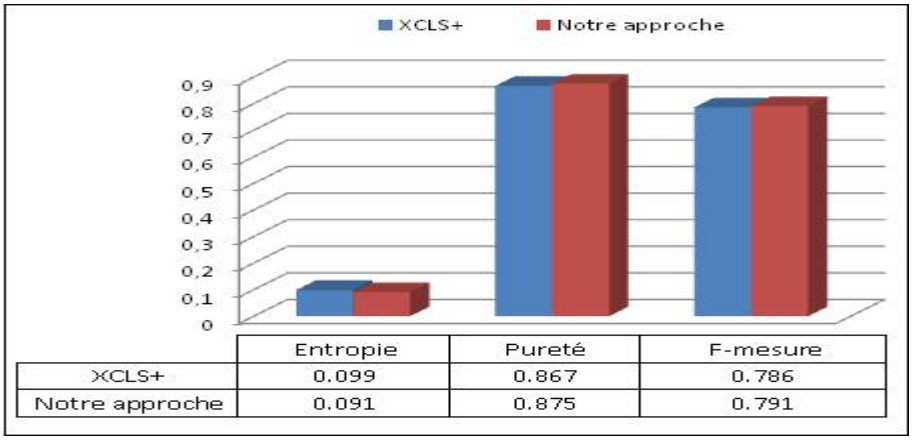
	Entropie	Pureté	F-mesure
XCLS+	0.237	0.701	0.764
Notre approche	0.215	0.741	0.784

FIGURE 4.14 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.75 et r =2).



	Entropie	Pureté	F-mesure
XCLS+	0.166	0.795	0.768
Notre approche	0.162	0.795	0.771

FIGURE 4.15 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.80 et r =2).

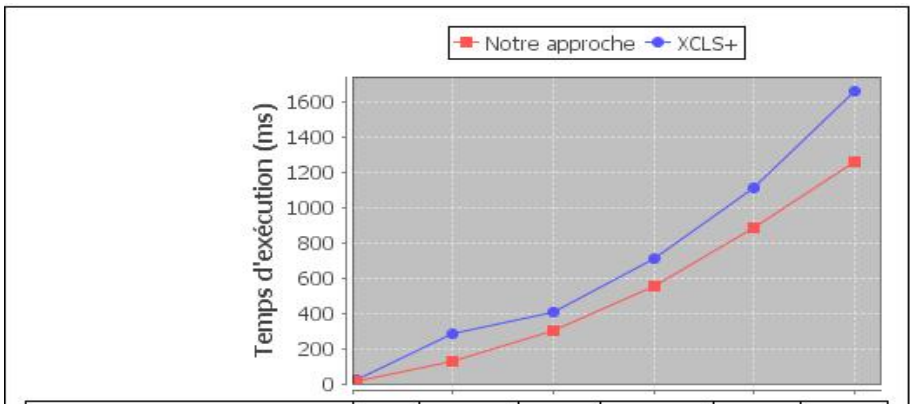


	Entropie	Pureté	F-mesure
XCLS+	0.099	0.867	0.786
Notre approche	0.091	0.875	0.791

FIGURE 4.16 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.85 et r =2).

4.5.1.3 Evaluation du temps d'exécution

Nous notons que l'algorithme XCLS+ a une complexité théorique en temps linéaire. Les différentes expériences effectuées sur le temps d'exécution et les résultats obtenus sont représentés dans la figure 4.17.



Nombre de documents XML	5	100	200	300	400	500
Notre approche	15	128	302	555	886	1247
XCLS+	20	280	410	707	1113	1662

FIGURE 4.17 – Temps d'exécution du clutering obtenus par notre approche versus XCLS+ (Structure).

Comme prévu, les résultats dans la figure 4.17 affirment que les temps d'exécution trouvés en utilisant le résumé d'arbre (notre approche) sont les meilleurs et cela est dû à la réduction de la taille de l'ensemble des nœuds de l'arbre correspondant au document ce qui permet de trouver les éléments communs et de calculer la similarité plus rapidement. Alors le résumé d'arbre a amélioré la performance de la procédure de regroupement en terme de temps. Où l'on a pu atteindre une accélération de l'ordre de 1/4.

4.5.2 Les expériences sur l'approche basée sur la structure et le contenu

Afin de valider l'approche proposée basée sur la structure et le contenu des documents XML et d'évaluer l'intérêt d'ajouter le contenu qui est limité par la structure du résumé d'arbre dans notre approche, nous essayons d'appliquer cette approche dans un premier temps sans l'utilisation de résumé d'arbre puis avec le résumé en mettant l'accent dans les deux cas sur la qualité des clusters obtenus et le temps d'exécution.

4.5.2.1 Evaluation de la qualité du clustering

Dans un premier temps, c'est l'approche proposée basée sur la structure et le contenu qui a été évaluée mais sans l'utilisation du résumé d'arbre où nous avons essayé d'exécuter l'algorithme XCLS+ avec des valeurs différentes de *seuil* et de *r*. Le tableau 4.7 reporte les résultats d'expérimentation et les évaluations sur la qualité des clusters obtenus sur l'ensemble de documents XML utilisée.

Algorithme	Facteur r	Seuil	Entropie	Pureté	F-mesure
XCLS+ + contenu	1.5	0.12	0.163	0.805	0.854
		0.15	0.116	0.856	0.817
	2	0.12	0.202	0.767	0.852
		0.15	0.133	0.853	0.819

TABLE 4.7 – *Résultats sur la qualité du clustering obtenus par l'approche XCLS+ (structure et contenu).*

Dans un second temps, nous avons répété la même procédure de regroupement mais avec l'utilisation du résumé d'arbre comme représentation où le rôle du résumé d'arbre est de limiter aussi le contenu des documents XML utilisé. La table 4.8 montre les résultats d'expérimentation sur la qualité des clusters obtenus sur l'ensemble de documents XML utilisé. Ces résultats sont comparés avec ceux de l'approche précédente. Les figures 4.18, 4.19, 4.20 et 4.21 illustrent cette étude comparative.

Algorithme	Facteur r	Seuil	Entropie	Pureté	F-mesure
Notre approche (S.C)	1.5	0.12	0.127	0.865	0.851
		0.15	0.078	0.913	0.820
	2	0.12	0.174	0.799	0.831
		0.15	0.122	0.859	0.807

TABLE 4.8 – *Résultats sur la qualité du clustering obtenus par notre approche (structure et contenu).*

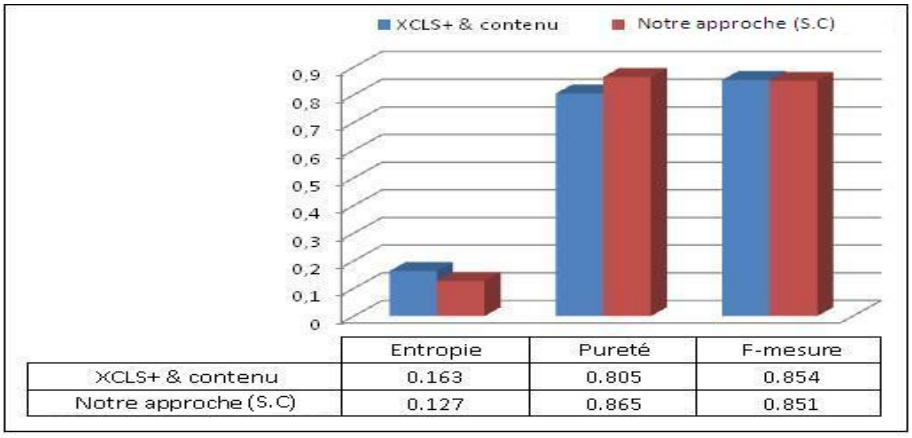
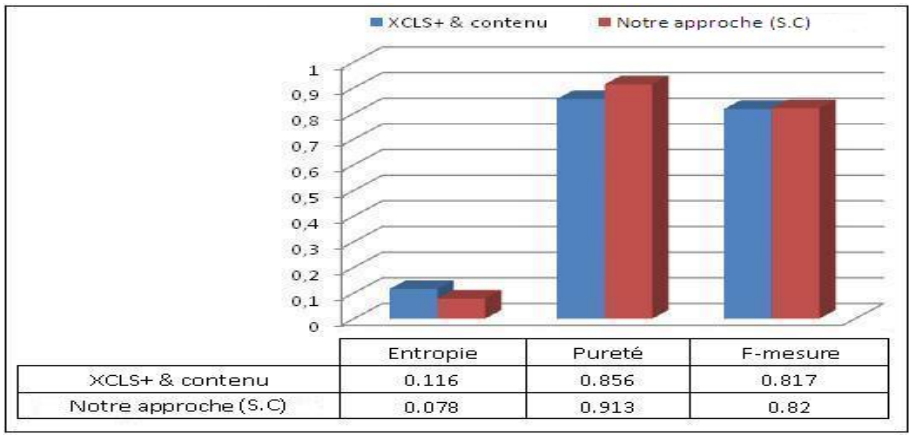
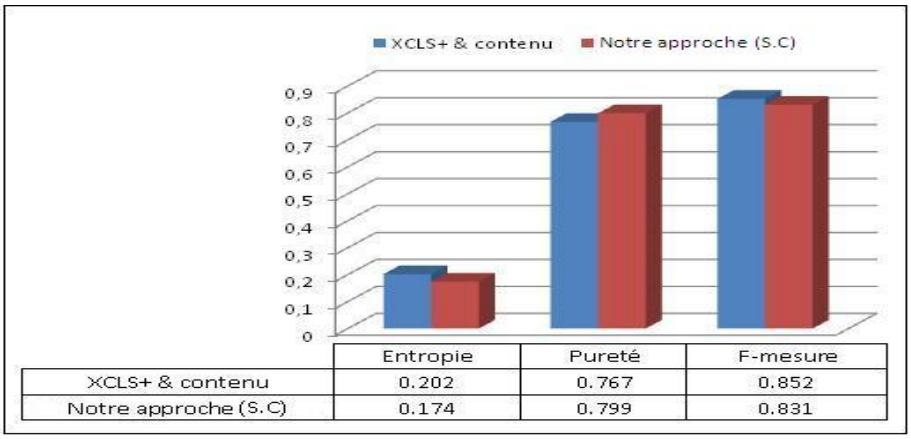


FIGURE 4.18 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.12 et r =1.5).



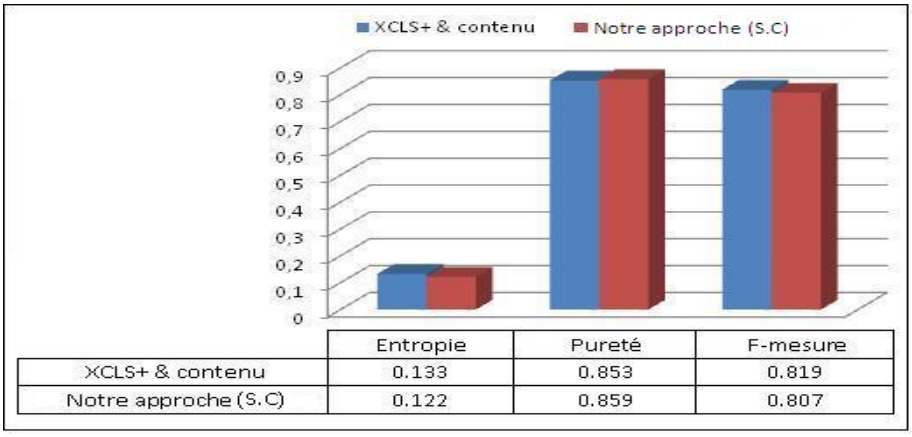
	Entropie	Pureté	F-mesure
XCLS+ & contenu	0.116	0.856	0.817
Notre approche (S.C)	0.078	0.913	0.82

FIGURE 4.19 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.15 et r =1.5).



	Entropie	Pureté	F-mesure
XCLS+ & contenu	0.202	0.767	0.852
Notre approche (S.C)	0.174	0.799	0.831

FIGURE 4.20 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.12 et r =2).



	Entropie	Pureté	F-mesure
XCLS+ & contenu	0.133	0.853	0.819
Notre approche (S.C)	0.122	0.859	0.807

FIGURE 4.21 – Comparaison des valeurs de l'entropie, la pureté et la F-mesure obtenues par les deux approches (seuil = 0.15 et r =2).

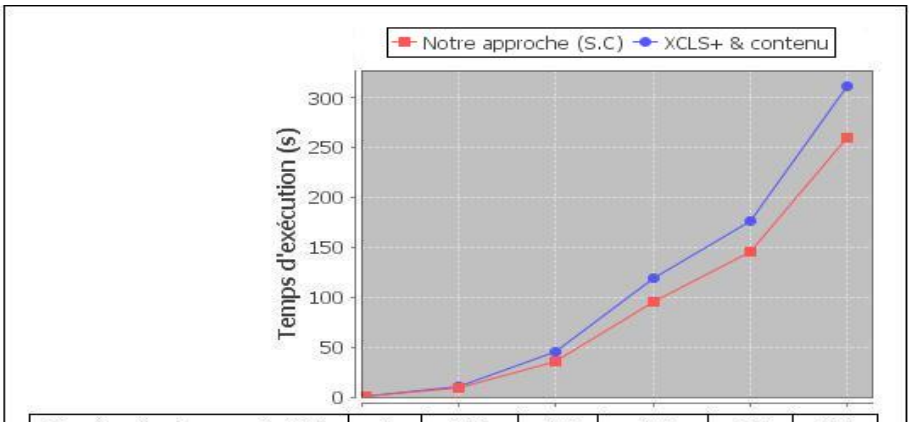
D'après les tables 4.7 et 4.8 et les graphiques comparatifs des résultats en figures 4.18, 4.19, 4.20 et 4.21 nous pouvons souligner que :

- Les résultats obtenus après le calcul des différents critères d'évaluation à savoir l'entropie, la pureté et la F-mesure montrent que l'approche de clustering proposée basée sur la structure et le contenu a réussi, de manière générale, à regrouper les documents XML avec une bonne qualité dans les deux cas (sans et avec le résumé d'arbre) surtout quand le facteur $r = 1.5$.
- Cependant, lorsque l'on utilise le résumé d'arbre les valeurs de l'entropie et la pureté trouvées sont toujours supérieures et au pire égale à celles de l'approche sans le résumé.

4.5.2.2 Evaluation du temps d'exécution

La figure 4.22 montre les différentes expériences effectuées et les améliorations en terme de temps d'exécution obtenus.

Le graphique comparatif des résultats en figure 4.22 montre que l'utilisation du résumé d'arbre dans l'approche proposée basée sur la structure et le contenu permet d'améliorer le temps d'exécution et cela est justifiée par la limitation de contenu des documents XML utilisé (l'ensemble des mots) ainsi la réduction de la taille de l'ensemble des nœuds de l'arbre correspondant au document ce qui permet de trouver les éléments communs et de calculer la similarité plus rapidement. Le facteur d'accélération atteint sur 500 documents est mesuré de 20%.



Nombre de documents XML	5	100	200	300	400	500
Notre approche (S.C)	0.08	9	36	95	146	260
XCLS+ & contenu	0.094	11	45	119	177	312

FIGURE 4.22 – Comparaison des temps d'exécution du clutering obtenus par notre approche avec et sans le résumé (structure et contenu).

4.6 Conclusion

Dans ce chapitre, nous avons présenté deux approches de clustering pour les documents XML qui sont basées sur la structure, et la structure et le contenu respectivement, et reposant sur l'approche de clustering proposée par Alishahi et al. [Alishahi et al., 2010].

Une étude expérimentale a ainsi été menée et portait sur des jeux de données réelles extraites de corpus INEX. Les résultats ont été comparés à ceux obtenus avec l'approche de clustering de [Alishahi et al., 2010].

La principale motivation de l'approche de clustering basée sur la structure est de proposer une approche qui permet de pouvoir facilement regrouper les documents XML à l'aide d'une représentation réduite "résumé d'arbre". Les résultats obtenus sont significativement encourageants et ont montré l'intérêt de l'approche de regroupement proposée fondée sur le résumé d'arbre qui a permis de maintenir la qualité de clustering avec un temps d'exécution meilleur.

Pour la deuxième approche de clustering, nous avons ajouté le contenu des éléments dont le but est d'améliorer la qualité de clustering. Dans ce cas, la similarité est calculée en fonction de la structure et le contenu. Les résultats obtenus montrent que l'approche a réussi à regrouper les documents XML sans dégradations des performances.

CONCLUSION ET PERSPECTIVES

Le travail présenté dans ce mémoire se situe dans le contexte du XML mining. Nous nous positionnons plus particulièrement dans le cadre de clustering des documents XML.

Le XML mining est un domaine de recherche très actif basé sur l'utilisation des techniques classiques du data mining afin de découvrir les connaissances cachées dans les documents XML. Il se distingue du text mining par le fait qu'il traite des contenus semi-structurés.

La naissance de XML mining est due à la prolifération rapide des documents XML dans le Web et la diminution de la capacité d'acquérir des connaissances à partir de sources XML en raison de leur hétérogénéité et de leurs structures irrégulières. Nous nous sommes intéressé dans ce mémoire à une des techniques de XML mining qui est le clustering dans le but de diviser cette masse importante de documents en groupes utilisables. Dans ce contexte, nous avons proposé deux approches dans le but de répondre à deux importants problèmes. La taille des documents XML est souvent importante ainsi une représentation arborescent, la plus naturelle, peut facilement faire défaut au problème de clustering. L'autre problème réside dans le fait que le contenu des documents XML est très peut considérée dans les travaux de clustering à comparer avec sa structure.

Notre première approche est une approche de clustering de documents XML par la structure. Elles est basée sur l'approche XCLS+ et l'utilisation des résumés d'arbres afin de diminuer les tailles des documents. Les résultats obtenus par nos expériences sur la collection INEX comparés à ceux de l'approche XCLS+ montrent que le résumé d'arbre a maintenu la qualité de clustering et avec des améliorations dans le temps d'exécution et la taille du cluster représentant.

Dans le deuxième approche, nous avons ajouté à l'approche précédente les caractéristiques du contenu des documents XML. Nous avons proposé une nouvelle représentation structure et contenu de niveau qui décrit la structure et le contenu de document XML, ainsi qu'une nouvelle formule de similarité LevelSimSC. L'algorithme XCLS+ est appliqué. Les résultats obtenus sur la collection INEX montrent que l'approche proposée basée sur la structure et le contenu a réussi à regrouper les documents XML avec une bonne qualité.

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

- Spécifier le seuil de manière automatique par l'algorithme.
- Prise en compte des autres éléments structurels à savoir les attributs des documents XML.
- Prendre en compte la similarité sémantique entre les tags car ils peuvent exister des balises différentes mais ils décrivent les mêmes choses. En utilisant des dictionnaires comme par exemple WordNet qui permet d'organiser les termes en ensembles des synonymes et définir des relations sémantiques entre ces ensembles.
- Prendre en compte aussi la similarité sémantique entre les mots.

BIBLIOGRAPHIE

[Aggrawal et al., 2007] C. C. AGGRAWAL N. TA et J. WANG, « XProj :A Framework for Projected Structural Clustering of XML Documents », *KDD'07*, California, USA, 12-15 Août 2007, p. 46-55.

[Agrawal et al., 1993] R. AGRAWAL, T. IMIELINSKI, et A. SWAMI, « Mining Association Rules Between Sets of Items in Large Databases », *SIGMOD Conference*, New York, USA, 1993, p. 207-216.

[Aïtelhadj et al., 2009] A. AITELHADJ, M. MEZGHICHE, et F. SOUAM, « Classification de Structures Arborescentes : Cas de Documents XML », *Conférence en recherche d'informations et applications*, Coria, 2009, p. 301-317.

[Alilaouar, 2007] Abdeslame ALILAOUAR, « Contribution à l'interrogation flexible de données semi-structurées », Th Doctorat, Université Paul Sabatier, Toulouse, 2007, 144 p.

[Alishahi et al., 2010] M. ALISHAHI, M. NAGHIBZADEH, et B.S. ASKI, « Tag Name Structure-based Clustering of XML Documents », *International Journal of Computer and Electrical Engineering*, Volume 2, Numéro 1, Février 2010, p. 119-126.

[Algergawy et al., 2008] A. ALGERGAWY, E. SCHALLEHN, et G. SAAKE, « A schema matching-based approach to XML schema clustering », *Proceedings of iiWAS*, 2008, p. 131-136.

[Anand et al., 1998] S.S. ANAND, A.R. PATRICK, J.G. HUGHES, et D.A. BELL, « A Data Mining methodology for cross-sales », *Knowledge Based Systems Jouranl*, Volume 10, 1998, p. 449-461.

[Antonellis et al., 2008] P. ANTONELLIS, CH. MAKRIS, et N. TSIRAKIS, « XEdge : Clustering Homogeneous and Heterogeneous XML Documents Using Edge Summaries », *SAC*, Fortaleza, Ceará, Brazil, 16-20 Mars 2008, p. 1081-1088.

[Ben Aouicha, 2009] Mohamed BEN AOUICHA, « Une approche algébrique pour la recherche d'information structurée », Th Doctorat, Université Paul Sabatier, Toulouse, 2009, 168 p.

[Ben Hassena et Miclet, 2008] A. BEN HASSENA et L. MICLET, « Les techniques d'appariement entre arbres », Rapport Bibliographique, Université de Rennes, 2008, 19 p.

[Berry et Linoff, 2000] M.J.A. BERRY et G. LINOFF, « Mastering Data Mining The Art and Science of Costumer Relationship Management », New York, Wiley Computer Pub., 2000, 494 p.

[Braga et al., 2003] D. BRAGA, A. CAMPI, S. CERI, M. KLEMETTINEN, et P. LANZI, « Discovering interesting information in xml data with association rules », *Proceedings of the 2003 ACM symposiumon Applied computing*, NewYork, USA, 2003, p. 450-454.

[Brillant, 2007] Alexandre BRILLANT, « XML cours et exercice », Eyrolles, 2007, 284 p.

[Carton, 2010] Olivier CARTON, « L'essentiel de XML », Cours XML, 2010.

[Chatti, 2006] Noureddine CHATTI, « Documents multi-structurés : de modélisation vers l'exploitation », Th Doctorat, INSA Lyon, 2006, 155 p.

[Chawathe, 1999] S. S. CHAWATHE, « Comparing hierarchical data in external memory », *Proceedings of the VLDB Conference*, Edinburgh, Scotland, UK, 1999, p. 90-101.

[Chawathe et al., 1996] S. S. CHAWATHE, A. RAJARAMAN, H. GARCIA-MOLINA, et J. WIDOM, « Change Detection in Hierarchically Structured Information », *Proceedings of the ACMSIGMOD Conference*, USA, 1996, p. 493-504.

[Chi et al., 2004] Y. CHI, Y. Yang, Y. XIA, et R.R .MUNTZ, « CMTreeMiner : Mining Both Closed and Maximal Frequent Subtrees », *The Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, Mai 2004, p. 63-73.

[Cios et al., 2005] K.J CIOS et L.A. KURGAN, « Trends in Data Mining and Knwoledge Discovery », In N.R.Pal et L.Jain, *Advanced Techniques in Knowledge Discovery and Data Mining*, Springer, 2005, p. 1-26.

[Cios et al., 2007] K.J. CIOS, W. PERDYCZ, R.W. SWINIARSKY et L.A. KURGAN, « Data mining : A Knowledge Discovery Appraoch », New york, Springer, 2007, 606 p.

[Costa et al., 2004] G. COSTA, G. MANCO, R. ORTALE, et A. TAGARELLI, « A Tree-based Approach to Clustering XML Documents by Structure », *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2004, p. 137-148.

[Cristianini et al.2002] N. CRISTIANINI, J. SHAWE-TAYLOR et H. LODHI, « Latent semantic kernels », *Journal of Intelligent Information Systems (JJIS)*, Volume 18, Issue 2/3, 2002, p. 127-152.

[Dalamagas et al., 2005] T. DALAMAGAS, T. CHENG, K.J WINKEL, et T. SELLIS, « Clustering XML document using structural summaries », *Current Trends in Database Technology-EDBT 2004 Workshops*, 2005, p. 547-556.

[Dalamagas et al., 2006] T. DALAMAGAS, T. CHENG, K.J WINKEL, et T. SELLIS, « A Methodology for Clustering XML Documents by Structure », *Information Systems*, Volume 31, Issue 3, 2006, p. 187-228.

[Denoyer et Gallinari, 2007] L. DENOYER et P. GALLINARI, « The Wikipedia XML Corpus », In Advances in XML Information Retrieval and Evaluation : Fifth Workshop of the INitiative for the Evaluation of XML Retrieval (INEX'06), Dagstuhl, Germany, 2007.

[Devi et Sreevani, 2010] B. N. DEVI et O. SREEVANI, « Dynamic Modelling Approach for Web Usage Mining Using Open Web Resources », *International Journal of Engineering Science and Technology*, Volume 2, 2010, p. 5605-5610.

[Doucet et A-Myka, 2002] A. DOUCET et H. AHONEN-MYKA, « Naive clustering of a large XML document collection », *INEX 2002 Workshop Proceedings*, 2002, p. 81-87.

[Etzioni, 1996] Oren ETIZION, « The Word Wide Web : quagmire or gold mine ? », *Communication of ACM*, Volume 39, numéro 11, 1996, p. 65-68.

[Fayyad et al., 1996a] U. FAYYAD, G. P-SHAPIRO, et P. SMYTH, « Knowledge Discovery and Data Mining : Towards a Unifying Framework », *Proceeding of the second international conference on knowledge discovery and data mining*, Portland, Oregon , 2-4 Août 1996, p. 82-88.

[Feldman et Dagan, 1995] R. FELDMAN et I. DAGAN, « Knowledge Discovery in Textual Databases (KDT) », *Proceeding of the First International Conference on Knowledge Discovery(KDD)*, 1995, p. 112-117.

[Flesca et al., 2002] S. FLESCA, G. MANCO, E. MASCIARI, L. PONTIERI, et A. PUGLIESE, « Detecting structural similarities between XML documents », *Proceedings of the Fifth International Workshop on the Web and Databases*, 2002, p. 55-60.

[Francesca et al., 2003] F. De FRANCESCA, G. GORDANO, R. ORTALE, et A. TAGARELLI« Distance-based Clustering of XML Documents », Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences, 2003, p. 75-78.

[Guha et al., 1999] S. GUHA, R. RASTOGI, et K. SHIM, « ROCK : A Robust Clustering Algorithm for Categorical Attributes », *Proceedings of the 15th International Conference on Data Engineering*, Washington, DC, USA, 1999, p. 512-.521.

[Gupta et al., 2007] G. GUPTA, G. HANS, et T. SEHGAL, « Applications and Trends in Data Mining », *Proceedings of Challengs and Opportunities in information technology*, 2007.

[Hand, 1998] David J. HAND, « Data mining : statistics and more ? », *American Statistician*, 1998, p. 112-118.

[Iyer et Simovici, 2008] I. SWAMI et D. A. SIMOVICI,« Structural Classification of XML Documents using MULTISETS », *WSPC-IJAIT*, Volume 17, Numéro 5, 29 septembre 2008, p. 1003-1022.

[Jeong et al., 2006] B. JEONG, D. Lee, J. LEE, et H. CHO, « Towards XML Mining : The Role of Kernel Methods », *Proceedings of the 2006 Fall Data Mining Conference*, Seoul, South Korea, Novembre 2006.

[Júnior et Gong, 2005] M.G.d-C. JUNIOR et Z. Gong, « Web Structure Mining : An Introduction », *Proceedings of the 2005 IEEE International Conference on Information Acquisition*, Hong Kong and Macau, China, 27 Juin- 03 Juillet 2005, p. 590-595.

[Karypis, 2002] George KARYPIS, « CLUTO - A Clustering Toolkit », Rapport technique, University of Minnesota, USA, 29 Avril 2002.

[Kim, 2009] Woosaeng KIM, « XML Documents Clustering based on Representative Path », *Proceedings of the WSEAES 13th international conference on computers*, 2009, p. 180-114.

[Kim, 2010] Woosaeng KIM, « XML Clustering by Bit Vector », *Proceedings of the 14th WSEAS international conference on Computers : part of the 14th WSEAS CSCC multiconference-Volume I*, 2010, p. 137-142.

[Kutty et al., 2007] S. KUTTY, R. NAYAK, et Y. LI, « PCITMiner-Prefix-based Closed Induced Tree Miner for finding closed induced frequent subtrees », *The Sixth Australasian Data Mining Conference (AusDM 2007)*, Gold Coast, Australia, 3-4 Décembre 2007, p. 147-156.

[Kutty et al., 2008] S. KUTTY, T. TRAN, R. NAYAK, et Y. LI, « Clustering XML Documents Using Closed Frequent Subtrees : A Structural Similarity Approach », *The 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007*, Germany, 17-19 Décembre 2007, p. 183-194.

[Kutty et al., 2009] S. KUTTY, R. NAYAK, et Y. LI, « HCX : an efficient hybrid clustering approach for XML documents », *The 9th ACM Symposium on Document Engineering : DocEng 2009*, Munich, Germany, 15-18 Septembre 2009, p. 94-97.

[Lee et al., 2001] J. LEE, K. LEE, et W. KIM, « Preparations for semantics-based XML mining », *Proceedings of IEEE International Conference on Data Mining (ICDM 2001)*, Novembre 2001, p. 345-352.

[Lee et P-Sauvagnat, 2010] M.D. LEE, K. P-SAUVAGNAT, « Utilisation de la distance d'édition pour l'appariement sémantique de documents XML », *Atelier GAOC - Conférence EGC*, Hammamet, Tunisie, 26-29 Janvier 2010.

[Lee et al., 2002] M. L. LEE, L. H. YANG, W.HSU, et X. YANG, « XClust : Clustering XML Schemas for Effective Integration », *CIKM'02*, Virginia, USA, 4-9 Novembre 2002, p. 292-299.

[Lian et al., 2004] W. LIAN, D.W. CHEUNG, N. MAMOULIS, et S.M. YIU, « An Efficient and Scalable Algorithm for Clustering XML Documents by Structure », *IEEE transactions on knowledge and Data Engineering*, volume 16, numéro 1, 2004, p. 82-96.

[Maimon et Rikach, 2005] O. MAIMON et L. RIKACH, « The Data Mining and Knowledge Discovery Handbook », Spinger Science et Business Media.INC, 2005, 1383 p.

[Mazuran et al., 2009] M. MAZURAN, E. QUINTARELLI, et L. TANCA, « Mining tree-based association rules from XML documents », *SEBD* , Italy, 2009, p. 109-116.

[Nayak, 2005] R. NAYAK, « Discovering Knowledge from XML Documents », in Wang, John, Eds. *Encyclopedia of Data Warehousing and Mining*. Idea Group Reference (IGI Global), 2005, p 372-376.

[Nayak, 2008] R. NAYAK, « XML Data Mining : Process and Applications », in Song, Min et Wu, Yi-Fang, Eds. *The Process and Applications of XML Data Mining*. Idea Group Inc. / IGI Global, 2008.

[Nayak et al., 2002] R. NAYAK, R. WITT, et A. TONEV, « Data Mining and XML documents », *Proceedings of the 2002 International Conference on Internet Computing*, Nevada, USA, 24-27 Juin 2002.

[Nayak et Xu, 2006] R. NAYAK et S. XU, « XML Documents Clustering by Structures », *INEX 2005 Workshop on Mining XML documents*, 2006, p. 432-442.

[Nierman et Jagadish, 2002] A. NIERMAN et H.V JAGADISH, « Evaluating structural Similarity in XML Documents », *Proceedings of the Fifth International Workshop on the web and Databases (WebBD 2002)*, 2002, p. 61-66.

[Rusty, 2003] Elliotte RUSTY HAROLD, « Processing XML with Java : a guide to SAX, DOM, JDOM, JAXP, and TrAX », chapitre 14 (JDOM), Addison-Wesley, 2003, p. 1071.

[Saitta, 2007] Sandro SAITTA, « Data Mining, des données à la connaissance », *Flash Informatique 10*, 18 décembre 2007, p. 4.

[Selkow, 1977] Stanley M. SELKOW, « The Tree-to-Tree Editing Problem », *Information Processing Letters*, 1977, p. 184-186.

[Stumme et al., 2006] G. STUMME, A. HOTHO, et B. BERENDT, « Semantic web mining : State of the art and future directions », *Web Semantics : Science, Services and Agents on the World Wide Web*, Volume 4, Numéro 2, Juin 2006, p. 124-143.

[Tran et al., 2008] T. TRAN, R. NAYAK, et P. BRUZA, « Combining structure and content similarities for XML document clustering », *The 7th Australasian Data Mining Conference*, Glenelg, South Australia, 27-28 novembre 2008, p. 219-226.

[Tufféry, 2007] Stéphane TUFFERY, « Data mining et statistique décisionnelle : l'intelligence des données », Paris, Edition Technip, 2007, 509 p.

[Vercoustre et al., 2006a] A.M. VERCOUSTRE, M. FEGAS, Y. LECHEVALLIER, et T. DESPEYROUX, « Classification de documents XML à partir d'une représentation linéaire des arbres de ces documents », *Actes des 6eme journées Extraction et Gestion des Connaissances (EGC 2006), Revue des Nouvelles Technologies de l'Information (RNTI-E-6)*, Lille, France, Janvier 2006, p. 433-444.

[Vercoustre et al., 2006b] A.M. VERCOUSTRE, M. FEGAS, S. GUL, et Y. LECHEVALLIER, « A flexible structured based representation for XML document mining », *Proccedings of the 4th International Workshop of the Initiative for Evaluation of the XML Retrieval, INEX'05*, Schloss Dagstuhl, Germany, 5 juillet 2006, p. 443-457.

[Wan et Dobbie, 2003] W.W. WAN et G. DOBBIE, « Extracting association rules from xml documents using xquery », *Proceedings of the 5th ACM international workshop on Web information and data management*, NewYork, USA, 2003, p. 94-97.

[Wanger et Fischer, 1974] R.A. WAGNER et M.J. FISCHER, « The String-to-String Correction Problem », *Journal of the ACM*, Volume 21, Numéro 1, 1974, p. 168-173.

[Witten et al., 2003] I.H. WITTEN, K.J. DON, M.DEWSNIP, et V. TABLAN, « Text mining in a digital library », *International Journal on Digital Libraries*, Volume 4, Numéro 1, 2003, p. 56-59.

[Yang et al., 2005] J. YANG, W. CHEUNG, et X. CHEN, « Learning the Kernel Matrix for XML Document Clustering », *In IEEE International Conference on e-Technology, e-Commerce and e-Service*, 2005, p. 353-358.

[Yoon et al., 2001] J. YOON, V. RAGHAVAN, et V. CHAKILAM, « BitCube : Clustering and Statistical Analysis for XML Documents », *Journal of Intelligent Information Systems*, Volume 17, 2001, p. 241-254.

[Zaki et Aggarwal, 2003] M.J. ZAKI et C.C. AGGARWAL, « XRules : An Effective Structural Classi?er for XML Data », *SIGKDD'03*, New York, USA, 24-27 Août 2003, p. 316-325.

[Zhang et Shasha, 1989] K. ZHANG et D. SHASHA, « Simple fast algorithms for the editing distance between trees and related problems », *SIAM Journal on Computing*, Volume 18, Issue 6, Décembre 1989, p.1945-1962.

[Zhang et al., 2002] Z. ZHANG, R. LI, S. CAO, et Y. ZHU, « Similarity Metric for XML Documents », *Workshop in knowledge and Experience Management, FGWM 2003*, Karlsruhe, Allemagne, 2002.

ANNEXE A : JDOM

JDOM est une API open source Java. Elle permet de représenter et manipuler un document XML plus simplement qu'avec les API classiques ainsi de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. Par exemple, JDOM utilise des classes plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe. En septembre 2004 JDOM est disponible en version 1.0 et est compatible avec les versions 1.1 du JDK et supérieures.

JDOM n'est pas un parseur : il a d'ailleurs besoin d'un parseur externe de type SAX ou DOM pour analyser un document et créer la hiérarchie d'objets relative à un document XML. Dans la mesure où l'API JDOM ne fait pas partie de la machine virtuelle Java, il faut, pour pouvoir l'utiliser, l'importer dans votre projet : import jdom.jar.

JDOM propose plusieurs fonctionnalités :
- Création de documents XML ;
- Encapsulation d'un document XML sous la forme d'objets Java de l'API ;
- Exportation d'un document dans un fichier, un flux SAX ou un arbre DOM ;
- Support de XSLT ;
- Support de XPath.

A.1 Les différentes entités de JDOM

Pour traiter un document XML, JDOM définit plusieurs entités qui peuvent être regroupées en trois groupes :
- les éléments de l'arbre : la classe Document, la classe Element, la classe Comment, la classe Attribute, etc ...
- les entités pour obtenir un parseur : les classes SAXBuilder et DOMBuilder.
- les entités pour produire un document : les classes XMLOutputter, SAXOutputter, DOMOutputter.

Les classes clefs présentées ci-après sont Document, Element et Attribut.

A.1.1 La classe Document

La classe Document encapsule l'arbre dans lequel JDOM stocke le document XML. Un document a trois propriétés :
- ***Un Element root*** : peut être nul temporairement ;
- ***Un "DocType object" (DTD)*** : peut être nul ;
- ***Une liste*** contenant l'élément root et, éventuellement, des instructions de traitement ou des "comments" dans le prologue et l'épilogue du document XML.

La classe Document possède plusieurs méthodes dont les principales sont :
`Document addContent(Comment)` : ajouter un commentaire au document ;
`List getContent()` : renvoyer un objet List qui contient chaque élément du document ;
`DocType getDocType()` : renvoyer un objet contenant les caractéristiques Doctype du document ;
`Document setDocType()` : définir le DocType du document ;
`Element getRootElement()` : renvoyer l'élément racine du document ;
Document setRootElement(Element) : définir l'élément racine du document ;
`Boolean hasRootElement()` : renvoyer un booléen qui indique si le document possède un élément racine ;
`Element detachRootElement()` : détache l'élément racine du document ;
`Document addContent(ProcessingInstruction)` : ajouter une instruction de traitement ;
`Document addContent(Comment)` : ajouter un commentaire ;
`Document removeContent(ProcessingInstruction)` : supprimer une instruction de traitement ;
`Document removeContent(Comment)` : supprimer un commentaire ;

A.1.2 La classe Elément

La classe Element est la classe la plus volumineuse de l'API JDOM et surtout la plus importante car l'élément est la base de la structure d'un document XML. La classe Element est le moyen privilégié par lequel un programme parcourt l'arbre pour trouver un contenu " Content " particulier. Un objet Element possède plusieurs propriétés :

- ***Name*** : un String initialisé à la construction de l'Element ; ne peut jamais être null ou vide ; est accessible via les méthodes setName() et getName() ;
- ***Namespace*** : l'espace de nommage de l'élément. Il y a toujours un objet de type Namespace pour chaque élément : si l'élément ne possède pas d'espace de nommage alors la propriété vaut Namespace.NO_NAMESPACE. Il est accessible via les méthodes setNamespace() et getNamespace() ;
- ***Content*** : collection de type List des éléments fils de l'élément
- ***Parent*** : élément père de l'élément : peut être null si l'élément est l'élément racine ou si l'élément n'est pas ajouté dans un document.
- ***Document*** : document qui contient cet élément : peut être null si l'élément n'est pas ajouté à un document ; accessible par la méthode getDocument() ;
- ***Attributes*** : une collection de type List qui encapsule les attributs de l'élément. La liste est accessible via les méthodes getAttributes() et setAttributes(). Les articles de la liste peuvent être lus et/ou modifiés par les méthodes getAttribute(), getAttributeValue(), et setAttribute().

La classe Element possède de nombreuses méthodes pour obtenir, ajouter ou supprimer une entité de l'élément (un élément enfant, le texte, un attribut, ...) :

Parcours d'arbre total

La méthode getContent() est le moyen de naviguer au sein d'un arbre JDOM. Cette méthode renvoie toutes les entités (commentaires, texte, ...) y compris les éléments fils sous la forme d'une collection. Le parcours total de l'arbre se fait par appel récursif de getContent().

Parcours d'arbre en ne s'intéressant qu'aux Elements - Tous les enfants

La class Element dispose de six méthodes permettant d'opérer uniquement sur les enfants de type Element :
`List getChildren()` : renvoyer une liste qui contient tous les éléments enfants directement rattachés à l'élément ;
`List getChildren(String)` : renvoyer une liste qui contient tous les éléments enfants directement rattachés à l'élément dont le nom est fourni en paramètre ;
`List getChildren(String, Namespace)` : renvoyer une liste qui contient tous les éléments enfants

directement rattachés à l'élément dont le nom et l'espace de nommage sont fournis en paramètre ;
`Boolean removeChildren()` : Supprimer tous les éléments enfants ;
`Boolean removeChildren(String)` : Supprimer tous les éléments enfants dont le nom est fourni en paramètre ;
`Boolean removeChildren(String, Namespace)` : Supprimer tous les éléments enfants dont le nom et l'espace de nommage sont fournis en paramètres.
Il est important de se rappeler que lorsqu'un Element est supprimé, ses fils et autres descendants sont supprimés en même temps que lui.

Parcours d'arbre en ne s'intéressant qu'aux Elements - Un seul enfant à la fois
Les méthodes permettent d'accéder directement a un enfant sont :
`Element getChild(String)` : renvoyer le premier élément enfant dont le nom est fourni en paramètres ;
`Element getChild(String, Namespace)` : renvoyer le premier élément enfant dont le nom et l'espace de nommage sont fournis en paramètres ;
L'utilisation de ces méthodes présente deux problèmes. La première est que, s'il existe plus d'un enfant possédant le même nom, seul le premier est retourné. Le second problème est que, s'il existe pas d'enfant, alors getChild() retourne null, menant ainsi à une NullPointerException.
`Boolean removeChild(String)` : supprimer l'élément enfant dont le nom est fourni en paramètre ;
`Boolean removeChild(String, Namespace)` : supprimer l'élément enfant dont le nom et l'espace de nommage sont fournis en paramètres.
Les méthodes removeChild() partagent avec getChild() le même problème : elles ne s'intéressent qu'au premier enfant. Cependant, après avoir supprimé le premier enfant, le second est maintenant le premier... Après avoir supprimé celui-ci, le troisième déviant alors à son tour le premier. Il existe donc une option qui ne fonctionne pas avec getChild() : vous pouvez simplement appeler removeChild() jusqu'à ce que null soit retourné, indiquant qu'il n'y a plus d'enfants.

Lire et écrire le texte d'un élément
Les méthodes permettant de lire le texte d'un élément sont :
`String getText()` : renvoyer les données au format texte contenues dans l'élément ;
`String getTextTrim()` : retourne la même chose que la première méthode en ayant supprimé les espaces précédents et suivants le texte ;
`String getTextNormalize()` : fait de même et supprime également, au sein du texte, tous les espaces consécutifs de façon à assurer que les mots ne sont séparés que par un espace unique ;
`Element setText(String)` : mettre à jour les données au format texte de l'élément.

Lire le texte d'un enfant
JDOM fournit six méthodes d'extraction du texte des éléments enfants :
`String getChildText(String name)` : renvoyer le texte du premier élément enfant dont le nom est fourni en paramètre ;
`String getChildText(String, Namespace)` : renvoyer le texte du premier élément enfant dont le nom et l'espace de nommage sont fournis en paramètre ;
`String getChildTextTrim(String name)` ;
`String getChildTextTrim(String name, Namespace namespace)` ;
`String getChildTextNormalize(String name)` ;
`String getChildTextNormalize(String name, Namespace namespace)` ;

Filtres
Il est possible de paramètrer la méthode getContent() en lui passant un objet org.jdom.filter.Filter de façon à limiter le contenu retourné par la méthode. Cette interface determine si un objet peut être ajouté, supprimé ou inclus dans une liste particulière. Pour le parcours et la recherche, seule

la méthode matches() importe. Elle détermine si un objet particulier est inclus ou non dans la List retournée par getContent(). JDOM propose deux implémentations de l'interface Filter :
ContentFilter : permet de filtrer sur le type de nœuds.
ElementFilter : permet de filtrer sur le nom et/ou l'espace de nommage des éléments.
La classe ContentFilter possède plusieurs constructeurs :
`ContentFilter()` : filtre acceptant tous les types de noeuds ;
`ContentFilter(int)` : filtre acceptant uniquement les noeuds précisés dans le masque (le masque utilise les constantes définies dans la classe ContentFilter) ;
`ContentFilter(boolean)` : filtre acceptant ou non tous les types de noeuds selon la valeur du paramètre.
Le classe ElementFilter possède plusieurs constructeurs :
`ElementFilter()` : filtre qui ne renvoie que les noeuds de type Element ;
`ElementFilter(String)` : filtre qui ne renvoie que les éléments dont le nom est fourni ;
`ElementFilter(Namespace)` : filtre qui ne renvoie que les éléments dont l'espace de nommage est fourni ;
`ElementFilter(String, Namespace)` : filtre qui ne renvoie que les éléments dont le nom et l'espace de nommage sont fournis.

Ajouter ou supprimer des enfants

L'ajout et la suppression de nœud à un Element se fait par l'appel aux méthodes :
`Element addContent(String text)` : ajouter des données sous forme de texte à l'élément ;
`Element addContent(Element)` : ajouter un élément fils à l'élément ;
`Element addContent(Comment)` : ajouter un commentaire à l'élément ;
`boolean removeContent(Comment)` : supprimer le commentaire fourni en paramètre ;
`boolean removeContent(Element)` : supprimer l'élément fourni en paramètre.
Ces méthodes ajoutent ou suppriment leur argument à la liste d'enfants de l'Element. Il est également possible de retrouver la List par la méthode getContent() et de supprimer les éléments en utilisant les méthodes de List remove() et removeAll().

Parents et ancêtres

Le parcours de l'arbre peut aussi se faire en remontant. Il suffit pour cela de demander le parent du parent du parent... jusqu'à ce que la méthode getParent() retourne null. Ceci indique qu'on est face à la racine de l'arbre, ou face à son noeud le moins profond. La classe Element propose les méthodes :
`Element getParent()` : renvoyer l'élément père de l'élément ;
`Boolean isRootElement()` : renvoyer un booléen qui indique si l'élément est l'élément racine du document ;
`Boolean isAncestor()` : renvoyer un booléen indiquant si l'élément est un ancêtre de l'élément fourni en paramètre.

Attributs

La classe Element possède plusieurs méthodes qui lisent et écrivent les valeurs des différents attributs de l'élément. Hormis cas particulier, ces méthodes suffisent à traiter les attributs et il n'est pas nécessaire d'utiliser directement la classe Attribute.
`Attribute getAttribute(String)` : renvoyer l'attribut dont le nom est fourni en paramètre ;
`Attribute getAttribute(String, Namespace)` : renvoyer l'attribut dont le nom et l'espace de nommage sont fournis en paramètres ;
`List getAttributes()` : renvoyer une collection qui contient tous les attributs ;
`String getAttributeValue(String)` : renvoyer la valeur de l'attribut dont le nom est fourni en paramètres ;
`String getAttributeValue(String, Namespace)` : renvoyer la valeur de l'attribut dont le nom et

l'espace de nommage sont fournis en paramètres ;
`Boolean removeAttribute(String)` : supprimer l'attribut dont le nom est fourni en paramètre ;
`Boolean removeAttribute(String, Namespace)` : supprimer l'attribut dont le nom et l'espace de nommage sont fournis en paramètres ;
`Element setAttribute(Attribute)` : ajouter un attribut ;
`Element setAttribute(String, String)` : ajouter un attribut dont le nom et la valeur sont fournis en paramètres ;
`Element setAttribute (String, String, Namespace)` : ajouter un attribut dont le nom, la valeur et l'espace de nommage sont fournis en paramètres ;

A.1.3 La classe Attribut

Cette classe encapsule un attribut d'un élément. La classe Attribut possède plusieurs propriétés :
- ***name*** : le nom de l'attribut accessible par les méthodes setName() et getName().
- ***namespace*** : l'espace de nommage accessible via setNamespace() et getNamespace(). Tout attribut non préfixé est initialisé par Namespace.NO_NAMESPACE.
- ***value*** : la valeur de l'attribut est un String contenant la valeur normalisée de l'Attribut. Les méthodes d'accès sont getValue() et setValue(). Il existe également des méthodes permettant de lire la valeur de l'attribut sous forme de double, float, long, int, ou boolean.
- ***parent*** : l'élément qui contient l'attribut, accessible via les méthodes getParent() et setParent(). Un attribut ne peut avoir plus d'un parent. Avant d'attacher un Attribut à un nouveau parent, vous devez invoquer detach() pour séparer l'Attribut de son parent en cours.
- ***Type*** : le type de l'attribut (par défaut Attribute.UNDECLARED_ATTRIBUTE), est accessible via les méthodes getAttributeType() et setAttributeType().

En plus des méthodes mentionnées précédemment, la classe possède les méthodes suivantes :
`clone()` : retourne un clone de cet Attribut ;
`detach()` : détache l'attribut de son parent et le retourne ou ne fait rien si l'attribut n'a pas de parent ;
`equals(Java.lang.Object ob)` : retourne un Boolean indiquant l'égalité ou non entre l'Attribut et Object ob ;
`getBooleanValue()` : renvoie la valeur de l'attribut sous forme d'un boolean ou une DataConversionException si la conversion ne peut être réalisée ;
`getDocument()` : renvoie le Document auquel se rattache l'attribut, null si l'attribut ne se rattache à aucun Document ;
`getDoubleValue()` : renvoie la valeur de l'attribut sous forme d'un double ou une DataConversionException si la conversion ne peut être réalisée ;
`getFloatValue()` : renvoie la valeur de l'attribut sous forme d'un float ou une DataConversionException si la conversion ne peut être réalisée ;
`getIntValue()` : renvoie la valeur de l'attribut sous forme d'un int ou une DataConversionException si la conversion ne peut être réalisée ;
`getLongValue()` : renvoie la valeur de l'attribut sous forme d'un long ou une DataConversionException si la conversion ne peut être réalisée ;
`getNamespacePrefix()` : renvoie sous forme de String le "namespace prefix " de l'attribut.
`getNamespaceURI()` : renvoie le URI sous forme de String ;
`hashCode()` : retourne le hash code (int) de l'attribut ;
`setAttributeType(int type)` : initialise le type de l'attribut ;
`toString()` : retourne une représentation sous forme de String de l'attribut (pratique pour le débogage).