F-CheX : Une approche de fouille dans les documents XML

Amina MADANI*, Omar BOUSSAID**
Hafida ABED*

*Université Saad Dahlab de Blida (Algérie)
a_madani@univ-blida.dz, hafidabouarfa@hotmail.com

**Université Lumière Lyon2 (France)
Omar.Boussaid@univ-lyon2.fr
http://eric.univ-lyon2.fr/~boussaid/

Résumé. Nous présentons dans cet article une approche de fouille dans les documents XML qui prend en compte la structure et le contenu. Notre approche consiste à effectuer un *clustering* sur les documents XML. Ces derniers sont représentés par des ensembles de chemins conservant la structure arborescente des éléments. Les ensembles de chemins sont mappés dans une matrice sur laquelle une méthode de *clustering* est appliquée. L'approche proposée utilise un thésaurus créé au préalable pour gérer l'aspect sémantique des mots. Une évaluation de notre approche est effectuée à travers une étude expérimentale sur deux collections de documents XML.

1 Introduction

Le développement du document électronique et du web ont permis l'émergence de formats semi-structurés permettant la représentation et le stockage de documents textuels ou multimédias. Différents formats comme le XML sont aujourd'hui très populaires et sont en train de s'imposer. Ils permettent de représenter l'information sous une forme enrichie et adaptée à des besoins spécifiques. Ces types de formats permettent de représenter conjointement les données et une information sur leur structure.

En outre, les techniques d'analyse traditionnelles ne permettent pas une exploration adaptée à ce genre de documents. En effet, ces derniers contiennent des données semi-structurées et des métadonnées qui sont des informations sur celles-ci. Les techniques de fouille de données (*data mining*), plus particulièrement le *text mining*, ne tiennent pas compte des aspects spécifiques des documents XML.

Différents travaux sur le *XML mining* sont alors proposés. D'une manière générale, la fouille des documents XML s'appuie sur les techniques classiques de fouille et notamment le classement (*classification*) et la classification (*clustering*). La fouille des documents XML s'avère être un volet de recherche récent et assez peu exploré.

Dans cet article, nous présentons une approche de fouille des documents XML que nous proposons. Notre approche prend en considération le contenu textuel, qui représente le

contenu des balises, en plus du contenu structurel, qui représente la structure du document, c'est-à-dire l'ensemble des éléments (balises).

Cet article est organisé comme suit : la section 2 présente un état de l'art sur les approches de fouille dans les documents XML. Dans la section 3, nous présentons notre approche. L'étude expérimentale est exposée dans la section 4. Enfin, nous terminons cet article par une conclusion et des perspectives.

2 Etat de l'art

Différentes approches de fouille dans la structure et dans le contenu sont proposées dans la littérature. Dans (Madani et al., 2009), nous avons déjà proposé une analyse approfondie de l'état de l'art du domaine de *XML mining*.

Les tableaux 1 et 2 présentent une synthèse des approches existantes sur la fouille de documents XML dont les références sont citées dans (Madani et al., 2009). Ces comparaisons sont faites en utilisant les critères suivants:

- *Doc. XML* : le type de documents XML utilisés (instances XML ou schémas XML).
- *Entrée* : la représentation du document XML en entrée.
- *Tâche de fouille* : la tâche de fouille accomplie (*clustering*, *classification* ou *association*).
- *Modèle* : 5 modèles de représentations des documents XML sont exploités :
  - *Modèle basé sur les vecteurs* : qui transforme un document XML en un vecteur.
  - *Modèle basé sur la similarité* : qui utilise des mesures de similarité.
  - *Modèle basé sur les réseaux de neurones* : exemple le modèle *Self Organizing Map*.
  - *Modèle d'arbres fréquents* : qui utilise un ensemble fréquent d'items d'un arbre XML.
  - *Modèle basé sur les réseaux bayésiens*.
- *A priori* : la spécification des paramètres est faite par les experts du domaine ;
- *Sémantique* : prise en compte de l'aspect sémantique des éléments, tel que la gestion des mêmes polysémies, les synonymes, etc.
- *Échantillonnage* : utilisation d'un échantillon pour assurer la phase d'apprentissage.
- *Sortie* : définition des résultats finaux obtenus.

Nous remarquons que la fouille est généralement appliquée sur les instances des documents XML bien que les grammaires XML (DTD ou schéma XML) sont mieux adaptés pour la fouille dans la structure. Dans cette dernière, le modèle basé sur la similarité est le plus utilisé avec des mesures de similarité telles que la distance d'édition même si elle est très couteuse par rapport à d'autres distances.

La fouille dans le contenu reste encore moins traitée. Cependant, les techniques de *text mining* peuvent étendre les techniques de *data mining* pour le contenu textuel.

Plusieurs travaux reposent sur une méthodologie de *clustering*. Ce dernier est un très bon moyen pour organiser automatiquement des grandes collections de documents XML en des petits sous ensembles homogènes regroupant ensemble les documents les plus similaires. La classification est aussi utilisée pourtant elle nécessite un échantillon important pour un bon apprentissage. Une riche collection d'exemples doit être traitée afin de faciliter la prédiction des classes dans la phase de test. L'association reste la moins utilisée dans la fouille des documents XML.

	Approche	Doc. XML	Entrée	Tâche de fouille	Modèle	A priori	Sémantique	Échantillonnage	Sortie
Structure	**Termier et al. 2002**	Instances	Arbre étiqueté	Clustering	Arbres fréquents	*			Clusters
	Francesca et al. 2003	Instances	Arbre enraciné étiqueté	Clustering	Similarité				Clusters
	Aggarwal et al. 2003	Instances	Règles structurelles	Classification	Arbres fréquents	*		*	Classes
	Boussaid et al. 2004	Instances	Matrice booléenne et DTD minimale	Association	Arbres fréquents	*			Doc. XML
	Dalamagas et al. 2004	Instances	Graphe connexe	Clustering	Similarité				Clusters
	Lian et al. 2004	Instances	Structure SG de bits	Clustering	Similarité	*			Clusters
	Candillier et al. 2005	Instances	Ensembles de paires d'attribut-valeur	Classification	Vecteurs	*		*	Arbre de décision
				Clustering					Hiérarchie de clusters
	Hagenbuchner et al. 2005	Instances	Vecteurs	Clustering	Réseaux de neurones			*	Clusters
	Nayak et al. 2005	Instances	Structures niveaux	Clustering	Similarité	*			Clusters
	Garboni et al. 2006	Instances	Vecteurs de séquences	Classification	Similarité			*	Clusters
	Nayak et al. 2007	Schémas	Matrice de similarité	Clustering	Similarité	*	*		Arbre de clusters

TAB. 1 – Comparaison des approches de fouille dans la st ructure XML (Madani et al., 2009).

	Approche	Doc. XML	Entrée	Tâche de fouille	Modèle	A priori	Sémantique	Échantillonnage	Sortie
Contenu	**Yang et al. 2002**	Instances	Structured Link Vector	Clustering	Vecteurs	*			Clusters
	Braga et al. 2002	Instances	Table relationnelle	Association	Arbres fréquents	*			Doc. XML
	Denoyer et al. 2004	Instances	Arbres	Classification	Réseaux bayésiens			*	Classes
	Vercoustre et al. 2006	Instances	Ensembles de chemins	Clustering	Vecteurs	*	*		Clusters
	Xing et al. 2006	Instances Schémas	Arbre étiqueté ordonné et grammaire NRHG	Classification	Similarité			*	Classes
	Doucet et al. 2006	Instances	Vecteurs	Clustering	Vecteurs	*			Clusters
	Knijf 2007	Instances	Arbre d'attributs enraciné étiqueté ordonné	Classification	Arbres fréquents	*		*	Arbre de décision

TAB. 2 – Comparaison des approches de fouille dans le co ntenu XML (Madani et al., 2009).

Certains aspects demeurent ignorés dans la majorité des travaux. Un aspect primordial pour la qualité de la fouille surtout dans le contenu textuel est l'aspect sémantique. Celui-ci est ignoré dans la majorité des travaux existants.

Malgré son importance, la structure arborescente des éléments composant un document XML (la notion du chemin…) est parfois ignorée.

Dans la section suivante, nous présentons notre approche de fouille dans les documents XML. Cette approche basée sur le modèle des vecteurs, permet la fouille aussi bien dans la structure que dans le contenu des documents XML. Elle consiste à appliquer le *clustering* sur les documents XML, ces derniers sont représentés par des ensembles de chemins conservant la structure arborescente des éléments. L'approche permet de gérer l'aspect sémantique des mots.

3 F-CheX, une approche de fouille par clustering des Chemins XML

L'approche est constituée de trois grandes phases dont chacune est composée de différentes étapes (voir figure 1). Elle s'inspire des travaux présentés dans (Vercoustre et al., 2006) qui ont proposé le *clustering* des documents XML représentés par des ensembles de chemins générés à partir de leurs arbres en ajoutant le contenu textuel et les attributs comme des nœuds de l'arbre. L'originalité de notre approche réside dans l'utilisation d'un thésaurus pour gérer l'aspect sémantique des mots présents dans la collection XML. Le thésaurus est créé à partir de deux sacs de mots générés de la collection XML en se basant sur l'approche de (Doucet et al., 2006). C'est la combinaison des concepts de thésaurus, de sacs de mots et de chemins XML, qui assure en réalité l'originalité de notre approche.

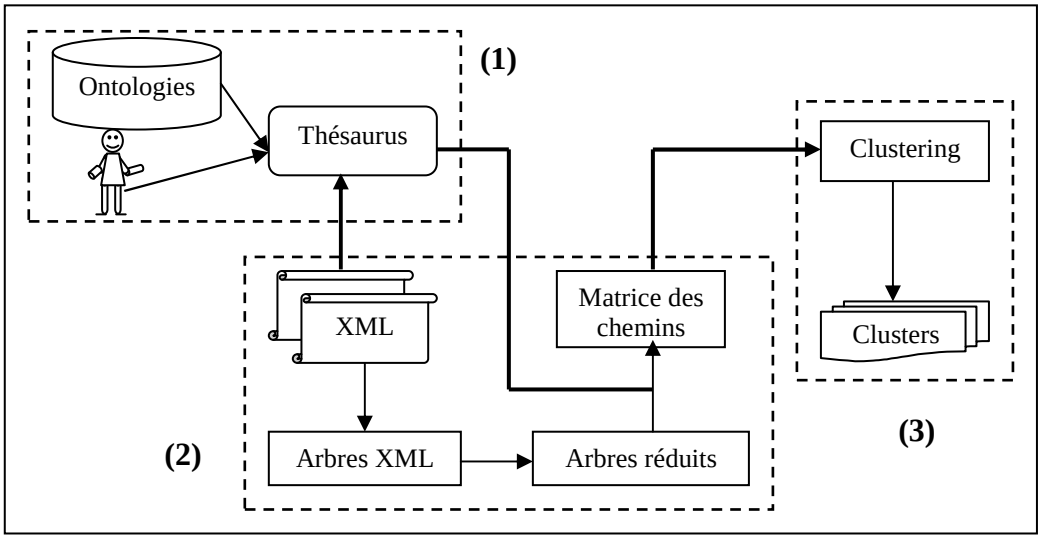

FIG. 1 – Architecture globale de l'approche.

3.1 Construction d'un thésaurus

(Doucet et al., 2006) proposent plusieurs représentations des documents XML telles que l'utilisation d'un sac de balises (*bag of tags*), d'un sac de mots (*bag of words*) et la combinaison du texte et des balises (*text+tags*) pour représenter les documents XML. Ils appliquent directement l'algorithme *k-means* sur ces représentations.

Dans notre cas, nous proposons de construire deux sacs de mots à partir de toute la collection des documents XML. Ces deux sacs contiennent les mots des éléments (structure) et ceux des valeurs d'éléments (contenu) :

- *Sac de structure* : contient des éléments qui représentent les termes de toutes les balises présentes dans la collection ;
- *Sac de contenu* : contient des éléments qui sont le contenu textuel entre les balises.

Les éléments des deux sacs peuvent être composés d'un ou de plusieurs termes. Par exemple, l'élément « Bibliothèque » est composé d'un seul terme alors que l'élément « XML cours et exercices » contient quatre termes.

Les deux sacs de mots permettent de constituer un thésaurus excluant toute sorte de redondances telle que chaque élément n'apparaisse qu'une seule fois dans un sac de mots. Le traitement des éléments fait appel à une ontologie du domaine si elle existe, ou à d'autres ontologies telles que *WordNet*[1] (Fellbaume, 1988), ou à des avis d'experts du domaine. Le thésaurus contient trois types d'éléments que nous définissons comme suit (figure 2) :

- *Élément maître* : les éléments différents et ayant la même sémantique sont regroupés sous un seul élément appelé maître. L'élément maître possède un ou plusieurs éléments synonymes.
- *Élément esclave* : les éléments synonymes qui sont englobés sous un élément maître sont appelés des éléments esclaves.
- *Élément neutre* : un élément qui n'est ni maître ni esclave est appelé un élément neutre.

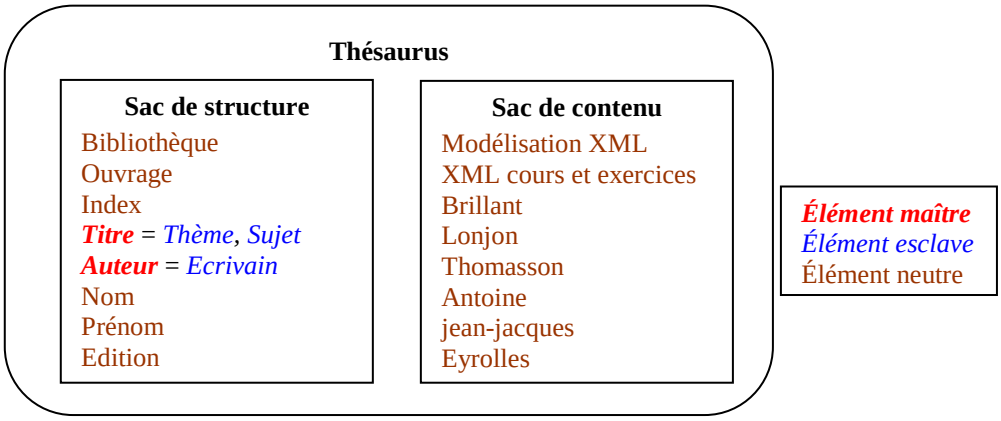

FIG. 2 – Un exemple d'un thésaurus.

[1] *WordNet* est une ressource lexicale de langue anglaise, disponible sur internet, qui regroupe des termes dénotant un concept donné (noms, verbes, adjectifs et adverbes) en ensembles de synonymes appelés *synsets* (Safar et al., 2007).

Nous présentons maintenant la deuxième phase de notre approche, qui consiste à générer une matrice des chemins.

3.2 Génération de la matrice des chemins

Puisque les documents XML ont une structure arborescente, nous représentons un document XML par un arbre enraciné étiqueté ordonné (Francesca et al., 2003), (Aggarwal et al., 2003), (Dalamagas et al., 2004), (Xing et al., 2006), (Knijf 2007) en prenant en considération la structure et le contenu.

Un arbre enraciné étiqueté ordonné T est un n-uplet $T = <n_0, N, A, \delta>$ (Francesca et al., 2003), où :

- n_0 est le nœud racine de l'arbre T.
- $N \subseteq E$ est l'ensemble des nœuds de l'arbre T dont E est l'ensemble d'étiquettes regroupant les noms des balises, les termes du contenu textuel et les attributs. Une étiquette d'un nœud est un élément d'un alphabet fini $\sum$.
- $A \subseteq N \times N$ est l'ensemble des arcs.
- δ est une fonction étiquetée qui assigne à chaque nœud $n \in N$ de l'arbre T une étiquette $e \in \sum$ tel que $\delta(n) = e$.

Nous représentons les nœuds structurels (les balises) par des cercles, les nœuds textuels (contenu) par des rectangles et les nœuds attributs par des triangles (figure 3).

Nous proposons ensuite de réduire chaque arbre enraciné étiqueté ordonné en étendant la notion de résumé structurel utilisée dans les travaux de (Dalamagas et al., 2004), où un arbre représente seulement la structure d'un document XML. Le résumé de l'arbre dans notre cas consiste à construire un arbre réduit en prenant en considération les trois types de nœuds : les nœuds structurels, les nœuds textuels et les nœuds attributs. L'idée est d'éliminer tous les nœuds dupliqués de l'arbre enraciné étiqueté ordonné construit en utilisant l'algorithme *ReduiceTree*. Un nœud dupliqué est un nœud qui possède un chemin dupliqué. Celui-ci doit commencer du nœud racine de l'arbre. Avant de supprimer l'un des deux nœuds dupliqués, il faut vérifier si le nœud à supprimer possède des nœuds fils ; Dans ce cas, il faut les affecter à l'autre nœud (figure 3).

L'algorithme qui permet de réduire un arbre enraciné étiqueté ordonné est donné ci-dessous ; Il est dénommé *ReduiceTree*.

Algorithme *TraitEC*

```
Entrée
    Arbre T
Sortie
    Arbre réduit
Variables
    n0 : nœud racine
    ni , nj : nœud
Début
    Parcourir T
    Si il existe deux nœuds égaux ni et nj Alors
        Vérifier les chemins des deux nœuds jusqu'à la racine n0
        Si les chemins sont égaux Alors
            Si ni est un nœud feuille Alors
                Supprimer ni
            Sinon affecter les fils de ni à nj
                Supprimer ni
            Fin Si
        Fin Si
    Fin Si
Fin
```

Après avoir transformé un arbre XML en un arbre réduit, nous proposons de générer un ensemble des chemins en s'inspirant des travaux de (Vercoustre et al., 2006). L'ensemble des chemins est généré en préservant la structure hiérarchique de l'arbre. Le parcours de l'arbre se fait en pré-ordre[1]. Un ensemble de chemins est de la forme suivante :

$$EC = < c_1, c_2, \ldots, c_n > = < (a/ec_1), (a/ec_2), \ldots, (a/ec_n) >$$

Où *a* est le nœud racine de l'arbre qui est un nœud structurel. Et ec_i est un sous chemin du nœud *a*. Un sous chemin comporte des nœuds structurels, des nœuds textuels notés entre des doubles cottes (" ") et des nœuds attributs préfixés par @ (nous conservons les mêmes notations de (Vercoustre et al., 2006)). Les chemins générés peuvent être de trois types :

- *Chemin structurel* : c'est un chemin qui se termine par un nœud antécédent d'un nœud feuille de l'arbre réduit (nœud structurel).
- *Chemin textuel* : c'est un chemin qui se termine toujours par un nœud feuille de l'arbre réduit (nœud textuel).
- *Chemin attribut* : c'est un chemin qui se termine par un nœud attribut de l'arbre réduit.

[1] Parcourir l'arbre de gauche à droite.

Le fait de réduire l'arbre à l'avance permet d'éviter la génération de chemins identiques lors de la transformation de l'arbre en un ensemble des chemins.

Dans l'ensemble des chemins de la figure 3, (A/B/D) est un chemin structurel, (A/B/D/"E") et (A/B/D/"F") sont deux chemins textuel, et le dernier chemin (A/@C) est un chemin attribut.

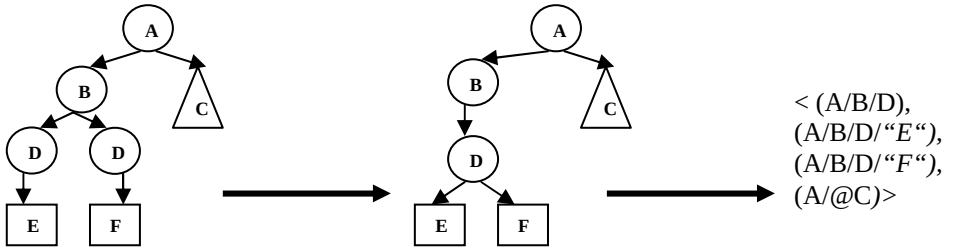

FIG. 3 – Un exemple d'un arbre réduit et son ensemble des chemins.

Dans l'ensemble des chemins, nous remplaçons tous les éléments esclaves par leurs éléments maîtres en utilisant le thésaurus. Pour ce faire, nous appliquons l'algorithme *TraitEC* présenté ci-dessous.

À la fin de cette étape, l'ensemble des chemins de chaque document est mappé dans une matrice booléenne. La matrice des chemins est définie pour calculer la fréquence d'apparition des chemins dans chaque document XML tel que les *m* lignes sont les documents et les *n* colonnes sont les chemins de la collection XML.

$$MC = \begin{bmatrix} & c_1 & \cdots\cdots & c_n \\ d_1 & 1 & \cdots\cdots & 0 \\ \vdots & \vdots & \cdots\cdots & \vdots \\ d_m & 0 & \cdots\cdots & 1 \end{bmatrix}$$

La dernière phase de l'approche consiste à appliquer le *clustering* sur la matrice afin d'identifier des groupes ou des clusters de documents similaires.

Algorithme *TraitEC*

```
Entrées
    Les ensembles de chemins EC de la collection XML, Thésaurus
Sorties
    Les ensembles de chemins traités
Variables
    EC_i : ensemble des chemins
    c_i : chemin
    e_i : élément
Début
    Pour chaque EC_i Faire
        Pour chaque c_i Faire
            Pour chaque e_i de c_i Faire
                Si e_i commence par " Alors
                    Si e_i est un élément esclave dans le sac de contenu Alors
                        Remplacer e_i par son élément maître
                    Fin Si
                Fin Si
                Si e_i ne commence ni par @ ni par " Alors
                    Si e_i est un élément esclave dans le sac de structure Alors
                        Remplacer e_i par son élément maître
                    Fin Si
                Fin Si
            Fin Pour
        Fin Pour
    Fin Pour
Fin
```

3.3 Clustering

Nous cherchons à cibler les groupes homogènes de documents, c'est-à-dire à identifier des groupes tels que les documents possédant les ensembles de chemins les plus similaires appartiennent au même groupe et que les documents possédant les ensembles des chemins les plus différents soient séparés dans différents groupes. La notion de similarité étant le plus souvent ramenée à une fonction de distance entre paires de documents.

Pour appliquer le *clustering*, l'algorithme *k-means* (MacQueen, 1967) est utilisé avec *k* un paramètre défini en entrée qui indique le nombre de clusters désirés. La distance euclidienne est utilisée pour mesurer la distance entre deux documents où *n* est le nombre de chemins. Elle est calculée par la formule suivante :

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_j)^2}$$

Les étapes de *k-means* sont les suivantes :

1. Choisir aléatoirement *k* documents qui formeront l'ensemble des centroïdes initiaux représentant les *k* clusters recherchés.
2. Assigner chaque document au cluster dont le centroïde le plus proche selon la distance *d* (si un minimum de *d* est trouvé entre deux objets).
3. Si aucun document ne change de cluster d'une itération à l'autre alors arrêt et sortir les clusters. Sinon, mettre à jour les centroïdes des clusters en fonction des objets qui leur sont associés.
4. Aller à 2.

Dans la section suivante, nous présentons le prototype d'évaluation, les collections de tests, les mesures d'évaluation et les résultats de l'évaluation expérimentale de l'approche proposée en haut.

4 Etude expérimentale

4.1 Prototype d'évaluation

Pour valider l'approche, un prototype, que nous appelons F-CheX (Fouille dans les Chemins XML), est créé avec le langage Java. F-CheX assure les tâches suivantes (figure 4) :

- Importer la collection de documents XML.
- Générer à partir d'une collection XML le thésaurus.
  - Générer le sac de structure et le sac de contenu.
  - Epurer les deux sacs (suppression des doubles).
  - Enregistrer les deux sacs.
- Générer l'arbre enraciné étiqueté ordonné pour un document XML.
- Générer l'ensemble des chemins pour un document XML.
- Générer à partir d'une collection XML la matrice des chemins.
  - Générer les ensembles de chemins pour toute la collection de documents XML.
  - Traiter les ensembles de chemins (en unifiant les mots avec le thésaurus).
  - Enregistrer les ensembles de chemins.
  - Générer et enregistrer la matrice des chemins.

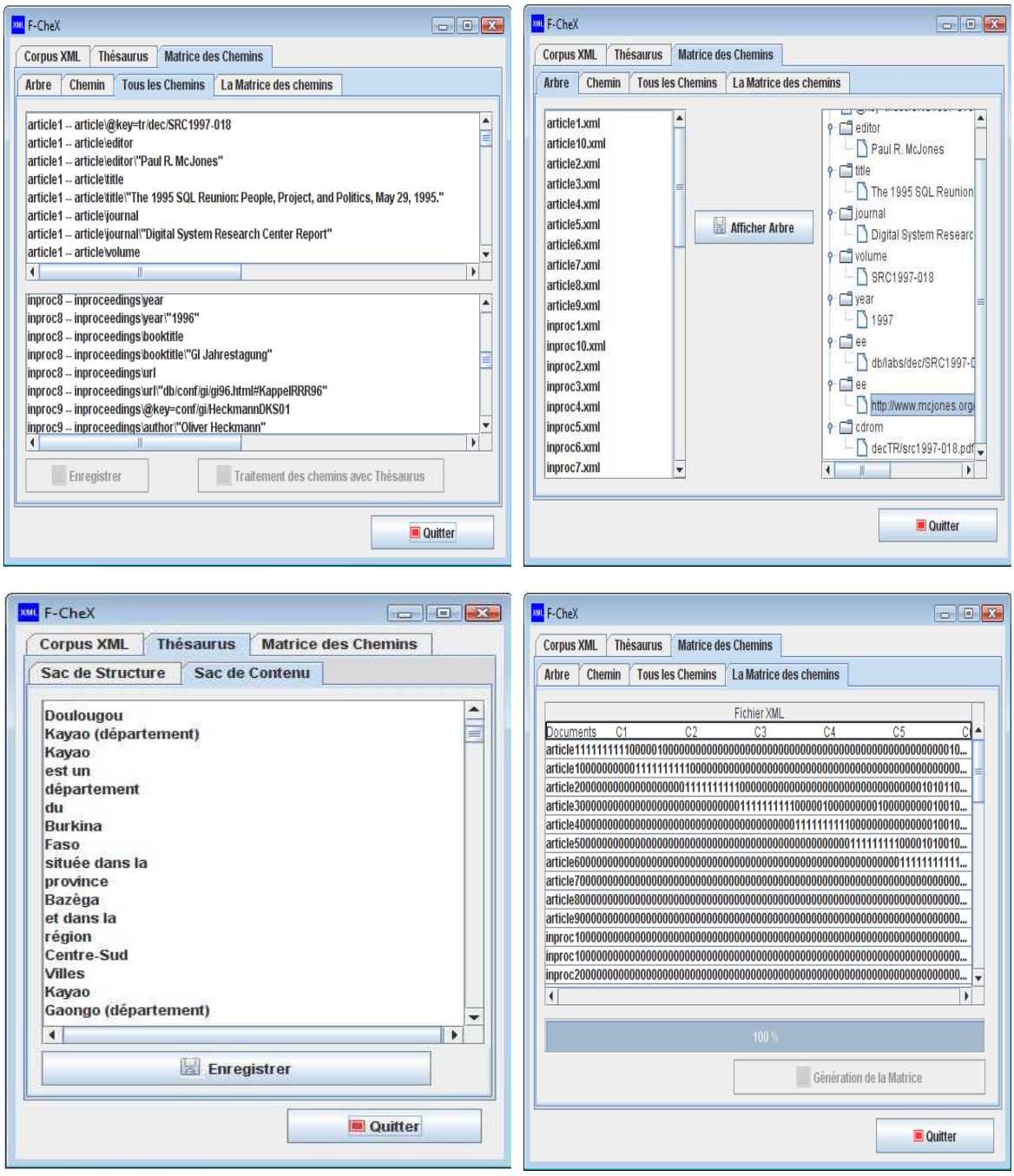

FIG. 4 – Le prototype F-CheX.

Nous utilisons la version 1.4.31 de *Tanagra*[1] (c'est un logiciel gratuit de *data mining* destiné à l'enseignement et à la recherche) pour appliquer l'algorithme de *clustering k-means* sur les matrices des chemins générées à partir des collections XML utilisées pour faire nos tests.

4.2 Collections de tests

Deux collections de documents XML sont utilisées pour l'évaluation de notre approche. La première comporte des documents de *DBLP*[2] *Computer Science Bibliography* contenant les références bibliographiques des publications de conférences scientifiques et de journaux en anglais. La deuxième collection est celle de *XML Wikipédia*[3], qui est en langue Française.

À partir de chaque collection, nous constituons 4 jeux de tests. Nous augmentons graduellement le nombre de documents XML composant les jeux de tests. Les 4 jeux de tests *jeu1*, *jeu2*, *jeu3* et *jeu4* sont composés respectivement de 50, 100, 500, et 1000 documents.

Chaque jeu comporte cinq types de documents XML. Les jeux contiennent un même nombre de documents pour chacun des types. Par, exemple *jeu2* contient 100 documents XML avec 20 documents pour chacun des 5 types.

Pour la première collection de documents XML (DBLP), les 5 types sont : «*articles*», «*inproceedings*», «*phdthesis*», «*proceedings*» et «*www*». Pour la deuxième collection (Wikipédia), les 5 types sont : «*départements*», «*foot*», «*peintres*», «*références*» et «*rivières*».

Puisque les deux collections ne disposent pas d'une ontologie de domaine, nous définissons manuellement les éléments maîtres et esclaves des sacs de mots qui constituent le thésaurus. C'est pour cette raison que nous ne traitons au maximum que 1000 documents XML. La construction manuelle d'un tel thésaurus est assez fastidieuse. Il est possible d'imaginer un moyen pour produire automatiquement un thésaurus d'un grand nombre de documents XML.

4.3 Résultats

Le tableau ci-dessous représente les résultats de 2 jeux de tests (*jeu1* et *jeu4*) des 2 collections, qui contiennent respectivement 50 et 1000 documents XML. Ces résultats désignent le nombre de nœuds, d'éléments du thésaurus construit, de chemins générés, les profondeurs minimale et maximale des chemins générés et enfin le nombre de lignes et de colonnes des deux matrices de chemins. Ces chiffres sont obtenus lors du déploiement de notre approche F-CheX.

Le rappel (*Recall*), la précision (*Precision*) et la F-mesure (*Fscore*) sont utilisés pour évaluer les résultats obtenus. Une précision élevée signifie une justesse élevée du *clustering* alors qu'un rappel faible signifie qu'il y a plusieurs documents XML qui ne sont pas dans le cluster approprié. Une précision et un rappel élevés indiquent que la qualité du *clustering* est excellente (Dalamagas et al., 2004). La F-mesure mesure un équilibre entre la précision et le rappel. Elle calcule la moyenne harmonique des deux valeurs précédentes.

---

[1] http://eric.univ-lyon2.fr/~ricco/tanagra/

[2] DBLP XML records, http://dblp.uni-trier.de/xml/

[3] http://www-connex.lip6.fr/~denoyer/wikipediaXML/

		DBLP		Wikipédia	
		Jeu1	Jeu4	Jeu1	Jeu4
Les nœuds	Nombre total de nœuds (non dupliqués)	366	1549	287	2553
	Nombre de nœuds attributs	51	1001	105	2100
	Nombre de nœuds structurels	23	23	20	22
	Nombre de nœuds textuels	292	525	162	431
Les éléments	Nombre total d'éléments du thésaurus	315	548	182	43
	Nombre d'éléments maîtres	10	25	20	32
	Nombre d'éléments esclaves	28	82	31	41
	Nombre d'éléments neutres	277	441	131	380
Les chemins	Nombre total de chemins	719	14208	1048	20155
	Nombre de chemins structurels	309	6180	281	5620
	Nombre de chemins textuels	350	6828	485	8895
	Nombre de chemins attributs	60	1200	282	5640
	Profondeur minimale d'un chemin	02	02	02	02
	Profondeur maximale d'un chemin	03	03	07	07
La matrice des chemins	Nombre de lignes	50	1000	50	1000
	Nombre de colonnes	402	677	336	814

TAB. 3 – Résultats pour jeu1 et jeu4 des deux collectio ns de test.

En appliquant *k-means* à l'aide du logiciel *Tanagra* sur les deux matrices de chemins calculées pour chaque jeu de tests des deux collections, et en mettant le nombre de clusters *k* à 5, cinq clusters sont construits avec un nombre de documents équivalent à la partition initiale. Comme l'indique le tableau 4, selon les mesu res d'évaluation nous pouvons dire qu'avec notre approche les résultats de *clustering* obtenus sont de haute qualité. Le rappel et la précision ont atteint de bonnes valeurs pour lesdeux collections.

		Nombre de clusters	Rappel	Précision	F-mesure
DBLP	Jeu1	5	1.00	1.00	1.00
	Jeu2	5	1.00	1.00	1.00
	Jeu3	5	1.00	1.00	1.00
	Jeu4	5	1.00	1.00	1.00
Wikipédia	Jeu1	5	1.00	1.00	1.00
	Jeu2	5	0.99	0.99	0.99
	Jeu3	5	0.99	0.99	0.99
	Jeu4	5	0.99	0.99	0.99

TAB. 4 – Les résultats du clustering avec k-means.

Nous constatons aussi que les résultats sont toujours bons avec :

- de petits arbres XML (profondeur = 3) ou avec des arbres larges (profondeur = 7).
- un nombre important de chemins attributs ou avec un petit nombre.
- l'information de structure est plus grande que l'in formation de contenu ou l'inverse.

En termes de temps d'exécution, nous remarquons qu'au-delà de 500 documents XML, le temps de calcul de la matrice devient de plus en plus lent.

5 Conclusion et perspectives

Dans cet article, nous avons proposé une nouvelle approche de fouille dans les documents XML qui permet de prendre en compte la structure et le contenu. L'approche consiste à appliquer le *clustering* sur les chemins générés des documents XML. L'algorithme *k-means* est utilisé avec *k* un paramètre défini arbitrairement en entrée qui indique le nombre de clusters désirés. La distance euclidienne est utilisée pour mesurer la distance entre paires de documents. L'originalité de notre approche réside dans l'utilisation d'un thésaurus créé au préalable pour gérer l'aspect sémantique et unifier tous les mots présents dans les ensembles des chemins générés de la collection XML. Le thésaurus est créé à partir de deux sacs de mots construits de la collection XML : un sac de structure et un sac de contenu.

Avec le prototype F-CheX, nous avons évalué notre approche en utilisant deux collections différentes : *DBLP* en anglais et *Wikipédia* en français. Le *clustering* avec *k-means* est appliqué sur les deux matrices générées par F-CheX avec le logiciel *Tanagra*. De bonnes valeurs des mesures d'évaluation ont été obtenues. Toutefois, il sera intéressant :

- d'améliorer le temps d'exécution et plus précisément le temps de calcul de la matrice des chemins générés à partir d'une collection XML ;
- d'évaluer l'approche sur des collections de tests comportant un nombre plus élevé de documents, et de tester si la performance de l'approche est toujours maintenue quand le nombre de clusters devient de plus en plus grand (passage à l'échelle) ;
- d'un autre coté, il est clair, que dans notre approche, le thésaurus est utilisé pour gérer l'aspect sémantique en faisant appel soit à *WordNet* soit à des experts de domaine. Il serait donc plus judicieux de tester l'approche sur une collection de documents XML possédant déjà une ontologie de domaine ou bien en lui créant une ontologie propre ;
- d'appliquer d'autres méthodes de *data mining* sur les ensembles de chemins est une perspective incontournable. Parmi ces méthodes, nous citons la classification supervisée en générant le pattern des chemins communs pour les documents de chaque cluster prédéfini et en appliquant la classification des documents selon ces patterns.

Références

Aggarwal, C. C., et M. J. Zaki (2003). *XRules: An Effective Structural Classifier for XML Data*, SIGKDD 03, pp. 316–325, ACM Press, New York.

Boussaid, O., A. Duffoux, S. Lallich, et F. Bentayeb (2004). *Fouille dans la structure de documents xml*. EGC 04, Revue des Nouvelles Technologies de l'Information, volume 2, pages 519-524, Clermont-Ferrand, France.

Braga, D., A. Campi, S. Ceri, M. Klemettinen, et PL. Lanzi (2002). *A Tool for Extracting XML Association Rules from XML Documents*, Research paper in Proceedings of IEEE-ICTAI 2002, pp. 57-64, Washington DC, USA.

Candillier, L., I. Tellier, et F. Torre (2005). *Transforming XML trees for efficient classification and clustering*. In Proceedings of the 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, pp. 469-480, INEX'05.

Dalamagas T., T. Cheng, K. Winkel, et T. Sellis (2004). *Clustering XML Documents using Structural Summaries*, In Proc. of ClustWeb - International Workshop on Clustering Information over the Web in conjunction with EDBT 04, pp. 547–556, Crete, Greece.

Denoyer, L., et P. Gallinari (2004). *Bayesian Network Model for Semi-Structured Document Classification*. Revue Information Processing & Management, Special Issue on Bayesian Networks and Information Retrieval, Pages 807-827, Elsevier.

Doucet, A., et M. Lehtonen (2006). *Unsupervised classification of text-centric XML document collections*, In: Workshop of the INitiative for the Evaluation of XML Retrieval.

Fellbaum, C. (1988). *WordNet : An Electronic Lexical Database*, MIT Press.

Francesca, D. F., G. Gordano, R. Ortale, et A. Tagarelli (2003). *Distance-based Clustering of XML Documents*, ECML'03 and PKDD'03 Cavtat-Dubrovnik, Croatia.

Garboni, C., F. Masseglia , B. Trousse (2006). *Sequential pattern mining for structure based XML document classification*. In: INEX 2005 Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 458–468.

Hagenbuchner, M., F. Trentini, A. Sperduti, A. Tsoi, F. Scarselli, et M. Gori (2005). *Clustering XML Documents using Self-Organizing Maps for Structures*. INEX 2005 Workshop on Mining XML documents, pp. 432–442.

Knijf, J. De. (2007). *FAT-CAT: Frequent Attributes Tree Based Classification*, In Fuhr, N., Lalmas, M., and Trotman, A., editors, Comparative Evaluation of XML Information Retrieval Systems, INEX 2006, pages 485–496.

Lian, W., et D. W. Cheung (2004). *An Efficient and Scalable Algorithm for Clustering XML Documents by Structure*, in IEEE Transactions on Knowledge and Data Engineering, pp. 82-96.

MacQueen, J.B. (1967). *Some methods for classification and analysis of multivariate observations*, In Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability, volume I : Statistics, pages 281–297.

Madani, A., Boussaid, O., Abed, H., et S. Loudcher (2009). *Fouille dans les documents XML : Etat de l'art*, Quatrième Atelier des Systèmes Décisionels (ASD 2009), Jijel, Algérie, pages 53-65.

Nayak, R., et S. Xu (2005). XML *documents clustering by structures with XCLS*. In: Workshop of the INitiative for the Evaluation of XML Retrieval INEX, pp. 432–442.

Nayak, R., et W. Iryadi (2007). *XML schema clustering with semantic and hierarchical similarity measures*, Knowledge-Based Systems, Volume 20, Issue 4, Pages 336-349.

Safar, B., Reynaud, C., et F.E. Calvier, (2007). *Techniques d'alignement d'ontologies basées sur la structure d'une ressource complémentaire*, 1ères Journées Francophones sur les Ontologies, JFO'2007, Sousse, Tunisie.

Termier, A., M. C. Rousset, et M. Sebag (2002). *TreeFinder : a First Step towards XML Data Mining*. International Conference on Data Mining ICDM'02, Maebashi, pp. 450-457, Japon.

Vercoustre, A.-M., M. FEGAS, S. Gul, et Y. Lechevallier (2006). *A flexible structured-based representation for XML document mining*, INEX'05, Schloss Dagstuhl, Germany, LNCS (3977), pp. 443-457, Springer.

Xing, G., et Z. Xia (2006). *Classifying XML documents based on structure/content similarity*, In: Workshop of the INitiative for the Evaluation of XML Retrieval, pp. 444-457, Springer.

Yang, J., et C. Xiaoou (2002). *A semi-structured document model for text mining*, Journal of Computer Science and Technology archive, Volume 17(5), 603-610.

Summary

We present in this article a new approach of XML mining combining the two categories: XML structure mining and XML content mining. The approach consists in applying the clustering on the XML documents, these last are represented by a set of paths keeping the tree structure of the elements composing these documents. The sets of paths are mapped into a matrix of paths on which the clustering with k-means is applied. The approach use a thesaurus created beforehand to manage the semantic of the words. We evaluate our approach by experimentation on two XML documents collections.