Apprentissage de conversion de documents semi-structurés à partir d'exemples.

Francis Maes, Ludovic Denoyer, Patrick Gallinari

prénom.nom@lip6.fr
Laboratoire d'Informatique de Paris 6 (LIP6), Université Pierre et Marie Curie

RÉSUMÉ. *Une majorité de travaux de Recherche d'Information dans les collections de documents semi-structurés se focalise sur le traitement de bases homogènes et ne sont pas utilisables sur des corpus de documents hétérogènes issus du Web par exemple. Nous présentons ici la méthode ISM (Incremental Structure Mapping) permettant la conversion de documents XML issus de sources hétérogènes dans un schéma de médiation. ISM est centrée document et permet la prise en compte simultanée de la structure et du contenu des documents. Elle ne nécessite pas de spécifier des correspondances entre schéma manuellement et utilise des méthodes d'apprentissage automatique, l'utilisateur n'ayant qu'à fournir au système un ensemble de documents exprimés conjointement dans leur schéma initial et dans le schéma de destination. Contrairement aux méthodes existantes, ISM possède une complexité très faible et permet de traiter de grands corpus de documents. Les résultats des expériences sur différents corpus montrent que l'algorithme est capable d'apprendre des transformations complexes, notamment pour la tâche de conversion du format HTML vers un format XML sémantiquement riche.*

ABSTRACT. *We propose here the method called ISM -Incremental Structure Mapping- which allows one to convert XML documents from heterogeneous sources to a mediated schema. Unlike existing methods, ISM is document centric and takes into account both the structural information and the content information. It does not need to specify manually correspondences between schema and is based on Machine Learning methods in order to transform documents to the mediated schema. The method learns the transformation using a set of documents expressed in both the input schema and the output schema. At last, ISM has a low complexity and can be used with very large XML collections. This methods is experienced here on a set of corpus and different tasks. The results show that the algorithm is able to learn complex transformations and to transform large corpora, particularly for the conversion of HTML documents to semantically rich XML documents.*

MOTS-CLÉS : *Apprentissage Structuré, XML, Transformation de Documents*

KEYWORDS: *Structured Output Classification, XML, Structure Mapping*

1. Introduction

Les données sémantiquement riches comme les documents textuels ou multimédia sont aujourd'hui représentées à l'aide de formats semi-structurés. Les éléments y sont organisés en fonction d'une structure qui reflète les relations logiques, sémantiques ou syntaxiques entre les différentes parties du document. Le format XML, et dans une moindre mesure le format HTML, permettent d'identifier les éléments d'un document (comme le titre ou les sections) et de décrire les relations entre ces éléments ; par exemple, nous pouvons connaître l'auteur ou la date d'une partie du document. De plus, des informations additionnelles tel que les meta-données ou les annotations sont souvent ajoutées au contenu afin d'en enrichir la description.

Pour un grand nombre d'applications, les données semi-structurées proviennent de sources hétérogènes qui ne partagent pas le même schéma. La gestion de cette hétérogénéité est un problème majeur pour l'exploitation de données semi-structurées qui a déjà été adressé depuis quelques années dans le domaine des Bases de Données (BD) notamment avec l'émergence de bases de données XML et de systèmes pair-à-pair. Plus récemment, ce problème est apparu dans le contexte du Traitement Automatique de Documents ainsi que dans celui de la Recherche d'Information (RI).

Nous considérons ici le problème de la conversion automatique de documents (*Structure Mapping*) qui consiste à apprendre à transformer des documents semi-structurés exprimés dans des formats hétérogènes vers un schéma de médiation prédéfini. Nous étudions ici cette problématique avec une approche *centrée documents* contrairement à d'autres travaux (notamment issus de la communauté BD) qui abordent ce problème avec une approche *centrée données*. Cela signifie que, contrairement aux approches centrées données, la sémantique du document est importante et que le processus de conversion doit prendre en considération simultanément une information de contenu et une information de structure.

La définition de correspondances entre schémas est un problème majeur pour le développement d'applications exploitant les documents semi-structurés. Ces correspondances sont habituellement définies manuellement en utilisant des langages complexes comme XSLT par exemple. Bien que des outils soient développés et existent pour aider les utilisateurs dans l'écriture de programmes de conversion, ce processus n'en demeure pas moins complexe et nécessite des compétences d'expert. Ce n'est donc pas adapté à des situations où les sources sont nombreuses et changent fréquemment. De plus, les schémas pour les grandes collections sont souvent pauvres et ne spécifient que très peu de contraintes sur les documents valides[1]. Dans ce cas, le schéma lui-même ne fournit pas assez d'informations et l'écriture de feuilles XSLT pour la transformation de document est très difficile, voir impossible, et très coûteuse en terme de temps. Finalement, beaucoup de sources, particulièrement sur le Web, ne possèdent même pas de schéma ou n'en permettent pas l'accès. La définition automatique de telles transformations est devenue rapidement un défi important. Plusieurs ap-

1. c'est le cas pour le corpus Wikipedia XML par exemple (Denoyer *et al.,* 2006)

proches ont été explorées comme l'utilisation de méthodes syntaxiques, les méthodes de transformations de grammaires, les transducteurs d'arbres ou des méthodes statistiques. La plupart d'entre elles reposent fortement sur des heuristiques spécifiques. Les approches existantes pour la transformation de documents sont donc limitées à un seul type de transformation ou à un type de documents. Ces méthodes ne considèrent souvent que la structure des documents et n'exploitent pas le contenu des noeuds. De plus, cette information de structure est sous-exploitée et seules quelques relations sont prises en compte par les modèles. Enfin, la plupart des méthodes ne passent pas à l'échelle et ne permettent pas le traitement d'une grande masse de documents.

Dans cet article, nous proposons le système ISM (*Incremental Structure Mapping*) qui permet la conversion automatique de documents semi-structurés dans un schéma de médiation. Cette conversion n'est pas spécifiée manuellement mais elle est apprise à partir d'exemples à l'aide de méthodes d'apprentissage automatique. Au lieu d'écrire des scripts de transformation, l'utilisateur va fournir au système un ensemble de documents d'entrées et leurs conversions respectives dans le schéma de médiation. Le système va alors apprendre automatiquement à convertir de nouveaux documents dans le schéma de médiation. Les documents peuvent provenir de sources structurées hétérogènes ou être des documents plats (sans structure). Cela nous permet de considérer des problèmes où le schéma d'entrée n'est pas explicite ou trop général. Il est important de noter que la conversion de documents vers un schéma de médiation est une tâche générique avec différentes applications dans différents domaines. C'est aussi un problème clef pour l'enrichissement sémantique de documents HTML dans les sources du Web 2.0 comme les forums, les blogs, les wiki, *etc*.

Le papier est organisé comme suit : l'état de l'art est décrit dans la partie 2. On donne ensuite les idées générales du modèle proposé dans la partie 3. Le modèle est ensuite décrit en détail dans la partie 4. Finalement, de nombreux résultats expérimentaux sont donnés dans la partie 5.

2. Etat de l'art

Plusieurs approches pour automatiser la transformation de documents ont été proposées. La plupart d'entre elles ne considèrent que la structure des documents et n'exploitent pas leur contenu. Ces méthodes proviennent de trois principaux domaines : la RI, la communauté document électronique et la communauté BD.

2.1. *Les modèles de RI et de gestion de documents électroniques*

La transformation de documents semi-structurés est une tâche relativement nouvelle dans la communauté document. Les travaux dans ce domaine ne considèrent que les transformations de schéma et requièrent donc que le schéma d'entrée soit explicite. L'information de contenu est ignorée. Cette classe de méthodes est donc restreinte à des collections précises au schéma connu comme les bases bibliographiques par

exemple. Elles ne s'appliquent pas ou difficilement aux très grands corpus. Leinonen propose par exemple une approche syntaxique basée sur des transducteurs d'arbres à états finis (Leinonen, 2003). Le système génère automatiquement des correspondances entre schémas si l'utilisateur fournit manuellement des correspondances entre les feuilles des documents. Su et al. dans (Su *et al.*, 2001) proposent un algorithme de mise en correspondance d'arbres basé sur la décomposition de la correspondance en une séquence d'opérations élémentaires. Cet algorithme utilise une heuristique permettant l'exploration de l'espace des actions. Boukotaya et al. dans (Boukottaya *et al.*, 2005) proposent aussi un algorithme ad-hoc qui combine plusieurs critères (relations entre les différentes étiquettes des noeuds, compatibilité entre les types des données, similarités entre les chemins, *etc*.) La tâche d'annotation de documents qui consiste à transformer des documents exprimés dans un format de rendu (HTML, PDF, *etc*.) vers un format sémantiquement riche représenté par un schéma XML a conduit au développement de différentes méthodes. Yip Chung et al. dans (Chung *et al.*, 2002) considèrent la tâche de conversion HTML vers XML. Ils utilisent des méthodes d'apprentissage non-supervisé et définissent manuellement des règles afin de découvrir des régularités dans la structure d'arbre des documents d'entrée. Plus proche de notre problématique, le modèle de Chidlovskii et al. dans (Chidlovskii *et al.*, 2005) que nous avons utilisé comme modèle de référence dans cet article, considère la conversion de documents HTML ou PDF vers un schéma XML. Ils utilisent des classifieurs et des grammaires probabilistes afin d'étiqueter les différents éléments d'un document d'entrée afin d'obtenir le document désiré. Cependant, la complexité de ces méthodes limite leur utilisation à des corpus de petite taille (comme le corpus Shakespeare présenté ici).

Récemment, les auteurs de (Gilleron *et al.*, 2006) on également proposé une méthode pour convertir des documents XML. Leurs méthodes est basé sur le formalisme des champs conditionnels aléatoires.

2.2. *Les modèles de Bases de Données : Le Schema Matching*

Dans la communauté des BD, l'intégration automatique ou semi-automatique de données hétérogènes connue sous le nom de *Schema Matching* a été un sujet de recherche actif pendant plusieurs années et de nombreux travaux ont été proposés. Un état de l'art exhaustif peut être trouvé dans (Rahm *et al.*, 2001) et (Shvaiko *et al.*, 2005). Nous résumons brièvement trois familles de modèles :

– Les modèles basés sur le schéma s'intéressent uniquement à la mise en correspondance de schémas et ignorent le contenu des documents (Palopoli *et al.*, 1998),(Castano *et al.*, 2001).

– Certains modèles considèrent simultanément l'information de schéma et la signification des différents éléments (les étiquettes) du schéma. Ils utilisent cette information additionnelle pour en déduire des correspondances (Doan *et al.*, 2001),(Li *et al.*, 2000). Certaines de ces approches utilisent des méthodes simples d'apprentissage statistique.

– Enfin, les modèles de fusion utilisent différents sous-modèles de mise en correspondance et agrège les résultats obtenus par ces sous-modèles afin de calculer la correspondance entre deux schémas (Doan *et al.*, 2001),(Embley *et al.*, 2001). Certains de ces travaux ont explorés l'utilisation de méthodes d'apprentissage automatique pour l'inférence de correspondances. Une série de travaux ont été développés par Halevy, Doan et leurs collègues. Dans (Doan *et al.*, 2003), les auteurs proposent une méthodologie qui combine différentes sources d'évidence comme les étiquettes, les types des données, les structures locales, *etc*. à l'aide d'un modèle de régression appris sur un ensemble d'apprentissage. Cette méthode à été utilisée sur différentes instances de problèmes et nous l'avons utilisée comme modèle de référence pour nos expériences sur le corpus *RealEstate*.

Les travaux sur le *Schema Matching* ont été développés pour différentes applications spécifiques et il n'existe pas aujourd'hui de *benchmark* permettant une réelle comparaison des méthodes existantes.

3. Idée générale du modèle ISM

Nous introduisons dans cette partie les principales idées qui ont menées au modèle ISM (*Incremental Structure Mapping*).

3.1. *Apprentissage structuré*

Nous nous plaçons ici dans le cadre de la classification (ou prédiction) structurée. Le but de la prédiction structurée est d'apprendre une fonction qui associe des entrées $\mathbf{x} \in \mathcal{X}$ à des sorties $\mathbf{y} \in \mathcal{Y}_\mathbf{x}$, où les entrées et les sorties sont des objets munis d'une structure interne. De tel objets peuvent par exemple être des chaînes de symboles, des arbres ou encore des graphes étiquetés. $\mathcal{X}$ est l'espace de toutes les entrées possibles et $\mathcal{Y}_\mathbf{x}$ est l'espace de toutes les sorties possibles, pour une entrée donnée $\mathbf{x}$. On note $\mathcal{Y} = \cup_{\mathbf{x}\in\mathcal{X}}\mathcal{Y}_\mathbf{x}$ l'ensemble de toutes les sorties possibles. Dans notre cas, les entrées x et les sorties y seront des arbres XML. Dans ce cadre, on considère que l'on dispose d'un ensemble d'exemples d'apprentissage $D = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i\in[1,n]}$ ainsi que d'une fonction permettant de mesurer la qualité d'une solution prédite notée $\Delta(\hat{\mathbf{y}}, \mathbf{y}^*)$ où $\hat{\mathbf{y}}$ est la sortie prédite et $\mathbf{y}^*$ la sortie désirée. Ces modèles sont usuellement paramétrés par un vecteur de paramètres $\theta \in \mathcal{R}^d$ et on note f_θ la fonction de prédiction correspondant aux paramètres θ. Ces paramètres seront appris sur la base d'apprentissage.

Un certain nombre de méthodes génériques d'apprentissage ont été développées dans ce contexte. Ces méthodes définissent une fonction $F(\mathbf{x}, \mathbf{y}; \theta)$ qui mesure la compatibilité entre une entrée et une sortie donnée. Muni d'une telle fonction, le problème d'inférence devient un problème de recherche de la sortie ayant la meilleure compatibilité avec l'entrée :

$$f_\theta(\mathbf{x}) = \underset{\mathbf{y}\in\mathcal{Y}_\mathbf{x}}{\operatorname{argmax}} F(\mathbf{x}, \mathbf{y}; \theta)$$

L'utilisation et l'apprentissage d'une telle fonction de compatibilité est à la base des modèles les plus récents de prédiction structurée comme par exemple SVM-ISO (Tsochantaridis *et al.,* 2004) et M^3N (Taskar *et al.,* 2003). Cependant, ces méthodes partagent un certain nombre de limites. La plus importante concerne l'inférence qui demande de parcourir l'espace des solutions pour trouver la meilleure d'entre elles. Intuitivement, le nombre de sorties possibles est la plupart du temps exponentiel en fonction de la taille des données considérée. Par exemple, le nombre d'arbres XML recouvrant un certain contenu et respectant un certain schéma est en général exponentiel en fonction du nombre de noeuds du document. Ce problème de l'inférence est généralement traité à l'aide d'algorithmes de programmation dynamique. Cependant, la complexité de ces algorithmes peut être trop élevée pour permettre un vrai passage à l'échelle. C'est le cas par exemple, avec les méthodes d'inférence d'arbres qui ont, pour la plupart, une complexité cubique en fonction du nombre de feuilles. Ceci n'est pas acceptable dans le cas d'arbres XML contenant des milliers de feuilles.

Afin d'éviter ce problème de complexité, le modèle ISM repose sur l'idée qui consiste à directement apprendre le processus d'inférence. Au lieu de modéliser *à quoi ressemble une bonne solution* puis de chercher cette solution, nous proposons plutôt de modéliser *comment construire la bonne solution* directement. En cela, nous adoptons une approche similaire à celles des récents algorithmes LaSO (Daumé III *et al.,* 2005) et Searn (Daumé III *et al.,* 2006).

3.2. *Cadre formel : les Processus de Décision Markoviens*

Notre méthode repose sur la modélisation du processus de construction d'arbres XML. Pour cela, elle repose sur le formalisme des *Processus de Décision Markoviens* (PDM) qui permet de modéliser des problèmes de prise de décisions séquentielles. Un PDM est défini par un ensemble d'états, un ensemble d'actions, une fonction de transition et une fonction de récompense. On appelle *agent*, l'entité qui évolue dans un PDM. A tout instant, un agent se trouve dans un état du PDM (un élément de l'ensemble d'états du PDM). Le but de l'agent est alors de choisir une action parmi un ensemble d'actions possibles. Une fois cette action exécutée, la fonction de transition définit le nouvel état dans lequel se trouve l'agent et celui-ci perçoit éventuellement une récompense. Le but d'un agent dans un PDM est de maximiser la somme des récompenses perçues. Pour cela, il doit optimiser sa procédure de sélection d'actions. Les domaines de l'apprentissage par renforcement (Sutton *et al.,* 1998) et de la programmation dynamique approchée (J. Si *et al.,* 2004) fournissent plusieurs algorithmes pour apprendre une telle procédure de sélection d'actions. Dans nos travaux, nous modélisons le processus de construction d'un arbre XML grâce à un PDM. L'état initial du PDM correspond à l'arbre XML vide. Chaque action consiste à rajouter un noeud au document XML en fonction du document d'entrée jusqu'à l'obtention d'une solution complète. Le PDM est donc dirigé par l'arbre d'entrée : il y a une étape de conversion par feuille du document d'entrée. Les états du PDM contiennent le document d'entrée, la feuille courante de ce document et l'arbre en cours de construction.

Chaque étape de construction consiste à déterminer la position de la feuille courante dans l'arbre de sortie. Pour cela, le PDM inclut une action par chemin possible dans l'arbre de sortie. Un chemin dans l'arbre de sortie est une séquence d'étiquettes et de positions. Une position spécifie soit un nœud existant, soit elle spécifie un emplacement pour insérer un nouveau nœud. L'ensemble des chemins possibles est construit à partir des documents de sortie de la base d'apprentissage. Un chemin est possible si il apparaît au moins une fois dans cet ensemble. On considère également une action `SKIP` qui permet de ne pas construire de nouveaux noeuds dans l'arbre de sortie. Cela permet donc de supprimer des feuilles de l'arbre d'entrée. Nos agents sélectionnent les actions sur la base d'une fonction de *score d'action*. Une telle fonction donne un score à chaque action possible, l'idée étant ensuite de sélectionner l'action ayant le score le plus élevé. La figure 1 illustre un exemple de la façon dont un document HTML est converti en XML.

4. Le modèle ISM

Dans cette partie, nous décrivons étape par étape notre méthode de conversion de documents semi-structurés. Cette méthode repose sur le Processus de Décision Markovien décrit plus haut. On commence par décrire l'algorithme d'inférence qui correspond à l'exécution séquentielle d'un ensemble d'actions. Les actions seront choisies en fonction d'une fonction de score décrite ensuite. Enfin, nous détaillerons l'espace de représentation qui correspond aux descriptions vectorielles des états et des actions utilisées par la fonction de score. Finalement, nous donnerons l'algorithme d'apprentissage de l'agent qui repose sur des méthodes connues dans le domaine de l'apprentissage par renforcement.

4.1. *Algorithme d'Inférence*

L'inférence permet de prédire un arbre XML de sortie $\mathbf{y} \in \mathcal{Y}$ étant donné l'arbre XML d'entrée $\mathbf{x} \in \mathcal{X}$. Le document d'entrée est vu comme une séquence de feuilles $\mathbf{x} = (x_1, \ldots, x_n)$. Chaque feuille a un contenu textuel et des informations de contexte, tel que les étiquettes des parents de cette feuille. On note $\mathcal{F}_X$ l'ensemble de toutes les feuilles possible. En plus du document d'entrée, l'algorithme 1 prend en paramètre une fonction score d'action. $fonctionScore(x_i, a, \mathbf{y})$ évalue la *qualité* de l'action a concernant la feuille x_i étant donnée l'arbre partiel $\mathbf{y}$. L'algorithme d'inférence commence avec un arbre de sortie vide (ligne 1). Il effectue ensuite une itération par feuille du document d'entrée. Pour chaque feuille, l'algorithme énumère l'ensemble des actions de constructions candidates (ligne 3). Il s'agit ici d'un ensemble des chemins possibles dans l'arbre de sortie. L'algorithme utilise ensuite la $fonctionScore$ pour déterminer la meilleure action, c'est à dire l'action ayant le score le plus élevé. Si cette meilleure action consiste à ne pas prendre en compte la feuille courante (action `SKIP`), alors on passe à la feuille suivante (ligne 6). Sinon, on ajoute le contenu de la feuille dans l'arbre partiel $\mathbf{y}$ (ligne 8). Une fois que toutes les feuilles x_i ont été

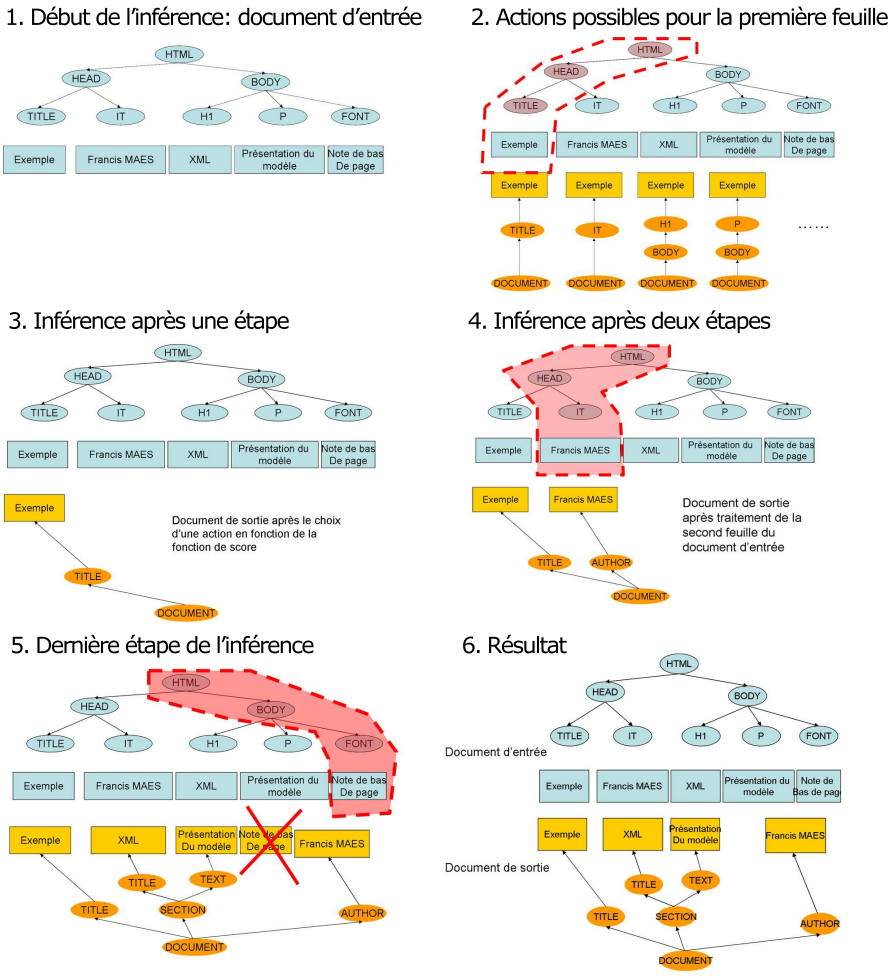

Figure 1. *Idée générale de la méthode ISM. 1) Au début de l'inférence, on commence avec un arbre de sortie vide. 2) Première étape : on se concentre sur la première feuille d'entrée (dont le texte est « exemple »). Toutes les actions possibles pour cette feuille sont énumérées. L'algorithme se base sur une fonction score d'action qui donne un score à chaque action possible. 3) Fin de la première étape : l'élément « exemple » a été rajouté dans le document en cours de construction, en choisissant l'action de meilleure score. 4) Fin de la deuxième étape : la deuxième feuille (« Francis Maes ») vient d'être rajoutée à l'arbre en cours de construction. 5) La dernière étape illustre l'action* `SKIP` *qui permet de ne pas garder certains éléments de contenu. 6) État final du processus de construction (après 5 étapes), la conversion complète est finie et l'arbre prédit (en jaune) peut être renvoyé à l'utilisateur.*

Algorithm 1 Inférence d'un arbre XML

$\mathbf{x} \in \mathcal{X}$: le document d'entrée
$fonctionScore : \mathcal{F}_X \times \mathcal{A} \times \mathcal{Y} \rightarrow \Re$: la fonction de score d'action

```
1: y ← ϵ
2: for x_i ∈ x do
3:     actions ← ensembleDesActionsCandidates(x_i,y)
4:     meilleureAction ← argmax_{a∈actions} fonctionScore(x_i, a, y)
5:     if meilleureAction = SKIP then
6:         continuer (avec la prochaine feuille d'entrée)
7:     else
8:         y ← ajouterNoeud(x_i, meilleurAction, y)
9:     end if
10: end for
11: return y
```

traitées, on peut retourner l'arbre construit $\mathbf{y}$. Le résultat de l'algorithme d'inférence dépend entièrement de la $fonctionScore$ utilisée. Cette fonction est apprise durant la phase d'apprentissage sur l'ensemble des couples de documents d'entrée et de sortie de la base d'apprentissage.

4.2. *Fonction Score d'Action*

Pour calculer le score d'une action, nous décrivons tout d'abord l'action sous forme vectorielle. Cela est réalisé grâce à une fonction de description $\phi : \mathbf{F}_X \times \mathcal{A} \times \mathbf{y} \rightarrow \Re^d$. On utilise $\phi(x_i, a, \mathbf{y})$ pour décrire l'action a concernant la feuille x_i sachant l'arbre partiel de sortie $\mathbf{y}$. Les détails de la fonction de description vectorielle sont donnés dans la partie suivante. Une fois que l'action est décrite sous forme d'un vecteur, on obtient le score en réalisant un produit scalaire de la description avec un vecteur de paramètres $\theta \in \Re^d$:

$$fonctionScore(x_i, a, \mathbf{y}) = \sum_{f=1}^{d} \theta_f \phi(x_i, a, \mathbf{y})_f$$

Notre fonction de score est donc paramétrée par le vecteur θ. L'apprentissage consiste à régler θ de manière à obtenir une fonction de score permettant d'effectuer les bonnes actions.

Description Vectorielle La fonction d'action vue plus haut se base sur une représentation vectorielle des actions. Cette représentation est constituée d'un ensemble de caractéristiques décrivant conjointement un aspect de l'action (le chemin de destination de la feuille courante) et un aspect de l'état (la feuille courante). Nous utilisons

Descripteur	Valeur
Nous sommes sur la première feuille d'entrée ET le chemin de sortie a les étiquettes *DOCUMENT TITLE*.	1
La feuille d'entrée à l'étiquette *IT* ET le chemin de sortie a les étiquettes *DOCUMENT AUTHOR*.	1
La feuille d'entrée est à hauteur de 3 ET le chemin de sortie insère le noeud entre un *TITLE* et une *SECTION*.	0
Le dernier mot de la feuille d'entrée est "footnote" ET l'action est `SKIP`.	1
Le dernier mot de la feuille d'entrée est un symbole de ponctuation ET le chemin de sortie a les étiquettes *DOCUMENT SECTION TEXT*.	0
...	...

Figure 2. *Quelques exemples de descripteurs de la fonction* $\phi(x_i, a, \mathbf{y})$ *En fonction du corpus considéré, on obtient entre* 10^3 *et* 10^6 *descripteurs distincts.*

un ensemble de *template* de caractéristiques pour faciliter la définition de la représentation vectorielle. Voici deux exemples de telles *templates* :

– *étiquette de la feuille courante* ET *étiquettes du chemin de sortie considéré*

– *premier mot du texte de la feuille courante* ET *étiquettes du chemin de sortie considéré*

Chaque *template* définit un ensemble de caractéristiques possible, dont la figure 2 donne quelques exemples. La prise en compte du contexte dans les caractéristiques permet ainsi de faire la distinction entre les différents chemins d'insertions possible.

4.3. *Algorithme d'apprentissage*

L'algorithme 2 donne la méthode d'apprentissage du vecteur de poids θ. L'algorithme prend deux paramètres : la base d'apprentissage D constituée d'un ensemble de couples de documents d'entrée et de sortie, et la fonction coût Δ mesurant une distance entre les documents de sortie prédits et les documents de sortie désirés. L'apprentissage est réalisé en modifiant les paramètres θ de manière à minimiser l'espérance du coût entre les sortie prédites et les sorties désirées. Pour cela, l'algorithme procède de manière itérative. Chaque itération commence par le choix d'un couple de documents d'apprentissage (ligne 3). Les paramètres θ sont ensuite utilisés pour faire l'inférence à partir du document d'entrée. On obtient ainsi un document de sortie prédit (ligne 4). Cela permet d'évaluer la distance entre le document prédit et la sortie correcte (ligne 5). L'algorithme effectue ensuite une étape d'apprentissage, en revisitant toutes les feuilles du document d'entrée (ligne 6-8). Cette étape d'apprentissage modifie légèrement les poids θ qui sont intervenus pendant l'inférence. Les détails de la procédure d'apprentissage sont omis ici pour la clarté. Dans nos expériences,

Algorithm 2 Apprentissage du processus de conversion XML

$D = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i \in [1,n]}$: Base d'apprentissage
$\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$: Fonction de coût

$\theta \leftarrow \mathbf{0}$
repeat
  $\mathbf{x}, \mathbf{y} \leftarrow$ choisirUnExample(D)
  $\bar{\mathbf{y}} \leftarrow$ Inférence($\mathbf{x}$, $fonctionScore_\theta$)
  coût $\leftarrow \Delta(\bar{\mathbf{y}}, \mathbf{y})$
  for $x_i \in \mathbf{x}$ **do**
    $\theta \leftarrow$ appliquerCorrection(θ, x_i, coût)
  end for
until convergence du vecteur θ
return θ

nous utilisons l'algorithme SARSA(0)-approché qui est un algorithme classique du domaine de l'apprentissage par renforcement (Sutton *et al.,* 1998). Plus de détail sur cette méthode peuvent être trouvé dans (Maes *et al.,* 2007).

5. Expériences

Nous avons décrit le contexte et le principe de la méthode ISM. Nous proposons ici un ensemble d'expériences afin de mesurer la qualité de ce modèle pour la tâche de conversion vers un schéma de médiation. Les expériences ont été réalisé à l'aide de la librairie d'apprentissage *Nieme*, disponible sur le site *http ://nieme.lip6.fr*.

5.1. *Jeux de données*

Afin de valider notre méthode dans un maximum de situations différentes, nous avons utilisé de nombreux corpus différents. Chaque corpus propose des documents exprimés dans différent schémas et issus de différentes sources. Nos expériences ont consisté à apprendre automatiquement les conversions entre schémas et à mesurer leur qualités.

– `Films` est un corpus composé de plus de 13 000 descriptions de films dans trois versions : la version HTML du site Allociné[2], la version HTML du site IMDB[3] et une version XML dans un format médiateur définit par nos soins. Ceci correspond à un scénario où deux sites Web différents doivent être convertis vers un format médiateur prédéfini.

2. http ://www.allocine.com
3. http ://www.imdb.com

Corpus	Tâche	Taille	Nœuds internes	Feuilles	Etiquettes
Films	XML → XHTML XHTML → XML et XHTML → XHTML	13,038	≈ 60	≈ 40	40
INEX IEEE	Plat → XML	12,017	≈ 200	≈ 500	139
Wikipedia	XHTML → XML	12,000	≈ 40	≈ 160	256
RealEstate	XML → XML	2,367	≈ 15	≈ 19	37
Shakespeare	Plat → XML	60	≈ 20	≈ 85	7

Figure 3. *Description des corpus utilisés dans nos expériences de conversion automatique. De gauche à droite : le nom du corpus, le type de tâche auquel il correspond, le nombre de documents dans le corpus, le nombre moyen de nœuds internes par document, le nombre moyen de nœuds feuilles par document et le nombre d'étiquettes XML distinctes.*

– `INEX IEEE` est un corpus XML proposé dans le cadre de la recherche d'information structurée (Fuhr *et al.*, 2002). Ce corpus est composé de 12 017 articles scientifiques dans un format XML. Chaque document provient d'un journal (18 journaux différents). Les documents sont donnés dans deux formats : une version segmentée plate et une version en XML. Le but est d'apprendre à prédire la structure XML, en ne connaissant que la segmentation du texte plat.

– Le corpus `Wikipedia` est constitué de 12 000 documents exprimé dans deux formats : le HTML original tel qu'il est disponible sur le web[4] et une version XML spécifique au wiki-texte issu du corpus Wikipedia XML (Denoyer *et al.*, 2006). Pour limiter le nombre d'action de ISM sur corpus, nous n'avons gardé que les chemins qui apparaissaient plus de 5 fois en apprentissage.

– Le corpus `RealEstate`, proposé par Anhai Doan[5], se compose de 2 367 documents XML orienté données. Les données sont exprimées dans un format XML d'entrée et un format XML médiateur de sortie. Les documents de ce corpus contiennent peu de texte et ont une structure très régulière.

– Pour comparer notre approche à un modèle de base, nous avons aussi fait des expériences sur le corpus `Shakespeare`[6]. Comme dans (Chidlovskii *et al.*, 2005), nous avons aléatoirement choisi 60 scènes de la collection. Nous nous comparons au modèle de (Chidlovskii *et al.*, 2005) qui est basé sur les grammaires hors contexte probabilistes et les classificateurs de maximum d'entropie (PCFG+ME). En raison de sa complexité (cubique en fonction du nombre de feuilles d'entrées, la où notre approche est linéaire), le modèle de (Chidlovskii *et al.*, 2005) est inapplicable aux autres corpus.

4. http ://wikipedia.org
5. http ://www.cs.wisc.edu/ anhai/
6. http ://metalab.unc.edu/bosak/xml/eg/shaks200.zip

La figure 3 synthétise les propriétés de nos corpus. Dans toutes nos expériences, nous utilisons 50% des données en apprentissage et 50% des données en test.

5.2. *Mesures d'évaluation*

Il est très difficile d'évaluer la tâche de conversion automatique dans l'absolu car celle-ci est destinée à être utilisé par une application cible. Pour bien évaluer la qualité d'un document prédit par notre système, il faudrait voir la perte que l'application cible subit en remplaçant les documents corrects par les documents prédits. Cela demandant une étude approfondie et dépendante d'une application particulière, nous avons plutôt fait le choix d'utiliser des distances générales entre arbres. Nous avons utilisé trois mesures : $F_{contenu}$, F_{chemin} et $F_{structure}$, qui reflètent respectivement la qualité de l'étiquetage des feuilles d'entrée, la qualité de la structure interne prédite et la qualité des actions choisi par l'agent de construction XML. Ces trois mesures sont la moyenne d'une score F_1 calculé pour tous les couples (document prédit, document correcte). Elles sont donc toutes trois comprises dans l'intervalle $[0, 1]$, où deux arbres identiques obtiennent des scores de 1. Pour le score $F_{contenu}$, nous calculons le score F_1 entre les étiquettes des feuilles. C'est une mesure équivalente au *Word Error Ratio* utilisé en traitement du langage naturel. La mesure F_{chemin} est directement liée au fonctionnement de la méthode ISM. Elle correspond au score F_1 entre tous les chemins de l'arbre prédit et tous les chemins de l'arbre correct. Il y a un chemin par feuille et deux chemins sont identiques si ils couvrent le même texte avec la même séquence d'étiquette de la racine à la feuille. La fonction de coût $\Delta(\bar{\mathbf{y}}, \mathbf{y})$ utilisée en apprentissage est dérivée de la mesure F_{chemin} par la relation $\Delta(\bar{\mathbf{y}}, \mathbf{y}) = 1 - F_{chemin}(\bar{\mathbf{y}}, \mathbf{y})$. La mesure $F_{structure}$ est basée sur le score F_1 entre tous les sous-arbres. La similarité $F_{structure}$ entre deux documents XML est calculée de la manière suivante :

1) Construire l'ensemble de tous les sous-arbres de chacun des deux documents. Il y a un sous-arbre par nœud du document.

2) Calculer le rappel et la précision sur ces sous-arbres. Deux sous-arbres sont identiques si ils ont la même étiquette, les mêmes fils et le même texte.

3) Calculer les score F1 : $F1 = \frac{2 * Rappel * Precision}{Rappel + Precision}$.

La mesure $F_{structure}$ à la particularité de décroître très rapidement avec peu d'erreurs. Par exemple, sur nos corpus, une seule erreur d'étiquetage sur une feuille amène typiquement à score $F_{structure}$ de $\approx 80\%$.

5.3. *Résultats*

La figure 1 donne les résultats obtenus sur les différents corpus. Tous les scores $F_{contenu}$ sont supérieurs à 75% et le score $F_{structure}$ – plus difficile – reste au dessus de $\approx 60\%$. Ces résultats doivent être contrastés avec la difficulté intrinsèque de la tâche de conversion automatique. Par exemple pour le corpus INEX IEEE, les seuls indices pour prédire une classe parmi plus de 100 sont donnés par le texte. Le tableau

Corpus	Méthode	$F_{contenu}$	F_{chemin}	$F_{structure}$
INEX IEEE Plat → XML	ISM	75.8 %	74.4 %	67.5 %
Films XML → Allocine	ISM	86.4 %	86.4 %	52.4 %
Films XML → IMDB	ISM	92.1 %	92.0 %	63.2 %
Allociné → Films XML	ISM	80.3 %	76.8 %	65.8 %
Allociné → IMDB	ISM	86.5 %	82.8 %	58.9 %
IMDB → Films XML	ISM	75.4 %	75.4 %	57.0 %
IMDB → Allocine	ISM	73.3 %	71.5 %	48.2 %
Films mélangés → Films XML	ISM	79.2 %	77.8 %	64.5 %
Wikipedia HTML → XML	ISM	80.2 %	74.3 %	65.6 %
RealEstate XML 1 → XML 2	ISM	99.9 %	99.9 %	99.9 %
RealEstate XML 1 → XML 2	PCFG+ME	99.9 %	7 %	49.8 %
RealEstate XML 1 → XML 2	LSD (Doan *et al.*, 2003)	-	*80 %*	-
Shakespeare Plat → XML	ISM	94.4 %	93.2 %	87.5 %
Shakespeare Plat → XML	PCFG+ME	98.7 %	97	94.7 %

Tableau 1. *Ce tableau synthétise les résultats de nos expériences de conversion automatique. La méthode proposé est la méthode ISM. La méthode de référence PCFG+ME ne fonctionne que sur les corpus RealEstate et Shakespeare à cause de sa complexité prohibitive.*

Corpus	Apprentissage	Inférence
INEX IEEE	$\approx$ 2 jours	$\approx$ 2 s / doc
Films	$\approx$ 20 min	< 100 ms / doc
Shakespeare	$\approx$ 20 min	$\approx$ 0.02 s / doc

Tableau 2. *Temps approximatifs d'apprentissage et d'inférence avec notre méthode sur un ordinateur standard de 3.2Ghz.*

2 montre les temps d'apprentissage et d'inférence de notre approche. On peut voir que la plupart des documents sont traités en moins d'une seconde. Ceci signifie que ISM pourrait être déployé sur des corpus grande échelle contenant des milliers ou des millions de documents.

Traitement de l'hétérogénéité Le corpus Films est constitué de descriptions de films en trois formats différents. Pour illustrer la capacité de ISM à traiter cette hétérogénéité, nous avons appliqué la méthode aux 6 conversions possibles entre formats. On peux voir que certaines conversions sont plus facilement apprises que d'autre (scores $F_{contenu}$ entre 73.3 % et 92.1 %). De plus, nous avons imaginé le scénario où la source (IMDB ou Allociné) est inconnue. Pour cela, nous avons constitué la base Films mé-

langés qui est un mélange aléatoire de films issus d'IMDB et de Allociné. On peux voir que les performances de conversions dans ce cas sont similaires au cas où les sources sont connues.

Comparaison avec l'existant La base de données de Shakespeare montre une comparaison entre notre approche et une méthode de référence appelé PCFG+ME (voir plus haut). Nous obtenons des scores moins bon que ceux du modèle de référence. Une première explication est que PCFG+ME fait une optimisation globale en utilisant la programmation dynamique. Ceci doit être contrasté avec notre approche qui construit incrémentalement la solution. Nous suspectons également notre méthode de souffrir du sur-apprentissage sur les corpus de petites tailles. Néanmoins, notre approche est environ mille fois plus rapide en inférence que la méthode de référence. De plus, en raison de sa complexité, ceci ne peut pas être appliqué à nos corpus du monde réel. Nous pensons que dans le contexte de la recherche d'informations hétérogènes, le temps d'inférence est un point beaucoup plus important que la perfection de la structure reconstruite.

Capacité à changer l'ordre des nœuds La conversion du corpus RealEstate nécessite de changer l'ordre des feuilles du document. La méthode PCFG+ME ne permet pas de faire cela, ce qui explique les score très faible obtenus pour F_{chemin} et $F_{structure}$. L'ordre des nœuds ne pose pas de problème à la méthode ISM.

6. Conclusion

Nous avons présenté la méthode ISM qui permet l'apprentissage de transformations complexes et permet de convertir des documents semi-structurés issus de sources hétérogènes vers un schéma de médiation unique. Cette méthode est apprise à partir de couples de documents exprimés simultanément dans leur schéma d'entrée et le schéma de médiation et ne nécessite pas de spécifier de mises en correspondances entre schéma manuellement. Enfin, ISM, qui est basée sur une méthode incrémentale d'inférence approchée, permet le traitement de très grands corpus de documents contrairement aux méthodes actuelles qui souffrent d'une complexité très élevée. Les résultats sur différents corpus montrent que cette méthode est capable d'apprendre différentes classes de transformations sur différents documents et différents formats (HTML, XML ou documents plats). En plus de pouvoir servir pour la conversion de documents, cet algorithme peut être utilisé pour convertir *a priori* un ensemble de documents hétérogènes dans un schéma de médiation pour ensuite utiliser une méthode de RI dans les documents structurés. Nous envisageons à l'avenir d'évaluer la qualité des transformations effectuées indirectement sur différentes tâches de RI et notament sur la tâche de recherche documentaire.

7. Bibliographie

Boukottaya A., Vanoirbeek C., « Schema matching for transforming structured documents. », *ACM DOCENG*, p. 101-110, 2005.

Castano S., Antonellis V. D., di Vimercati S. D. C., « Global Viewing of Heterogeneous Data Sources. », *IEEE Trans. Knowl. Data Eng.*, vol. 13, p. 277-297, 2001.

Chidlovskii B., Fuselier J., « A probabilistic learning method for XML annotation of documents », *IJCAI*, 2005.

Chung C. Y., Gertz M., Sundaresan N., « Reverse Engineering for Web Data : From Visual to Semantic Structure. », *ICDE*, p. 53-63, 2002.

Daumé III H., Langford J., Marcu D., « Search-based Structured Prediction », 2006.

Daumé III H., Marcu D., « Learning as Search Optimization : Approximate Large Margin Methods for Structured Prediction », *ICML*, ACM Press, Bonn, Germany, 2005.

Denoyer L., Gallinari P., « The Wikipedia XML Corpus », *SIGIR Forum*, 2006.

Doan A., Domingos P., Halevy A., « Learning to Match the Schemas of Data Sources : A Multistrategy Approach », *Maching Learning*, vol. 50, p. 279-301, 2003.

Doan A., Domingos P., Halevy A. Y., « Reconciling Schemas of Disparate Data Sources : A Machine-Learning Approach. », *SIGMOD Conference*, p. 509-520, 2001.

Embley D. W., Jackman D., Xu L., « Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. », *Workshop on Information Integration on the Web*, p. 110-117, 2001.

Fuhr N., Govert N., Kazai G., Lalmas M., « INEX : Initiative for the Evaluation of XML Retrieval », *SIGIR 2002 Workshop on XML and IR*, 2002.

Gilleron R., Jousse F., Tellier I., Tommasi M., « XML Document Transformation with Conditional Random Fields », *INEX 2006*, number 4518, http ://www.springerlink.com/, 2006.

J. Si A. G. Barto P. W. B., II D. W., *Handbook of Learning and Approximate Dynamic Programming*, Wiley&Sons, INC., Publications, 2004.

Leinonen P., « Automating XML document structure transformations. », *ACM DOCENG*, p. 26-28, 2003.

Li W.-S., Clifton C., « SEMINT : A tool for identifying attribute correspondences in heterogeneous databases using neural networks. », *Data Knowl. Eng.*, vol. , p. 49-84, 2000.

Maes F., Denoyer L., Gallinari P., « Sequence Labelling with Reinforcement Learning and Ranking Algorithms », *ECML*, Varsaw, Poland, 2007.

Palopoli L., Saccà D., Ursino D., « Semi-Automatic Semantic Discovery of Properties from Database Schemas. », *IDEAS*, p. 244-253, 1998.

Rahm E., Bernstein P. A., « A survey of approaches to automatic schema matching. », *VLDB J.*, vol. , p. 334-350, 2001.

Shvaiko P., Euzenat J., « A Survey of Schema-Based Matching Approaches. », , vol. , p. 146-171, 2005.

Su H., Kuno H. A., Rundensteiner E. A., « Automating the transformation of XML documents. », *WIDM*, p. 68-75, 2001.

Sutton R., Barto A., *Reinforcement learning : an introduction*, MIT Press, 1998.

Taskar B., Guestrin C., Koller D., « Max-Margin Markov Networks. », *NIPS*, 2003.

Tsochantaridis I., Hofmann T., Joachims T., Altun Y., « Support vector machine learning for interdependent and structured output spaces », *ICML*, ACM Press, New York, NY, USA, 2004.