Extraction de relations dans les documents Web

Rémi Gilleron * , Patrick Marty * , Marc Tommasi * , Fabien Torre*

* Projet Mostrare Inria Futurs & Université de Charles de Gaulle - Lille III
59653 Villeneuve d'Ascq CEDEX FRANCE
prenom.nom@univ-lille3.fr

Résumé. Nous présentons un système pour l'inférence de programmes d'extraction de relations dans les documents Web. Il utilise les vues textuelle et structurelle sur les documents. L'extraction des relations est incrémentale et utilise des méthodes de composition et d'enrichissement. Nous montrons que notre système est capable d'extraire des relations pour les organisations existantes dans les documents Web (listes, tables, tables tournées, tables croisées).

1 Introduction

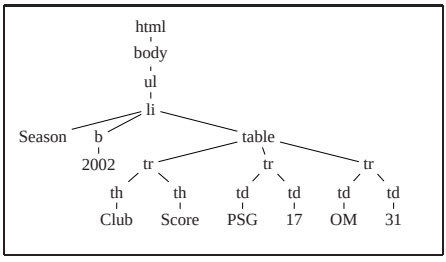
Le développement d'Internet comme source d'informations a conduit à l'élaboration de programmes nommés *wrappers* pour collecter de l'information sur les sites Web. Ces programmes sont difficiles à concevoir et à maintenir. Deux approches sont alors envisageables : la première consiste à assister l'utilisateur, c'est le cas du système Lixto (Baumgartner et al., 2001) dans lequel on spécifie le wrapper dans un langage logique avec l'aide d'un environnement visuel ; la seconde consiste à générer automatiquement le wrapper en limitant l'intervention de l'utilisateur à l'annotation des informations à extraire sur quelques documents. Cette approche est fondée sur le fait que la plupart des documents sur Internet sont générés par programme et présentent des régularités exploitables par les méthodes d'apprentissage automatique.

Les premiers systèmes d'induction de wrappers n'utilisaient que l'aspect textuel des documents (Hsu et Dung, 1998; Kushmerick, 1997). Avec l'apparition de XML, ces approches textuelles sont devenues insuffisantes. Les systèmes actuels utilisent la structure arborescente des documents du Web (Carme et al., 2005; Cohen et al., 2003; Kosala et al., 2002; Muslea et al., 2003; Thomas, 2003). Nous nous inscrivons dans cette démarche en proposant un système d'induction qui utilise à la fois les vues textuelle et arborescente. Beaucoup de systèmes d'induction de wrappers sont conçus pour des tâches unaires. Un wrapper unaire extrait un ensemble de valeurs, par exemple l'ensemble des noms de produits disponibles sur un site marchand. Un wrapper n-aire extrait les instances d'une relation n-aire, par exemple les couples (nom du produit, prix). Il existe deux approches pour induire un wrapper n-aire : soit combiner n wrappers unaires, soit apprendre directement le wrapper n-aire. La première approche nécessite l'obtention d'un modèle pour la combinaison, ou une intervention de la part de l'utilisateur (Jensen et Cohen, 2001; Muslea et al., 2003), ou encore l'utilisation d'heuristiques. La seconde approche est illustrée par les systèmes WIEN (Kushmerick, 1997) et SOFT MEALY (Hsu et Dung, 1998) utilisant des délimiteurs textuels pour repérer les composantes des tuples et le système LIPX (Thomas, 2003) basé sur la logique du premier ordre.

Nous proposons un système d'induction de wrappers n-aire pour les documents du Web utilisant les vues textuelle et arborescente. Pour cela, nous étudions à la section 2 les différentes représentations arborescentes de tables dans les documents Web. Nous proposons ensuite un système basé sur les deux principes suivants : l'extraction est incrémentale, c'est-à-dire que le système extrait l'ensemble des premières composantes, puis l'ensemble des couples pour les deux premières composantes, jusqu'à l'ensemble des n-uples ; pour extraire l'ensemble des tuples de longueur i, des informations sur les tuples de longueur $i-1$ sont utilisées. Les algorithmes d'extraction et d'induction sont présentés dans la section 3. Le système est évalué sur des jeux de données réels dans la section 4. Les résultats montrent que notre système peut appréhender l'extraction de relations dans les documents Web pour des organisations fréquentes que les systèmes existants ne sont pas capables de traiter.

2 Représentations arborescentes d'une relation n-aire

Nous considérons que les données sont stockées dans des documents au format `XML` ou `XHTML` qui peuvent être considérés selon plusieurs vues. La *vue `DOM`* considère le document comme un arbre (arbre d'analyse) ; la *vue séquentielle* comme un flux de caractères ; la *vue feuillage* comme la séquence des feuilles textes de la vue `DOM`. Ces différentes vues sur les données sont illustrées par la Figure 1.



<html> <body>   Season 2002 <table> <tr> <th> Club </th> <th> Score </th> </tr> <tr> <td> PSG </td> <td> 17 </td> </tr> <tr> <td> OM </td> <td> 31 </td> </tr> </table>   </body></html>

Season 2002 Club Score PSG 17 OM 31

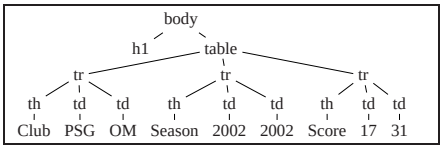
FIG. 1 – *Vue `DOM` (gauche), vue textuelle (en haut à droite), vue feuillage (en bas à droite).*

Une étude des documents du Web nous a amené à distinguer les cas suivants :

Cas 1. Les données sont dans une table dont la première ligne contient les noms des composantes et les lignes suivantes contiennent les données. Dans la vue `DOM`, il existe un plus petit sous arbre contenant chaque tuple et seulement lui (Figure 1).

Cas 2. Une autre représentation est celle d'une liste où les tuples sont stockés séquentiellement. Pour les vues textuelle et arborescente, il peut être difficile de retrouver les tuples sans information auxiliaire surtout dans le cas de valeurs manquantes.

Cas 3. Les données sont dans une table tournée dont la première colonne contient les noms des composantes et les colonnes suivantes contiennent les données. Les tuples sont entrelacés dans les vues textuelle et feuillage. Dans la vue `DOM`, les composantes d'un même tuple ont le même numéro de fils dans des arbres différents de racine `tr`. Ceci est illustré par la relation ternaire (Season, Club name, Score) de la figure 2 où les tuples à extraire sont `(2002,PSG,17)` et `(2002,OM,31)`.

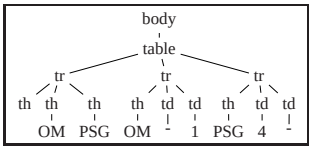


Club	PSG	OM
Season	2002	2002
Score	17	31

FIG. 2 – *Table tournée*

Cas 4. Pour éviter les répétitions, on peut avoir recours à des tables à pivots et de manière générale à des valeurs factorisées. Par exemple dans la Figure 1, la valeur `2002` apparaît seulement une fois mais est partagée par plusieurs tuples : `(2002,PSG,17)` et `(2002,OM,31)`.

Cas 5. Les tables croisées permettent de représenter des relations où les composantes sont presque toutes dépendantes entre elles. Cela correspond aux tables de distances ville à ville, aux tables de conversions de devises, mais aussi au cas des tableaux générés à partir de tableurs ou d'outils de reporting. Cette organisation cumule la complexité des deux cas précédents (voir figure 3).



	OM	**PSG**
OM	-	1
PSG	4	-

FIG. 3 – *Table croisée*

3 Apprendre à extraire une relation n-aire

Les processus d'extraction et d'induction sont incrémentaux : on extrait les premières composantes, puis les tuples de longueur 2, jusqu'aux tuples de longueur n pour une relation n-aire cible. Pour la première composante appelée *graine*, la tâche d'extraction considérée consiste à extraire certaines feuilles textes d'un document arborescent (nous ne considérons pas les cas où les données à extraire sont situées dans une feuille texte ou sur plusieurs feuilles textes). Nous utilisons un codage attribut-valeur des feuilles : le **codage de base d'un noeud** p est fourni par les attributs suivants : label, position dans la séquence des fils du père de p, profondeur et hauteur, nombre de fils, taille du sous arbre enraciné en p, label du frère gauche et du frère droit de p, valeurs éventuelles des attributs `XHTML class` et `id`. La **représentation d'une feuille** l est donnée par l'application du codage de base à l, à ses 5 ancêtres dans la vue `DOM`, puis par le contenu brut des feuilles textes précédente et suivante dans la vue feuillage. Ainsi une feuille est représentée par 57 attributs. Une fois codées en vecteurs d'attributs, les feuilles sont fournies à un classifieur qui classe chacune d'entre elles. Une feuille est extraite si elle est classée comme à extraire.

Considérons maintenant une relation cible n-aire et sa restriction aux i premières composantes. Le codage rep_i de la relation d'arité i va utiliser des informations sur les composantes 1 à $i-1$, il est définie par : la **représentation d'un tuple** d'arité i se fait par le codage de base de la i-ième composante notée l et de la description des dépendances entre l et la graine

Algorithm 1 Extraction

Input: un document d ; n classifieurs c_i à valeurs dans $\{-1, +1\}$ travaillant sur les représentations rep_i.
Notation: L est l'ensemble des feuilles de d, S_i est l'ensemble des tuples candidats à l'étape i, T_i est l'ensemble des tuples extraits à l'étape i sélectionnés dans S_i.
1: $S_1 = \{(l) \mid l \in L\}$; $T_1 = \{t_1 \in S_1 \mid c_1(rep_1(t_1)) = +1\}$
2: **for** $i = 2$ **to** n **do**
3: $S_i = \{t_i \mid t_i = (t_{i-1}, l),\ t_{i-1} \in T_{i-1},\ l \in L\}$
4: $\forall t_i \in S_i$, si $c_i(rep_i(t_i)) = +1$, ajouter t_i à T_i
5: si tous les t_i basés sur le même t_{i-1} sont classés comme négatif par c_i, ajouter $(t_{i-1}, \texttt{null})$ à T_i
6: **end for**
Output: T_n l'ensemble des tuples n-aire extraits

ainsi qu'entre l et la composante $i - 1$. Le codage de la dépendance entre deux feuilles p et m est constituée du codage de leur plus petit ancêtre commun a, des longueurs des plus courts chemins dans la vue DOM entre p et a, m et a et entre p et m, du nombre de feuilles textes se trouvant entre p et m dans la vue feuillage, et pour $1 \leq k \leq 5$ de la différence entre la position (relative à leur père) du k-ième ancêtre de p et du k-ième ancêtre de m. Un tuple est représenté par 220 attributs quel que soit i différent de 1.

L'ordre des composantes étant déterminé par la relation cible fournie par l'utilisateur du système, l'extraction est réalisée par l'algorithme 1. On peut noter qu'il est possible d'extraire plusieurs tuples de longueur i pour un même tuple de longueur $i-1$ (ligne 4) et que les valeurs manquantes sont gérées en ligne 5.

Du point de vue de l'apprentissage, la tâche d'induction d'un programme d'extraction n-aire est définie par l'algorithme 2 et correspond à n problèmes de classification supervisée, chaque problème consistant à classer des tuples d'arité i comme étant à extraire ou pas. L'algorithme de classification supervisée utilisé est C5.0 (Quinlan, 2004). Pour apprendre le classifieur c_i, il faut fixer l'ensemble d'apprentissage (ligne 4). Les exemples positifs (ensemble S_i^+) sont la projection des tuples annotés (ensemble S^+) sur les i premières composantes. Lors de la première itération, est considéré comme exemple négatif (ensemble S_i^+) toute feuille qui n'est pas sélectionnée comme première composante d'un tuple annoté comme à extraire. Pour les étapes suivantes, l'ensemble S_i^- des exemples négatifs est construit à partir de S_{i-1}^+ en ajoutant une feuille l à $t_{i-1} \in S_{i-1}^+$ tel que le tuple obtenu n'est pas la projection d'un tuple annoté sur les i premières composantes.

4 Expériences

Toutes les expériences sont réalisées avec deux documents annotés en apprentissage. La qualité de notre système est évaluée à travers la f-mesure (F) des wrappers produits. Les critères d'évaluation sont stricts : un tuple extrait est correct si toutes les composantes sont égales exactement aux composantes du tuple à extraire.

Des expériences ont été réalisées sur les jeux de données classiques du domaine, en l'occurrence RISE[1]. Les organisations correspondent aux deux premiers cas de la section 2. Notre système réussit parfaitement sur tous les jeux de données à l'exception de S1 et IAF. Sur S1,

[1] http://www.isi.edu/info-agents/RISE/index.html

Algorithm 2 Apprentissage

Input: un échantillon S de documents où les tuples à extraire sont annotés.

1: $S^+ = \{t = (l_1, \ldots, l_n)\}$ l'ensemble des n-uplets de feuilles à extraire dans S.
2: $C = \emptyset$
3: **for** $i = 1$ **to** n **do**
4: $\quad S_i^+ = \{t_{|\{1,\ldots,i\}} \mid t \in S^+\}$ # projection des n-uplets à extraire sur les i premières composantes
5: $\quad S_i^- = \{t' = (t_{i-1}, l) \mid t_{i-1} \in S_{i-1}^+, l \in L$, et $t' \notin S_i^+\}$ # sélection des négatifs
6: $\quad T_i^+ = rep_i(S_i^+)$ et $T_i^- = rep_i(S_i^-)$
7: $\quad$ soit c_i le classifieur appris par C5.0 avec l'échantillon $T_i^+ \cup T_i^-$
8: $\quad$ ajouter c_i à la séquence C de classifieurs
9: **end for**

Output: C

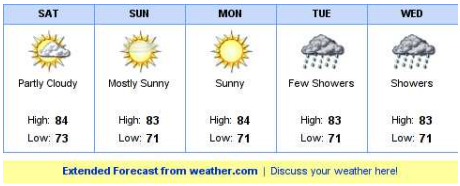


FIG. 4 – *Une page du site* EXCITE WEATHER *: un cas de table tournée.*

nous souffrons de ne pouvoir utiliser la partie textuelle des feuilles ($F = 88.61$). Sur IAF, nous obtenons $F = 64.22$ qui améliore les performances des systèmes existants.

Nous avons complété nos expériences par une évaluation du système sur des corpus[2] construits à partir des résultats du championnat français de football. Ces corpus contiennent les différentes organisations présentées en section 2. Les expériences montrent que notre système est capable d'appréhender toutes ces organisations.

Enfin, nous évaluons le système sur des sites présents sur le Web en visant les organisations les plus difficiles. Tout d'abord le site EXCITE WEATHER [3] et la relation (*town*, *day*, *weather*, *high*, *low*). La composante *town* est factorisée, elle apparaît dans l'entête de page. Les autres composantes sont présentées dans une table tournée à 5 colonnes comme le montre la figure 4. Nous obtenons $F = 100$, ce qui montre la capacité de notre système à gérer ce type d'organisation. Ensuite le site *Bureau of Labor Statistics*[4] et la relation (*value*, *year*, *quantile*). Il s'agit ici d'un exemple d'organisation en table croisée. Nous obtenons $F = 98.52$ à cause de la présence de valeurs manquantes dans certaines tables.

5 Conclusion

Nous avons présenté un système capable d'induire automatiquement des programmes d'extraction n-aire à partir de documents du Web. Ce système présente les avantages suivants :

[2] http://www.grappa.univ-lille3.fr/~marty/corpus.html
[3] www.weather.excite.com
[4] www.bls.gov

l'extraction des composantes et la construction des tuples sont réalisées simultanément ; apprentissage et extraction sont rapides et peu de documents annotés sont nécessaires pour générer un wrapper efficace ; notre système gère des organisations complexes mais fréquentes, telles que les tables croisées, qui sont hors de portée des systèmes existants. Nous travaillons à l'amélioration de ce système selon les directions suivantes : prendre en compte le contenu textuel des feuilles, apprendre à partir de documents partiellement annotés. Une perspective concerne le Web caché pour extraire derrière des formulaires.

Références

Baumgartner, R., S. Flesca, et G. Gottlob (2001). Visual web information extraction with lixto. In *28th International Conference on Very Large Data Bases*, pp. 119–128.

Carme, J., R. Gilleron, A. Lemay, et J. Niehren (2005). Interactive learning of node selecting tree transducer. In *IJCAI Workshop on Grammatical Inference*.

Cohen, W., M. Hurst, et L. Jensen (2003). *Web Document Analysis : Challenges and Opportunities*, Chapter A Flexible Learning System for Wrapping Tables and Lists in HTML Documents. World Scientific.

Hsu, C.-N. et M.-T. Dung (1998). Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems 23*(8), 521 – 538.

Jensen, L. S. et W. Cohen (2001). Grouping extracted fields. In *Proceedings of IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*.

Kosala, R., J. V. D. Bussche, M. Bruynooghe, et H. Blockeel (2002). Information extraction in structured documents using tree automata induction. In *6th International Conference Principles of Data Mining and Knowledge Discovery*, pp. 299 – 310.

Kushmerick, N. (1997). *Wrapper Induction for Information Extraction*. Ph. D. thesis, University of Washington.

Muslea, I., S. Minton, et C. Knoblock (2003). Active learning with strong and weak views : a case study on wrapper induction. In *IJCAI 2003*, pp. 415–420.

Quinlan, R. (2004). Data mining tools see5 and c5.0. http ://www.rulequest.com/see5-info.html.

Thomas, B. (2003). Bottom-up learning of logic programs for information extraction from hypertext documents. In Springer-Verlag (Ed.), *In proceedings of ECML/PKDD 2003*.

Summary

We study how n-ary relations are encoded in tree structured documents like XML or XHTML. Then we present a wrapper induction system for n-ary relations. The extraction process is iterative. At a given step i, the $i-th$ wrapper extracts the $i-th$ component with the knowledge about the $i-1$ first components. Experimental results show that our system outperforms existing ones for many tree representations schemes in the case of factorized values and in the case of missing values.