Extraction de données à partir de pages HTML par création semi-automatique de règles XSLT

N.Georgiev[1], J.M.Labat [2], J.L.Minel [3], L.Nicolas[4]

[1] CRIP5, Nielsen // NetRatings
Nikolay.Georgiev@netvalue.fr
[2] CRIP5, Université Paris 5
Jean-Marc.Labat@math-info.univ-paris5.fr
[3] LaLICC, UMR 8139 CNRS - Université Paris 4
jean-luc.minel@paris4.sorbonne.fr
[4] Nielsen // NetRatings
67, rue Anatole France
92 300 Levallois-Perret, France
Laurent.Nicolas@netvalue.fr

Résumé : L'extraction des données à partir des sources web suscite un intérêt particulier ces dernières années. Cependant il n'existe aucun standard, car les sources d'information Web restent très hétérogènes. Il y a quand même un point commun – elles sont toutes disponibles en format HTML pour être visualisées dans le navigateur client. Cet article présente une méthodologie et les outils associés d'extraction de données à partir de documents HTML. Notre approche est basée sur des technologies XML pour effectuer l'extraction des données, notamment XHTML et XSLT. Afin de valider notre méthodologie, nous avons créé une application nommée XPFE WRAPPER, un générateur semi-automatique de «wrappers[1]» qui a été développé sur la plate-forme XPFE de Mozilla [1].

Mots-clés : Wrappers, Données Semi-structurées, Extraction de Données, Transformations XSL, XSLT, XML ;

1 Introduction

Si l'information disponible sur Internet augmente de manière exponentielle, elle reste aujourd'hui largement inexploitable par les moyens informatiques en raison de sa nature textuelle. Certes, à moyen terme, la migration vers le format XML [19] devrait permettre un accès plus facile car ce langage rend les documents non seulement lisibles mais aussi compréhensibles et exploitables par les machines. De même, à plus long terme, les recherches portant sur le web sémantique [2] sont également très prometteuses.

[1] Le Wrapper présente une procédure conçue pour extraire des contenus à partir d'une source d'information spécifique. Dans le contexte Web un wrapper convertit l'information implicitement stockée dans des documents HTML vers une information explicitement stockée d'une manière structurée.

Il n'en reste pas moins qu'à l'heure actuelle, nous avons à faire à deux types de sites : les premiers présentent une énorme quantité d'informations non exploitables automatiquement, faute de technologies adaptées (*pages web statiques*), les seconds même développés en XML, n'interagissent avec l'utilisateur que par des pages HTML dynamiques (*le serveur assurant la transformation XML vers HTML avant d'envoyer les pages*). Ils sont de plus inaccessibles aux systèmes d'indexation et de recherche (*voir la notion de « Deep Web »* [3]).
La problématique "*Comment accéder à l'information disponible sous forme de pages HTML ?*" garde toute sa pertinence, au moins à court terme, et représente un enjeu énorme pour les années à venir. La conception et le développement de systèmes d'extraction de données reste donc une nécessité forte, en particulier pour des entreprises telles que NetRatings dont l'objet est d'évaluer finement le comportement des internautes à partir d'un panel de plusieurs milliers de personnes, ce qui représente des centaines de milliers de pages visitées à analyser.

2 Contexte de travail

Les travaux présentés dans ce texte, réalisés au sein de la société Nielsen // NetRatings, font partie d'un projet global d'analyse de contenu de pages web. Ce premier travail porte sur l'analyse des achats effectifs sur les sites de e-commerce. En l'occurrence, notre but est d'élaborer des techniques d'extraction des données à partir des pages web. Par "*données*", nous entendons, les informations dynamiques liées à un utilisateur particulier (*i.e. son nom, le numéro d'identification et le montant de la transaction,…*). Une caractéristique importante des sites de e-commerce est qu'ils sont riches en données et restreints en contenu ontologique [15,16]. Nous n'avons pas à traiter des documents contenant des expressions langagières syntaxiquement correctes (*par ex. des phrases entières*), mais plutôt de l'information présentée en style télégraphique. Souvent ce sont des données sous forme de tableaux ou de listes donnant des informations sur le(s) produit(s), les paniers d'achats, les transactions électroniques, etc.

Par ailleurs, il est fondamental de pouvoir obtenir des résultats avec un taux de confiance important. Donc la procédure d'extraction doit être à la fois fiable, car les résultats fournis par NetRatings ne doivent pas être contestables, et robuste, car la structure des pages web est susceptible d'évoluer souvent (*par exemple, par l'introduction d'une publicité*). Voilà pourquoi l'algorithme doit rester relativement indépendant du reste du contenu du document.

Le processus d'extraction des données de pages web nécessite leur récupération préalable, ce qui pose des problèmes d'une part de navigation dans la structure du site web et d'aspiration des pages, d'autre part de simulation d'une session complète. Ces problèmes ne sont pas abordés car dans le cadre de notre travail, on dispose déjà d'un ensemble de pages web téléchargées.

3 Contexte technologique

HTML est un langage à balises [21] ne portant pas d'information sur les données contenues dans la page, mais seulement sur leur mise en forme. Nous sommes alors face à un corpus semi-structuré [9]. De plus, dans notre cas, il est très difficile d'appliquer les techniques de TALN[2] par le simple fait que souvent la page Web ne contient que peu d'éléments textuels. Fréquemment même, nous sommes confrontés à des phrases incomplètes et syntaxiquement incorrectes. C'est pourquoi nous avons adopté le langage de transformation XSLT[3] [21] qui effectue des transformations sur des documents XML pour les convertir vers d'autres formats comme RDF, RTF, XHTML, ou tout simplement vers d'autres documents XML. Dans le cadre de notre projet, nous utilisons ce langage puissant, pour faire des transformations de (X)HTML vers XML.

Les premiers systèmes d'extraction de données et d'information n'exploitaient pas la structure de la page [9]. Ils utilisaient une approche linéaire et séquentielle qui manquait de puissance. Récemment, la recherche dans le domaine s'est focalisée sur l'exploitation de la structure interne des pages web. Il s'agit d'une décomposition de la page et d'une traduction de son contenu semi-structuré vers un format bien défini. Cela implique la construction de wrappers composés de règles d'extraction. La génération d'un wrapper se fait soit à la main soit avec des outils interactifs facilitant le processus de création.

Il existe parallèlement un autre type de techniques d'extraction développé en intelligence artificielle basée sur des algorithmes d'auto-apprentissage [9,10]. Ce sont des systèmes capables d'analyser un ensemble de pages web avec des caractéristiques communes afin de produire automatiquement des règles d'extraction capables de ressortir des données à partir de pages similaires [10].

A terme, les deux approches devront être utilisées ensemble pour construire des systèmes d'extraction fiables car même le meilleur système automatique n'est pas capable de remplacer l'être humain dans la validation et la supervision du processus [7].

4 Travaux similaires

Il y a plusieurs systèmes déjà disponibles qui se ressemblent plus au moins par les méthodes utilisées. Chaque système utilise son propre langage interne pour décrire ses règles d'extraction. Certains utilisent des langages propriétaires comme le W4F[4] [6] qui a nommé son langage HEL[5]. D'autres utilisent des langages tiers conçus

[2] TALN Traitement Automatique de la Langue Naturelle

[3] XSLT (eXtensible Stylesheet Language Transformations)

[4] W4F (WysiWyg Web Wrapper Factory) Générateur de wrappers avec GUI

[5] HEL (HTML Extraction Language) Langage d'extraction de données à partir de pages HTML

spécialement pour extraire des données comme le DEL[6], WebL[7], Elog[8] [9]. Tous ces langages utilisent des règles décrivant le chemin vers l'élément contenant les données et s'appuient souvent sur des expressions régulières pour spécifier le résultat.

Un inconvénient de ces langages est qu'ils sont basés sur la structure globale de la page. Des changements, même mineurs, dans la mise en forme ou dans la composition, nécessitent la génération d'un nouveau wrapper [13]. Tous ces langages sont aussi relativement compliqués et il est très difficile de générer leur syntaxe d'une manière automatique à partir de manipulations visuelles (*drag-and–drop, point-and-click*).

Les systèmes les plus connus qui suscitent les débats dans le domaine sont le W4F [6,8], ANDES [13] et le XWrap [5]. Le W4F (*développé au sein de l'Ecole Nationale Supérieure des Télécommunications, Paris*) utilise des chemins absolus pour retrouver des données dans les pages HTML. Cette approche ne permet pas de prendre en compte le changement de la mise en forme de la page. Le XWrap est basé sur la signification sémantique des balises HTML (*les tableaux et les en-têtes*) et la façon dont ils sont utilisés pour la mise en forme de la page. Cette approche n'est pas efficace pour des paragraphes peu structurés. Enfin, le système ANDES de IBM utilise comme langage d'extraction XSLT mais il se base explicitement sur le repérage des mots clefs en ignorant complètement la structure dans laquelle ils apparaissent. Cela pourrait conduire à des résultats peu fiables, car il est possible d'avoir des mots clefs qui se répètent dans la page.

5 La méthodologie

Le contexte technologique décrit précédemment implique que les règles d'extraction de données soient spécifiques à chaque site analysé. L'objectif est donc de construire un système qui facilite la création et la maintenance de ces règles afin de réduire les coûts tout en analysant de manière fiable des centaines de milliers de pages. Le système doit également stocker les résultats dans une base de données pour produire ultérieurement des statistiques.

Notre système d'extraction des données supervisé intègre trois modules principaux : (i) un module de proposition, (ii) un module de génération de règles d'extraction (*wrapper générateur*[9]) et (iii) un moteur d'extraction des données.

Le module de proposition analysera automatiquement les pages web en cherchant les données pertinentes de la page traitée. Ce module sera basé sur une approche ontologique d'identification des données [16]. Les règles d'identification dépendront du contexte dans lequel l'extraction des données s'effectuera et elles seront appliquées selon des heuristiques prédéfinies. Il utilisera une base des meta données enrichie

---

[6] DEL (Data Extraction Language) Langage de conversation vers XML inspié par XSLT qui utilise des expressions XPath

[7] WebL – langage d'extraction des données propriétaire à Compaq

[8] Elog – langage d'extraction déclaratif

[9] Wrapper générateur est la système qui génère le wrapper[1] souvent de manière semi-automatique.

automatiquement par le module de création de règles suite à l'interaction avec l'opérateur. Dans cette base de 'connaissances' seront sauvegardés des « patterns modèles » (*structures partielles*), des mots clefs et des attributs avec des significations sémantiques (*par rapport au contenu de la balise, comme : name, id, etc.*) , qui seront utilisés pour faire les propositions. Ce module n'a pas encore été abordé.

A partir des propositions du module précédent, le module de génération de règles demandera à un opérateur de les valider, puis créera les procédures d'extraction (*des wrappers*). Ce module peut aussi être exécuté de manière interactive, en demandant à l'utilisateur de sélectionner les données qu'il souhaite extraire ainsi que les mots clés introduisant les données. Le module génère alors automatiquement les règles XSLT. C'est cette partie que nous avons réalisée et dont la description est l'objet du paragraphe suivant.

Enfin, le moteur d'extraction récupère les données en exécutant les "wrappers" sur des pages HTML et remplit une base de données avec les données recueillies.

Il faut noter que les pages à partir desquelles nous extrayons des données sont des pages dynamiques, derrière lesquelles l'information est stockée dans des bases de données. Selon [8], 80% du Web est basé sur des données dynamiques.

Les pages générées à la volée et non pas créées à la main sont plus structurées et moins susceptibles de changer complètement (*au niveau de structure*) que des pages statiques. Souvent, les modifications détectées dans des pages de ce type sont seulement des changements de style ou de restructuration suite à l'insertion d'une publicité ou de nouveaux blocs d'information. Même si des changements plus radicaux sont effectués, il est fort possible que les sous-structures enfermant les données pertinentes pour notre application ne soient pas affectées par les changements opérés.

C'est pourquoi nous avons décidé de nous appuyer sur la structure locale à l'intérieur de la page qui réunit les données à extraire et non pas sur la structure de la page elle-même. Cette structure partielle englobant les données peut être répétée dans la page plusieurs fois, car le nombre des noms des éléments impliqués est restreint (*par la spécification des balises HTML*). Cela veut dire que l'on va avoir des structures comme des tableaux, listes, paragraphes, etc., très semblables. Pour repérer la structure contenant les données pertinentes pour notre application, nous avons besoin d'une sorte d'ancre qui désigne de manière non ambiguë cette sous-structure. Notre ancre s'appuie sur le contenu de la page, car nous nous sommes aperçus qu'il y a toujours un ou plusieurs mots clés qui accompagnent les données à extraire. Ces mots clefs définissent le contexte dans lequel l'extraction s'effectue. Par exemple si on extrait des données à partir d'une page traitant d'appareils numériques, on est dans un contexte ontologique spécifique au domaine technique de la photographie numérique [16]. Les mots clés seront des éléments dans l'ontologie propre au domaine : les caractéristiques de l'appareil (*résolution, capacité, zoom, etc*). Il est probable que chacun de ces mots clés soit physiquement très proche des données à extraire, probablement dans la même sous-structure.

Pour l'instant notre système permet de sélectionner interactivement des mots-clefs et de les associer avec les données à extraire. L'approche ontologique sera réellement exploitée dans le module de proposition.

La méthodologie, telle que nous l'avons définie pour le moment, repose sur la notion de *couple d'information*. Un couple est constitué d'un type de données auquel l'opérateur donne un nom et est caractérisé par un ou plusieurs mots-clefs, et les relations entre eux. Par exemple, la sélection de l'identifiant de la transaction conduit à la génération d'un couple formé du mot "*identifiant"* et des mots clefs "*transaction*" et "*numéro*", ainsi que la structure locale englobant.

Nous avons également défini la notion de *patron* qui est l'ensemble des *couples d'information* formant un enregistrement des données. Un *patron* représente la structure partielle dans laquelle les données sont situées, les relations entre les couples, et se définit par son élément racine dans la hiérarchie DOM[10] [18] (*i.e. un ancêtre commun à toutes les données de la structure*). Nous appelons cet ancêtre commun l'ancre du patron, puisque la structure partielle est attachée à cet élément. Un *patron* contient au moins un couple avec un mot-clef pour pouvoir fixer l'ancre du *patron*.

Nous avons choisi XSLT comme langage pour coder notre *patron*. Les relations entres les éléments sont décrites grâce à des expressions et fonctions XPath[11] [17].

Nous avons adopté une construction XSLT relativement simple et en même temps suffisante pour répondre à notre besoin. Le principe utilisé est le suivant : Notre programme construit un ensemble de règles qui décrivent nos couples d'information et notre patron afin de construire une sorte de wrapper en XSLT. Le patron identifie dans la page HTML la sous-structure qui contient les données. Pour cela, il utilise les marques lexicales des couples d'information. Une fois la structure localisée, l'extraction des données s'effectue par des chemins relatifs par rapport à l'ancre grâce aux transformations XSL.

Un problème surgit à ce niveau du fait que XSLT travaille sur des sources XML et pas sur du HTML mal formaté. Ce problème est en partie résolu à l'aide des outils comme Tidy [4, 21] qui sont capables de nettoyer le code et de le convertir en XHTML qui est un format bien structuré. Les "erreurs" découlant d'HTML sont réparées, les tableaux et les balises sont fermés proprement, les attributs répétés plusieurs fois ne sont pas autorisés, etc.

Une fois cette étape accomplie, il faut analyser (*"parser"*) la structure HTML de la page web. Après avoir expérimenté plusieurs analyseurs, nous nous sommes vite aperçus que les outils disponibles dans les langages de programmation standard ainsi que ceux proposés par des tiers ne sont pas encore au point. La seule possibilité pour obtenir une analyse correcte est d'utiliser un navigateur pour la décomposition HTML. Cela nous a conduit au choix de la plate-forme Mozilla [11,1,20] qui est à la fois un navigateur et un environnement de développement très puissant appelé Cross-Platforme Front End ou XPFE.

Choisir des technologies standards (*XHTML, XSLT, XPFE*) permet à notre système d'évoluer avec eux, ce qui est très important car les procédés changent en permanence et rendent obsolète les techniques utilisées. Par exemple, le système W4F n'est plus supporté car il est très difficile à maintenir.

---

[10] DOM (Document Object Model) – la structure de la page (X)HTML ou XML décomposée en éléments.

[11] XPath (XML path language)

Notre système a besoin d'un bon analyseur, d'un composant de visualisation du HTML, et surtout d'un langage d'extraction fiable. La plate-forme XPFE suit le progrès technologique, elle est capable de parser et visualiser tout type de pages WEB, ce qui justifie notre choix. Si cette plateforme nous garantit la comptabilité avec les futures pages web, au moins pour quelques années, le choix du langage XSLT, nous assure une manipulation des ressources XHTML et XML également.

6 Un exemple de construction des règles d'extraction

Nous allons illustrer la construction des règles d'extraction à partir d'un exemple. Considérons la page Web issue du site EasyJet.com, indiquant la confirmation de l'achat d'un billet. Les données à récupérer sont le numéro de confirmation, le nom du client et le montant total payé (*cf. figure 1 -partie centrale où les données à récupérer sont encadrées*-). L'opérateur sélectionne à la souris ces données, leur donne un nom et sélectionne les marques lexicales pour former les couples d'information comme suit :

- d'abord l'utilisateur crée de manier visuelle un nouveau couple et lui donne un nom (à droit sur l'image écran, fig1.);
- il choisit les données pour le couple en cliquant dans la page web visualisée ;
- ensuite l'opérateur sélectionne un ou plusieurs mots clés définissant la donnée ;
- les opérations 1-2-3 sont répétées pour créer autant de couples que nécessaire ;
- finalement le système crée des règles XSLT et les applique ;
- l'utilisateur valide les résultats ;

Au cours de ces manipulations visuelles, le système construit en plan des expressions XPath pour chaque couple d'information, selon la décomposition de la page fait par XPFE et les regroupe afin d'engendrer le patron pour la page donné.

La figure 1 -partie gauche- montre l'arborescence DOM de cette page web, telle qu'elle est chargée en mémoire par XPFE. Les rectangles en pointillés marquent la structure locale pour chaque couple d'information. Ils peuvent être codés en expressions XPath par :

Expressions pour les mots-clefs en XPath :

```
Pour ancre-couple1 : p/em[contains(.,'confirmation')]
Pour ancre-couple2 : p/em[contains(.,'Passager:')]]
Pour ancre-couple3 : p[em[contains(.,'total:')] and
                        strong[contains(.,'EUR')]]
```

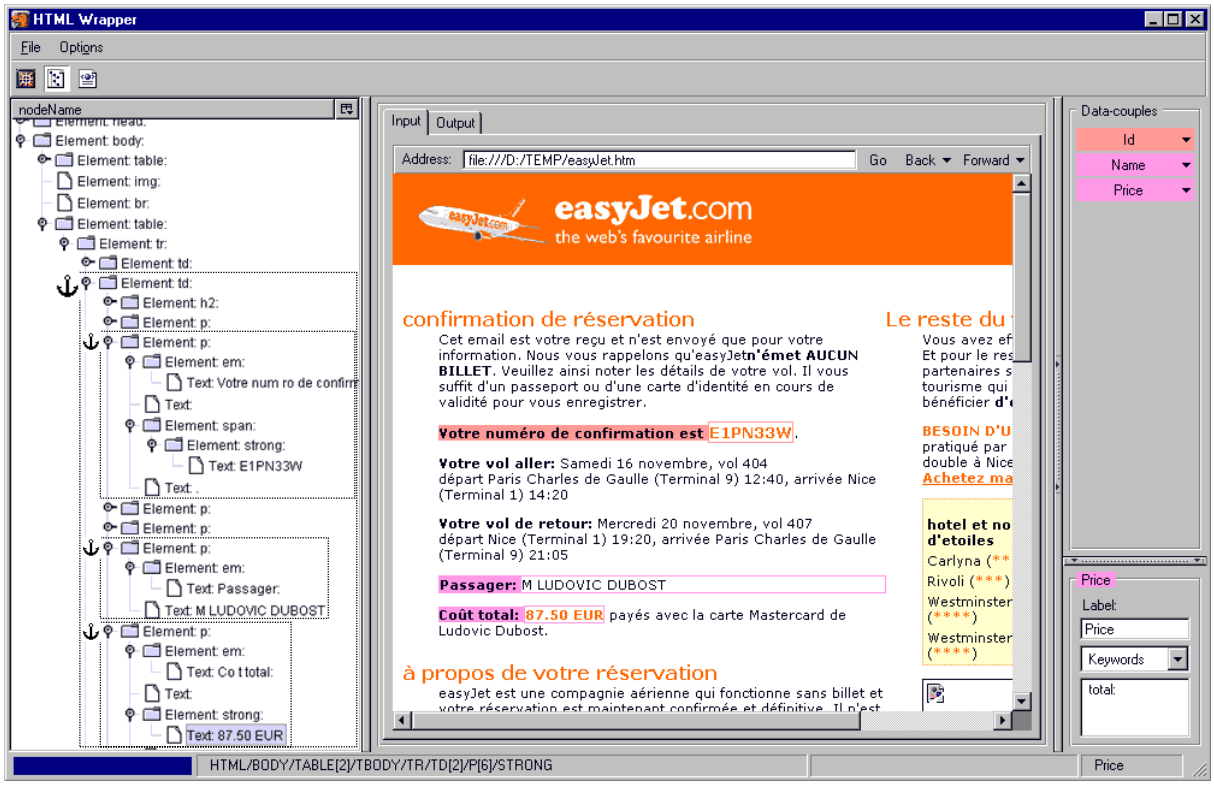

Fig.1 XPFE Wrapper

Le chemin vers les éléments dans le couple est fixé par le premier élément qui contient les autres, il joue le rôle d'une ancre pour le couple. Dans notre exemple c'est l'élément « **p** ». Les marques lexicales sont marquées en gras. Par exemple, la troisième expression peut être paraphrasée en langue française comme suit :

*Trouve-moi un élément **p** père d'un élément **em** qui contient la sous-chaîne "**total**" et en même temps père d'un autre élément **strong** qui lui de son côté contient la sous-chaîne "**EUR**".*

Après avoir fixé les ancres des couples d'information, on peut accéder aux données en faisant référence à l'élément textuel qui les contient par rapport à l'ancre déjà définie grâce à des variables (préfixées par $ en XSL).

```
Pour ancre-couple1 : $ancre-couple1/span/strong/text()
Pour ancre-couple2 : $ancre-couple2/text()
Pour ancre-couple3 : $ancre-couple3/strong/text()
```

L'ancre du patron est le premier ancêtre commun aux 3 ancres des couples. En l'occurrence, il s'agit de la balise **td**. Elle est obtenue avec l'opérateur de descente récursive (//) (*chemin de localisation relatif*) :

```
Pour ancre-patron : //td[ p/em[contains(.,'confirmation')]
                          and  p/em[contains(.,'Passager:')]
                          and p[em[contains(.,'total:')]
                                 and em[contains(.,'EUR')] ] ]
```

En réalité nous appliquons plusieurs transformations XSL (*dans notre cas deux*) pour effectuer l'extraction. La première sert à la récupération de la structure locale contenant les données (*l'expression de l'ancre du patron*) et la deuxième sert à effectuer l'extraction proprement dite des données. Il faut tenir compte du positionnement exact des éléments à extraire par rapport à l'ancre du couple ou par rapport à l'ancre du patron. Par exemple, pour le premier couple, les règles d'extraction possibles sont les suivantes:

```
$ancre-couple1/span/strong/text() ou
$ancre-patron/p[2]/span/strong/text()
```

Remarque : la deuxième expression s'applique pour les cas où il n'y pas de marques lexicales définies pour le couple.

Finalement nous pouvons construire le wrapper en syntaxe XSLT en regroupant les règles de la manière suivante :

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:variable name="patron" select="//td[ p[ em[ contains(.,'confirmation') ] ]
                                        and p[ em[ contains(.,'Passager:') ] ]
                                        and p[ em[ contains(.,'total') ]
                                            and strong[ contains(.,'EUR') ] ] ]"/>
     <xsl:variable name="id" select="$patron/p[2]/span/strong/text()"/>
     <xsl:variable name="name" select="$patron/p[5]/text()"/>
     <xsl:variable name="price" select="$patron/p[6]/strong/text()"/>

     <xsl:template match="/">
        <id><xsl:value-of select="$id"/></id><br/>
        <name><xsl:value-of select="$name"/></name><br/>
        <price><xsl:value-of select="substring-before($price,'EUR')"/></price><br/>
     </xsl:template>
</xsl:stylesheet>
```

Notre wrapper est un script XSLT qui transforme la partie à récupérer du document HTML en un document XML contenant les données à extraire dans un format bien structuré. Grâce aux fonctions classiques manipulant des chaînes des caractères (*substring*, *contains*,…), il est possible de raffiner les données et de les séparer des autres informations textuelles présentes éventuellement dans le même élément. C'est une solution provisoire en attendant la standardisation de la version 2.0 de XPath qui apportera la puissance des expressions régulières dans les feuilles XSL.

Le résultat de l'exécution de l'exemple au-dessus est le suivant :

```
<id>E1PN33W</id>
<name>M LUDOVIC DUBOST</name>
<price>87.50</price>
```

Ainsi, notre méthodologie permet d'exploiter la structure de la page Web tout en restant indépendante des changements mineurs dans la structure globale de la page.

7 Conclusion et travaux à venir

Notre objectif est de construire une méthode fiable et résistante aux changements dans la page. Pour ce faire, nous présentons dans cet article une méthodologie d'extraction des données qui s'appuie sur des technologies standard et une normalisation préalable du HTML vers XHTML.

Cette normalisation, qui demande l'utilisation de techniques de filtrage et de nettoyage du code HTML, est une difficulté temporaire, car si la migration complète vers XML va durer plusieurs années (*due aux différents problèmes non uniquement techniques*), le web a déjà adapté le format XHTML. Notre système est donc conçu pou rester compatible (*grâce aux procédures de filtrage et de nettoyage du code HTML*) avec les pages web courantes. Nous utilisons les langages XSL et XPath qui se sont imposés comme les principales techniques de manipulation de ressources web. XSL répond exactement aux besoins de notre méthodologie qui est basée sur la recherche d'une structure locale repérée par une ancre en utilisant des marquer flottants (*mots clés, attributs sémantico-significatif, …*). Une fois la structure avec les données localisée, l'extraction peut être effectuée facilement par des expressions relatives à l'ancre grâce à XSLT.

Le principal problème à résoudre reste l'élaboration automatique des scripts XSLT avec une syntaxe correcte. Nous avons accompli un premier pas avec notre prototype appelé XPFE WRAPPER qui génère automatiquement ces règles grâce à de simples manipulations visuelles (*drag-and-drop, point-and-click*) à partir d'une interface graphique.

Il reste à expérimenter le système avec des pages web d'origine diverse pour valider la méthodologie et les outils développés. Nous serons certainement confrontés à des exceptions pour lesquelles nous seront obligés d'élaborer des règles XSLT spécifiques. La méthodologie décrite doit être encore développée afin de produire des patrons de manière semi-automatique. Enfin, des tests extensifs doivent être réalisés pour calculer la précision d'extraction et la montée en charge de la technologie utilisée.

Bibliographie

[1] Mozilla, XPToolkit Project: http://www.mozilla.org/xpfe/

[2] Laublet P., Reynaud C., Charlet J., Sur Quelques Aspects du Web Sémantique, Assises 2002 GdR I3, Nancy http://sis.univ-tln.fr/gdri3/

[3] M. K. Bergman, Deep Web : http://www.press.umich.edu/jep/07-01/bergman.html

[4] Dave Ragget, Tidy : D.Raggett. Clean Up Your Web Pages with HTML TIDY, 1999, http://www.w3c.org/People/Raggett/tidy/

[5] Ling Liu, Calton Pu, XWRAP, Liu, L. Pu, C. and Han, W.: XWrap – An XML-enabled Wrapper Construction System for Web Information Sources, Proceedings of the 16th International Conference on Data Engi-neering (*ICDE'2000*), San Diego CA, *2000*, http://www.cc.gatech.edu/projects/disl/XWRAPElite/

[6] A.Sahuguet, Fabien Azavant, W4F, A.Sahuguet, Fabien Azavant, Building light-weight wrappers for legacy Web data-sources using W4F, *Ecole Nationale Supérieure des Télécommunications, Paris, France, fabien.azavant@enst.fr* , http://cheops.cis.upenn.edu/W4F

[7] Alberto H. F. Laender, Berthier A. RibeiroNeto, A Brief Survey of Web Data Extraction Tools, *CiteSeer 2002,* http://citeseer.nj.nec.com/laender02brief.html

[8] Sahuguet, A. and Azavant, F.: Web Ecology – Recycling HTML pages as XML documents using W4F, in: ACM International Workshop on the Web and Databases (*WebDB'99*), Philadelphia, Pennsylvania, USA, June 1999, http://cheops.cis.upenn.edu/W4F

[9] Lin Eikvil (1999). Information Extraction from the WWW, Scientific Literature Digital Library, http://citeseer.nj.nec.com/eikvil99information.html

[10] Kushmerick, N.: Wrapper Induction for Information Extraction, Dissertation 1997, Dept of Computer Science & Engineering, Univ. of Washington, Tech. Report UW-CSE-97-11-04, *Oct. 2002,* http://www.cs.ucd.ie/staff/nick/home/research/download/kushmerick-phd.ps.gz

[11] R. Allen Wyke, Jason D. Gilliam (1999). Pure JavaScript, *SAMS*

[12] Michael Kay (2001). XSLT La référence du programmeur, *Wrox Press France SARL*

[13] Marlboro College Graduate Center MSIE 2002: DB2, Data on the Web, Session 6, Data handling in an HTML world, http://bob.marlboro.edu/~msie/2002/db2/s6.html

[14] Marlboro College Graduate Center MSIE 2002: IPL. Java. 15 June 02. Web Services, *http://bob.marlboro.edu/~msie/2002/ipl/java/lectures/jun15/ipl3-4.html*

[15] D.W.Embley, D.M.Campbell, A Conceptual-Modeling Approach to Extracting Data from the Web, *CiteSeer 1998,* http://citeseer.nj.nec.com/24307.html

[16] H. Snoussi, L.Magnin, J.Y.Nie, Heterogeneous Web Data Extraction using Ontology, *CiteSeer*, http://citeseer.nj.nec.com/550153.html

[17] World Wide Web Consortium (*1999*) : XPath XML Path Language, http://www.w3.org/TR/xpath

[18] World Wide Web Consortium: The Document Object Model, http://www.w3.org/DOM

[19] World Wide Web Consortium (*1998*) : XML Extensible Markup Language, http://www.w3.org/XML

[20] David Boswell (2002). Creating Applications with Mozilla, *O'Reilly*

[21] World Wide Web Consortium (*2000*) : A Reformulation of HTML 4 in XML 1.0, http://www.w3.org/TR/xhtml1