République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA
Faculté des Sciences Exacts Département d'informatique

Mémoire

Présenté par

BARR Youcef

Pour l'obtention du diplôme de Magister

Filière : Informatique

Option : Cloud Computing

Thème

Proposition et Implémentation d'un mécanisme d'extraction et intégration de données d'un entrepôt médical

Soutenu le : 10/04/2016 Devant le Jury composé de :

Nom et Prénom	Grade		
Mr. AIT SAIDI Ahmed	Professeur	Univ. de Béjaia	Président
Mr. TARI Abdelkamel	Professeur	Univ. de Béjaia	Rapporteur
Mr. BOUKERAM Abdellah	Professeur	Univ. de Béjaia	Examinateur
Mme.TAHAKOURT Zineb	Maître Assistant A	Univ. de Béjaia	Invitée

Année Universitaire : 2015 /2016

Remerciements

Avant tout, je remercie Dieu, tout puissant, qui m'a aidé à réaliser ce travail. Je remercie très sincèrement Dr AbdelKamel TARI, Professeur à l'Université de Béjaia, Responsable de l'école doctorale, et Directeur du Laboratoire LIMED d'avoir accepté d'être rapporteur de ce mémoire. Je le remercie beaucoup, sans lui l'école doctorale n'aurait jamais eu le succès.

Je remercie vivement Mme. Z. TAHAKOURT Née YOUNSI, Professeur à l'Université de Béjaia, pour avoir accepté de Co-encadrer ce travail, et pour me Permettre de découvrir un domaine Passionnant de l'informatique, sans elle je n'aurais jamais eu l'idée de lancer ce mémoire, pour sa patience et sa confiance, pour m'avoir fait partager sa passion pour la recherche .Ses critiques constructives, ses explications, ses suggestions pertinentes, et ses conseils précieux, m'ont conduit à l'aboutissement de ce travail.

Merci encore à tous qui m'ont entouré pendant mon séjour à Béjaia. Au sein de la promotion 2012, bien sûr, mais aussi plus largement. Je voudrais en particulier citer AITE SEDDIK Hamza et MOUKTAFI Mouhand.

Je tiens à exprimer ma profonde gratitude aux membres de mon jury qui me font le grand honneur d'avoir accepté de juger mon travail.

A Ma Grand-Mère

Résumé

La croissance de l'internet fût accompagnée d'un intérêt particulier à l'analyse des données web. L'extraction de ces données est reconnue actuellement comme une tâche cruciale pour exploiter les informations issues du web, notamment dans un projet d'entreposage. La plupart des méthodes actuelles d'extraction ont été conçues pour traiter une seule source à la fois, cependant la prise en compte de l'aspect multi-source est un élément déterminant dans le domaine d'extraction d'information à partir du web.

Le travail présenté dans ce mémoire s'inscrit essentiellement dans une double orientation de recherche dont les objectifs sont d'une part l'extraction multi-source d'information à partir du web et d'autre part la construction d'un entrepôt médical. Nous proposons une approche, pour l'extraction d'information à partir du web, basée sur un système d'inférence qui permet de suggérer une méthode d'extraction adéquate pour chaque source web dans un projet d'extraction multi-source. Le système proposé exploite les connaissances contenues dans une ontologie. Cette dernière regroupe : les caractéristiques des méthodes d'extraction, celles de la source web en question, et le niveau de manipulation d'utilisateur exploitant notre système. Cette approche est très efficace, car elle permet de tirer les avantages de chaque méthode. L'expérimentation réalisée montre l'intérêt de cette démarche, ainsi l'extraction a été effectuée à partir de deux sources web pour alimenter un entrepôt XML de médicaments.

Mots-clés : extraction d'information à partir du web, entrepôt de données, ontologie, intégration de données, inférence, Web content mining, multi-sources, méthode d'extraction.

Abstract

The growth of the Internet was accompanied by a special interest in the analysis of web data. The extraction of this data is currently recognized as a crucial task to exploit the information issues from the web, especially in a data warehouse project. Most current extraction methods were designed to treat only one source at a time, however, considering the multi-source aspect is a key factor for information extraction from the web.

The work presented in this paper is essentially part of a double line of research which aims on one hand to multi-source information extraction from the web and, on the other hand, the construction of a medical data warehouse. We propose an approach for the extraction of information from the web, based on an inference system to suggest a suitable extraction method for each source in a multi-source web information extraction project. The proposed system exploits the knowledge contained in ontology. This last includes: characteristics of extraction methods, those of the web source, and the user level witch handling our system. This approach is very effective because it can draw the benefits of each method. Experiments shows interest of this approach, and the extraction was performed from two web sources to supplying an XML data warehouse of medicaments.

Keywords: information extraction from the web, data warehouse, ontology, data integration, inference, Web content mining, multi-source, extraction method.

ملخص

لقد واكب تطور شبكة الإنترنت اهتمام خاص بتحليل المعطيات الموجودة فيها .كما يعتبر استخراج هذه البيانات ،حاليا ، مرحلة جد مهمة لاستغلال معلومات هذه الشبكة، خصوصا في مشاريع مستودع البيانات .وقد صممت معظم طرق الاستخراج الحالية لعلاج مصدر واحد فقط في كل مرة. ومع ذلك، فإن الأخذ بعين الاعتبار خاصية (تعدد المصادر) هو عامل رئيسي في مجال استخراج المعلومات من الإنترنت.

إن العمل المقدم في هذه المذكرة يندرج أساسا في مجالين من البحث اللذين يهدفان من جهة إلى استخراج المعلومات من مصادر متعددة عن طريق شبكة الإنترنت وأيضا إلى إنشاء مستودع بيانات طبي .لقد اقترحنا من خلال هذا العمل نهجا جديدا لاستخراج المعلومات من شبكة الإنترنت، إن الطريقة المقترحة تقوم على أساس نظام استدلال. هذا الأخير يقوم بإيجاد طريقة استخراج مناسبة لكل مصدر ويب ، وذلك ضمن مشروع استخراج المعلومات متعددة المصادر .تعتمد طريقة عمل النظام المقترح على استغلال المعطيات الموجودة في الأنطولوجيا .هذه الأخيرة تشمل: خصائص طرق الاستخراج، خصائص مصدر الويب، وكذا معلومات تخص مستوى مستخدم النظام المقترح . إن هذه الطريقة فعالة جدا لأنها تمكن من استغلال فوائد كل طرق الاستخراج. لقد أظهرت التجارب المجرات فائدة هذه الطريقة، كما تم استخراج المعلومات من مصدرين اثنين من اجل إنشاء مستودع بيانات خاص بالأدوية.

الكلمات المفتاحية: استخراج المعلومات من شبكة الإنترنت، مستودع البيانات، الأنطولوجيا، تكامل البيانات، الاستدلال تحليل محتوى الويب، متعددة المصادر، طريقة استخراج المعلومات

Table des matières

Liste des Figures

Chapitre 1 : Fouille Du Contenu Web

Chapitre 2 : Extraction D'information A Partir Du Web

Chapitre 3 : Intégration De Données

Chapitre 4 : Proposition D'une Méthode Pour L'IEW Multi-Sources

Chapitre 5 : Expérimentation

ANNEXE A : Implémentation De La Méthode STALKER

Liste des Tables

INTRODUCTION GÉNÉRALE

1. Contexte

Depuis son invention en 1989 par Tim Berners-Lee, le Web propose un nombre sans cesse croissant de sources d'informations. L'exploitation des données de ces sources très hétérogènes est devenue, ces dernières années, une problématique majeure pour de nombreux chercheurs. En effet, il n'est pas évident d'en exploiter le contenu de manière efficace et intelligente : les pages HTML ont deux inconvénients majeurs [J.R Meurisse, 2004] : d'une part, elles sont constituées d'un mélange de données et de consignes de présentation et, d'autre part, elles sont pratiquement dépourvues d'informations relatives à la structure et à la signification des données.

Afin d'exploiter les données de sources web, il est donc intéressant d'élaborer des méthodes pour séparer les données de leur présentation, les extraire, les interpréter et retrouver leur structure invisible dans les pages HTML. Cet intérêt a motivé les chercheurs à penser à des solutions, d'où l'apparition de l'extraction des données à partir du Web.

Les entrepôts de données intègrent les informations en provenance de différentes sources, souvent réparties et hétérogènes et qui ont pour objectif de fournir une vue globale de l'information aux analystes et aux décideurs. Contrairement aux entrepôts de données classiques qui ont pour vocation l'analyse de l'activité d'une entreprise dans le but de mieux la piloter ; les entrepôts de données de santé ont une réelle spécificité. En effet, il existe aujourd'hui une demande grandissante, plus scientifique, d'utilisation de données médicales, à des fins de recherche clinique ou épidémiologique. Ces entrepôts de données médicales (non administratives), issues par exemple de la biologie, de la génétique, ou bien d'activités cliniques, peuvent se construit à partir du web.

Dans ce mémoire, nous abordons l'aspect multi-sources dans un contexte d'extraction d'information à partir du Web, pour construire un entrepôt de données médical. Dans cette section, nous tâcherons de poser la problématique, en suite nous allons présenter notre contribution.

2. Problématique

L'objectif de l'extraction d'information à partir du Web est de construire des programmes capables, d'extraire d'une page Web une partie de son contenu et de le mettre dans un format structuré. Plusieurs méthodes d'extraction ont été proposées, dont la majorité sont conçues pour être appliquées sur une seule source web à la fois. La prise en compte du besoin de l'extraction multi-sources devra être un élément déterminant dans les travaux de recherche dans ce domaine. En effet, l'extraction d'information de plusieurs sources engendre, en plus, deux types de problèmes :

- Chaque source web est basée sur un schéma (Template) pour présenter ses pages, elle peut aussi utiliser une technologie différente (langages de programmation web….etc.). D'où le problème d'hétérogénéité au niveau de la syntaxe des pages web de différentes sources.
- Chaque source web est fondée sur une base de données, d'où le problème classique, d'hétérogénéité des données issues de différentes bases de données.

Certaines méthodes [D.W. Embley et al, 1999] [H. Snoussi et al, 2001] [Li. LIU et al, 2010] sont proposées pour répondre au problème de l'extraction multi-sources. Elles exploitent les connaissances décrites dans une ontologie pour extraire des informations à partir de pages du même domaine. Les connaissances stockées dans l'ontologie se référant aux caractéristiques des données et à leur présentation dans les pages web. Cependant les méthodes proposées souffrent d'un inconvénient major, l'étape essentielle et commune pour ces méthodes est la construction d'avance des connaissances de domaine d'intérêt (ontologie), cette tâche est très couteuse. Notamment, recenser l'ensemble des formes de présentations possibles d'informations nécessite une analyse de l'ensemble des sites, à partir desquelles l'utilisateur souhaite extraire l'information, afin de les intégrer dans la connaissance du domaine.

L'objectif de notre travail est de proposer une nouvelle approche pour l'WIE multi-sources. L'idée de cette approche est de développer un système d'inférence qui suggère une méthode d'extraction pour chaque source. Ce système est basé sur les critères des méthodes d'extraction, et les caractéristiques des sources web, et en prenant en compte aussi le niveau de manipulation de l'utilisateur de ce système. Le but est d'effectuer l'extraction via plusieurs méthodes dans un projet d'extraction multi-sources. Dans ce contexte, plusieurs questions se posent :

1. Quels sont les critères permettant de différencier les méthodes d'extraction ?
2. Quelles sont les caractéristiques des sources web, qui favorisent d'utiliser une méthode par rapport à une autre. Et comment les obtenir automatiquement en analysant les sources web?
3. Comment effectuer le processus d'inférence puis faire l'extraction multi-sources ?

3. Contribution

Notre contribution dans ce mémoire consiste en la proposition d'une méthode d'extraction multi-sources d'informations à partir du web. Notre stratégie consiste à développer un système d'inférence qui, à partir de plusieurs sources web, suggère la méthode adéquate pour chaque source, en suite appeler cette méthode pour effectuer l'extraction. Cette démarche est très efficace, car elle permet de tirer les avantages de chaque méthode. Le système proposé est basé sur la construction d'une ontologie d'application. Les données introduites dans l'ontologie sont issues de trois ressources : les critères des méthodes d'extraction, les caractéristiques des sources web, et celles concernant l'utilisateur. Nos apports dans ce mémoire sont:

- ✓ Nous avons fait une étude comparative des méthodes d'extraction, l'objectif de cette étude est de comprendre la manière de travail pour chacune, ainsi de fixer les critères permettant de comparer les méthodes d'extraction.
- ✓ Nous avons construit l'ontologie d'application. Il s'agit d'une ontologie OWL DL, les règles d'inférence sont écrites en langage SWRL.
- ✓ Pour l'analyse des sources web, Nous avons proposé une nouvelle méthode, en deux étapes :
  1. L'idée de la première étape consiste à classifier les pages web en clusters suivant la structure DOM de chaque page. En parcourant les arbres DOM en largeur, l'opération de clustering se fait pour chaque niveau parcouru. Ce processus continu jusqu'à atteindre le niveau des feuilles. Cette technique permet de repérer la moindre différence entre les structures arborescentes. Pour chaque niveau parcouru, une procédure d'élimination des blocks bruyants est appelée, les nœuds ayant un contenu commun pour les pages du même cluster sont considérés comme blocks bruyants, ils sont donc éliminés.
  2. Pour détecter les caractéristiques de la source web, nous avons introduit une étape de re-clustering. En prenant une page exemple pour chaque cluster obtenu en (1), on compare les pages deux à deux ; des heuristiques sont proposées pour indiquer l'existence de chaque caractéristique (attribut multi-ordre, attributs à valeurs manquantes, …etc.).

     Outre que les étapes (1) et (2), d'autres heuristiques sont proposées pour tirer d'autres caractéristiques, comme (le type de page : texte, structurée, ou semi-structurée, nombre d'enregistrements dans une page (soit multi-enregistrements, ou page détaillée…etc.)
- ✓ Nous avons aussi proposé une architecture pour le projet d'extraction multi-sources basé sur le système d'inférence, nous avons défini, en outre les fichiers de configuration du projet, et le processus d'extraction multi-sources.

4. Organisation du contenu

Ce mémoire est structuré en cinq chapitres:

Chapitre 1 : Nous précisons, dans ce premier chapitre, le cadre de notre sujet en présentant les concepts de fouille web. Nous décrivons les tâches principales de la fouille web. Nous présenterons, en suite, la taxonomie de ce concept général qui se divise, selon le type de données à exploiter, en trois catégories principales: Web Content, Web Structure et Web Usage Mining. Nous nous intéressons, en particulier, au concept du Web Content Mining.

Chapitre 2 : Dans ce chapitre, nous présentons un état de l'art pour le domaine de l'extraction d'information à partir du web. Dans un premier temps, nous présentons quelques notions de base pour ce domaine. Nous mettons également l'accent sur les défis de l'extraction d'information à partir du web. Les différentes classifications des méthodes d'extraction sont ensuite présentées. Nous analysons aussi les métriques d'évaluation pour une tâche d'extraction. En fin, nous présentons un tour d'horizon des méthodes pertinentes. Ainsi que les travaux de recherche récents dans ce domaine.

Chapitre 3 : Le chapitre trois dresse un état de l'art pour l'intégration de données. Il aborde, la problématique d'intégration de données, ainsi l'hétérogénéité des données et les types de conflits qui peuvent êtres rencontrés dans une tâche d'intégration. Ensuite, nous présentons le concept du système d'intégration : son architecture et ces tâches, ainsi que le processus d'intégration de données. Nous conclurons par une présentation, selon différents critères, des approches d'intégration.

Chapitre 4 : Ce chapitre présente notre approche d'extraction d'information qui nous permet d'extraire les informations à partir de plusieurs sources web pour alimenter un entrepôt de données médical. L'approche proposée est basée sur la conception d'un moteur d'inférence qui permet de suggérer une méthode adéquate pour chaque source web, en suite lancer l'extraction, via la méthode inférée, pour chaque source. Dans notre contexte nous considérons que le jugement d'une méthode d'*WIE* pour chaque source se fait vis-à-vis les contraintes des méthodes d'extraction, les critères des sources web, et les compétences de l'utilisateur.

Chapitre 5 : Dans le cinquième chapitre nous présentons la mise en œuvre de la méthode proposée, où nous présentons les expérimentations, visant à valider notre approche, réalisées.

CHAPITRE 1

FOUILLE DU CONTENU WEB

Sommaire

1.1. Introduction

L'information sur le web est disponible avec de grosse quantité, ce qui rend son traitement très difficile et complexe. Ainsi, avec l'apparition du web les utilisateurs ont bénéficié d'un accès à de multiples informations appartenant aux différentes sources. Ces informations sont souvent complètes et disponibles en grandes masses. Cependant, le web est basé sur un paradigme de navigation qui rend très difficile de collecter ces informations.

Pour répondre à ces besoins de récupération et 1' intégration d'informations, un ensemble d'architectures, de démarches et d'outils, certains nouveaux, d'autres existants depuis longtemps, a été regroupé sous le terme « Fouille Web ; *Web mining* en anglais».

Ce premier chapitre présente les concepts de fouille Web, où ses différentes tâches sont décrites. Le processus du Web Mining se divise, selon le type de données à exploiter, en trois catégories principales: Web Content, Web Structure et Web Usage Mining. Chaque catégorie a son propre processus d'exécution et ses différentes techniques d'application. Dans notre cas, nous nous intéressons en particulier au concept du Web Content Mining ; une grande partie de ce chapitre est consacrée pour ce dernier concept.

1.2. Fouille Web

1.2.1. Notions de base

a) Data Mining : «Data mining est l'analyse des ensembles de données d'observation (souvent volumineuses) pour trouver des relations implicites et pour résumer les données de différentes manières qui sont à la fois compréhensibles et utiles au propriétaire de données» [H. David et al, 2001].

b) Web mining

Définition : Bing Liu défini le web mining comme : « Le web mining vise à découvrir les connaissances utiles à partir de la structure des hyperliens web, du contenu des pages, et d'usage de données web. Bien que le web mining utilise plusieurs techniques du data mining, il s'agit pas de l'application pure de data mining, à cause de la nature des données web : hétérogénéité, semi-structuré, et non structurée » [Bing Liu, 2007].

Le processus du web mining est similaire au processus du data mining. La différence principale réside dans l'étape de collection de données. En effet, pour le data mining traditionnel, les données sont déjà extraites via les bases de données et sauvegardées dans un entrepôt de données. Par contre, pour le web mining : la collection de données est une tâche substantielle, qui implique un parcours d'un nombre grand de liens et le traitement des pages web ayant une nature semi-structurée.

1.2.2. Difficultés de la Fouille Web

Le Web a beaucoup de caractéristiques propres, ce qui rend l'extraction des informations et des connaissances utiles une tâche difficile. Nous allons présenter ces caractéristiques, trouvées dans les travaux de [B. Liu et K. C. C Chang, 2004] et [Bing Liu, 2007]:

- La quantité de données / informations sur le Web est énorme et toujours en croissance. On peut trouver des informations à propos presque de n'importe quelle chose sur le Web.
- Les Informations sur le Web sont hétérogènes, En raison de diversité des auteurs de pages Web. plusieurs pages Web peuvent présenter des informations identiques ou similaires en utilisant des syntaxes différentes ou même des formats complètement différents, ce qui rend intégration de l'information provenant de plusieurs pages un problème difficile.
- Une quantité importante d'informations sur le Web est liée. Il ya des liens parmi les pages dans un site, et à travers les différents sites. Ces liens servent comme un outil d'organisation de l'information et aussi comme indications de confiance / autorité dans les pages et les sites liés.
- Tous les types de données peuvent exister sur le Web, par exemple, des tableaux structurés, textes, données multimédia (par ex. images, audio, et vidéo), etc.
- Le Web est bruyant. Les bruits viennent de deux sources principales :
    1. Tout d'abord, une page Web contient plusieurs blocks, par exemple, le contenu principal de la page, les annonces, les panneaux de navigation, et les mentions de copyright, etc. Pour exploiter une page web, seulement une partie de page web est utile. Le reste est considéré comme bruits. Pour perfectionner l'analyse des pages web et l'extraction de données, le bruit doit être supprimé.
    2. Deuxièmement, du fait que la qualité de l'information sur Web n'est pas contrôlée, à savoir, on peut écrire presque tout ce qu'on veut, une grande quantité d'informations sur le Web est de faible qualité, erronée, voire trompeuse.
- Le Web est également conçu pour fournir des services au gens. De nombreux sites Web et pages permettent aux gens d'effectuer des opérations avec les paramètres d'entrée, par exemple : l'achat des produits, de payer des factures et de remplir des formulaires. Autrement dit, ils fournissent des services.
- Le Web est dynamique : les informations sur le Web se changent fréquemment. Garder l'observation avec la moindre modification sur une source web est une condition importante pour le web mining.

➢ Avant tout, le Web est une société virtuelle. Il est non seulement conçu pour présenter des informations et des services, mais aussi pour offrir des interactions entre les personnes, les organisations et les systèmes automatiques : On peut communiquer avec les gens à n'importe où dans le monde facilement et instantanément, et aussi exprimer nos points de vue à propos n'importe quelle chose dans les forums internet, les blogs et les sites d'examen.

➢ Une grande partie d'informations sur le Web est semi-structuré en raison de la structure hiérarchique du code HTML, et la nécessité des designers des pages Web de présenter des informations dans un mode simple et régulier pour faciliter la visualisation et la navigation humaine.

➢ Le Web se compose de deux niveaux : le niveau visible (ou le web de surface), et le niveau invisible (ou le web de profond) :

1. Le niveau visible est composé des pages qui peuvent être parcourues en utilisant un navigateur Web normal. Ce niveau est également consultable via moteurs de recherche populaires.
2. Le niveau invisible est principalement composé de bases de données qui ne sont accessibles que par le biais de requêtes paramétrées en utilisant les formulaires.

1.2.3. Tâches de la Fouille Web

D'après [R. Malarvizhi et K. Saraswathi, 2013] et [S. Balan et P. Ponmuthuramalingam, 2013], le processus du web mining doit être décomposé en ces sous-tâches :

1. *Parcours des ressources web* : La tâche de récupération de documents Web destinés.
2. *Sélection de l'information et prétraitement* : Sélection automatique et prétraitement des informations spécifiques à partir de documents Web récupérées.
3. *Généralisation* : découvre automatiquement les schémas généraux sur les sites Web individuels, ainsi que sur plusieurs sites.
4. *Analyse* : Validation et / ou l'interprétation des connaissances minées. La figure 1.1 montre le processus du Web mining :

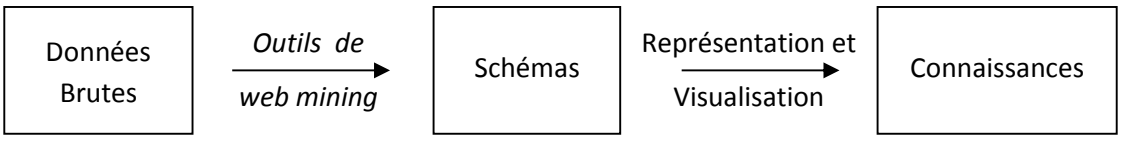


Fig 1.1. Processus web minig [S. Balan et P. Ponmuthuramalingam, 2013].

1.2.4. Taxonomie de la Fouille Web

En général, le web mining peut être devisé en trois catégories distinctes selon le type de données à explorer. La figure 1.2 illustre une telle taxonomie [J. Srivastava et al, 2005]

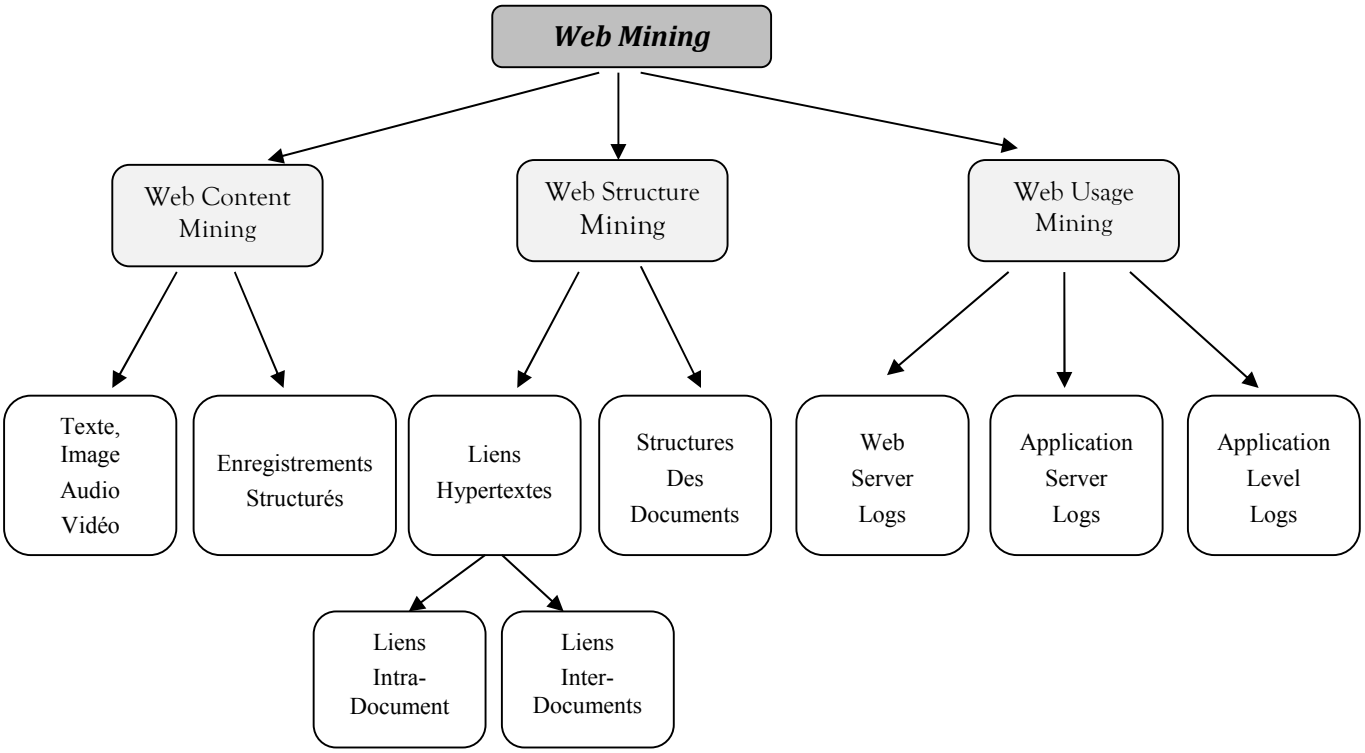


Fig 1.2. Taxonomie du Web mining.

Nous fournissons un bref aperçu des catégories : web usage mining et web structure mining, la catégorie web content mining sera bien détaillée.

A. <u>Web usage mining</u> : est l'application des techniques du data mining pour la découverte des modèles d'usage à partir des données de web, dans une perspective de comprendre et mieux servir les besoins des applications à base de web. Les données d'usage capturent l'identité des utilisateurs web ainsi que leurs comportements de navigation [S. BELLAOUAR, 2009].

Le web usage mining peut être classifié selon le type des données d'usage [J. Ananthi, 2014]:

- Les données d'un serveur web.
- Les données d'un serveur d'application.
- Les données au niveau des applications.

❖ *Le processus d'application* : Suivant [R. Malarvizhi et K. Saraswathi, 2013], Le WUM[1] comporte quatre étapes principales : La Collection de données, le prétraitement, le Clustering, et l'analyse (l'interprétation) des motifs :

1) La Collection de données : qui comprend les fichiers logs[2] de serveur Web, de serveur proxy et de navigateur. Et aussi les profils utilisateur, les sessions ou les transactions, les requêtes, les données d'enregistrement, les Cookies, les clics et défilements de la souris, ou n'importe quelle autre donnée comme résultat de l'interaction.
2) Le prétraitement : il s'agit de la fusion des données obtenu dans l'étape précédente, nettoyage et structuration, et parfois stockage dans une base de données.
3) Le Clustering : Une fois les données transformées en séquences d'URLs ou actions, les méthodes classiques de la fouille des données peuvent être appliquées. Parmi les méthodes de data mining, le Clustering est le plus utilisé dans les différents projets par des chercheurs pour trouver les modèles d'utilisation ou profils d'utilisateurs. Les algorithmes de clustering sont utilisés pour trouver les groupes d'utilisateurs.
4) Analyse et interprétation : Les résultats d'un processus de fouille de données sont, par définition, impossibles à estimer à l'avance, car l'objectif d'un tel processus est de découvrir des "nouvelles connaissances". Dans l'étape d'analyse et d'interprétation qui clôt le processus WUM, les patrons de visite découverts par la fouille de données doivent être analysés (idéalement de façon automatique) afin de garder les plus pertinents.

Techniques principales: nous détaillerons, dans cette section, les techniques principales de fouille de données utilisées dans le processus du Web Usage Mining :

- Les règles d'association : Il s'agit par exemple d'un ensemble de requêtes au cours d'une même session, exprimées sous forme d'implications logiques, ayant une probabilité p ; par exemple : X% des utilisateurs ayant visité les pages P1, P2 ont aussi visité P3. L'utilisation de ces résultats peut alors varier, selon la pertinence des règles obtenues et les besoins du propriétaire du site.
- Les motifs séquentiels : La recherche de motifs séquentiels présente de fortes similitudes avec celle des règles d'association. En effet, elle est dérivée de celle des règles d'association par l'introduction du temps (le temps qui sépare les événements entre eux et qui donne un aspect séquentiel aux informations contenues dans la base de données).

[1] *Web Usage Mining*
[2] *Un fichier log web contient les requêtes faites au serveur web, enregistrées en ordre chronologique.*

L'introduction de cette notion de temps peut être, selon le point de vue adopté, un avantage ou un inconvénient : l'ordre induit peut masquer des règles d'association, comme celles-ci peuvent détruire le caractère séquentiel de la visite.

- La classification : Elle est différente des deux précédentes méthodes, la classification a pour objectif de grouper en classes les utilisateurs ou les sessions d'un fichier log2. Le but peut être de permettre au plus tôt la catégorisation d'un utilisateur, afin de lui proposer, dès sa visite, des aides adaptées à son profil.

B. Web Structure Mining : la structure d'un graphe du web consiste en pages web en tant que nœuds et les hyperliens en tant que arcs reliant les différentes pages. Le web structure mining est le processus de découverte de l'information de la structure à partir du web [S. BELLAOUAR, 2009].

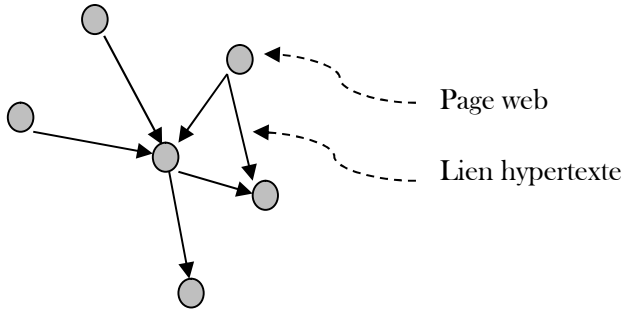


Fig 1.3. Le graphe du Web.

Lien hypertexte : c'est une unité structurelle qui connecte un emplacement dans une page web à un autre, soit dans la page elle-même, soit dans une autre page. Le lien qui connecte à une partie différente d'une même page est appelé *lien intra-document*, et celui qui connecte deux différentes pages est appelé *lien inter-document*.

Les buts du Web Structure Mining sont :

- ✓ Trouver des structures web intéressantes.
- ✓ Savoir la qualité d'une page (ex. nombre des liens vers elle).
- ✓ Trouver les pages reliées.
- ✓ Détection des pages dupliquées.

1.3. Fouille Du Contenu Web

Dans la section précédente, nous avons présenté le web usage mining, et le web structure mining. Notre thème de recherche entre dans le cadre du web content mining, cette catégorie est présentée dans le reste de ce chapitre.

1.3.1. Définition

Le contenu d'un document correspond aux informations dont la page web est conçue pour les transférer aux utilisateurs.

Le web content mining est le processus d'extraction d'information utile à partir du contenu des documents web. Ça peut être du texte, des images, des vidéos ou des enregistrements structurés comme les listes et les tables [G.M. Upadhyay et K. Dhingra, 2013].

1.3.2. Approches

Deux approches sont utilisées pour le web content mining [R. Malarvizhi et K. Saraswathi, 2013] et [S. Balan et P. Ponmuthuramalingam, 2013]: approche basée-agents, et approche de base de données.

L'approche basée-agents consiste à utiliser trois types d'agents pour faire le web content mining :

1. Agents de recherche intelligente : qui, suivant une requête, recherche automatiquement les informations en utilisant les caractéristiques du domaine et les profils d'utilisateur.
2. Agent pour filtrage/ catégorisation d'informations : qui utilise un ensemble de techniques pour filtrer les informations.
3. Agents de web personnalisé : ils sont utilisés pour apprendre les préférences d'utilisateurs, en suite découvrir les informations suivant ces préférences.

La seconde approche (dite de base de données) vise à modéliser les données sur le Web en une forme plus structurée, afin d'appliquer les mécanismes standards d'interrogation de base de données, et les applications de fouille de données pour l'analyser.

1.3.3. Domaines de recherche

Beaucoup de travaux de recherche ont été réalisé dans le domaine du web content mining, dans cette section on va présenter en bref les différents domaines de recherche dans le cadre du web content mining [B. Liu et K. C. C. Chang, 2004] [B. Liu, 2007] [S. Balan et P. Ponmuthuramalingam, 2013]:

1.2.3.1. Aspiration du web (Web Crawling, en anglais)

Un aspirateur web est un programme automatisé (ou un script) qui va scanner (« to crawl » en anglais) l'ensemble des pages Internet (ou d'un site particulier) dans le but de créer un index de pages le plus exhaustif possible. On entend parfois parler de « web spider », « web robot », « crawler » ou encore « automatic indexer » pour désigner l'aspirateur web. Les applications du web content mining ont recours aux aspirateurs web pour collecter les pages web d'une ou de plusieurs source(s).

Fonctionnement : De façon conceptuelle, l'algorithme permettant d'implémenter un crawler est très simple. Dans sa version la plus élémentaire, celui-ci n'utilise qu'une seule structure de données (une file d'attente) qui contient un ensemble d'adresses web connues. La structure ne contient au départ qu'un petit nombre d'adresses web appelées candidats initiaux (peut être une seule adresse). Pour chacune de ces adresses, le programme télécharge la page web associée et identifie les liens qu'elle contient. Ce processus est récursif : pour chaque page analysée, les adresses nouvelles trouvées sont rajoutées à la file d'attente. Le processus continu jusqu'à satisfaire la condition d'arrêt : celle-ci peut être : le parcours de toutes les pages d'une source web, de plusieurs sources, ou autres conditions.

Dans la littérature, plusieurs travaux sont réalisés dans ce domaine. Par exemple : [A. Billard et al, 2013], [L.J. Cui et al, 2013], et plusieurs versions d'aspirateurs sont issues : des aspirateurs peuvent analyser le code HTML simple, d'autres sont conçus pour pouvoir analyser en plus les morceaux AJAX[1]. Des aspirateurs téléchargent la page web en suite font l'analyse, d'autres font le téléchargement et l'analyse en parallèle. Des aspirateurs utilisent un seul thread pour l'aspiration, et d'autres travail en multithread. La figure suivante présente une architecture générale d'un aspirateur web :

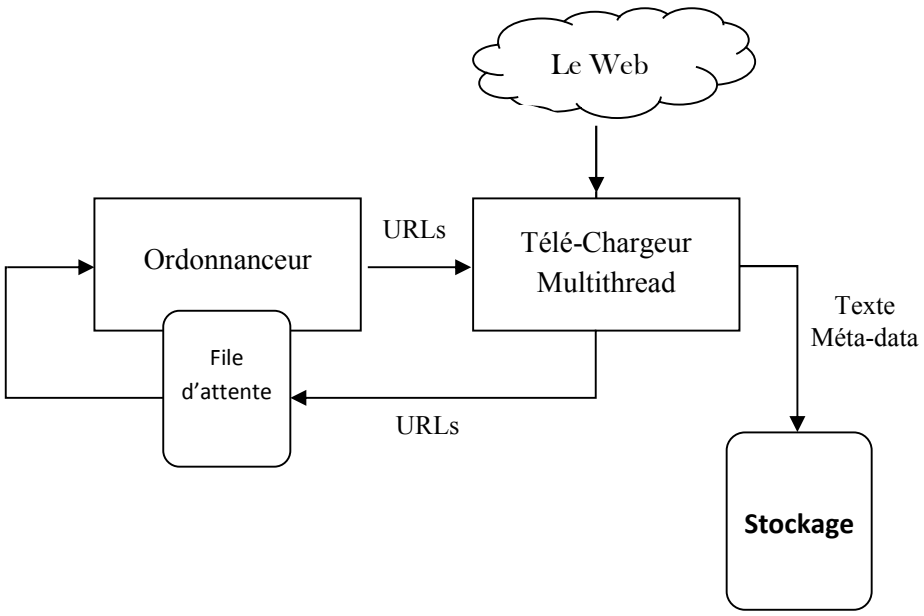


Fig 1.4. Composants logiciels d'un crawler [A. Billard et al, 2013].

Cette représentation assez haut niveau représente la base requise pour créer un crawler extensible et efficace. L'utilisation d'un ordonnanceur est pour prioriser les URLs à visiter, permet de mieux gérer les différents temps morts entre le téléchargement des pages.

[1] *AJAX (Asynchronous Javascript And XML) : C'est la technologie qui permet de modifier le contenu des pages web, en évitant la transmission et l'affichage d'une nouvelle page complète.*

1.2.3.2. Extraction d'information structurée

C'est le sujet de recherche le plus largement étudié du Web content mining. Une des raisons de son importance et de popularité est que les données structurées sur le Web sont souvent très importantes car elles représentent l'essentiel du contenu des pages web, par exemple, des listes de produits et services. L'extraction de telles données permet de fournir des services importants. Les données structurées sont également plus faciles à extraire par rapport aux textes non structurés. Le principe d'extraction se base sur la génération d'adaptateur (Wrapper en anglais), il s'agit d'un programme qui génère les règles d'extraction, ces dernières sont appliquées sur les pages web, et donnent comme résultat l'information utile qui se trouve dans les pages HTML. Ce sujet d'extraction d'information structurée sera abordé en détail dans le chapitre suivant.

1.2.3.3. Extraction d'information non structurée

La plupart des pages Web peuvent être considérés comme des documents texte. L'extraction informations de documents Web a également été étudiée par de nombreux chercheurs. Ce domaine est lié à la recherche d'information et le traitement du langage naturel.

1.2.3.4. Intégration de l'information Web et mise en correspondance des schémas

L'explosion du nombre de sources web multiplie les besoins de techniques d'intégration de ces sources autonomes et hétérogènes. L'intégration des données est le processus par lequel plusieurs sources de données autonomes, réparties et sous forme hétérogène (où chaque source est associée à un schéma local) sont intégrées sous forme de source unique représentée par un schéma global. Deux axes de recherche se présentent dans le cadre de l'intégration des sources web : (1) intégration des interfaces d'interrogation (Formulaires) web, pour pouvoir interroger plusieurs bases de données web. (2) Mise en correspondance des schémas, il s'agit du problème classique d'intégration des bases de données selon l'approche matérialisée. Dans ce cas l'intégration se fait après l'extraction des données pour chaque source web. Le chapitre trois sera consacré pour étudier le problème d'intégration de données web.

1.2.3.5. Extraction d'opinion à partir de sources en ligne

Le web a changé la façon dont les gens expriment leur avis. Ils peuvent désormais poster des critiques de produits sur les sites marchands et expriment leurs points de vue dans les forums Internet, des groupes de discussion, blogs, etc., qui sont communément appelés le *contenu généré par l'utilisateur* ou le *média généré par utilisateur*. Ce comportement en ligne représente des sources nouvelles d'informations, mesurables avec de nombreuses applications pratiques. En effet, les entreprises veulent toujours savoir les opinions du public ou des consommateurs à propos de leurs

produits et services. Les clients potentiels veulent aussi connaître les opinions des utilisateurs existants avant qu'ils utilisent un service ou achètent un produit. En outre, l'exploitation des opinions peut également fournir une information précieuse pour le placement de publicités dans les pages Web. Si par exemple, les gens expriment dans une page des opinions ou des avis positifs à propos d'un produit, alors déposer une annonce du produit est une bonne idée.

Des techniques sont en cours d'élaboration pour exploiter des sources d'opinions pour aider les entreprises et les individus à acquérir ces informations facilement et efficacement. Dans cette section on va présenter quelques méthodes pour le traitement des documents exprimant des avis :

1. Classification des avis : dans cette méthode, la fouille d'opinions sur le Web est considérée comme un problème de classification de texte. Elle classe un texte d'évaluation comme étant positive ou négative. Cette méthode ne traite que les avis, les informations concernant les gens ne sont pas prises en compte.

2. Fouille d'opinions basée- caractéristiques : Cette méthode traite le niveau phrase pour découvrir les détails, à savoir, quels sont les aspects d'un objet que les gens aimaient ou détestaient. L'objet peut être un produit, un service, un sujet, un individu, une organisation, etc. Par exemple, dans un forum de discussion pour un produit, cette méthode identifie les caractéristiques de produit qui ont été commentés par les clients et détermine si les commentaires sont positifs ou négatifs.

3. Comparaison des phrases et détection de relations : la comparaison est un autre type d'évaluation, qui compare directement un objet avec un ou plusieurs autres objets similaires. Par exemple, la phrase suivante compare deux médicaments : « le médicament **A** traite la grippe mieux que le médicament **B** ». On veut identifier ces phrases et d'en extraire les relations de comparaison.

1.2.3.6. Construction d'hiérarchie des concepts

En raison de la taille énorme des données sur le Web, l'organisation de l'information est évidemment une question importante. Bien qu'il soit difficile à organiser tout le Web, il est possible d'organiser les résultats de la recherche sur le Web suite à une requête donnée. Une liste linéaire de pages classées produites par les moteurs de recherche est insuffisante pour de nombreuses applications. La méthode standard pour l'organisation de l'information est le concept d'hiérarchie et / ou catégorisation. Les ontologies jouent un rôle primordial dans l'organisation de l'information.

Par exemple, dans [A. J. Shaikh, V. L. Kolhe, 2013] les auteurs ont proposé un Framework pour le web content mining basé sur la sémantique, en utilisant les ontologies et le SPARQL [1]. Il

[1] *SPARQL est un langage de requête et un protocole qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponibles à travers Internet*

s'agit d'un système de recherche intelligent qui construit une base de données ontologique puis l'interroger via le langage SPARQL. L'idée de base derrière ce Framework est de réduire le fossé entre les pages web présentant un contenu du langage naturel et les raquettes d'utilisateurs. La figure suivante présente l'idée générale du Framework proposé :

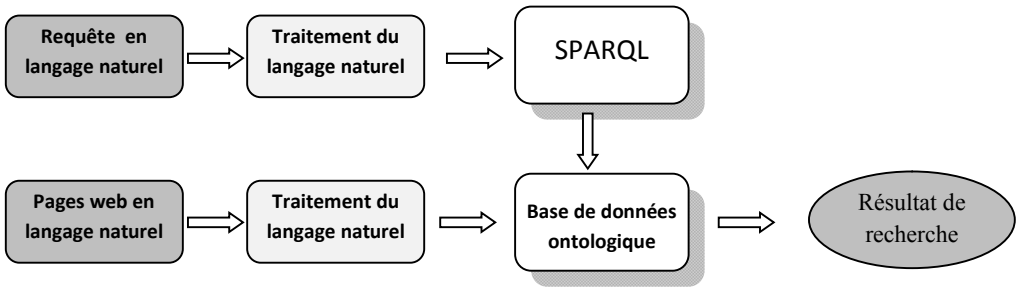


Fig 1.5. Le schéma général du Framework proposé.

1.2.3.7. La segmentation des pages Web et détection de bruits

Une page Web typique se compose de plusieurs blocs ou zones, par exemple : les zones de contenu principal, les zones de navigation, des publicités, etc. Il est utile de séparer automatiquement ces zones pour plusieurs applications pratiques. Par exemple, dans la fouille web, l'identification des zones de contenu principal, et l'élimination des blocs bruyant (les publicités, les panneaux de navigation, etc.) permet d'obtenir des meilleurs résultats.

Plusieurs méthodes ont été proposées pour segmenter une page web en régions ou blocs. Par exemple : dans l'approche de segmentation à base de DOM[1], un document HTML est représenté sous forme d'un arbre DOM. Ce dernier fournit une structure utile pour une page Web. Les nœuds de cet arbre sont des balises HTML telles que TABLE et P. Une autre méthode intuitive pour la segmentation des pages web est basée sur le schéma visuel d'une page web. De cette manière, une page Web est généralement séparée en cinq régions: haut, bas, gauche, droite et centre. Plusieurs travaux ont été proposés dans le cadre de la segmentation des pages et détection de bruits. Ci-dessous nous allons présenter un des travaux :

Dans [R. Song et al, 2004], les auteurs ont proposé un modèle pour attribuer automatiquement des valeurs d'importance aux blocs dans une page Web. Ils ont défini l'estimation de l'importance des blocs en tant que un problème d'apprentissage. Dans ce qui suit nous allons présenter leur travail :

[1] *DOM (Document Object Model) il s'agit d'une description des documents XML sous frome hiérarchique.*

- Tout d'abord, ils ont effectué un sondage pour étudier l'importance des blocks dans une page web, pour ce faire ils ont effectué une étude sur un nombre considérable de pages de 405 site web, ils ont demandé aux gens d'effectuer un niveau d'importance aux blocks pour chaque page web (cinq niveau d'importance sont définis).
- Puis les caractéristiques spatiales (comme position et de taille) et les caractéristiques de contenu (tel que le nombre d'images et de liens) sont extraites pour construire un vecteur de caractéristiques pour chaque bloc.
- Basé sur ces caractéristiques, des algorithmes l'apprentissage sont utilisés pour former un modèle à accorder une valeur d'importance aux différents segments de la page Web. Il s'agit d'une fonction de correspondance entre les caractéristiques et l'importance pour chaque block. Cette fonction est formalisée comme :

  < Paramètres de block > → < importance de block >

- Dans leurs expériences, le meilleur modèle peut atteindre une performance avec Micro-précision de 85,9%, ce qui est assez proche de la vision d'une personne.

1.3.4. Techniques du Web content mining

Plusieurs techniques sont utilisées pour collecter les données à partir du contenu des ressources web. Les principales techniques sont : l'extraction des modèles d'association, le clustering (regroupement) des documents web et la classification des pages web. Cependant, Plusieurs recherches ont été publiées pour chaque technique. Dans cette section nous allons présenter ces techniques et quelques travaux pour chacune.

1.3.4.1. Clustering (Regroupement)

Le Clustering est l'opération qui consiste à regrouper les individus d'une population en un nombre limité de groupes. Les clusters (ou partitions) ont deux propriétés : D'une part, ils ne sont pas prédéfinis, mais découverts automatiquement au cours de l'opération, contrairement aux classes de la classification. D'autre part, les clusters regroupent les individus ayant des caractéristiques similaires et séparent les individus ayant des caractéristiques différentes (homogénéité interne et hétérogénéité externe).

Il s'agit d'une tâche d'apprentissage "non supervisée" car on ne dispose d'aucune autre information préalable que la description des exemples. Après application de l'algorithme et donc lorsque les groupes ont été construits, d'autres techniques ou une expertise doivent dégager leur signification et leur éventuel intérêt.

Une formulation du Clustering est présentée dans [R. Etemadi et N. Moghaddam, 2010] : Considérons l'ensemble $X = \{x_1, x_2, ..., x_n\}$ comprenant n objets, le but de Clustering est de grouper les objets en k Clusters comme $C = \{c_1, c_2, ..., c_k\}$ Tel que :

(1) $c_1 \cup c_2 \cup ..... \cup c_k = C$

(2) $\cap c_i = \emptyset$ $_{i=1....k}$

Les documents web sont divisés en groupes suivant les métriques de similarité, ces dernières sont des caractéristiques à extraire à partir du contenu des pages web. Dans [S. Alcic et S. Conrad, 2011] les auteurs ont classé ces métriques en trois classes :

A. Distance à base de l'arbre DOM : dans cette classe, chaque document est présenté par son arbre DOM, la distance entre deux arbres est calculée en fonction de correspondance entre leurs nœuds.
B. Distance géométrique : dans cette classe, chaque document est partitionné en blocks, où chaque block est représenté par un rectangle. Ce dernier est présenté par ses coordonnées (coordonnées du centre, ou coordonnées des deux points diagonaux). La distance entre deux documents est calculée en fonction des distances entre les points représentants les différents blocks.
C. Distance sémantique : dans cette classe, chaque document est représenté par son contenu textuel. Pour calculer la similarité entre deux textes, plusieurs métriques de similarité existent dans la littérature. Par exemple : on peut considérer le niveau lexical, dans ce cas la distance est calculée en fonction de la correspondance entre les termes de deux documents. Le WordNet[1] est utilisé pour faire la correspondance entre synonymes.

Ces trois distances peuvent êtres combinées pour former une seule métrique de similarité.

Les algorithmes de Clustering existants peuvent être classés en deux grandes catégories : *Clustering par partition* et *Clustering hiérarchique*. Les algorithmes de Clustering par partition tentent à déterminer k Cluster, où il n'y a pas de points communs entre les Clusters ; chaque individu appartient exactement à un seul groupe (Cluster). Au contraire, le Clustering hiérarchique est une séquence de partitions dans lequel chaque partition est imbriquée dans la partition suivante. Un algorithme *d'agglomération* pour le regroupement hiérarchique commence avec l'ensemble disjoint des Clusters, qui place chaque individu dans un Cluster. Les paires d'individus ou de Clusters sont ensuite successivement fusionnées jusqu'à ce que le nombre de Clusters réduise à K. [M. N. Garofalakis et al, 1999].

[1] *WordNet : est une base de données lexicale consultable en ligne. Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise.*

1.3.4.2. Classification

La classification est une technique importante dans le data mining. Elle a été largement étudiée par la communauté de systèmes experts comme une solution possible au problème d'*acquisition* ou *d'extraction des connaissances.* Une définition de la classification des pages web est présentée dans [M.X. QI ET B.D. DAVISON, 2009] comme : La classification des pages web, ou comme reconnue la catégorisation des pages web, est le processus d'affectation des pages web à des catégories prédéfinies. Elle est traditionnellement posée comme un problème d'apprentissage supervisé dans lequel un ensemble de pages web classées sont utilisés pour apprendre un classificateur qui peut être ensuite utilisé pour classifier de nouvelles pages web.

Le problème général de classification des pages web peut être divisé en problèmes plus spécifiques : Classification des sujets (en thèmes), Classification fonctionnelle, et Classification des opinions. La classification en thèmes consiste à spécifier le thème (sujet) pour chaque page traitée, par exemple : Arts, Commerce, Sports…etc. La classification fonctionnelle consiste à décider le rôle de la page, par exemple : page d'accueil, page d'admission, page de recherche, ou page présentant un objet… etc. La classification d'opinions consiste à classifier l'opinion présentée dans une page web.

En se basant sur le nombre de catégories pour un problème, la classification peut être divisée en deux types : Classification binaire (le nombre de catégories est deux), ou classification multi-catégories (le nombre de catégories est supérieur à deux). En se basant sur l'organisation des catégories, la classification peut aussi être divisée en deux types : Classification plat, et Classification hiérarchique.

Plusieurs techniques et algorithmes sont utilisés pour la classification des pages web, dans [R. Malarvizhi et K. Saraswathi, 2013] les auteurs ont présenté les principaux algorithmes comme suivant :

- Arbre de décision (Decision Tree) : La technique de l'arbre de décision est employée en classement pour détecter des critères permettant de répartir les individus d'une population en n classes prédéfinies. On commence par choisir la variable (attribut) qui, par ses modalités, sépare le mieux les individus de chaque classe, de façon à avoir des sous-populations, que l'on appelle nœuds, puis on réitère la même opération sur chaque nouveau nœud obtenu jusqu'à ce que la séparation des individus ne soit plus possible ou plus souhaitable. Par construction, les nœuds terminaux (les feuilles) sont constitués d'individus d'une seule classe. L'ensemble des règles de toutes les feuilles constitue le modèle de classement. Cette méthode à été employée dans [M. Tsukada et al, 2001] pour catégoriser les pages web en sujets, les auteurs ont appliqué leur méthode dans Yahoo ! JAPAN.

- KNN (k-Nearest Neighbor) : La méthode des plus proches voisins (PPV en bref, Nearest Neighbor en anglais) est une méthode dédiée à la classification. Il s'agit d'une méthode de raisonnement à partir de cas. Elle part de l'idée de prendre des décisions en recherchant un ou des cas similaires déjà résolus en mémoire. Contrairement aux autres méthodes de classification, il n'y a pas d'étape d'apprentissage consistant en la construction d'un modèle à partir d'un échantillon d'apprentissage. C'est l'échantillon d'apprentissage, associé à une fonction de distance et d'une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle de classement. Dans [J.J. Lee et al, 2009] les auteurs ont utilisé un algorithme KNN pour classifier les pages web.

- Naive Bayes : La **classification naïve bayésienne** est un type de classification Bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. En termes simples, un classificateur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. La mise en œuvre de cette technique passe par les étapes suivantes :

1. L'ensemble des classes est construit préalablement, chaque classe doit contenir un ensemble d'individus. Pour chacun de ces derniers, l'ensemble des caractéristiques est extrait. En suite l'espérance de chaque caractéristique est calculée pour chaque classe.
2. Pour chaque nouvel individu, ses caractéristiques sont extraites en suite la probabilité que cet individu soit appartient à chaque classe est calculée en fonction de ses caractéristiques suivant la formule de Bayes. En fin l'individu est affecté dans la classe correspondant à la plus grande probabilité. Cette méthode a été employée dans [M. Ceci et D. Malerba, 2007] pour la classification des documents web en hiérarchie de catégories.

- Les réseaux de nuerons (Neural Network) : sont des outils très utilisés pour la classification. Ils sont issus de modèles biologiques, ils sont constitués d'unités élémentaires (les neurones) organisées selon une architecture calquée sur celle du cerveau, organisée en neurones et synapses connectés entre eux, chaque variable prédictive (attribut) correspondant à un nœud d'un premier niveau, appelé couche d'entrée, et chaque variable prédictive catégorique correspondant à un nœud de la couche de sortie. De façon générale, les étapes dans la mise en œuvre d'un réseau de neurones pour le classement sont : 1. L'identification des données en entrée et en sortie, 2. la normalisation de ces données, 3. la constitution d'un réseau avec une structure adaptée, 4. l'apprentissage du réseau, 5. le test du réseau, 6. L'application du modèle généré par l'apprentissage, 7. La dénormalisation des données en sortie. Une étude récente [S. Gupta et K.K. Bhatia, 2013] a été publiée, elle est basée sur les réseaux de neurones pour l'identification du domaine et la classification des pages web.

1.4. Conclusion

Nous avons présenté dans ce premier chapitre introductif les notions liées à la fouille Web. Nous avons abordé le concept général de fouille de données web. Nous avons introduit les caractéristiques qui font la fouille web une tâche difficile, ensuite les différentes étapes de la fouille web sont présentées. Nous avons mentionné qu'il ya trois catégories de la fouille web, suivant le type de données web à traiter : la fouille de structure web, la fouille d'usage web, et le web content mining.

Une grande partie de ce chapitre a été consacrée pour présenter le concept de fouille de contenu web (web content mining), à savoir : la définition, les approches, les domaines de recherche, et les techniques utilisées pour le web content mining. Cette présentation, bien que n'étant pas exhaustive car ce domaine est assez vaste aujourd'hui, introduit les éléments nécessaires qui nous permettent d'aborder les prochains chapitres.

Dans le chapitre suivant de ce travail, nous allons présenter le domaine de l'extraction d'information à partir du web, qui fait le sujet de recherche pour ce mémoire.

CHAPITRE 2

EXTRACTION D'INFORMATION A PARTIR DU WEB

SOMMAIRE

2.1. Introduction

Pour que les applications puissent exploiter les diverses et nombreuses informations disponibles sur le Web, ces informations doivent être extraites et transformées aux formats de représentation appropriés. Cette tâche est appelée Extraction d'Information à partir du Web (EIW). Le programme permettant d'effectuer une telle tâche est appelé extracteur, ou plus couramment ***WRAPPER***. Il s'agit d'une procédure fondée sur le principe du «correspondance des schémas, ou pattern-matching en anglais» qui, étant donné un ensemble de règles d'extraction, effectue une recherche de patterns (motifs) dans le code HTML de la page web en question. Ces motifs servent à déterminer une portion du code HTML qui représente l'information concernée par l'extraction.

Plusieurs méthodes ont étés proposée pour extraire de l'information du web, ou plus exactement pour générer des ***WRAPPERS*** : en rédigeant les règles d'extraction manuellement, en intégrant les techniques d'apprentissage (approches semi-automatiques), ou en exploitant les régularités dans les documents web (approches automatiques).

Dans ce chapitre, nous nous intéressons au problème de l'extraction de l'information provenant du Web. Le but derrière le processus de l'extraction est de définir un accès automatique aux données web. Dans un premier temps, nous présentons quelques notions de base pour ce domaine avant d'évoquer un bref historique. Nous mettons également l'accent sur les défis de l'extraction d'information à partir du web. Les différentes classifications des méthodes d'extraction sont ensuite présentées. Nous analysons aussi les métriques d'évaluation pour une tâche d'extraction. En fin, nous présentons une synthèse des méthodes pertinentes, ainsi que les travaux de recherche récents dans ce domaine.

2.2. Notions de base

2.2.1. Information présentée sur le web

Le Web offre une source importante d'informations, on peut trouver des médias (audio, vidéo), des documents de différents formats (TXT, DOC, PDF…etc.). Ou encore l'information présentée par les pages web et visualisée sur les navigateurs (HTML). Dans notre cas, nous nous intéressons par l'information présentée directement sur les pages web. Cette information est souvent semi-structurée, le contenu visuel est mélangé avec les symboles du langage de mise en format web.

Le langage HTML est le langage de mise en forme des pages Web. Il permet la création de documents plus riches que du texte plat, en décrivant à la fois la structure du document, son contenu et sa présentation. L'information des pages HTML est structurée par des paires de balises de mise en forme. Chaque paire est constituée d'une balise ouvrante, et d'une balise fermante.

Les pages Web sont produites manuellement ou automatiquement par programme. Dans ce cas elles intègrent souvent des informations provenant d'une base de données, comme par exemple les pages de résultats d'un moteur de recherche ou les pages d'un site de commerce. Lors de la transformation des enregistrements de la base en un (ou plusieurs) document(s) HTML, la structure des données est soit perdue soit rendue implicite par le balisage de mise en forme.

Les pages HTML possèdent une structure interne que nous appelons structure hiérarchique intra-page, grâce à l'utilisation de balises HTML qui permettent de définir des éléments de différentes granularités. Par exemple, la balise <P> définit un paragraphe, les balises <H1>, <H2>, <H3>, <H4>, <H5>, et <H6> définissent des sections…etc. La figure 2.1 montre un exemple simple de structure hiérarchique d'une page HTML :

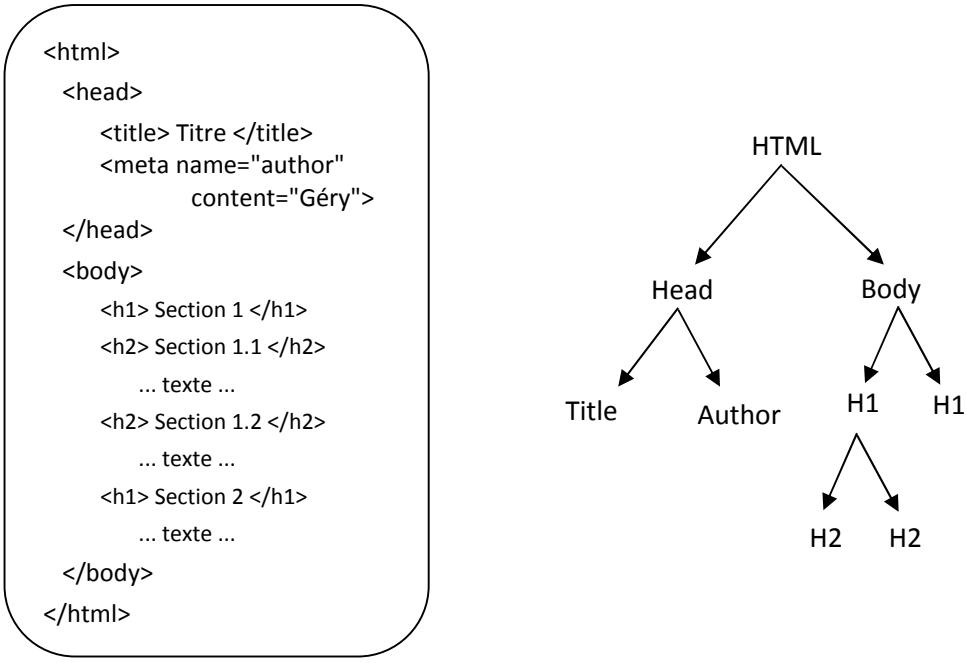


Fig 2.1. Structure hiérarchique d'une page HTML.

Donc, les documents Web sont différents des documents (textuel) traditionnellement utilisés dans l'extraction d'information. En effet, le volume est très grand, la syntaxe et le contenu des documents web se changent souvent, une grande partie de ces documents est structurées ou semi-structurées, et les documents peuvent contenir des informations de lien hypertexte. Par conséquent, les documents Web ont fournit de nouveaux défis pour le domaine d'extraction d'information.

2.2.2. Extraction d'information

Définition **:** Dans la littérature scientifique on rencontre plusieurs définitions de l'extraction d'information. [S. Tan et al, 2009] propose une définition plus précise et focalisée sur le Web : Pour les auteurs, L'extraction d'information est la technologie qui consiste à caractériser, dans les documents web structurés ou semi-structurés, les informations souhaitées par l'utilisateur, les mettre dans un format structuré, ensuite les intègre. C'est la tâche qui rend l'information web réutilisée.

Extraction d'information ou recherche d'information ?

Le but de la recherche d'information est de sélectionner un sous-ensemble de documents, dits pertinents, provenant d'une collection plus large suite à une requête de l'utilisateur. L'utilisateur doit en suite parcourir les documents retournés pour obtenir l'information désirée. Tandis que l'extraction d'information consiste à extraire une partie du contenu à partir de documents. Par conséquent, les deux techniques sont complémentaires [L. Eikvil, 1999].

2.2.3. Adaptateur

Le programme permettant de générer un ensemble de règles d'extraction, de les expérimenter, et de les appliquer sur un document Web est appelé adaptateur (Extracteur) (ou *WRAPPER* en anglais). Son conception est une technologie clé pour l'extraction et l'intégration d'information Web.

[E. Ferrara et al, 2014] définit un adaptateur Web comme: Une procédure, qui pourrait mettre en œuvre une ou plusieurs classes différentes d'algorithmes, qui cherche et trouve des données requises par un utilisateur humain, en les extraire à partir de sources Web non-structurée (ou semi-structurés), et de les transformer en données structurées, en suite fusionner et unifier ces informations pour un traitement ultérieur. Ces tâches s'effectuent d'une manière semi-automatique ou entièrement automatique.

[S. Zhang et P. Shi, 2009] propose une définition formelle d'adaptateur comme suite: Soit un ensemble de pages Web P = $\{p_2, p_1 \ldots . p_n\}$ de la source web S, alors un *WRAPPER* est un mappage W entre P et un résultat R (collection de données contenues dans P) W : P → R.

Le noyau d'adaptateur est les règles d'extraction, il s'agit des formules ou primitives qui s'appliquent sur un ensemble de pages pour localiser, dans ces pages, des morceaux précis. Étant donné que les règles d'extraction utilisées par les adaptateurs sont généralement dépendantes du Template de site web, c à d elles sont liées au niveau syntaxique du code HTML, alors les changements de Template des pages web conduisent au mal fonctionnement de l'adaptateur actuel,

Ce qu'est habituellement appelé le problème de maintenance. Dans ce qui suit, nous allons présenter le cycle de vie d'adaptateur :

Le cycle de vie d'adaptateur constitué de trois étapes [E. Ferrara et al, 2014]:

- *Génération d'adaptateur*: l'adaptateur est défini selon une méthode d'extraction.
- *Exécution d'adaptateur*: l'adaptateur est en marche, et extrait l'information du web de façon continue.
- *Maintenance d'adaptateur*: en cas de changement du Template de la source web, l'adaptateur doit être adapté pour qu'il continuer à travailler correctement.

La figure suivante présente le cycle de vie d'adaptateur :

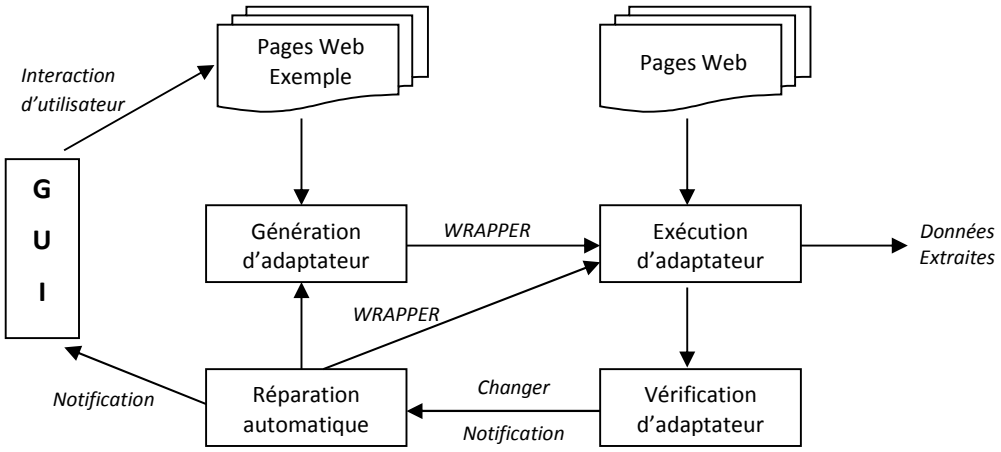


Fig 2.2. Cycle de vie d'adaptateur [S. Zhang et P. Shi, 2009].

A. Génération et exécution d'adaptateur :

La première étape dans le cycle de vie d'adaptateur est sa génération. Dans cette étape l'utilisateur connu la source web, à partir de laquelle, il veut extraire de l'information. Ensuite il choisit un outil d'extraction, parmi ceux qui existent. Il donne ensuite les entées de l'outil d'extraction : soit l'écrire manuelle des règles (pour les outils manuels), ou il fournit un ensemble de pages web étiquetées (pour les outils supervisés), soit il ne fait rien, sauf qu'il sélectionne la source web, (pour les outils purement automatiques). En fin, il donne l'ordre à l'outil pour générer l'adaptateur. Une fois ce dernier est généré et vérifié, il commence son exécution (extraction périodique d'informations à partir de la source web en question).

B. Maintenance d'adaptateur :

La génération d'adaptateur, quelle que soit la technique adoptée, est l'un des aspects du problème de l'extraction de données à partir du Web. D'autre part, la maintenance d'adaptateur est aussi importante, de tel sorte que l'extraction de données Web peut atteindre des niveaux élevés de

robustesse et de fiabilité. En fait, contrairement aux documents statiques, les pages web se changent et évoluent de façon dynamique, et leur structure peut se changer, parfois avec la conséquence que les adaptateurs existants ne font plus l'extraction des données avec succès.

Donc, on peut confirmer que la maintenance d'adaptateur est une étape critique dans le processus d'extraction de données web. Même si, cet aspect n'acquit pas beaucoup d'attention dans la littérature (Beaucoup moins que le problème de la génération d'adaptateurs). La première idée de maintenance d'adaptateurs s'était manuellement: les utilisateurs qui conçoivent généralement les *WRAPPERS* Web, font la mise à jour ou la réécriture de ces *WRAPPERS* à chaque fois la structure des pages Web est modifiée. La maintenance manuelle était bien pour les petits problèmes, mais devient impropre s'il s'agit d'un nombre important de sources web, ou si des changements fréquents se produisent sur ces dernières. Pour ces raisons, des tentatives d'automatiser la maintenance d'adaptateurs ont été confrontées dans la littérature récente.

2.3. Bref historique

Bien que la science de recherche d'information soit un domaine classique qui existe depuis longtemps aussi que les bases de données ont existé, le domaine de l'extraction d'information a été avancé dans les deux dernières décennies. Deux facteurs ont joué un rôle important pour le développement de ce champ [L. Eikvil, 1999] : la croissance exponentielle de la quantité de données textuelles, à la fois, en ligne et hors ligne. Et le ressort mis sur ce domaine par la Conférence (Message Understanding Conferences « MUC ») vers la fin des années 1980. L'histoire de l'extraction d'information, d'après [D.T. NGUYEN, 2006], peut être répartie en trois périodes ou intervalles temporelles : avant le programme MUC, pendant le programme MUC et après le programme MUC :

- **Première période : avant le programme MUC** : Initialement, l'extraction d'information ne concernait qu'un certain nombre de projets, comme par exemple le projet : LSP (Linguistic String Project), qui a été démarré au milieu des années 1960 et a durée jusqu'au début des années 1980 à l'université de New York. Ce projet consistait à développer une grammaire computationnelle de l'anglais pour créer des formes d'information régularisées (c'est-à-dire des motifs) dans le domaine médical.

  La caractéristique commune des projets de cette époque était l'application du remplissage des motifs avec l'information extraite à partir des textes en langues naturelles dont le traitement restait encore manuel, et propre aux domaines spécifiques.

- **Deuxième période : durant le programme MUC** : Dans cette époque, les recherches se sont concentrées sur l'extraction d'information à partir des données textuelles dans le but de résoudre des problèmes linguistiques. Trois tendances dominantes ont marqué cette deuxième période: l'adaptation de traitements linguistiques aux spécificités des systèmes, l'acquisition automatique des règles d'extraction et l'intégration de modules relativement indépendants. Cette évolution a été bien démontrée par les projets qui ont été présentés dans les différentes rencontres allant de MUC-4 (1992) à MUC-6 (1995).

- **Troisième période : après le programme MUC** : Dans cette période trois nouvelles tendances sont marqués: la portabilité des systèmes d'extraction d'information, l'extraction automatique des contenus et l'annotation pour le Web sémantique. Cette époque a connu une concurrence dans les recherches reposant sur le problème de l'extraction provenant du Web. Cependant, chacune de ces approches utilisent des méthodes différentes. Cette distinction réside dans la nature des données utilisées en entrée et le résultat de 1'adaptateur généré en sortie.

  Les premiers adaptateurs étaient construits manuellement. Dans ces approches, les règles d'extraction sont écrites par le concepteur humain. À cet effet, plusieurs outils ont été proposés afin de faciliter au concepteur la tâche de la construction des règles. Par exemple : Minerva, TSIMMIS, et W4F.

  La construction manuelle d'un adaptateur exige un haut niveau de précision et elle suppose que l'utilisateur soit un expert dans certains langages de programmation tels que Java ou Perl pour qu'il soit capable de créer les règles d'extraction. La tâche de la création des règles d'extraction peut devenir fastidieuse et très difficile à réaliser lorsqu'un domaine d'application nécessite un grand nombre de sources ou lorsque le format des données change au fur et à mesure dans le temps.

  Afin de répondre à ce problème, plusieurs approches sont apparues visant l'automatisation du processus de construction des règles. Chacune de ces méthodes utilise un outil assistant approprié pour la construction des règles d'extraction. Par exemple : SoftMealy, STALKER, IEPAD…etc.

2.4. Défis de l'extraction d'information du web

Dans sa formulation générale, le problème de l'extraction de données à partir du Web est difficile, car il est caractérisé par plusieurs contraintes. Les principaux défis que nous pouvons rencontrer dans la conception d'un système d'extraction d'information du Web peuvent être résumés comme suit [E. Ferrara et al, 2014]:

- Les techniques d'extraction de données Web nécessitent souvent l'aide d'experts humains. Un premier défi consiste de garantir un degré élevé d'automatisation, en réduisant les efforts humains autant que possible. L'interaction d'humain peut, cependant, jouer un rôle important pour élever le niveau de précision des données extraites.

  Un point critique consiste, par conséquent, à identifier un degré raisonnable entre la nécessité de réaliser une procédure hautement automatisée d'extraction d'information et l'obligation d'obtenir un degré élevé de performance, qui suppose que les informations extraites soient précises.

- Les techniques d'extraction de données Web doivent traiter de gros volumes de données dans un temps relativement court. Cette exigence est particulièrement stricte parce que l'analyse des informations extraites doit s'effectuer en temps opportun.

- Les applications Web dans le domaine social ou, plus en général, qui traitent des données à caractère personnel doit fournir des garanties de confidentialité solides. Par conséquent, le potentiel (même involontaire) tente de violer les informations privées d'utilisateur doit être rapidement identifié et bien neutralisé.

- Les approches reposant sur l'apprentissage machine exigent souvent un ensemble grand de pages Web manuellement étiquetées. En général, la tâche d'étiquetage de pages est coûteuse en temps, elle subit aussi des possibilités d'erreurs, ce qui implique la génération de faux adaptateurs.

- Souvent, un outil d'extraction de données Web doit extraire périodiquement des données provenant d'une source de données Web qui peut évoluer au fil du temps. Les sources Web sont toujours en évolution, et des changements structurels (au niveau de Template) se produisent sans préambule. Si de tels scénarios se produisent, alors les adaptateurs pourraient cesser de fonctionner correctement. Dans ce cas, la maintenance de ces adaptateurs sera indispensable.

2.5. Classification des méthodes d'extraction

Dans la littérature scientifique, plusieurs études ont été réalisées pour élaborer des taxonomies des différentes méthodes d'extraction. Chacune de ces études est basée sur un facteur pour classifier les méthodes d'extraction en catégories. Dans cette section, on va présenter les études principales dans ce contexte.

Dans [C.H. Chang et al, 2003] les auteurs ont comparé les méthodes d'extraction selon le degré d'automatisation de la méthode (ou le degré de l'intervention d'utilisateur pour générer l'adaptateur). En se basant sur ce facteur, quatre catégories sont définies : les méthodes manuelles, supervisées, semi-supervisées, et non-supervisées. Cette classification sera détaillée dans la section (2.8) de ce chapitre.

Dans [A.H.F. Laender et al, 2002] les auteurs ont proposé une taxonomie suivant la technique utilisée pour la génération d'adaptateur, à savoir : les langages de description, l'analyse de la structure du document Web, la modélisation de la structure du document Web, le traitement automatique des langues naturelles, le mécanisme d'induction, et à base d'ontologie.

✓ Adaptateurs à base de langues descriptives : Il est possible de définir des langues descriptives pour que l'utilisateur puisse rédiger des règles d'extraction directement à la main. Par exemple : Minerva, TSIMMIS, Web-OQL.

✓ Induction d'adaptateurs à partir des pages étiquetées : Ces méthodes dites supervisées semi-automatiques nécessitent une phase d'étiquetage de quelques pages exemples à utiliser pour la phase d'apprentissage. Celle-ci permet la construction des règles d'extraction généralisées. Pour induire de nouvelles règles d'extraction, la procédure compare les chaînes de caractères précédant et succédant les attributs de la donnée à extraire. Elle apprend donc les délimiteurs et les motifs d'extraction pour chaque attribut. Comme exemple, nous citons les systèmes : WIEN, STALKER, et SOFTMEALY.

✓ Génération d'adaptateurs par l'extraction de motifs-analyses de la structure du document : la stratégie employée par les méthodes de cette classe consiste à formater les pages HTML en arbres (ayant une structure hiérarchique), en suite, au constat de la régularité des séquences de balises suivant les mêmes formats dans la structure des documents, l'extraction de motifs via l'analyse de la structure des documents permet de générer des expressions (des motifs) décrivant le format général dans lequel se trouvent des données à extraire. Quelques systèmes peuvent être présentés dans cette approche comme par exemple W4F, XWRAP, ROADRUNNER et IEPAD.

✓ Génération d'adaptateurs par le traitement automatique des langues naturelles : Certains systèmes sont construits par l'utilisation des techniques de traitements automatiques des langues naturelles. Ces techniques sont appliquées dans les systèmes comme WHISK, RAPIER et SRV, pour apprendre des règles d'extraction des données existantes dans les textes. Ces règles sont basées sur des contraintes syntaxiques et sémantiques.

✓ Génération d'adaptateurs à partir des motifs : Cette approche consiste à construire un mappage entre des portions de données dans les documents web, et un schéma de données fourni par l'utilisateur. Deux méthodes se basent sur cette approche sont le système NoDoSE et le système DEByE.

✓ Génération d'adaptateurs à partir d'ontologie : Le principe pour cette approche est d'exploiter les connaissances décrites dans une ontologie pour extraire des informations à partir de documents web de même domaine. L'ontologie comporte un ensemble de termes du domaine, leurs caractéristiques lexicales, leur contexte d'apparition à l'aide de mots clés et de relations sémantiques entre les termes. La méthode BYU est la plus reconnue pour cette approche.

De plus de ces approches, [B. HABEGGER, 2004] cite les adaptateurs générés à partir de relations extraites.

✓ Adaptateurs générés à partir de relations extraites : Dans cette approche, l'objectif est de construire un ensemble de motifs permettant d'extraire un sous-ensemble des instances d'une relation donnée. Les motifs sont alors applicables sur l'ensemble des pages du Web dans sa globalité. Un exemple dans cette approche est le système DIPRE.

2.6. Evaluation des méthodes d'extraction

La nécessité des mesures d'évaluation pour l'extraction de l'information a été posée avec la Conférence (MUC). Trois mesures permettent d'évaluer une tâche d'extraction d'information [L. Eikvil, 1999] sont: Le *Rappel*, La *précision*, et Le *F-mesure*. Le rappel représente le pourcentage des informations correctes extraites, au total d'informations correctes. La précision représente le pourcentage des informations correctes extraites, au total d'informations extraites. La F-mesure évalue la qualité globale d'un extracteur en combinant sa précision et son rappel en une mesure unique. La formule des trois mesures est définie comme suit:

Soit :

EXT$_{CORRECTES}$: le nombre d'enregistrements corrects, extraits d'une page web.

EXT$_{INCORRECTES}$: le nombre d'enregistrements incorrects, extraits de cette page web.

Non-EXT$_{CORRECTES}$: le nombre d'enregistrements corrects, non extraits de cette page web.

Alors :

Définition (rappel): Le rappel de l'extracteur est défini par :

$$\mathbf{R} = \mathbf{EXT}_{CORRECTES} \ / \ (\mathbf{EXT}_{CORRECTES} + \mathbf{Non\text{-}EXT}_{CORRECTES})$$

Définition (précision) : La précision de l'extracteur est définie par :

$$\mathbf{P} = \mathbf{EXT}_{CORRECTES} \ / \ (\mathbf{EXT}_{CORRECTES} + \mathbf{EXT}_{INCORRECTES})$$

Par définition, ces deux mesures prennent leur valeur entre 0 et 1, mais par commodité on les exprime généralement sur une échelle de 0 à 100. Une précision et un rappel de 100 indique un extracteur parfait.

Lorsqu'on compare la performance des systèmes différents, la précision et le rappel doivent être considérés à la fois. Cependant, comme il n'est pas facile de comparer les deux paramètres en même temps, la F-mesure a été proposée. Une telle mesure combine la précision **P**, et le rappel **R**, en une seule mesure de la manière suivante:

$$F = \ (\beta^2 + 1)PR \ / \ (\beta^2 \ P + R)$$

Le paramètre β détermine le degré de favoriser le rappel sur la précision. Les chercheurs rapportent fréquemment le score F1 d'un système d'extraction d'informations où β = 1, qui implique un équilibre entre la précision et le rappel [L. Eikvil, 1999].

En plus de ces trois métriques, [W. Liu et al, 2010] a introduit une autre métrique pour mesurer la performance d'un algorithme d'extraction automatique d'information. Il s'agit de la métrique *Révision*, elle est définie par le pourcentage des bases de données Web dont les enregistrements ou les attributs ne sont pas parfaitement extraits, c'est à dire, soit la précision ou le rappel n'est pas à cent pour cent. Donc, cette mesure indique le pourcentage de bases de données Web pour lesquelles la solution automatisée n'obtient pas une extraction parfaite, et une révision manuelle est nécessaire pour compléter la solution.

Dans ce qui suite, nous allons illustrer l'importance de cette mesure via un exemple : Supposons qu'il existe trois méthodes (A1, A2, et A3) qui permettent d'extraire des enregistrements de données à partir pages Web, et elles utilisent le même ensemble de données (cinq bases de données Web, avec dix enregistrements dans chaque base de données Web). La table suivante présente les résultats obtenus :

Méthode	NB-enrg-extr Site1	NB-enrg-extr Site2	NB-enrg-extr Site3	NB-enrg-extr Site4	NB-enrg-extr Site5	*Rappel*	*Précision*	*F1*	*Révision*
A1	09	09	09	09	09	90%	100%	94,7%	100%
A2	11	11	11	11	11	100%	90,9%	95,2%	100%
A3	10	10	10	10	00	80%	80%	80%	20%

Table 2.1. Exemple d'illustration des métriques d'évaluation.

A1 extraits neuf enregistrements pour chaque site et ils sont tous corrects. Ainsi, la précision moyenne et le rappel pour A1 sont 100 et 90 pour cent, respectivement. A2 extrait 11 enregistrements pour chaque site et 10 sont corrects. Ainsi, la précision moyenne et le rappel pour A2 sont 90,9 et 100 pour cent, respectivement. A3 extrait 10 enregistrements pour quatre des cinq bases de données web et ils sont tous corrects. Pour le cinquième site, A3 n'extrait aucun enregistrement. Ainsi, la précision moyenne et le rappel pour A3 sont à la fois 80 pour cent. En se basant sur la précision et le rappel, A1 et A2 sont mieux qu'A3. Mais dans l'application réelle, A3 peut être le meilleur choix. Pour rendre la précision et le rappel cent pour cent, tous les adaptateurs générés par A1 et A2 doivent être réglés manuellement, alors que seulement un adaptateur généré par A3 doit être réglé manuellement. En d'autre terme, A3 a besoin d'une intervention manuelle minimale.

2.7. L'Extraction d'information combinée avec des domaines de recherche liés au web

Internet est un réseau très complexe, car il est constitué de millions de connexions utilisant des technologies très disparates. Ainsi, le monde d'internet offre un milieu idéal pour la recherche en informatique, ce qui a permis la naissance de nouveaux domaines et technologies. En combinant l'extraction d'information avec les dernières technologies innovantes et contemporaines dans le web, de nouvelles perspectives sont attendues. Dans cette section, on va présenter, en bref, la relation de l'extraction d'information avec le domaine de services web d'un côté, et avec le domaine de cloud computing d'un autre côté. Ainsi que les travaux réalisés dans ces champs.

2.7.1. L'extraction d'information et les Services Web

Un Service Web est souvent vu comme une application accessible à d'autres applications sur le Web. Il s'agit d'une entité logicielle permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués.

Définition :

Le consortium W3C [1] définit un Service Web comme étant : « une application, ou un composant logiciel qui vérifie les propriétés suivantes » [J. BOUHASSOUN et L. BOUHASSOUN, 2011]:

- Il est identifié par un URI [2];
- Ses interfaces et ses liens peuvent être décrits en XML ;
- Sa définition peut être découverte par d'autres Services Web ;
- Il peut interagir directement avec d'autres Services Web à travers le langage XML en utilisant des protocoles Internet standards.

[1] *World Wide Web Consortium*
[2] *Uniform Resource Identifier*

• *Modèle de base des Services Web* : Le modèle des Services Web repose sur une Architecture Orientée Service (SOA[1]). Celle-ci fait intervenir trois catégories d'acteurs : **le fournisseur de service**, **l'annuaire des services** et **le client**. Un fournisseur de service fournit un module logiciel accessible sur le réseau et définit une description pour ce Service Web. Ensuite, il le publie dans l'annuaire des services de telle sorte que le client peut le trouver. L'annuaire des services correspond à un registre de descriptions des Services Web offrant des facilités de publication des Services Web à l'intention des fournisseurs, ainsi que des facilités de recherche des Services Web à l'intention des clients. La description de service contient des informations telles que l'entrée du service, sa sortie et l'adresse où le service est situé. Le client interroge l'annuaire pour un certain type de service et récupère sa description. Ensuite, il utilise les informations dans la description du service pour se lier avec le fournisseur et d'invoquer le Service Web. Nous citons, notamment les standards émergents suivants :

- SOAP [2] : Définit un protocole de transmission de messages basé sur XML. Il permet l'échange d'informations à distance en utilisant le formalisme XML pour à la fois, définir les messages envoyés entre les applications et représenter les données échangées.
- WSDL [3] : Un standard fondé sur XML qui permet la description des Services Web.
- UDDI [4]: Fournit l'infrastructure de base pour la publication et la recherche des Services Web.

Le schéma suivant présente le principe de base des services web :

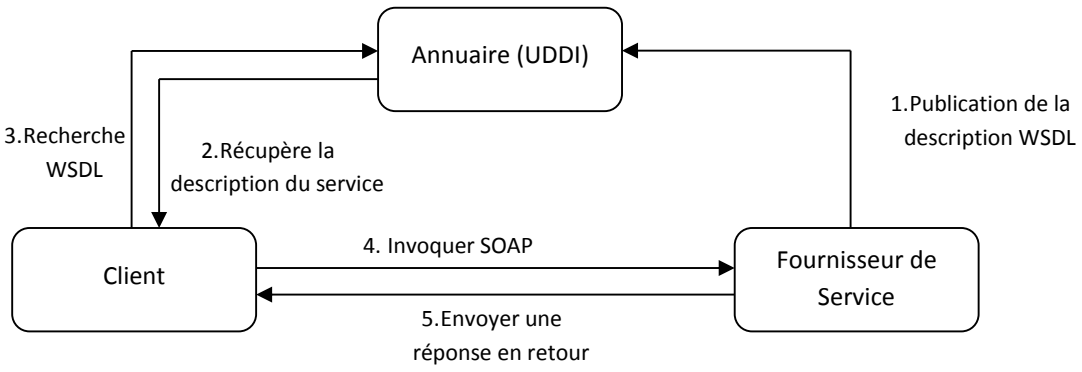


Fig 2.3. Modèle de base des Services Web

• *Services Web pour l'extraction d'information* : Dans l'intégration des Services Web et de l'extraction d'informations deux aspects sont intéressants [B. HABEGGER, 2004] : Le premier

[1] *Service Oriented Architecture*
[2] *Simple Object Access Protocol*
[3] *Web Service Description Language*
[4] *Universal Description Discovery and Integration*

consiste à considérer un Service Web comme une nouvelle source d'informations pouvant être exploitée dans l'exécution d'une tâche d'extraction d'informations. Le second consiste à considérer la tâche elle-même comme un Service Web.

1. Services Web comme source d'informations : Un Service Web peut être considéré au même titre qu'un site Web comme une source d'informations. Accéder à un Service Web consiste à construire un message SOAP correspondant à la requête soumise au Service Web et de poster ce message via un protocole de transport (HTTP, SMTP, etc.) à un endroit désigné. L'utilisation d'un document WSDL permet de s'abstraire des détails techniques de la sortie. Ainsi, avec une référence (URL) du WSDL du service, accéder à celui-ci revient à faire un appel de méthode avec un ensemble de paramètres.

   Pour permettre l'accès à un Service Web par une tâche d'extraction d'informations il suffit alors d'ajouter un opérateur *soap* faisant appel à un Service Web. Un tel opérateur se spécifie par les paramètres suivants :

   - une référence vers la description en WSDL du Service Web;
   - le nom de la méthode à appeler ;
   - un ensemble de valeurs pour les paramètres de la méthode.

   L'opérateur soap donne en sortie un document XML correspondant à l'enveloppe SOAP générée pour l'appel au Service Web. Celui-ci pourra alors être traité de la même manière qu'un document HTML obtenu par l'exécution d'une requête http qui consiste à sélectionner les éléments à extraire du document.

2. Services Web comme une tâche d'extraction d'informations : Un Service Web peut être simplement vu comme une bibliothèque de fonctions pouvant être appelées à distance. Or, une tâche d'extraction d'informations, peut être considérée comme l'application d'une fonction à un ensemble de données qui forment la requête de l'utilisateur. Ceci permet de rendre disponible l'exécution d'une tâche d'extraction d'informations à distance. Par exemple, dans [B. Habegger et M. Quafafou, 2004] les auteurs ont proposé un Framework pour décomposer une tâche d'extraction d'information et de construire un nouveau service web à partir de celle-ci. Ils ont aussi proposé un langage, basé sur XML, pour exprimer des services Web d'extraction d'information.

L'apparition des Services Web peut remettre en question la nécessité de systèmes d'extraction d'informations à partir du Web. En effet, si une information est disponible de manière structurée au

travers d'un Service Web, il est alors inutile de chercher cette même information sur un site Web classique. Toutefois, dans l'état actuel, la grande majorité des informations disponibles via le Web les sont au travers de documents HTML et non de Services Web. De plus, il n'est pas certain que tout le Web classique évolue vers un accès au travers de Services Web. En effet les Services Web sont orientés plutôt vers la communication entre organisations (Business to Business) alors que le Web classique reste la solution idéale pour communiquer avec l'utilisateur final (Business to Client). Ainsi, l'apparition des Services Web n'a pas pour vocation de remplacer les sources de données Web existantes [B. HABEGGER, 2004].

2.7.2. L'extraction d'information et le Cloud Computing

Le terme « Cloud Computing » se traduit littéralement par « informatique dans les nuages », ces nuages faisant référence à Internet et au Web. Il s'agit d'un concept apparu assez récemment, mais dont les prémices remontent à quelques années, notamment à la technologie des grilles de calcul, utilisée pour le calcul scientifique. Le Cloud Computing fait référence à l'utilisation de la mémoire et des capacités de calcul des ordinateurs et des serveurs répartis dans le monde entier, et liés par un réseau, tel Internet. Il permet donc d'accéder à des logiciels en ligne, sous forme d'abonnement, dans de nombreux domaines, mais aussi à des services de stockage et de calcul accessibles par Internet. Les utilisateurs du nuage pourraient ainsi disposer d'une puissance informatique considérable.

• *Utilisation des infrastructures du Cloud Computing pour l'extraction d'information* : la complexité incrémentée des procédures d'extraction d'information a poussé quelques auteurs à penser d'utiliser des services du Cloud Computing, comme par exemple EC2 d'Amazon, ce qui permet de rendre d'extraction d'information fiable et scalable. Pour cette raison, nous pensons que dans les prochaines années, un nombre croissant de plateformes d'extraction de données Web sera fondé sur les services de Cloud Computing [E. Ferrara et al, 2014].

Dans [Y. Shen et al, 2013], les auteurs ont proposé une approche parallèle avec une plateforme basée sur le Hadoop MapReduce pour l'extraction de données Web à grande échelle. Dans cette approche, le processus séquentiel d'extraction d'information est divisé en sous-tâches de telle sorte que ces tâches s'effectuent en parallèle. En premier temps la tâche d'extraction est partagée en sous-tâches (chaque sous-tâche concerne un ensemble de pages web à partir desquelles on veut effectuer l'extraction, ou un sous-ensemble des données recherchées…etc.), en suite chaque sou-tâche est affectée à un nœud dans le nuage. Chaque nœud effectue les trois étapes d'extraction (la navigation des pages, l'extraction d'informations et l'intégration des données). En fin, les résultats obtenus pour chaque nœud sont rassemblés. Cette technique permet d'accélérer le processus

d'extraction d'information à grande échèle, par exemple (pour un nombre de nœud égal à 10, les auteurs ont obtenu une accélération « SpeedUp » = 9.86).

2.8. Approches d'EIW selon le degré d'automatisation

Comme nous l'avons déjà dit, [C.H. Chang et al, 2003] a distingué quatre approches qui permettent d'élaborer un extracteur avec un degré décroissant de l'intervention de l'utilisateur. Dans cette section, on va présenter ces approches, ainsi quelques méthodes d'extraction pour chacune.

2.8.1. Construction manuelle d'adaptateur

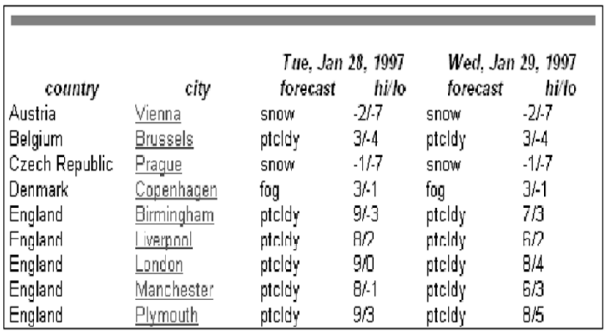
Elle consiste à construire manuellement l'ensemble des motifs ou règles d'extraction. Ceci est possible à l'aide d'un langage de programmation (comme Python ou java) et d'expressions régulières, qui permettent l'extraction de portions du code HTML. Les langages d'interrogation et de transformation, comme XSLT [1] et XQUERY[2], peuvent aussi êtres utilisés pour décrire les règles d'extraction manuellement. On peut citer comme exemple, les systèmes : TSIMMIS, WebOQL, et Minerva.

2.8.1.1. TSIMMIS

Dans [J. Hammer et al, 1997], les auteurs ont conçu un programme d'extraction d'information à partir de pages Web semi-structurées. Ce programme est configurable et il utilise en entrée un fichier de spécifications permettant de définir l'emplacement des données dans la page Web (source HTML), et comment ces données sont-elles stockées dans les objets. Ce fichier est composé de plusieurs commandes où chacune d'elle a la forme d'un triplé [*variables*, *source*, *motif*]; où : *source* représente le texte considéré contenant les informations à extraire, *variables* est l'ensemble des éléments à extraire, et *motif* les délimiteurs à appliquer pour extraire les différentes variables à partir de leur source.

Dans ce qui suit nous allons illustrer, par un exemple (présenté dans la même référence au-dessus), l'application de cette méthode. La *figure 2.4.a* présente la prévision des températures dans certaines villes. Il s'agit d'une page web au format HTML. Une partie de son code est présenté à coté dans la *figure 2.4.b*. Le fichier de spécification des règles d'extraction est présenté dans la *figure 2.4.c*, Chaque commande de ce fichier permet d'extraire une information pour la récupérer dans une variable réutilisable par la commande suivante. Prenons par exemple la première commande (lignes [l-2]) permettant d'extraire le contenu de la page Web et de l'enregistrer dans la variable root,

[1] *XSLT* (*eXtensible Stylesheet Language Transformations*) : *langage de transformation des documents XML*
[2] *XML Query ; Langage pour interroger les documents XML*



country	city	Tue, Jan 28, 1997 forecast	hi/lo	Wed, Jan 29, 1997 forecast	hi/lo
Austria	Vienna	snow	-2/-7	snow	-2/-7
Belgium	Brussels	ptcldy	3/-4	ptcldy	3/-4
Czech Republic	Prague	snow	-1/-7	snow	-1/-7
Denmark	Copenhagen	fog	3/-1	fog	3/-1
England	Birmingham	ptcldy	9/-3	ptcldy	7/3
England	Liverpool	ptcldy	8/2	ptcldy	6/2
England	London	ptcldy	9/0	ptcldy	8/4
England	Manchester	ptcldy	8/-1	ptcldy	6/3
England	Plymouth	ptcldy	9/3	ptcldy	8/5

Fig 2.4.a

```
1 [["root",
2 "get('http://www.intellicast.com/weather/europe/')","#"],

3 ["temperatures", "root",
4 "*<TABLE*<TABLE*</TR>#</TABLE>* "],

5 ["_citytemp",
6 "split(temperatures,'<TR ALIGN=left>')", "#"],

7 ["city_temp",
8 "_citytemp[1:0]", "#"],

9 ["country,c_url,city,weath_tody,hgh_tody,low_today,
10 weath_tomorrow,hgh_tomorrow,low_tomorrow",
11 "city_temp",
12 "*<TD>#</TD>*HREF=#>#</A>*<TD>#</TD>*<TD>#/#
13 </TD>*<TD>#</TD>*<TD>#/#*"  ]]
```

Fig 2.4.c

```
<HTML>
<HEAD>
<TITLE>INTELLICAST: europe weather</TITLE>
<A NAME="europe"></A>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=509>
<TR>  <TD colspan=11><I>Click on a city for local forecasts</I>
<BR></TD>  </TR>
<TR>  <TD colspan=11><I> temperatures listed in degrees celsius </I>
<BR></TD>  </TR>
<TR>  <TD colspan=11><HR NOSHADE SIZE=6 WIDTH=509>
</TD>  </TR>
</TABLE>
<TABLE CELLSPACING=0 CELLPADDING=0 WIDTH=514>
<TR ALIGN=left>
<TH COLSPAN=2><BR></TH>
 <TH COLSPAN=2><I>Tue, Jan 28, 1997</I></TH>
<TH COLSPAN=2><I>Wed, Jan 29, 1997</I></TH>
 </TR>
 <TR ALIGN=left>
<TH><I>country</I></TH>
<TH><I>city</I></TH> <TH><I>forecast</I></TH>
<TH><I>hi/lo</I></TH> <TH><I>forecast</I></TH>
<TH><I>hi/lo</I></TH>  </TR>
<TR ALIGN=left> <TD>Austria</TD>
<TD>
<A HREF=http://www.intellicast.com/weather/vie/>Vienna</A>
</TD>
<TD>snow</TD> <TD>-2/-7</TD>
<TD>snow</TD> <TD>-2/-7</TD></TR>
 <TR ALIGN=left> <TD>Belgium</TD>
<TD><A HREF=http://www.intellicast.com/weather/bru/>Brussels</A>
</TD>
<TD>fog</TD> <TD>2/-2</TD>
<TD>sleet</TD> <TD>3/-1</TD>
</TR>
.
.
</TABLE>
.
</HTML>
```

Fig 2.4.b

(Fig 2.4.a, Fig 2.4.b, Fig 2.4.c) Exemple d'application de la méthode TSIMMIS

Ensuite la seconde commande (lignes [3-4]) permet d'appliquer le motif "*<TABLE*<TABLE*</TR>#</TABLE>*" sur la source root pour extraire des nouvelles informations. Ces informations seront stockées dans la nouvelle variable temperatures, le symbole # représente la partie à extraire. La troisième commande (lignes [5-6]) permet de découper la source temperatures en fragments de texte selon le séparateur '<TR ALIGN=left>', ensuite la quatrième commande (lignes [7-8]) enregistre l'ensemble de fragments de texte dans le tableau city_temp. Finalement, la dernière commande (lignes [9-13]) applique le motif "*<TD>#<ITD>*HREF=#>#</A>*<TD>#<ITD>*<TD>#I#<ITD>*<TD>#<ITD>*<TD>#I# *"sur les différentes cellules du tableau afin d'extraire les informations élémentaires désirées (country,c_url ,city,weath_tody,hgh_tody,low_today,weath_tomorrow,hgh_tomorrow,low_to morrow) .

Cette méthode suit un processus hiérarchique récursif résidant dans le fait que l'extraction d'une telle information atomique nécessite l'extraction successive de l'ensemble des parties engendrant celle-ci.

2.8.1.2. WebOQL

Dans [G.O. Arocena et A.O. Mendelzon, 1998], les auteurs ont développé le système WebOQL. Il s'agit d'un langage fonctionnel qui peut être utilisé comme un langage de requêtes pour les données semi-structurées. La structure de données principal fournie par WebOQL est l'hyper-arbre. Ce dernier est un arbre ordonné, dont les arcs sont étiquetés. Il peut être utilisé pour modéliser une page web, une table relationnelle, une hiérarchie de répertoires… etc. L'exemple suivant, présenté dans [C.H. Chang et al, 2006], montre un cas d'utilisation de ce système. La figure (*Fig.2.5.b*) montre l'hyper-arbre pour la page web présentée dans la figure (*Fig.2.5.a*).

```
<html><body>
<b> Book Name </b> Data Mining
<b>  Reviews </b>
<ol>
< li>
<b> Reviewer  Name  </b> Jeff
<b> Rating </b> 2
<b> Text </b> ...
< /li>

< li>
<b> Reviewer Name </b> Jane
<b> Rating </b> 6
<b> Text </b> ...
< /li>
</ol>
</body></html>
```

Fig 2.5.a (*pe2.html*)

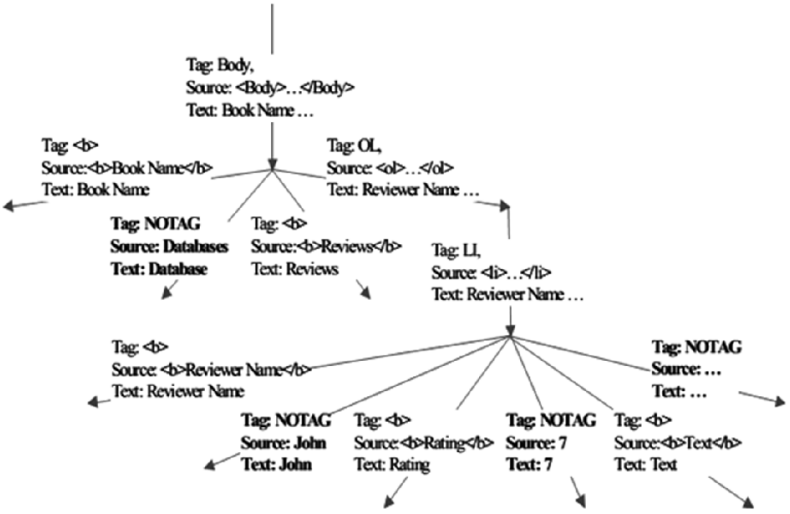


Fig 2.5.b (l'hyper-arbre pour la page dans **2.5.a**)

Comme le montre la figure, la structure de l'arbre est similaire à la structure d'arbre DOM où les arcs sont étiquetés avec trois attributs : Tag, Source, texte, correspondant au nom de balaise, le morceau de code HTML, et le texte hors balisage, respectivement. Le langage fourni par WebOQL est basé sur le paradigme (Select-From-Where). Le principe est donc de simuler toutes les opérations en algèbre relationnelle. A titre d'exemple, la requête suivante extrait les noms des examinateurs "Jeff" et "Jane" de la Page web présentée par (*Fig.2.5.a*), Où la quat (') et le point d'exclamation (!) représentent le premier sous-arbre et l'arbre de queue, respectivement. Les variables, suivant le cas, itèrent sur les arbres simples ou les arbres queues d'hyper-arbre spécifié après l'opérateur "in".

Select [Z! '. Texte]
From *x* in **browse** ("pe2.html") ', *y* in *x',* Z in *y*'
Where *x*.Tag = "ol" and Z.Text = "Reviewer Name"

L'écriture manuelle d'extracteurs nécessite une analyse des documents à traiter et de leur organisation, afin de résoudre le problème d'extraction d'information. C'est une tâche difficile, et présente aussi une source d'erreurs, qui nécessite des connaissances et une expertise qui n'est pas à la portée de n'importe qui. Enfin les changements dans la structure des documents du Web sont fréquents et rendent délicate la maintenance d'extracteurs conçus manuellement. Pour améliorer le développement des extracteurs et assister l'utilisateur dans le processus d'extraction d'information, une nouvelle approche, dite par spécification assistée, est apparue.

Les systèmes basés sur cette approche, comme par exemple : XWrap, W4F guident l'utilisateur au cours du processus de conception de l'extracteur. Par le biais d'une interface graphique, l'outil de spécification dialogue avec l'utilisateur qui lui désigne les éléments à extraire et lui livre également des connaissances sur les documents à traiter et leur organisation.

2.8.1.3. W4F

Dans [A. Sahuguet et F. Azavant, 2001], les auteurs ont développé le système W4F (*développé au sein de l'Ecole Nationale Supérieure des Télécommunications, Paris*). Il s'agit d'un kit à outils java permettant à l'utilisateur de parcourir le contenu html d'un document Web. La construction d'adaptateur consiste en trois couches indépendantes : la récupération, l'extraction, et le mappage, la figure suivante illustre l'architecture du système W4F.

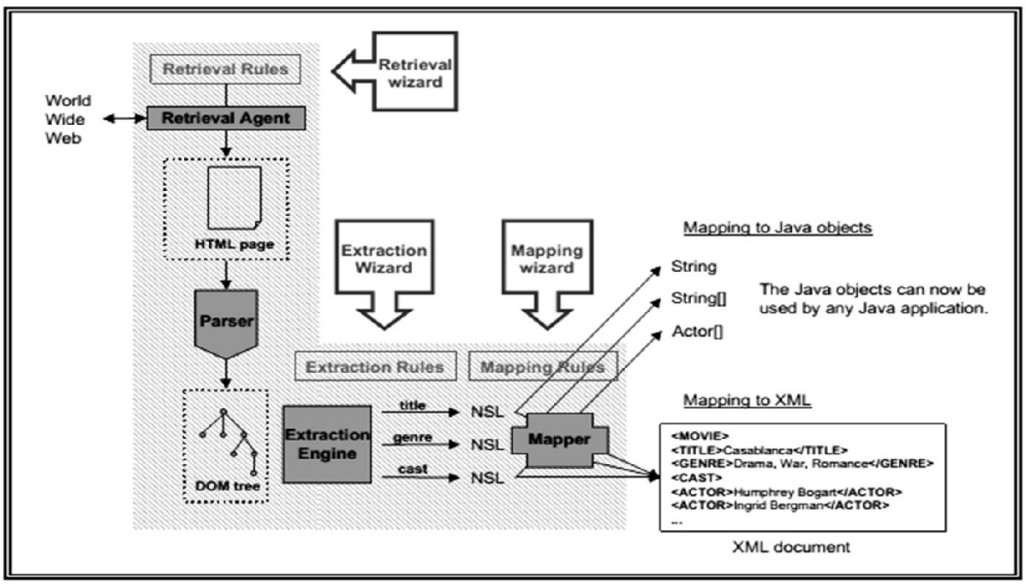


Fig 2.6. Architecture du système W4F [A. Sahuguet et F. Azavant, 2001].

La couche récupération a pour rôle de récupérer les documents à partir du Web via un protocole de communication HTTP. Les documents récupérés sont alors sélectionnés, puis transformés sous forme d'un arbre suivant le modèle DOM via le parseur HTTP.

La deuxième couche est la couche d'extraction qui permet l'extraction des données du document en appliquant les règles d'extraction correspondantes sur l'arbre obtenu durant la couche récupération. Pour spécifier les règles d'extraction utilisées dans cette couche, le langage HEL[1] est utilisé. Il s'agit d'un langage basé sur le modèle DOM, où les documents HTML sont représentés par un graphe étiqueté. Dans le système W4F, ce langage est utilisé pour la navigation dans les documents HTML, et l'extraction de données. L'extraction n'est pas limitée pour les données élémentaires, on peut capturer des structures de données complexes. De cet ongle, le langage HEL est différent du XPath[2] qui peut retourner uniquement un ensemble de nœuds.

Le langage HEL fournit l'opérateur ***Fork*** "#" (comme un constructeur d'enregistrements) pour construire des structures de données complexes basés sur des règles d'extraction. Le rôle de cet opérateur est de suivre un ensemble de sous-chemins simultanément et de concaténer les résultats obtenus. La figure suivante présente une règle d'extraction permettant de spécifier les différents attributs pour un (*Movie*) :

```
movie = html.body(
          ->h1.txt, match/(.*?) [(]/
        # ->h1.txt, match/.*?[(]([0-9]+)[)]/
        # ->td[i:0].a[*].txt
        # ->table[ii:0].tr[jj:*].td[0].txt, match/(\S+)\s(.*)/
                )
where html.body->td[i].b[0].txt = "Genre"
and   html.body->table[ii].tr[0].td[0].txt =~ "Cast"
and   html.body->table[ii].tr[jj].getNumberOf(td) = 3;
```

Fig 2.7. Exemple d'utilisation de l'opérateur ***Fork*** "#"

Ces données seront après enregistrées dans le système W4F sous la forme d'arbres de chaînes de caractères hiérarchiques NSL[3]. En observant la règle d'extraction dans la figure précédente, nous pouvons déduire que pour un film(Movie) le schéma NSL correspondant se compose de 4 éléments.

La troisième couche est la couche de mappage, elle consiste en la construction de règles d'extraction suivant le modèle d'arbre NSL construit précédemment. La construction d'une règle d'extraction est basée sur l'arbre NSL ainsi que le chemin qui mène à l'attribut à extraire.

Après finir les trois couches, nous pouvons maintenant exprimer un adaptateur dans une version complète. La figure suivant présente un adaptateur pour l'extraction des informations concernant les (movies) à partir d'une page web, où :

[1] *HTML Extraction Language*
[2] *XPath est un langage avec une syntaxe non XML, permettant d'adresser les différents nœuds ou groupes de nœuds particuliers d'un document XML.*
[3] *Nested String List*

✓ la section SCHEMA représente le mappage entre le schéma NSL de données et les règles d'extraction.

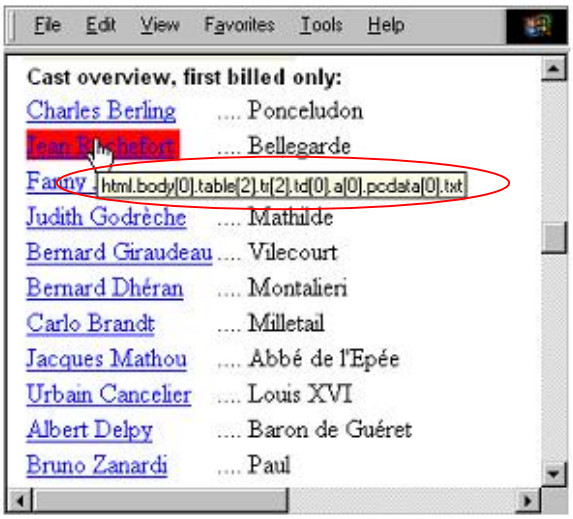
✓ La section EXTRACTION_RULES représente la spécification des règles d'extraction.

✓ La section RETRIEVIAL_RULES définit la méthode d'accès à la source web, où la chaine url doit être fournie.

```
SCHEMA
{
String title;
int year;
String[] genres;
String[][] cast;
}
EXTRACTION_RULES
{
title  = html.body->h1.txt, match/(.*?) [(]/;
year   = html.body->h1.txt, match/.*?[(]([0-9]+)[)]/;
genres = html.body->td[i:0].a[*].txt
  WHERE  html.body->td[i].b[0].txt = "Genre";
cast   = html.body->table[i:0].tr[j:*].td[0].txt, match/(\S+)\s(.*)/
  WHERE  html.body->table[i].tr[0].td[0].txt =~ "Cast"
    AND  html.body->table[i].tr[j].getNumberOf(td) = 3;
}
RETRIEVAL_RULES
{
   get(String url) { GET "$url$"; }
}
```

Fig 2.8. Une version complète d'un adaptateur en W4F.

Une motivation clé pour le système W4F est qu'il assiste l'utilisateur dans l'étape de création des règles d'extraction. Pour ce faire, la boîte à outils est conçue avec la technologie WYSIWYG (ce que vous voyez est ce que vous obtenez ; En anglais : What You See Is What You Get). Le navigateur fourni par W4F intègre cette technologie. La figure suivante présente un exemple :



Maintenant, quand l'utilisateur pointe sur le mot "Tean Rachefort", le chemin qui mène au texte est apparu à coté de la souris. Ceci permet de déterminer la règle d'extraction facilement.

Fig 2.9. Illustration de la spécification assistée du système W4F.

Tout comme l'écriture manuelle d'extracteurs, la spécification assistée impose, dans une moindre mesure, à l'utilisateur de réaliser une grande partie de l'expertise de la tâche d'extraction et de l'analyse des documents à traiter. De plus ce genre d'outils suppose une capacité d'abstraction et une démarche algorithmique qui le réserve encore à des experts. Pour décharger l'utilisateur de cette expertise, une nouvelle approche, intégrant des techniques d'apprentissage, est apparue.

2.8.2. Génération d'adaptateur basée sur l'apprentissage supervisé

Ces méthodes dites supervisées semi-automatiques nécessitent une phase d'étiquetage manuel de quelques pages exemples à utiliser pour la phase d'apprentissage. Celle-ci permet la construction des règles d'extraction généralisées. L'étiquetage des exemples d'apprentissage est fait par un humain via un éditeur de texte ou une interface graphique. Pour induire de nouvelles règles d'extraction, la procédure compare les chaînes de caractères précédant et succédant les attributs de la donnée à extraire. Elle apprend donc les délimiteurs et les motifs d'extraction pour chaque attribut.

Plusieurs méthodes sont apparues visant la construction d'adaptateurs avec apprentissage supervisé (appelée aussi l'induction d'adaptateurs). Chacune de ces méthodes utilise un outil assistant approprié pour l'annotation des exemples d'apprentissage, et un langage différent pour la construction des règles d'extraction. Nous présentons dans cette section des travaux de recherche autour des problèmes d'extraction d'informations (semi-automatique) à partir de documents web.

2.8.2.1. WIEN

Le premier système d'induction d'adaptateurs a été développé par Kushmerick dans l'Université de Washington, il s'agit du système WIEN [N. Kushmerick, 1997]. L'auteur a considéré le problème d'extraction comme une réponse à une requête. À cet effet, et à partir d'un ensemble de pages Web exemples étiquetées représentant des réponses liées à une requête fictive, l'algorithme d'induction permet de trouver un adaptateur candidat permettant d'extraire les données à partir des pages Web similaires de celles utilisées comme exemples.

Kushmerick a défini six classes différentes d'adaptateurs afin d'exprimer les structures des sites Web. Sa première apparition était avec la classe LR (left, right) dont 1'objectif était de déterminer le début et la fin de chaque attribut à extraire du document. Il a représenté un adaptateur sous la forme d'un modèle relationnel à 2*k-uplets (l1, r1, l2, r2, ..., lk, rk) où li, ri représentent respectivement le délimiteur gauche et le délimiteur droit pour l'attribut i. De plus, Kushmerick a introduit des contraintes sur les délimiteurs :

- Un délimiteur de début doit être un suffixe propre c'est-à-dire il ne doit pas se présenter plusieurs fois dans une chaîne de caractère, ainsi que ce délimiteur ne doit pas figurer dans la fin de la page à extraire.
- Un délimiteur de fin ne doit pas être une valeur d'un attribut à extraire, et ne doit pas faire partie de chaque valeur d'un attribut i.

Suivant la classe LR Kushmerick a étendu sa méthode, en introduisant d'autres classes d'adaptateurs, par exemple l'auteur a défini une nouvelle classe appelée OLRC en intégrant 2 nouveaux paramètres à la classe précédente : Supposant un tuple M comprenant K attributs, dans cette approche, un adaptateur est représenté sous la forme d'une relation de (2k+2)-uplets (o, li, ri, c) où 'o' représente le début du fragment de texte du tuple M, 'li', 'ri' représentent le délimiteur gauche, respectivement droit de l'attribut i, et 'c' représente la fin du tuple M .

Une autre classe appelée HLRT a été ajoutée, en intégrant 2 nouveaux paramètres à la classe LR, où 'h' et 't' sont ajoutés. Ces deux paramètres permettent de délimiter une zone à l'intérieur du document dans laquelle l'extraction de tuples aura lieu. Suivant le même principe, plusieurs d'autres classes d'adaptateurs du système WIEN ont été proposées. En combinant ces différentes méthodes, l'approche a pu répondre à 70% des sources utilisées [N. Kushmerick, 1997].

Le système WIEN dans ses différentes variations ne permet pas de traiter les documents ayant une structure hiérarchique. D'autre part les attributs d'un même tuple doivent être ordonnés et toujours présents. Toutefois, les données contenues dans une source Web ne sont pas souvent ordonnées, et généralement se présentent sous un schéma hiérarchique, ce qui a ouvert les portes à d'autres approches plus intéressantes tentant de relever ces défis.

2.8.2.2. STALKER

STALKER [I. Muslea et al, 1999] est un système d'apprentissage supervisé des règles d'extraction. Il s'agit d'un système plus expressif que le système WIEN. Il traite une page Web mise sous forme d'arbre et extrait l'information de manière hiérarchique. L'induction d'adaptateur est fondée sur :

1. Un catalogue (arbres-ECT « Embedded Catalog Tree ») qui décrit la structure hiérarchique et logique des données, construit à la main pour chaque ensemble de pages Web (site Web). Les éléments feuilles de la hiérarchie représentent des valeurs d'attributs. Son avantage est d'optimiser le processus d'extraction dans des documents hiérarchiques non tabulaires. Cet arbre est composé de nœuds, où chacun d'eux pourrait être :

Une feuille: Un élément primitif qui représente une information à extraire dans l'arbre.

Une liste : Un nœud comprenant 0 ou 1 fils.

Un tuple : Un nœud comprenant 1 ou plusieurs fils.

Il est à noter bien que l'ensemble de fils d'une liste ont le même type d'information, par contre ceux d'un nœud tuple n'ont pas forcément le même type d'information. La *figure 2.10.b* illustre l'arbre hiérarchique ECT du document de la *Figure 2.10.a*.

Fig 2.10.a. Capture d'une page web présentant des médicaments.

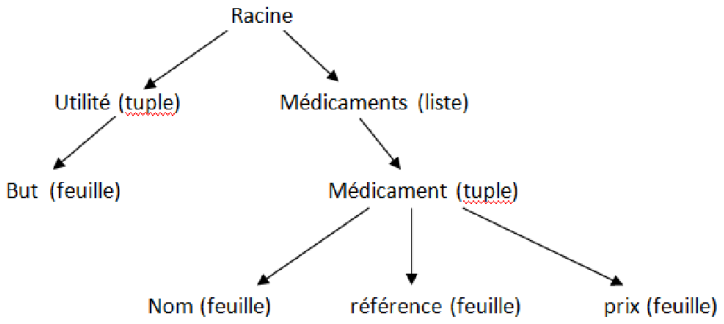


Fig 2.10.b. Arbre ECT représentant le document de la Figure 2.10.a

2. En s'inspirant de l'analyse lexicale du code html de la source, STALKER propose la transformation des documents en séquence de symboles, Cette abstraction introduit des symboles de trois types : les mots (chaîne de caractères alphanumériques), les symboles (caractères non alphanumérique) et les balises HTML.

Noté par *ALPH* l'ensemble de l'alphabet, et supposant *P* un fragment du texte représentant un document à traiter, donc *ALPH* s'écrit sous la formule suivante:

ALPH = words(*P*) U symbols(*P*) U balises(*P*) ; avec words(*P*) est l'ensemble des mots contenus dans *P*, symbols(*P*) est l'ensemble des symboles de ponctuation contenus dans *P*, et balises(*P*) l'ensemble des balises HTML contenues dans *P*.

3. En se basant sur le formalisme ECT, et un ensemble de pages exemples étiquetées, l'algorithme d'apprentissage permet de trouver les règles d'extraction associées à chaque nœud de l'arbre ECT. Le contenu d'une règle est représenté par une disjonction de délimiteurs dont chaque délimiteur est une chaîne de caractère définie dans un ensemble généralisée de l'alphabet *ALPH*. Une règle simple est de la forme **R = SkipTo(w1)** ; avec **w1**est un mot de l'alphabet *ALPH* ; **SkipTo(w1)** indique qu'il faut rechercher la première occurrence de **w1** dans le document à partir de la position courante. La position de fin pour le mot **w1**indique le début de l'attribut. Par exemple : le délimiteur suivant représente une règle simple pour l'extraction des occurrences du nœud Médicament :

SkipTo (**)**.

Une fois que la phase d'apprentissage est terminée, le système STALKER applique successivement les règles générées sur les pages Web pour extraire les informations.

Dans ce système, l'ordre des attributs d'un tuple n'est pas indispensable. De plus, extraire des informations provenant des sources Web ayant un manque d'informations est possible. Les auteurs ont donné une importance à l'expressivité des règles d'extraction en deux points :

- ✓ Le système intègre des règles disjonctives pour pallier au problème de variation dans les pages Web (règles disjonctives c'est-à-dire que plus d'un délimiteur peut être utilisée pour l'extraction des occurrences du même attribut). La forme d'une telle règle est :

  R = SkipTo(w_1) U **SkipTo(w_2)** U….U **SkipTo(w_n)**. C'est-à-dire qu'un des délimiteurs : **SkipTo(w_1)**,….. , **SkipTo(w_n)** est utilisé à chaque fois. Ce qui permet l'extraction d'un attribut avec des formats différents.

- ✓ Le système intègre aussi la notion du raffinement des délimiteurs (c'est-à-dire qu'un délimiteur peut être constitue de plusieurs sauts). La forme d'une telle règle est :

  R = SkipTo(w_1) SkipTo(w_2) …. **SkipTo(w_n)**. C'est-à-dire qu'il faut effectuer plusieurs sauts avant de procéder à l'extraction.

2.8.2.3. WHISK

WHISK [S. Soderland, 1999] est un système d'apprentissage supervisé qui nécessite un ensemble d'exemples d'entrainement étiquetés à la main. Le processus d'étiquetage est entrelacé avec l'apprentissage. A chaque itération, le système présente à utilisateur un ensemble d'instances à étiqueter, puis le système induit un ensemble de règles à partir de l'ensemble élargi d'exemples d'entrainement.

Ce système produit des règles d'extraction pour une grande variété de documents structurés ou non structurés. Les motifs d'extraction sont des expressions (spéciales) régulières ayant deux composantes : l'une décrit le contexte du motif, et l'autre indique les délimiteurs du motif à extraire. Selon la structure d'un document, WHISK génère des règles d'extraction qui reposent exactement sur l'un des deux composants cités ci-dessus. En particulier, dans le cas de texte libre, il utilise des règles basées sur le contexte, en cas de texte structuré il utilise des délimiteurs.

Une instance à annoter est la plus petite unité d'un document, comme une phrase pour les documents non structurés, ou une portion du code HTML pour les documents semi-structurés. Ces instances sont utilisées pour guider la création de règles et de tester la performance des règles proposées. Si une règle est appliquée avec succès à une instance, l'instance est considérée comme couverte par la règle. Tout d'abord, la règle la plus générale qui couvre toutes les instances est générée, en suite, à chaque fois cette règle est pondérée (des termes sont ajoutés à la règle) jusqu'à ce que les erreurs sont réduits à zéro (c'est-à-dire la règle couvre uniquement les exemples positifs).

Exemple d'application pour un document semi-structuré : la *figure 2.11.a* présente une partie du code source pour une page qui donne une annonce en ligne proposant des chambres en location. Le but est d'extraire le nombre de chambre et le prix à partir de ce code HTML:

```
Capitol Hill - 1 br twnhme. fplc D/W W/D.
Undrgrnd pkg incl $675. 3 BR, upper flr
of turn of ctry HOME. incl gar, grt N. Hill
loc $995 (206) 999-999 <br>
<i><font size=-2> (This ad last ran on 09/03/97.)
</font></i><hr>
```

Fig 2.11.a. Code HTML.

La règle la plus générale a la forme : *(*) * (*) *

```
Pattern : : *(Digit ) ' BR' * '$' (Number )
Output : : Rental {Bedrooms $1} {Price $2}
```

Fig 2.11.b. La règle d'extraction WHISK.

```
Rental:
        Bedrooms: 1          Price: 675
Rental:
        Bedrooms: 3          Price: 995
```

Fig 2.11.c. Résultat obtenu.

Exemple d'application pour un document non-structuré : les figures suivantes présentent un exemple de formation de règle d'extraction à partir d'un exemple en texte libre.

Énoncé :

C. Vincent Protho, chairman and chief executive officer of this maker of semiconductors, was named to the additional post of president, succeeding John W. Smith, who resigned to pursue other interests.

Fig 2.12.a. Texte libre.

Analyse syntaxique :

```
@S[
    {SUBJ @PN[C. Vincent Protho]PN , @PS[chairman
          and chief executive officer]PS of this
          maker of semiconductors,}
    {VB   @Passive was named @nam}
    {PP   to the additional post of @PS[president]PS , }
    {REL_V succeeding @succeed @PN[John W. Smith]PN
          , who resigned @resign to pursue @pursu
          other interests.}
    ]@S 8910130051-1
```

Fig 2.12.b. Analyse syntaxique.

Règle correspondante :

Pattern : : * (*Person*) * '@Passive' *F 'named' * {PP *F (*Position*) * '@succeed ' (*Person*)

Output : : Succession {PersonIn $1}{Post $2}{PersonOut $3}

Fig 2.12.c. Règle d'extraction WHISK.

Résultat (Succession) :

PersonIn: C. Vincent Protho

PersonOut: John W. Smith

Post: president

Fig 2.12.d. Résultat.

Pour le traitement des documents non-structurés, WHISK intègre une analyse syntaxique des énoncés qui permet d'identifier les rapports syntaxique de l'information avec son contexte et d'ajouter ce critère aux contraintes de la règle d'extraction. L'analyse syntaxique de cet exemple permet de distinguer les champs syntaxiques :

(SUBJ pour sujet, VB pour verbe principal, PP pour groupe prépositionnel et REL_V pour proposition relative rattachée à un verbe). Les indications précédées par @ sont d'ordre sémantique (PN pour nom de personne, PS pour poste, CN pour nom de société), syntaxiques (Passive pour passif) ou morphologiques (la racine d'un verbe : nam, succeed, resign, pursu). La règle qui en découle est une expression régulière. Les mots en italiques indiquent des classes sémantiques. Les @ permettent de requérir une indication morphologique (@succeed demande un verbe de radical succeed) ou syntaxique (@Passive exige un verbe à la voix passive). Le quantifieur * permet de ne pas prendre en considération un nombre illimité de caractères. S'il est suivi de F (*F).

2.8.2.4. SoftMealy

Dans [C.N. Hsu et M.T. DUNG, 1998] les auteurs ont proposé le système SoftMealy qui a pour but résoudre le problème des attributs multi-ordre et celui des attributs manquants. Dans ce système, le formalisme pour décrire un extracteur est celui des transducteurs à états finis. Ce dernier est un automate de mots qui produit un mot de l'alphabet de sortie, en fonction de l'état dans lequel il se trouve et de la lecture d'un mot d'entrée. Les règles d'extraction sont décrites par les transitions entre les états.

Ce système apporte une approche qui n'est plus basée sur des délimiteurs mais sur des séparateurs. En plus qu'il détermine les positions de début et de fin pour un attribut, le séparateur prend en compte aussi l'ensemble des symboles composant l'attribut (c'est-à-dire qu'il prend en compte le type de l'attribut à extraire « le contexte »). Cet aspect sémantique est vu comme un avantage capital par rapport aux systèmes WIEN et STALKER.

L'algorithme d'extraction est conçu dans un esprit similaire à celui des extracteurs *HLRT* du système WIEN. Dans ce cas, un adaptateur se compose de deux transducteurs :

- Un pour déterminer la zone concernée par l'extraction dans le document;
- Un autre pour extraire les attributs à partir de la zone déterminée par le premier transducteur.

Le premier transducteur a le même rôle que les délimiteurs *h* et *t* d'un extracteur *HLRT*. La sortie de ce transducteur est l'entrée du second. Ce premier transducteur est simple. Il est constitué de 3 états :

- un état initial qui est quitté lorsque la règle d'extraction gauche est détectée. Cette règle indique le début de la zone concernée par l'extraction
- un état intermédiaire dans lequel le transducteur reste tant que la règle d'extraction droite non encore détectée. Cette règle indique la fin de la zone concernée par l'extraction
- et un état final dans lequel on arrive quand le transducteur quitte l'état précédant.

Le deuxième transducteur (permettant l'extraction des attributs) est le corps d'adaptateur SoftMealy. Il est plus complexe que le précédent. Un transducteur permettant l'extraction d'une relation à $\boldsymbol{n}$ attributs aura $\boldsymbol{2n + 1}$ états : un état par attribut, un état entre chaque paire d'attributs successifs, un état de début et un état de fin. Formellement, ce transducteur est composé de 5-uplets ($\sum_1$, $\sum_2$, **Q**, **R**, **E**) où $\sum_1$ est l'ensemble de séparateurs d'entrée, $\sum_2$ est l'ensemble des caractères de sortie, **Q** est un ensemble fini d'états contenant:

- Un état initial ***b***, et un état final ***e***.
- ***k*** est un état s'il existe un attribut ***k*** à extraire.
- Pour chaque attribut ***k***, il existe un état ***-k*** correspond à un attribut factice représentant l'ensemble des symboles à consommer entre l'attribut ***k*** et l'attribut suivant.

R est l'ensemble des règles contextuelles (règles d'extraction) dont chaque règle est dénotée par ***s (i, j)*** représentant la classe de séparateurs figurant entre l'attribut ***i*** et l'attribut ***j***.

E = Q x **R** x $\sum_1$* x **Q** : est un ensemble de transitions, une transition de ***i*** vers ***j*** est possible s'il existe une transition **(*i*, r, o, *j*)** Є **E** tel que le prochain séparateur d'entrée satisfait la règle **r**. Une transition de ***i*** vers ***j*** montre que les attributs ***i*** et ***j*** sont adjacents dans les pages Web.

L'induction d'un extracteur se fait à partir d'un ensemble de documents complètement annotés. Elle consiste à construire les deux transducteurs la construction d'un transducteur revient à apprendre les règles de transition permettant de transiter d'un état à l'état suivant. Pour une relation à ***n*** attributs le transducteur d'attributs est initialisé à ***2n + 1*** états, en suite les symboles permettant les transitions sont appris via les exemples annotés.

Dans ce qui suit, nous allons illustrer le transducteur d'attributs par un schéma. Soit à extraire à partir des pages web les attributs de la relation suivante :**R (At_1, At_2,… At_k,… At_n);** où les At_i sont les attributs à extraire. Alors le transducteur aura la forme suivante :

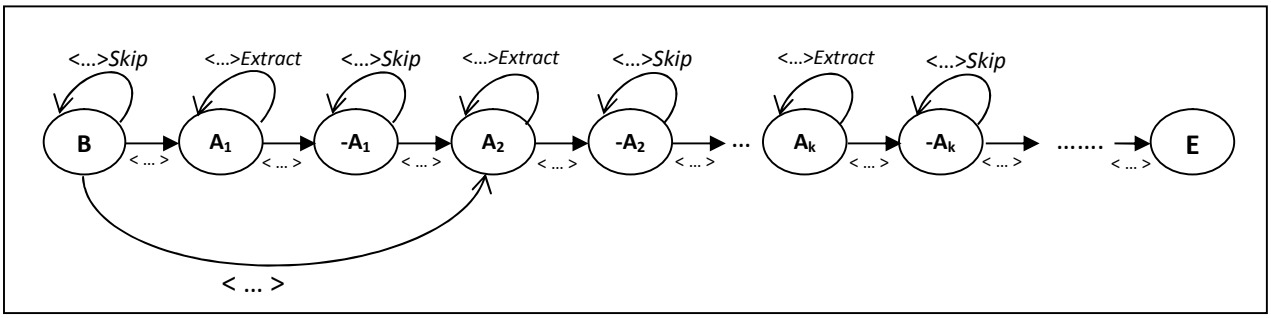


Fig 2.13. Transducteur SoftMealy.

Comme le montre la figure en dessus, pour chaque transition est associe une séquence de symboles <….> permettant de la déclencher. A chaque attribut At_i à extraire correspond deux états : l'état A_i dans lequel le transducteur restera, grâce à une boucle, pour extraire les occurrences de l'attribut et l'état -A_i, qui correspond à la séquence de symboles à ignorer avant de rencontrer une autre valeur à extraire. Les transitions qui arrivent dans un état A_i, sauf les boucles, correspondent à la lecture du séparateur de début pour l'attribut At_i. Les transitions qui sortent d'un état A_i, sauf les boucles, correspondent à la lecture du séparateur de fin pour l'attribut At_i. Les règles des boucles sur les états A_i assurent l'extraction de la valeur pour l'attribut At_i (ce qui permet d'extraire les attributs multi-valeurs). La transition directe entre l'état **B** et l'état A_2 signifie que l'attribut At_1 peut avoir des valeurs absentes. Le transducteur des attributs est appliqué tant que la zone d'extraction sélectionnée par le premier transducteur n'est pas intégralement parcourue. Chacune de ses applications assure l'extraction d'un enregistrement.

La critique fréquemment adressée aux systèmes d'induction supervisée d'extracteurs est l'importance de l'effort d'annotation de la part de l'utilisateur. Ces systèmes nécessitent des documents complètement annotés, c'est à dire toutes les données à extraire sont marquées. Dans le but de minimiser les efforts d'utilisateur, une nouvelle approche dite semi-supervisée est apparue. Dans la section suivante, nous allons présenter cette approche.

2.8.3. Méthodes semi-supervisées pour la construction d'adaptateur

Le but de cette approche est de réduire l'effort d'utilisateur dans la tâche d'extraction d'information. Plusieurs méthodes ont étés proposées dans ce contexte, chacune d'elles est basée sur l'un des deux principes :

- Soit assister l'utilisateur dans l'étape d'étiquetage des exemples d'entrainement. Dans ce cas l'étiquetage est entrelacé avec l'apprentissage. A chaque itération un ensemble minimal d'instances est étiqueté, en suit le système, à partir de ces instances, génère un ensemble de règles d'extraction. Ces dernières sont immédiatement appliquées sur l'ensemble des exemples d'entrainement, l'étiquetage dans l'itération suivante concerne uniquement les données non extraites en appliquant les règles générées dans les itérations précédentes. L'avantage est que l'étiquetage d'un seul exemple permet à chaque fois de générer un délimiteur.
- Soit le système procède à une détection automatique des motifs à partir d'un (ou plusieurs) page(s) web. Dans ce cas l'utilisateur est déchargé complètement de l'étiquetage des exemples d'entrainement, uniquement un post-traitement est nécessaire pour valider les motifs générés ou sélectionner quelques uns.

2.8.3.1. IEPAD

Le système IEPAD [C.H. Chang et S.C. Lui, 2001] ne nécessite pas un étiquetage des pages exemples. En effet, l'intervention de l'utilisateur permet uniquement de préciser les motifs d'extraction des enregistrements. Ce système consiste à découvrir automatiquement des règles d'extraction dans les pages Web. Le fait que la majorité des pages provenant du Web suivent une certaine régularité syntaxique, les informations à extraire issues de ces pages peuvent avoir des motifs communs qui séparent les informations situant dans les pages d'une même source web. Sur la base de cette idée, les auteurs se sont basés sur les motifs communs pour la génération des règles d'extraction. Le système peut automatiquement identifier la frontière entre les champs en se basant sur les motifs fréquents et sur l'alignement de séquence multiple.

IEPAD cherche à extraire des blocs d'informations intéressantes présentées de manière contigüe dans les pages. La découverte des motifs s'effectue en plusieurs étapes :

- La première étape consiste à encoder le document analysé en une séquence de symboles en faisant abstraction des parties textuelles. Il s'agit de recenser les balaises HTML contenant dans un document, chaque symbole est encodé avec une suite de chiffres binaires unique. Un seul code binaire est utilisé pour représenter les segments de texte.
- A partir de l'encodage obtenu dans l'étape précédente, cette étape consiste à construire un arbre PAT[1] afin de procéder à une recherche des répétitions maximales dans le document codé en binaire. Ces répétitions forment un ensemble de motifs candidats.
- Parmi les motifs candidats générés dans l'étape précédente, cette étape consiste à effectuer une sélection des motifs. La sélection se fait selon trois critères :
  - ✓ La régularité : ce paramètre vérifie si la dispersion du motif est organisée ou non.
  - ✓ La compacité : d'un motif est une mesure de densité du motif dans une zone délimitée par la position du premier motif et celle dernier motif.
  - ✓ La couverture : évalue le pourcentage de zone couvre par le motif par rapport à la longueur de la zone contenant les données.

  Un seuil est défini pour chaque paramètre. Les motifs qui présentent des paramètres dépassants le sont retenus.
- Comme étape Finale, les motifs ayant une faible densité sont alors étendus via l'utilisation de l'alignement multiple de séquences.

Après ces étapes un ensemble de motifs sont retenus. Il est nécessaire, en suite, de faire appel à l'utilisateur pour choisir parmi les motifs générés et corriger les motifs pour qu'ils puissent extraire entièrement toutes les valeurs sans toutefois en extraire de trop.

Ce système suppose que l'information traitée est décrite d'une manière rassemblée et régulière, ce qui n'est pas souvent le cas des informations contenues dans les pages Web. En outre, l'extraction d'information se limite uniquement sur des informations d'un même enregistrement ce qui est inapplicable dans le cas des données hiérarchiques. Ce système est utilisé seulement sur des pages Web contenant une liste d'enregistrements (au moins 2 enregistrements par page). Les résultats d'extraction de ce système ont montré que sur quatorze moteurs de recherche les plus populaires, les précisions d'extraction ont atteint 97% [C.H. Chang et S. Lui, 2001].

[1] *Un arbre PAT d'une chaînes est un arbre Patricia (Practical Algorithm to Retrieve Information Coded in Alphanumeric) Construit sur l'ensemble suffixes de s.*

2.8.3.2. OLERA

Ce système [C.H. Chang et S.C. Kuo, 2004] est considéré comme amélioration du système IEPAD, en intégrant une interface graphique pour choisir les motifs d'extraction. Il s'agit donc d'un système semi-supervisé qui ressemble au système IEPAD mais qui permet de lui ajouter une option permettant l'extraction des données provenant de pages Web ayant un seul enregistrement. Ce système permet la génération des règles d'extraction en commençant par un seul exemple qui représente un bloc des informations à extraire. Le principe est de trouver l'ensemble des blocs similaires à un exemple d'entrainement (où l'utilisateur marque uniquement un bloc d'information contenant un enregistrement à extraire). Les figures suivantes présentent ce principe :

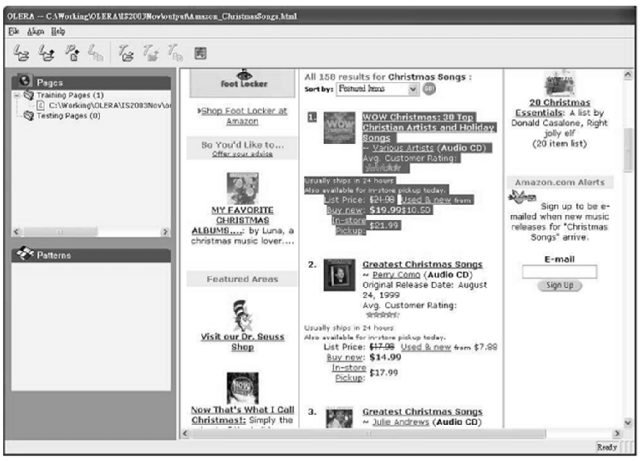


Fig 2.14.a étiquetage d'un enregistrement

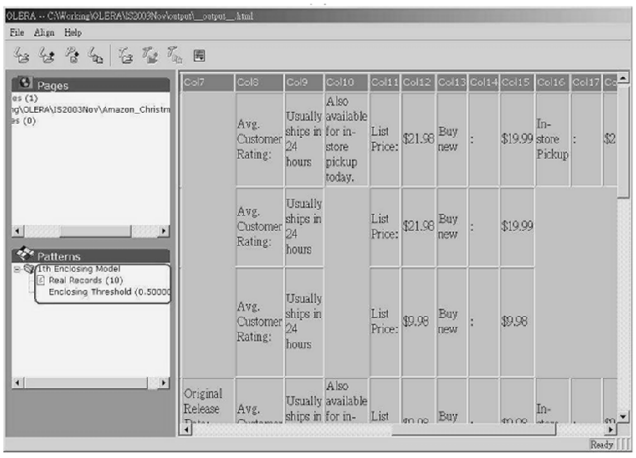


Fig 2.14.b Résultat obtenu.

Dans la figure à gauche, l'utilisateur a sélectionné une zone sur la page comme un enregistrement brute (c'est-à-dire l'utilisateur n'a pas déterminé les attributs de l'enregistrement). A partir de l'analysé de la zone sélectionnée, dix enregistrements (Seuls les quatre tops sont présentés ici) se trouvent et sont alignés dans un tableur (la figure à droite).

Le processus de construction des règles d'extraction est composé de trois phases :

1) Codage de la page d'apprentissage : En se basant sur la classification proposée dans la table ci-dessous, le contenu de la page est divisé sur plusieurs niveaux présentés en trois catégories. Pour chaque niveau est affecté un code significatif et un ensemble de délimiteurs.

Category	Encoding scheme	Delimiters
Markup-level	Block-level tag	block-level tags
	text-level tag	text-level tags
Text-level	Paragraph	NewLine, CarriageReturn, Tab
	Sentence	Period, Question Mark, Exclamation Mark
Word-level	Phrase	Colon, Comma, Semicolon, Bracket, Quotation Mark
	Others	(,), $, -, /, @, Blank, etc.

Table 2.2. Niveaux d'encodage des pages web [C.H. Chang et S.C. Kuo, 2004].

2) Détection de motif : Dans cette phase, le système permet de trouver le motif du bloc utilisé comme exemple. En comparant ce motif avec d'autres pages d'entraînement, le système peut détecter les enregistrements similaires à ce bloc.

3) Alignement des enregistrements : Il consiste à aligner l'ensemble des enregistrements découverts dans l'étape précédente en utilisant des techniques d'alignement de chaîne de caractères (Multiple String Alignment). Le résultat est présenté sous la forme d'un tableau (**Fig 2.14.b**) de rn enregistrements et n slots. Le slot représente un délimiteur qui peut être rectifié par l'utilisateur.

2.8.4. Méthodes non-supervisées pour la construction d'adaptateur

L'induction d'extracteur peut être complètement automatisée. Les méthodes de cette approche ont une caractéristique commune : l'absence d'intervention humaine dans le processus d'extraction d'information. Deux catégories d'outils semi-supervisés existent :

A. *Outils fondés sur l'extraction de régularités dans des pages* :

Ces outils d'induction déterminent quelles sont les données à extraire en analysant la structure et les régularités des documents. Dans le code HTML d'une page web, deux parties sont distinguées : une partie statique (les symboles responsable de la mise en forme des pages web) et une partie variable (les données visibles sur navigateur). Le schéma régulier *Template* d'une page correspond à la partie statique. Pour les pages, qui présentent des données issues d'une base de données web, la partie statique est commune. En analysant les régularités présentées par le Template des pages web, on peut déduire les motifs permettant l'extraction d'information. Comme exemple, nous citons les systèmes RoadRunner et DeLa.

B. *Outils d'extraction exploitant un modèle conceptuel (ontologie)* :

Ces outils consistent à exploiter les connaissances décrites dans une ontologie pour extraire des informations à partir de documents. Les connaissances stockées dans l'ontologie se réfèrent aux caractéristiques des données et à leur présentation dans les documents. Un des premiers outils d'extraction d'informations adoptant cette approche est BYU.

Dans cette section nous présentons les trois systèmes : RoadRunner, DeLa, et BYU.

2.8.4.1. RoadRunner

Ce système [V. Crescenzi et al, 2001] est basé sur l'exploitation les régularités structurelles des pages web. Il compare la structure du code HTML d'un certain échantillon de pages Web afin de chercher les correspondances entre les documents d'une même source pour construire les motifs. Ce qui permet de conclure le schéma des emplacements des données à extraire dans une page. La

construction d'un adaptateur dans ce système se réduit à un problème d'inférence d'une grammaire régulière sans union[1]. L'algorithme d'extraction produit une expression régulière qui accepte tous les attributs des pages de l'échantillon. La construction des règles est un processus complètement automatique.

L'expression régulière permettant de repérer les données est construite via la comparaison des pages web. Deux types de disparités peuvent se présenter dans cette comparaison :

1) Disparité de texte (*String mismatch*): Il y a une disparité de texte lorsque, dans les deux pages, un texte différent apparaît à la position courante. Ceci indique qu'une valeur d'un champ a été détectée. Dans la figure ci-dessous, la ligne 4 dans les deux pages présente une disparité de texte.

2) Disparité de balise (*Tag mismatch*): Dans ce cas, une option ou une répétition (itération) a été détectée. Comme par exemple, dans le cas de la ligne 6 sur les deux cotés : cette disparité indique qu'il s'agit d'une option (l'image présentée par la balaise <IMG src=…/> est optionnelle). Un cas deuxième : la ligne 19 sur le coté gauche et la ligne 20 sur le coté droite présentent une disparité de balaise. Cette fois il s'agit d'une itération (répétition) sur le coté droite.

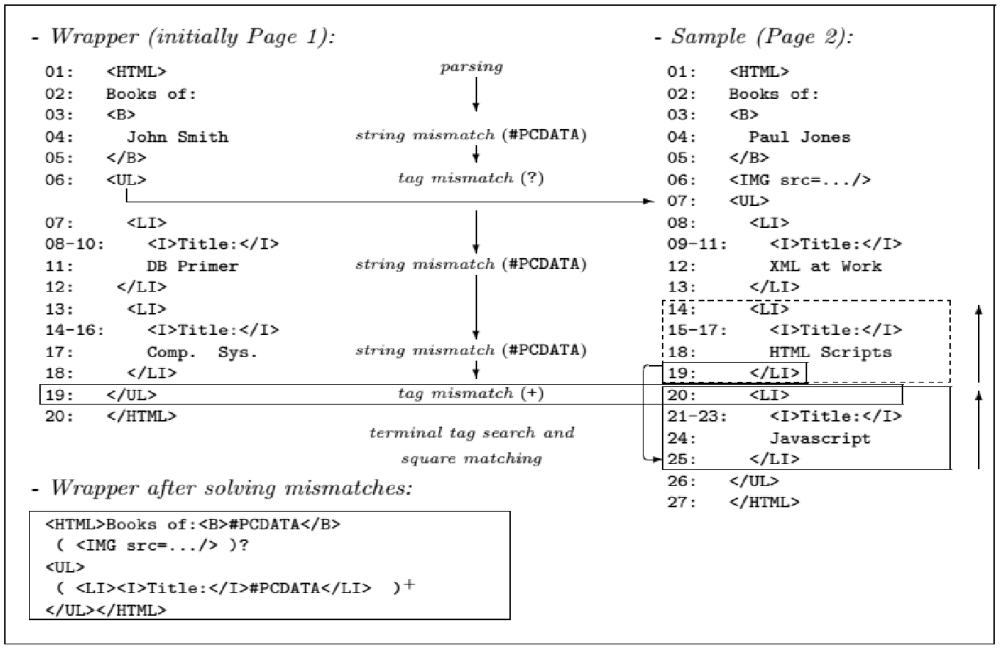


Fig 2.15 Exemple d'application du système RoadRunner.

L'absence d'intervention de l'utilisateur dans le processus d'extraction d'information est l'avantage principal de ce système. Cependant un nombre d'attributs d'options et d'itérations autant,

[1] *Une grammaire régulière sans union est une grammaire qui peut être décrite à l'aide d'une expression régulière avec les opérateurs habituels (. +, ?, (,), 1) sauf I . Une telle expression s'appelle une expression régulière sans union ou ERSU.*

exactement, le nombre de disparités détectées au niveau syntaxique des pages web. Ce qui met, encore, l'utilisateur en face d'un choix des motifs parmi plusieurs. D'autre part, bien que l'algorithme de correspondance découvre la structure, il n'a aucun moyen d'associer une étiquette sémantique aux différentes parties de la structure découverte. On peut confirmer que la structure des documents est de la forme A, (B, C, D?)+, mais on ne peut pas savoir que A est un nom d'auteur, que B est le titre d'un livre… etc.

2.8.4.2. DeLa

[J. Wang et F.H. Lochovsky, 2003] décrivent DELA (*Data Extraction and Label Assignment*), un système d'extraction automatique d'information. Pour la construction de leur système, les auteurs ont basé sur les deux remarques suivantes :

A. les pages web générées dynamiquement suite à une requête suivent généralement le même Template (c'est-à-dire ces pages partagent une structure commune des balaises HTML). Basé sur cette observation, il y a une possibilité pour générer automatiquement une expression régulière permettant d'extraire les données à partir de ces pages, en suite stocker ces données dans une table.

B. les formulaires, à partir desquelles on interroge les bases de données web, fournissent une information cruciale permettant de savoir une partie de la structure de la base de données web, ainsi que la sémantique des informations présentées sur les pages web résultats. Basé sur cette remarque, DeLa exploite les informations présentées par ces formulaires pour assigner automatiquement des étiquettes aux attributs de la table contenant les informations extraites.

Le processus d'extraction d'information est complètement automatique, ainsi l'intervention de l'utilisateur humain n'est pas nécessaire. Ce système permet, aussi, d'extraire des données provenant d'objets imbriqués. Les auteurs ont introduit deux techniques, une qui permet d'identifier les régions contenant les données à extraire, et l'autre pour identifier la structure des données dans ces régions. Le processus de la construction des règles est basé sur deux étapes importantes :

1- Algorithme d'extraction des régions : Cette étape consiste à comparer deux arbres DOM Représentant deux pages Web différentes d'un même site Web afin d'identifier les régions de textes contenant des informations à extraire. L'algorithme utilisé est l'algorithme de détection des régions DES (*Data-rich Section Extraction*).

2- Extracteur des motifs : C'est le composant principal de ce système, il permet de construire la représentation symbolique des arbres suffixes. Ensuite, il s'agit d'appliquer l'algorithme de la découve1ie de motif pour détecter les motifs des objets dans un même arbre suffixe.

D'après les auteurs, le système DELA est considéré à 100% automatique. La précision d'extraction atteint 90 %, et la précision d'assignement des étiquettes aux attributs atteint 80%.

2.8.4.3. BYU

A partir d'une ontologie [D.W. Embley et al, 1999], ce système peut concevoir et alimenter automatiquement une base de données relationnelle par des informations reconnues et extraites à partir des documents données en entrée. Ces documents sont supposés traiter du même domaine et contenir de multiples enregistrements (« Multiple-Record»).

L'ontologie est construite manuellement par un expert du domaine visé. Elle comporte les termes du domaine, leurs caractéristiques lexicales, leur contexte d'apparition à l'aide de mots clés et de relations sémantiques entre les termes. La figure suivante présente l'architecture de ce système.

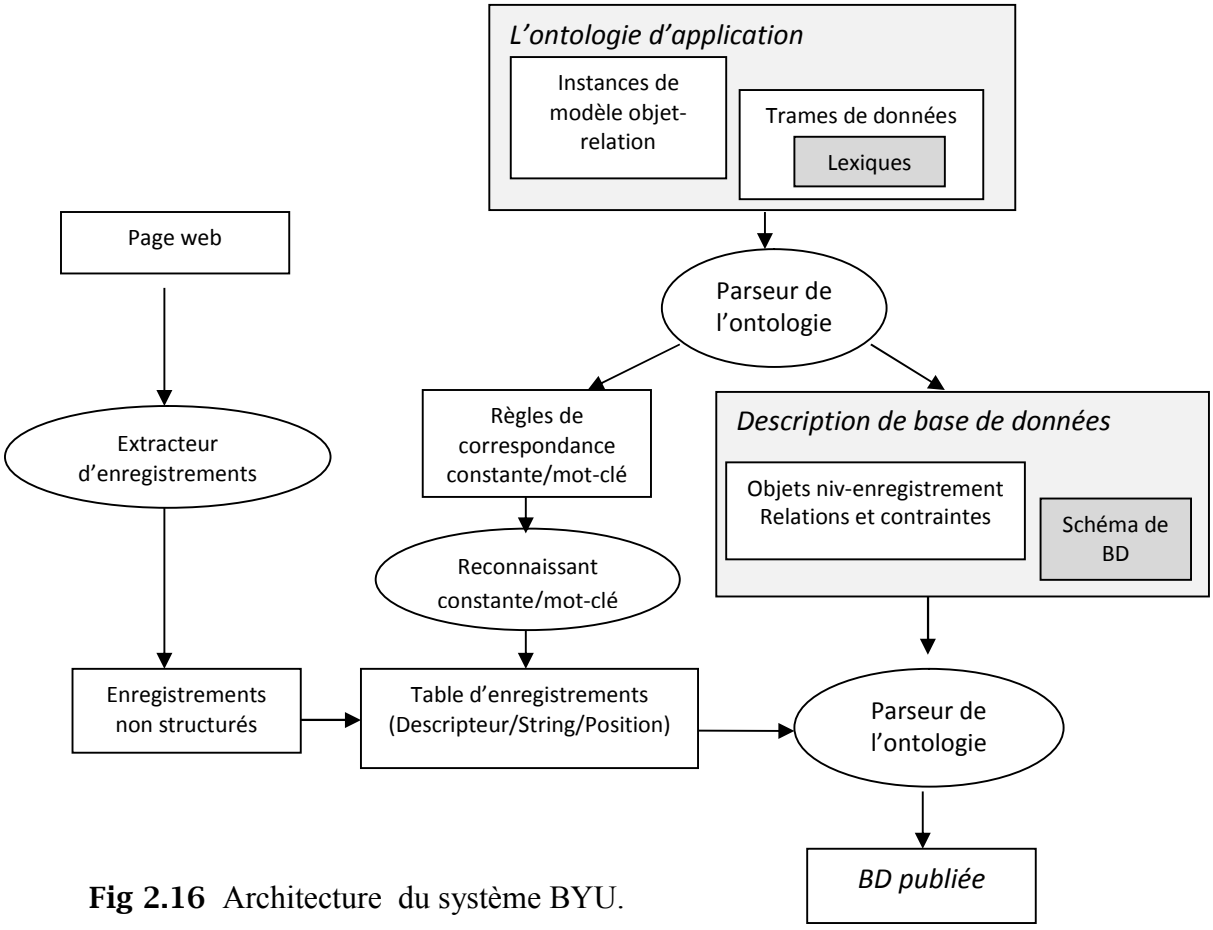


Fig 2.16 Architecture du système BYU.

Dans ce qui suit, on va présenter les modules du système BYU :

- *Spécification ontologique* : Comme indique la figure en dessus, l'ontologie de l'application se compose d'une instance de modèle objet-relation, des trames de données, et des lexiques.

Un parseur d'ontologie prend ces informations en entrée et produit les règles de correspondance constants / mots –clés, et une description de base de données en sortie. Les composants de l'ontologie sont :

A. Le modèle objet-relation : c'est un modèle conceptuel de données. Le but est de décrire les concepts du domaine traité et les relations entre eux. Pour le présenter le modèle des systèmes orienté-objet (OSM) est utilisé. Il peut être aussi décrit sous forme de descriptions textuelles.

B. Les trames de données et lexiques : En générale, une *trame de données* décrit les connaissances pertinentes sur un ensemble d'objets. Il s'agit ici de décrire : 1) un ensemble de termes indiquant la présence d'une valeur. 2) un ensemble d'expressions décrivant chacune un format (contexte) dans lequel peuvent apparaître les concepts (termes).

➢ *Extraction d'enregistrements non structurés* : Pour cette tâche, l'obtention des pages intérêt à partir du Web se fait en deux étapes : (1) Identifier les pages contenant plusieurs enregistrements d'intérêt par rapport à l'application donnée par l'ontologie. (2) Partitionner les pages obtenues en (1) en morceaux (enregistrements) de données. Un enregistrement est un bloc de données qui représente une instance de l'élément principal spécifié dans l'ontologie. Dans cette section on va concentrer uniquement sur l'étape (2).

Pour partitionner les pages web en enregistrements en procède comme suite :

A. construire l'arbre de la page en question,

B. appliquer des heuristiques pour chercher le nœud le plus susceptibles de contenir les Enregistrements.

C. appliquer des heuristiques pour trouver le séparateur d'enregistrements le plus probable.

Pour détecter la région dans la page contenant les enregistrements d'intérêt, nous cherchons la balises a pour le sous-arbre avec le plus grand *Fan-out*. Il s'agit du sous arbre ayant le max de nœuds au niveau 2 (à partir de la racine de ce sous arbre). Appelons ce sous-arbre *S*.

Pour extraire les enregistrements l'attention est limitée uniquement sur *S,* et nous concentrons sur balises HTML qui séparent les morceaux dans *S.* Les nœuds enfants directs de la racine de *S* sont appelés *balises candidats*. Nous rejetons toutes les balises candidats qui se produisent relativement peu de fois. Pour découvrir le séparateur d'enregistrements, des poids sont associés aux balises candidats en utilisant les Heuristiques :

❖ *HC* est l'heuristique de *compte plus haut.* : Il classe les candidats suivant le nombre d'occurrences. *HC* suppose que la balise de séparation se produit fréquemment quand il ya de nombreux enregistrements.

- *IS* est l'heuristique de *Séparateur Identifiable* : Il utilise une liste prédéterminée de balises HTML.
- *SD* est l'heuristique de *l'écart standard* : Il suppose que les enregistrements sont de taille similaire. Les Candidats sont classés par ordre de l'écart-type de texte brut compris entre les balises identiques.
- *RP* est l'heuristique de *schéma répété* : Il suppose que les divisions entre les enregistrements sont formées par plusieurs balises qui se produisent régulièrement dans le même ordre.
- *OM* est l'heuristique de *Correspondance Ontologique* : Il est basé sur le contenu ontologique d'un enregistrement. L'ontologie indique quels sont les champs d'un enregistrement qui sont attendus de se produire une seule fois. Ces champs sont appelé « *records identifier ». OM* compte le nombre de « *records identifier »* sur une page et calcule ensuite *X* : la moyenne du nombre d'occurrences de chaque « *records identifier »*. *OM* classe les candidats par la façon dont ils prennent leur nombre d'occurrences correspond à *X*.

➢ *Génération d'enregistrements de BD* : Avec la sortie de parseur d'ontologie et la sortie de l'extracteur d'enregistrements, le système procède à remplir la base de données en appliquant deux grandes étapes pour chaque enregistrement non structuré :

Etape 1 : produire une table d'enregistrements contenant un ensemble de tuples (descripteur / string / position) comme illustré dans la *figure 2.15* pour les constantes et les mots-clés reconnus dans l'enregistrement non structuré. Il s'agit d'une étape de découpage de l'enregistrement non structuré en petites chaines de caractères, pour chaque chaine on donne un descripteur, string et une position.

Etape 2 : Appliquer des heuristiques à cette table pour construire les tuples de la BD : Il s'agit ici de faire correspondre chaque chaine obtenue dans l'étape précédente à un champ de l'enregistrement.

Le résultat de l'application des deux étapes à un enregistrement non structuré est une liste d'instructions SQL (insertions).

Le système BYU est qualifie par la résilience et l'adaptabilité. On entend par la résilience et l'adaptabilité la capacité de traitement de nouvelles occurrences de documents ayant des changements dans la structure et dans les caractéristiques de mise en forme. Ceci constitue une difficulté intrinsèque pour les autres approches.

Cependant, Cette méthode présente deux points faibles. D'une part, le Fan-out des nœuds dans la structure arbre de page web est utilisé pour identifier la région cible (contenant les enregistrements). Cet heuristique ne suffit pas car, parfois, il existe des régions non cibles pourraient

avoir un fan-out supérieur à celui du nœud correspond à la région cible. D'autre part, l'étape essentielle est la construction d'avance d'une ontologie, cette tâche est couteuse.

La construction d'ontologie nécessite un effort signifiant par quelqu'un qui est habile avec les expressions régulières et la modélisation conceptuelle. Ce processus est un peu comme le développement de logiciels: il ya l'analyse, la conception, la mise en œuvre, et les étapes de test.

Les auteurs de la méthode BYU ont construit une ontologie d'extraction pour les décès [D.W. Embley et al, 1999], ils ont crée 19 objets, 17 relations, 21 trames de données, et 219 expressions régulières: écrite pour les mots clés, le contexte, substitution, et le filtrage.

Travaux proposés dans le contexte de la méthode BYU:

- Un environnement pour le développement d'ontologie intégrée pour l'extraction d'information [S.W. Liddle et al, 2003] : l'objectif derrière ce travail est la proposition d'un outil basé sur Java pour la création graphiquement et le teste d'ontologies d'IE. Cet outil est basé sur les normes telles que Java et XML pour fournir un environnement portable, extensible, et maintenable. Cet outil assiste les experts et simplifie la tâche de création d'ontologie. Cette dernière est présentée avec le modèle des systèmes orienté-objet (OSM). L'éditeur proposé se compose de trois composantes :
  - Un éditeur de modèle structuré pour définir les relations entre les objets.
  - Un éditeur de trames de données pour définir graphiquement les expressions régulières utilisées pour définir les constantes et mots-clés.
  - Un module pour visualiser les documents et présenter les expressions régulières.
- Un Framework généralisé pour les systèmes d'extraction d'information basés sur l'ontologie [A. Wessman et al, 2005] : l'objectif derrière ce travail est de standardiser quelques modules du système BYU. Le Framework proposé offre de nouveaux algorithmes et idées qui peuvent êtres incorporés au système d'IE. Le but est de limiter l'effort d'extraction d'information du web uniquement dans l'implémentation de l'ontologie du domaine en question.

2.9. Comparaison des méthodes d'extraction

Après avoir présenté une taxonomie des approches d'extraction suivant la technique utilisée pour la génération d'adaptateur (*Section 2.5*), et suivant le degré de la l'intervention d'utilisateur (*Section 2.8*), ainsi qu'un panorama des méthodes d'extraction reconnues, nous tentons dans cette section de présenter une comparaison des approches d'extraction. Sans êtres exhaustifs, les deux travaux : celui de [A.H.F. Laender et al, 2002], et celui de [C.H. Chang et al, 2006] ont présenté des comparaisons pertinentes dans ce domaine.

Comme nous l'avons présenté dans la Section 2.5, dans [A.H.F. Laender et al, 2002] les auteurs ont présenté une taxonomie des outils d'extraction d'information à partir du Web. Ils se sont basés sur la technique utilisée par chaque méthode pour la génération de l'adaptateur. À cet effet, ils ont considéré cinq techniques qui assistent au processus de la définition des règles d'extraction. Ces techniques sont : les langages de description, l'analyse de la structure du document Web (HTML-aware), la modélisation de la structure du document Web (Modeling-based), le traitement automatique des langues naturelles (NLP-based), le mécanisme d'induction (Induction), et à base d'ontologie (Ontology-based). Afin d'évaluer les performances de chacune des approches, les auteurs ont introduit un ensemble de critères et mesures, ainsi la table 2.3 présente un résumé de l'analyse qualitative des méthodes :

- ✓ Le degré de l'automatisation : Permet de mesurer l'intervention de l'être humain dans la phase de la création des règles d'extraction.
- ✓ La complexité de la structure de l'objet : évalue la capacité des méthodes pour le traitement des structures de données complexes.
- ✓ Le contenu de la page traitée : Page Web semi-structurée (SD) ou texte libre semi-structuré (ST).
- ✓ La facilité de l'utilisation de l'outil : Il s'agit de mesurer la complexité de l'outil en question.
- ✓ Sources non Html : Possibilité de traitement des documents provenant des sources non HTML.
- ✓ Sortie en XML : Possibilité pour produire des résultats en XML.

Outils		degré de l'automatisation	complexité de la structure	Facilité de l'utilisation	Sortie XML	Source non HTML	Contenu de page
Langages	Minerva	manuel	codage	+	oui	partiel	SD
	TSIMMIS	manuel	codage	+	non	partiel	SD
	WebOQL	manuel	codage	+	non	aucun	SD
HTML-aware	W4F	Semi-autom	codage	+	oui	aucun	SD
	XWRAP	automatique	oui	++++	oui	aucun	SD
	RoadRunner	automatique	oui	++++	non	aucun	SD
NLP-based	WHISK	Semi-autom	non	++	non	idéal	ST
	PAPIER	Semi-autom	non	++	non	idéal	ST
	SRV	Semi-autom	non	++	non	idéal	ST
Induction	WIEN	Semi-autom	non	++	non	partiel	SD
	SoftMealy	Semi-autom	partiel	++	non	partiel	SD
	STALKER	Semi-autom	oui	++	non	partiel	SD
Modeling-based	NoDoSE	Semi-autom	oui	+++	oui	partiel	SD
	DEByE	Semi-autom	oui	+++	oui	partiel	SD
Ontoloy-based	BYU	manuel	codage	++	non	idéal	ST/ SD

Table 2.3. Résumé de l'analyse qualitative des méthodes d'extraction [A.H.F. Laender et al, 2002].

Lés résultats publiés dans la table ci-dessus montre que le degré d'automatisation dans les méthodes basées sur les approches Html-aware est très élevé, ainsi que 1'utilisation de 1'outil est toujours facile pour 1'utilisateur. Par contre, ces méthodes ne s'appliquent pas sur les documents non HTML.

Les méthodes basées sur l'induction sont capable d'extraire des informations provenant de quelques documents non HTML. Elles peuvent traiter les données ayants une structure complexe. Le degré de l'automatisation est moyen suite à 1' intervention d'utilisateur uniquement pour l'étiquetage des exemples d'entraînement. Pour le cas des méthodes basées sur le modèle prédéfini (Modeling-based), le degré de l'automatisation est moyen et l'utilisation de l'outil est facile pour l'utilisateur.

Le fait de considérer le développement et le test de l'ontologie, les méthodes basées sur l'approche de l'ontologie sont considérées manuelles. Mais d'un point de vue d'utilisateur (pour lui l'ontologie, d'extraction d'information à partir des sources d'un domaine spécifique, est disponible), l'extraction d'information est considérée automatique. Les méthodes basées sur les langages de description sont toujours manuelles. La liberté de description des règles d'extraction permet de traiter les structures de données complexes.

Parmi les six approches de conception d'extracteurs présentées, seules les méthodes basées sur le traitement du langage naturel permettent de traiter uniquement les textes libres. Le degré de l'automatisation est moyen et ces méthodes peuvent extraire des informations provenant des sources non Html. Par contre, ces méthodes n'acceptent pas les documents contenant des objets complexes.

[A.H.F. Laender et al, 2002] a visualisé la classification de ces approches suivant deux axes : le degré d'automatisation, et l'adaptabilité aux différents formats. Ci-dessous est présenté ce schéma :

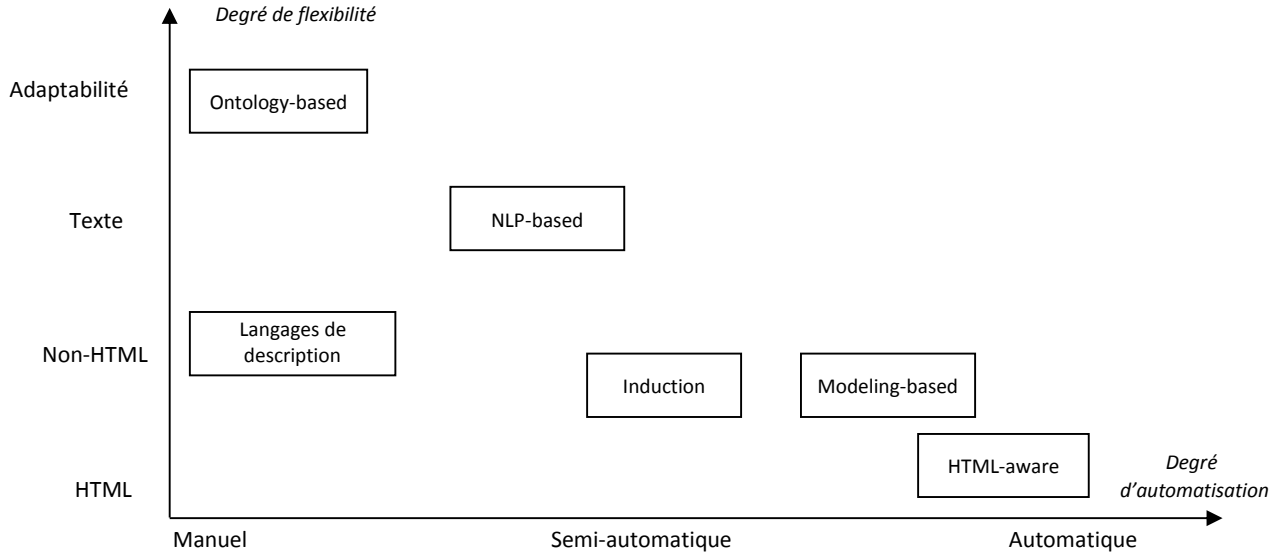


Fig 2.17 représentation graphique d'analyse qualitative des méthodes d'extraction.

Dans [C.H. Chang et al. 2006], les auteurs ont présenté une taxonomie pour les différentes méthodes d'extraction. Ils ont analysé les méthodes d'*WIE* sur la base de trois dimensions: les tâches de domaine, le degré de l'automatisation de la méthode et la technique d'extraction utilisée. Pour chaque, ils ont proposé un ensemble de critères pour évaluer les méthodes, ainsi qu'une table est dressée pour résumer les résultats obtenus pour chacune des axes citées. Cette étude sera détaillée dans le chapitre quatre, ainsi les résultats obtenus serrant utilisés pour la proposition de notre démarche d'*WIE*.

2.10. Conclusion

Tout au long de ce chapitre, nous avons abordé le sujet de l'extraction d'information à partir du web. Une tâche d'WIE consiste à construire un adaptateur (programme permettant l'extraction d'information à partir d'une source web), ainsi le cycle de vie de ce dernier a été présenté. Nous avons aussi expliqué comment évaluer une tâche d'WIE, les métriques permettant de réaliser une telle évaluation sont présentés, à savoir : le *Rappel*, la *Précision*, la *F-mesure*, et la métrique *Révision* (introduite récemment). Comme tous les domaines de recherche, l'WIE a été couplée avec d'autres domaines de recherche. Parmi ces derniers, nous avons décrit comment coupler le domaine d'WIE avec le celui des Services Web, d'un côté, et avec le domaine du Cloud Computing d'un autre côté.

L'objectif premier dans ce chapitre été d'explorer les différentes approches pour accomplir une tâche d'WIE. Pour ce faire, nous avons présenté les deux taxonomies pertinentes : celle basée sur le degré de l'intervention d'utilisateur pour réaliser l'extraction, et celle basée sur la technique utilisée pour décrire un adaptateur. Ainsi, dans une grande partie de ce chapitre, nous avons exploré les différentes méthodes d'WIE. En fin, une comparaison entres les approches d'extraction a été présentée.

CHAPITRE 3

INTEGRATION DE DONNEES

SOMMAIRE

3.1. Introduction

L'explosion du nombre de sources d'information accessibles via le Web multiplie les besoins de techniques d'accès automatique aux sources web autonomes et hétérogènes. Pour permettre un accès automatique multi-sources aux bases de données web, deux problèmes sont à résoudre : le premier concerne l'extraction d'information du web, et le deuxième concerne l'intégration des données. Dans le chapitre précédent, nous avons présenté l'extraction d'information du web. Ce chapitre traite le deuxième problème qu'est l'intégration de données.

Un système d'intégration est l'infrastructure via laquelle plusieurs sources de données autonomes, réparties et sous forme hétérogène (où chaque source est associée à un schéma local) sont intégrées sous forme de source unique représentée par un schéma global, en donnant, ainsi, l'impression à l'utilisateur qu'il utilise un système homogène.

Dans ce chapitre, l'étude de ce type de systèmes est abordée. Dans un premier temps la problématique d'intégration de données est présentée, ainsi l'hétérogénéité des données et les types de conflits qui peuvent êtres rencontrés dans une tâche d'intégration de données. En suite, nous présentons le système d'intégration : l'architecture et les tâche d'un système d'intégration, ainsi que le processus d'intégration de données. En fin, selon différents critères, serons présentées les approches d'intégration.

3.2. Problématique de l'intégration de données

Grâce à la révolution des nouvelles technologies de l'internet, les entreprises aussi bien que les individus disposent d'une grande quantité de données. Ces données sont stockées dans des sources qui sont dans la plupart du temps hétérogènes et autonomes. D'où, le problème d'intégration de données. Cette dernière a pour but, selon [M. Lenzerini, 2002], de combiner les données réparties dans différentes sources et de fournir à l'utilisateur une vue unifiée de ces données. Par sources de données on entend, les informations enregistrées dans des bases de données traditionnelles et accessibles par des langages de requête très puissants, d'autres simplement stockées dans des systèmes de fichiers ou de simples tableurs, d'autres encore sont les résultats d'applications plus ou moins complexes, enfin, certaines sont les données variables et contenues dans les pages web et présentées directement sur les navigateurs.

Plusieurs problèmes doivent être pris en compte pendant la conception de systèmes d'intégration. Ces problèmes résultent de l'hétérogénéité des données. En effet, les sources de données ont été conçues indépendamment par des concepteurs différents, ceci explique le fait que les données relatives à un même sujet sont représentées différemment sur des systèmes d'information distincts.

Cette hétérogénéité provient des choix différents qui sont faits pour représenter des faits du monde réel dans un format informatique. Les données des sources sont, donc, structurellement indépendantes mais sont toujours supposées relever de domaines similaires.

3.3. Hétérogénéité des données

Elle peut être de deux natures : hétérogénéité structurelle et hétérogénéité sémantique.

3.3.1. Hétérogénéité structurelle

Concerne la manière dont sont représentées les données. Les différences structurelles peuvent être regroupées dans différentes catégories :

- Choix de type de données : ces conflits se posent lorsqu'on utilise des types de données différents pour la même information. Par exemple, dans le domaine médical, la période d'infection pour une maladie est représentée par un entier dans une source S1, et par une chaîne de caractère dans une autre source S2.
- Choix du nombre de constructeurs : ces conflits se présentent lorsque le nombre de constructeurs modélisant une information est différent d'une source à une autre. Par exemple, l'attribut nom d'un patient, est modélisé par un seul attribut servant à stocker le nom et le prénom d'un patient dans une source S1, alors que deux attributs sont utilisés dans une autre source 2.
- Choix des informations représentées : ces conflits se posent lorsqu'une information est représentée dans des sources alors qu'elle ne l'est pas dans d'autres. Par exemple, l'adresse d'un patient n'est pas connue pour tous les patients d'une source S1, alors que c'est une donnée obligatoire dans une source S2.

3.3.2. Hétérogénéité sémantique

Elle est en rapport avec la signification des données (synonymes, homonymes, etc.). Plusieurs types de conflits dus à l'hétérogénéité peuvent être considérés dans l'établissement des correspondances entre schémas lors de l'intégration de données. De nombreuses taxonomies de conflits ont été proposées, , par exemple [C.H. GOH et al, 1999] proposent une classification des conflits qui peuvent apparaître lors de la mise en correspondance entre schémas :

- Les conflits de nommage : se produisent lors de l'attribution des noms dans des schémas qui diffèrent de manière significative. Les cas les plus fréquents sont les cas de présence de synonymes et d'homonymes.
- Les conflits de graduation : apparaissent lorsque différents systèmes de graduation sont utilisés pour mesurer une valeur. par exemple : Dollar ou Euro.

- Les conflits de confusion : se produisent lorsque les concepts paraissent avoir la même signification mais diffèrent en réalité. Il peut être causé par des contextes temporels différents par exemple. Par exemple le poids d'une personne dépend de la date où elle se pèse.
- Les conflits de représentation : se produisent quand deux schémas sources décrivent le même concept de manière différente. Par exemple, dans une source l'adresse peut être désignée par une chaîne de caractères tandis que dans une autre, l'adresse est une structure composée du numéro et du nom de la rue, du code postal et de la ville.

3.4. Systèmes d'intégration de données

Un système d'intégration a pour but de fournir un schéma cohérent des données provenant de multiples sources d'informations autonomes, réparties et hétérogènes, de manière à faciliter aux utilisateurs l'accès et l'interrogation de ces données, comme s'ils accédaient à une seule source de données. Dans cette section, nous présentons l'architecture d'un système d'intégration, ses tâches, ainsi que le processus d'intégration.

3.4.1. Définition

Un système d'intégration de données fournit une vue unifiée de données provenant de sources multiples et hétérogènes. Il permet d'accéder à ces données à travers d'une interface uniforme, sans se soucier de leur structure ni de leur localisation [A. BOUSSIS, 2008]. Il comporte, généralement, deux parties principales : une partie externe et une partie interne (comme illustré dans la figure suivante):

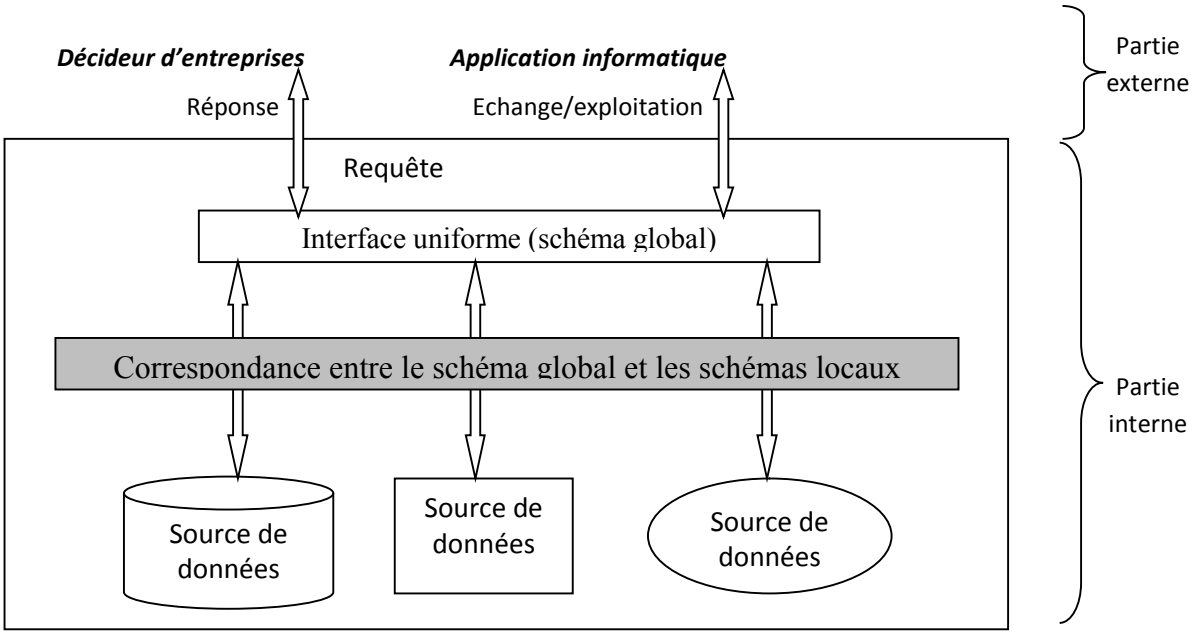


Fig 3.1. Architecture générale d'un système d'intégration [D.N. Xuan, 2006].

- La partie externe : correspond aux utilisateurs du système intégré comme, par exemple, les décideurs d'une entreprise, ou d'autres systèmes.
- La partie interne : comprend d'une part les sources d'informations participant dans le processus d'intégration et d'autre part une interface (schéma global) permettant aux éléments de la partie externe d'accéder d'une manière transparente aux sources de données. Cette interface peut être une couche sans données propres (approche médiateur) ou une couche contenant sous une forme qui lui est propre une duplication des données pertinentes des sources (approche entrepôt). Les éléments externes au système ne voient pas les sources internes. Celles-ci sont encapsulées, cachées par l'interface. Les éléments externes doivent pouvoir agir sur le système d'intégration d'une manière transparente via un schéma global.

3.4.2. Tâche d'un système d'intégration

On peut distinguer quatre tâches principales d'un système d'intégration. Les deux premières concernent la traduction de données provenant des sources différentes et résolvent le problème de l'hétérogénéité physique/logique des sources en fournissant une interface d'accès uniforme. Les deux dernières résolvent le problème de l'hétérogénéité sémantique en reliant chaque source au schéma global. Ces quatre tâches sont décrites ci-après [A. BOUSSIS, 2008] :

1. Transformation de données: Un exemple typique de cette tâche est la transformation de données relationnelles en XML et inversement. Les problèmes importants qui doivent être résolus à ce niveau sont la perte d'information, la taille des données générées et la performance des traitements sur ces données. L'exploitation de la structure de données à transformer (types, schémas) joue un rôle crucial dans ce contexte.

2. Traduction de requêtes: La traduction de requêtes d'un langage en un autre langage (exp. XQuery en SQL) est liée au problème de transformation de données. Elle doit également prendre en compte la puissance d'expression du langage cible et nécessite souvent des extensions spécifiques afin d'obtenir la puissance d'expression du langage source.

3. Réécriture de requêtes: Cette tâche est différente de la tâche précédente et généralement plus complexe car elle doit prendre en compte l'hétérogénéité structurelle et sémantique entre les schémas. Elle joue un rôle primordial dans l'intégration de données sur le Web.

4. Fusion de données: La fusion de données essaye de répondre au problème de la représentation multiple d'une même information dans différentes sources. Elle fait partie de la tâche de réécriture de requêtes, mais se pose également dans un environnement où les données sont matérialisées .

3.4.3. Processus d'intégration

Selon [C. Parent et S. Spaccapietra, 1996], Le processus d'intégration est ainsi décomposé en trois phases distinctes :

- ✓ Le pré intégration : Cette phase vise à préparer l'intégration des schémas en les rendant plus homogènes. Elle consiste principalement à traduire les schémas initiaux dans un modèle de données commun (réduction de l'hétérogénéité syntaxique). Elle s'attache également à enrichir leur sémantique.
- ✓ L'identification des correspondances : Durant cette phase, les correspondances entre les éléments des schémas source sont détectées et formalisées de même que les différents conflits.
- ✓ L'intégration : Cette phase finale produit le schéma intégré et fournit les règles de traduction permettant de passer des schémas source au schéma intégré et inversement « mapping ».

3.5. Les approches d'intégration

On s'appuie dans ce qui suit sur la classification des approches d'intégration proposée par [D.N. Xuan, 2006] qui classifie ces approches selon les trois critères orthogonaux suivants :

(1) la représentation des données intégrées (ou le degré d'automatisation)

(2) le sens de mise en correspondance entre schéma global et schémas locaux

(3) la nature du processus d'intégration

3.5.1. La représentation des données intégrées (degré d'automatisation)

Ce critère permet de distinguer les approches d'intégration selon la manière dont elles stockent les données. On distingue ainsi : l'approche virtuelle et l'approche matérialisée.

3.5.1.1. Approche virtuelle

Cette approche consiste à développer une application chargée de jouer le rôle d'interface entre les sources de données locales et les applications d'utilisateurs. Il repose sur deux composants essentiels : le médiateur et l'adaptateur.

✓ Le médiateur : chargé de la localisation des sources de données et des données pertinentes par rapport à une requête, il résout de manière transparente les conflits de données. Un ensemble de connaissances sur les sources permet au médiateur de générer un plan d'exécution pour traiter les requêtes d'utilisateurs.

✓ L'adaptateur : c'est un outil permettant à un (ou plusieurs) médiateur(s) d'accéder au contenu des sources d'informations dans un langage uniforme. Il fait le lien entre la représentation locale des informations et leur représentation dans le modèle de médiation.

Dans cette approche, les données ne sont pas copies, elles restent dans les sources de données. L'approche médiateur présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leur source d'origine.

Un système de médiation est donc constitué du médiateur, d'un ensemble d'adaptateurs et d'un ensemble de sources locales. Les sources locales sont intégrées dans le sens qu'un seul schéma global est construit à partir des schémas exportés par les sources locales.

Cette approche est basée sur un modèle de décomposition de requête, d'envoi de sous requêtes aux sources de données et de combinaison des résultats obtenus. Des requêtes sont ainsi posées sur le schéma global et traitées par le médiateur. Les schémas des sources sont construits par des adaptateurs qui établissent la correspondance entre la source et le médiateur. La *figure 3.2* présente le principe de cette approche.

Dans le cadre d'intégration de bases de données web suivant l'approche médiateur, on peut citer comme exemple les travaux [H. He et al, 2004] [J. Wang et al, 2004] [W. Rui et W. Nianbin, 2010] qui proposent des méthodes pour l'intégration des interfaces de recherche (formulaires) web.

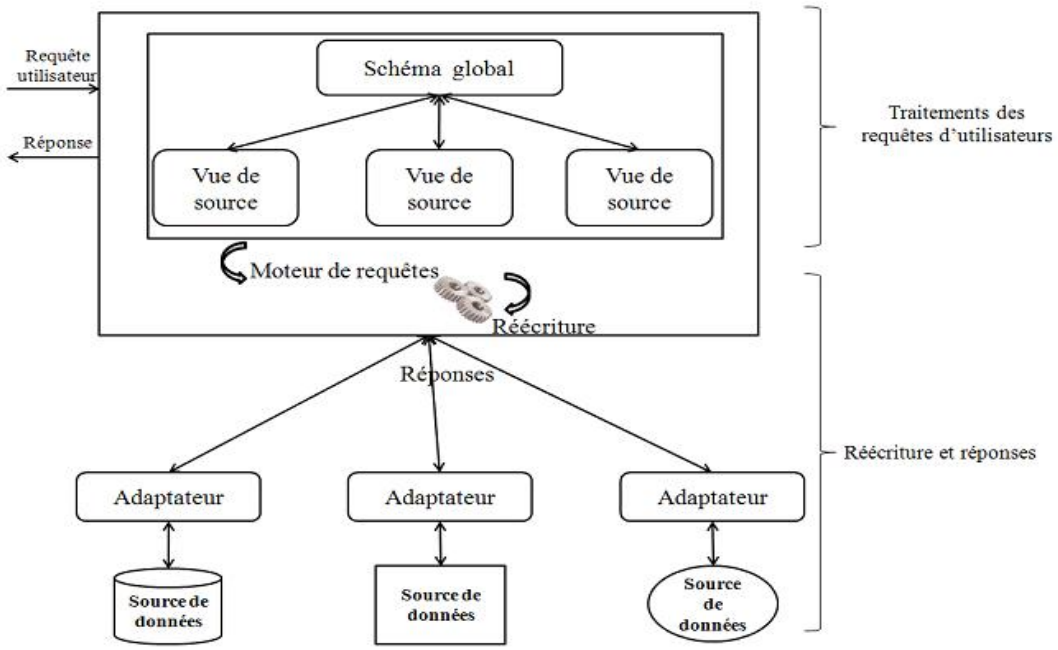


Fig 3.2. Intégration virtuelle [S. KHOURI, 2009].

3.5.1.2. Approche matérialisée

Cette approche consiste à centraliser physiquement, dans un entrepôt de données (il ya donc duplication des données), l'ensemble de données consolidées à partir de diverses sources (catalogues électroniques, bases de données relationnelles, Web, etc.).

Le processus de construction est décomposé en quatre étapes principales [L. Fatima, 2011]:

(1) l'extraction des données des sources de données opérationnelles
(2) la transformation des données aux niveaux structurel et sémantique
(3) l'intégration des données
(4) le stockage des données intégrées dans le système cible.

Comme dans le cas de l'approche virtuelle, chaque source de données est liée à un adaptateur, ou wrapper. Au niveau supérieur, un intégrateur est chargé d'alimenter la base cible avec une copie des données issues des sources selon le schéma global défini pour cette base. L'utilisateur interroge la base cible, et donc la copie des données intégrées des sources de données, sans avoir à connaître le schéma ou les méthodes d'accès propres à chaque source de données.

L'avantage principal d'une telle approche est que l'interrogation d'un entrepôt de données se fait directement sur les données de l'entrepôt et non sur les sources originales. Nous pouvons donc utiliser les techniques d'interrogation et d'optimisation des bases de données traditionnelles. La principale problématique liée à la matérialisation est la conservation des données de la base cible à jour. Selon les besoins, ces données sont généralement mises à jour sur des bases périodiques : mensuelles, hebdomadaires, quotidiennes, horaires, ….etc.

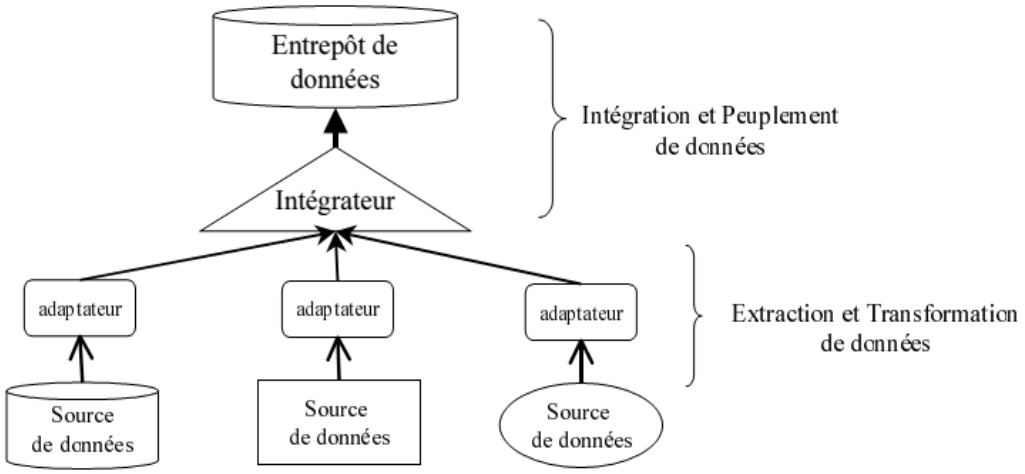


Fig 3.3. Intégration matérialisée [D.N. Xuan, 2006]

3.5.2. Le sens de mise en correspondance entre schéma global et schémas locaux

Ce critère permet de distinguer les approches d'intégration selon la liaison entre le schéma global et les schémas locaux des sources à intégrer. On distingue pour ce critère deux approches : GaV (Global as View) et LaV (Local as View).

3.5.2.1. Approche GAV

L'approche GAV a été la première à être proposée pour intégrer des informations. Elle consiste [L. Fatima, 2011] à définir à la main (ou de façon semi-automatique) le schéma global en fonction des schémas des sources de données à intégrer puis à le connecter aux différentes sources. Pour cela, les prédicats du schéma global, aussi appelés relations globales, sont définis comme des vues sur les prédicats des schémas des sources à intégrer. Comme les requêtes d'un utilisateur s'expriment en termes des prédicats du schéma global, nous obtenons facilement une requête en termes du schéma des sources de données intégrées, en remplaçant les prédicats du schéma global par leurs définitions.

Les principales caractéristiques de l'approche GAV sont les suivantes [C. Gueydan, 2010]:

- L'approche GAV n'est pas modulaire : l'ajout ou la modification des sources de données nécessite des modifications du schéma global.
- La reformulation de requête est simple, elle se réduit à un « dépliage » des vues.

3.5.2.2. Approche LAV

L'approche LAV est l'approche duale de GAV. Elle définit les schémas des sources à intégrer en fonction du schéma global. Les relations des schémas des sources (relations locales) sont définies comme des vues sur les relations globales [S. KHOURI, 2009].

L'avantage principal de l'approche LAV est la facilité offerte pour l'ajout d'une nouvelle source. Cependant l'évaluation des requêtes peut s'avérer complexe. Par ailleurs, tout changement au niveau du schéma global peut être difficile à répercuter au niveau des vues définissant les sources. C'est pourquoi cette approche est préconisée dans le cas où le schéma global n'est pas sujet à de fréquentes modifications [A. BOUSSIS, 2008].

L'approche GAV est en général plus appropriée aux systèmes utilisant des sources de données stables tandis que l'approche LAV convient à des systèmes utilisant beaucoup de sources de données relativement « inconnues » du système et nécessitant l'addition, la modification ou la suppression de sources [C. Gueydan, 2010].

3.5.3. La nature du processus d'intégration

Ce dernier critère permet de distinguer les approches d'intégration selon le degré d'automaticité du processus d'intégration. Notons que cette automaticité concerne la résolution ou l'élimination des conflits sémantiques entre les sources durant le processus d'intégration. On distingue les approches manuelles, semi-automatiques et automatiques.

3.5.3.1. Approche manuelle

Pour cette approche, les conflits sémantiques sont gérés manuellement et nécessitent la présence d'un expert humain pour interpréter la sémantique des données. Ces approches manuelles deviennent impraticables lorsque le nombre de sources de données à intégrer est important, ou lorsque les sources évoluent fréquemment.

3.5.3.2. Approche semi-automatique

Cette approche consiste à se baser sur une ontologie linguistique[1] pour la résolution des conflits sémantiques. Une ontologie linguistique permet de fournir la sémantique des concepts d'un domaine de manière formelle. Elle permet d'automatiser partiellement la gestion des conflits sémantiques. Les ontologies linguistiques traitent des termes, et non des concepts. Ceci peut créer des conflits de noms. ***MOMIS*** [D. Beneventano et al, 2000] est un exemple de projet reposant sur l'ontologie linguistique *WordNet* pour l'intégration des sources.

3.5.3.3. Approche automatique

Elle consiste à associer aux données des sources une ontologie conceptuelle qui en définit le sens. Une ontologie conceptuelle traite les concepts d'un domaine donné. Elle est définit comme « une spécification formelle et explicite d'une conceptualisation partagée d'un domaine de connaissance » [T.R. Gruber, 1993]. La sémantique du domaine est ainsi spécifiée formellement à travers des concepts, leurs propriétés ainsi que les relations entre les concepts. La référence à une ontologie permet d'éliminer automatiquement les conflits sémantiques entre les sources. La connexion entre les sources et les ontologies peut se faire de plusieurs manières [H. Wache et al, 2001]: définition des termes de la BD par les concepts ontologiques, annotation des données, enrichissement de la source par des règles logiques (logique de description).

[1] *Une ontologie linguistique définit le sens des mots utilisés dans un domaine d'étude. Ces mots sont essentiellement liés par des relations linguistiques telles que la synonymie, l'antinomie, l'hyperonymie, etc.*

3.6. Comparaison des approches d'intégration

Le tableau ci-dessous présente une comparaison entre l'approche médiateur et l'approche matérialisée pour l'intégration d'information.

Approche	Avantages	Inconvénients
l'approche médiateur	✓ l'alimentation d'une base cible n'est pas nécessaire (il n'existe pas de base cible) ✓ les données sont toujours à jour car les réponses aux requêtes de l'utilisateur sont directement tirées des bases sources	✓ les temps de réponse peuvent être longs ✓ il existe des possibilités d'incohérences dues à d'éventuelles données redondantes sur différentes sources de données ou encore à des traitements concurrents mal contrôlés ✓ la mise en place d'un médiateur est une tâche complexe ✓ l'approche est inadaptée aux requêtes nécessitant des agrégations
l'approche matérialisée	✓ les temps de réponses sont courts car l'utilisateur interroge une seule base de données (la base cible) ✓ les réponses sont sûres car il n'existe pas d'incohérence liée aux données redondantes sur différentes sources ou aux traitements concurrents ✓ les techniques sont éprouvées car cette approche, notamment l'entrepôt de données, est utilisée par les entreprises depuis plusieurs années (le terme « entrepôt de données » date du début des années 90) ✓ l'approche est adaptée aux requêtes nécessitant des agrégations	✓ le processus d'alimentation est long et coûteux en termes de ressources réseaux et système ✓ les données datent de la dernière alimentation (ou réalimentation) effectuée

Table 3.1. Comparaison des approches d'intégration Médiateur et matérialisée [C. Gueydan, 2010].

3.7. Conclusion

Dans ce chapitre nous avons expliqué le problème d'intégration de données, ainsi les niveaux d'hétérogénéité des données qui empêchent une intégration souple des données. Nous avons ainsi étudié les principales notions relatives au système d'intégration : définition, tâches, et processus d'intégration.

Les méthodes de conception des solutions d'intégration peuvent être classifiées, selon la représentation des données intégrées, en deux approches : celle basée sur le médiateur pour intégrer les données de différentes sources, et celle basée sur l'entrepôt de données. On a étudié les avantages et les inconvénients de chacune de ces approches.

CHAPITRE 4

PROPOSITION D'UNE METHODE POUR L'IEW MULTI-SOURCES

SOMMAIRE

4.1. Introduction

L'objectif du processus de l'extraction d'information à partir du Web est de construire des programmes permettant de générer un ensemble de règles d'extractions, de les expérimenter, et de les appliquer sur un document Web, ces programmes sont appelés adaptateurs. Dans la littérature, la majorité des méthodes permettent de construire des programmes applicables sur une seule source de données.

D'autres méthodes ont été proposées afin de répondre au problème de l'extraction multi-sources, elles exploitent les connaissances d'un domaine bien spécifique pour générer des adaptateurs capables d'extraire les informations provenant de différentes sources du même domaine. Cependant les méthodes proposées souffrent d'un inconvénient major. En effet, l'étape essentielle et commune pour ces méthodes est la construction d'avance des connaissances de domaine d'intérêt (ontologie), cette tâche est très couteuse. Notamment, recenser l'ensemble des formes de présentations possibles d'informations nécessite une analyse de l'ensemble des sites, desquelles l'utilisateur souhaite extraire d'informations, afin de les intégrer dans la connaissance du domaine.

Dans ce chapitre nous proposons une méthode pour l'extraction multi-sources. Elle est basée sur l'analyse des concepts et relations exploitées par les différentes méthodes d'extraction d'information du web, l'objectif étant de trouver la méthode d'*WIE* adéquate pour chaque source web. La méthode proposée est très utile, car elle permet d'optimiser le processus d'*WIE*. Dans notre contexte nous considérons que le jugement d'une méthode d'*WIE* pour chaque source se fait vis-à-vis les contraintes des méthodes d'extraction, les critères des sources web, et les compétences de l'utilisateur.

4.2. Aperçu général de la méthode proposée

Notre méthode d'*WIE* multi-sources est basée sur l'exploitation de la connaissance dans une ontologie, qui fournit des services d'inférence et des mécanismes de raisonnement puissants. Ce qui nous permet de développer un système expert, qui à partir d'un ensemble de sources web, inférer la méthode d'IE adéquate pour chaque source. Les données introduites dans l'ontologie sont classées suivant trois axes :

1) les critères des méthodes d'extraction, par ex : le type des pages considérées (structurée, semi-structurée, texte). Le niveau de traitement (niveau page, niveau enregistrement…) etc. Ces critères sont disponibles dans notre ontologie pour chacune des méthodes.

2) les critères concernant l'utilisateur de notre système, par ex : le niveau d'exploitation de l'utilisateur (programmation, étiquetage, novice …. etc.). Ces critères sont introduits par l'utilisateur de notre système.
3) les critères concernant les sources web, par ex : le type de pages web (Template, semi-structurée, texte), la forme des informations présentées : tableaux, listes … etc. Ces critères sont obtenus automatiquement, en appliquant des techniques de web content mining sur la source web.

Afin d'obtenir les critères concernant les sources web, nous avons basé sur l'observation suivante : les sources web qui présentent des informations issues d'une base de données web, génèrent des pages web dynamiquement suivant certain schéma (Template). Pour obtenir automatiquement les caractéristiques pour la source web, nous avons proposé une technique intelligente qui consiste en premier temps à classifier les pages de la source web en clusters, de telle sorte que les pages du même cluster ont le même schéma (ayant exactement la même structure arborescente partant de la racine jusqu'aux feuilles). La classification se base sur l'aspect structurel des pages web, ainsi, pour chaque niveau (de la structure arborescente) parcouru les pages sont classées suivant les balaises, décrivant l'aspect visuel des pages web, en clusters. Cette technique de classification nous permet d'éliminer les blocks bruyants qui se trouvent dans la page web, et de faciliter la détection des caractéristiques de la source web. Dans un deuxième temps, des pages à partir de clusters différents sont comparées, ainsi des heuristiques sont proposées pour la détection des caractéristiques de la source web.

Une étude préalable des méthodes d'WIE dans le chapitre 2 était nécessaire afin de comprendre les concepts et les mécanismes utilisés pour chaque méthode, dans [C.H. Chang et al. 2006], les chercheurs ont présenté une taxonomie pour les différentes méthodes d'extraction. Ils ont comparées les méthodes d'*WIE* sur la base de trois dimensions: les tâches de domaine, le degré de l'automatisation de la méthode et la technique d'extraction utilisée. À cet effet, ils ont proposé un ensemble de critères pour mesurer la fiabilité et l'avantage de chacune par apport à l'autre. Cette étude nous a permis donc de dégager des éléments importants, tels que les attributs, les types de données utilisées, la taille, etc. ces éléments aident à structurer les données nécessaires au futur système, notamment dans la conception de notre ontologie. Parmi les 19 méthodes étudiées nous avons choisi 11 méthodes à utiliser dans notre méthode, le tableau suivant présente les résultats obtenus.

Méthodes d'*WIE*		Type-page	Niv-extract	Contraintes d'attributs						Scan Pass	Features Used	Learning Algorithm	Token	User Expertise	Applica	Limitation
				MA / MVA	MOA	Nested	VF	CF	UTA							
Manuelle	WebOQL	Semi-S	Record	Y	Y	Y	Dj	BO	N	Single	Hyper-tr	None	Manual	Program	High	Not restricted
	W4F	Temp	Record	Y	Y	Y	SP	BO	Y	Single	DOM-tr	None	Tag Level	Program	Medium	Not restricted
Supervisée	WHISK	Free	Record	Y	Y	N	Dj	BO	Y	Single	Synta/ Semantic	Set Covering	Word Level	Labeling	Medium	Not restricted
	WIEN	Semi-S	Record	N	N	Lim	N	BO	N	Single	HTML / words	Ad-hoc	Word Level	Labeling	Medium	Not restricted
	STALKER	Semi-S	Record	Y	Y	Y	Bt	MC	N	Multi	HTML / words	Ad-hoc	Word Level	Labeling	Medium	Not restricted
	SoftMealy	Semi-S	Record	Y	Lim	Multi Pass	Dj	BO SgP	Y	Both	HTML / words	Ad-hoc	Word Level	Labeling	Medium	Not restricted
Semi-supervisée	IEPAD	Temp	Record	Y	Lim	Lim	Bt	BO	Y	Single	HTML tags	Pattern-M String-A	Multi-Level	Pos- lab / Pattern-S	Low	Mult-R- page
	OLERA	Temp	Record	Y	Lim	Lim	Bt	BO	Y	Single	HTML tags	String-A	Multi-Level	Partial-Lab	Low	Not restricted
Non-super	BYU	Semi-S	Record	Y	Y	Y	Y	MC	Y	Multi	DOM-tr	None	Word Level	Program-ontology	High	Mult-R- page
	DeLa	Temp	Record	Y	Lim	Y	Bt	BO	N	Single	HTML tags	Pattern-M	Tag Level	Pattern-S	Low	Mult-R- page, More-one-page
	RoadRunner	Temp	Page	Y	N	Y	N	BO	N	Single	HTML tags	String-A	Tag Level	Pattern-S	Low	More-one-page

Table 4.1. Les critères des méthodes d'extraction.

4.3. Architecture général de la méthode proposée

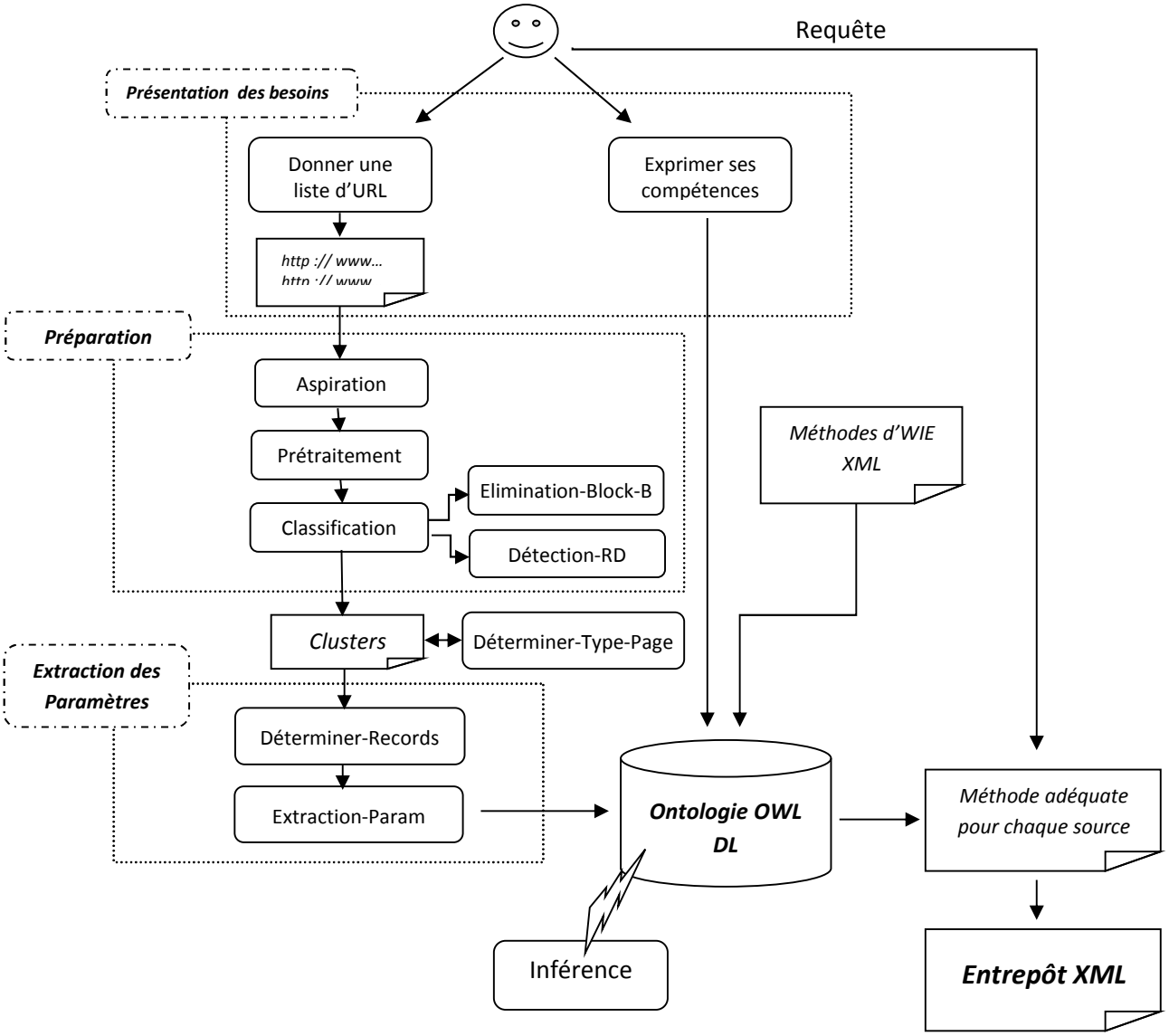


Fig 4.1. Architecture générale du système proposé.

4.4. Construction de l'ontologie

Dans cette section, nous construisons notre ontologie d'application qui concerne l'WIE multi-sources. A cette fin, nous suivrons les étapes du processus de construction d'ontologies [S. BOUARROUDJ, 2010] : spécification, conception, formalisation.

4.4.1 Spécification des besoins

- **Domaine :** Description des méthodes d'IE et de sources web selon leurs caractéristiques.

- **Objectif opérationnel (But):** l'ontologie regroupe des règles d'inférence utilisées dans un processus de raisonnement lors de la recherche de la méthode d'IE adéquate pour chaque source web.
- **Utilisateurs :** utilisateurs de notre système.
- **Porté (liste des termes importants) :** méthode d'IE, source web,….
- **Source de connaissances :** Classification des méthodes d'extraction (table..), les sources web.
- **Scénarios d'usage :** suggérer la méthode d'IE adéquate pour une source web.

4.4.2 Conception

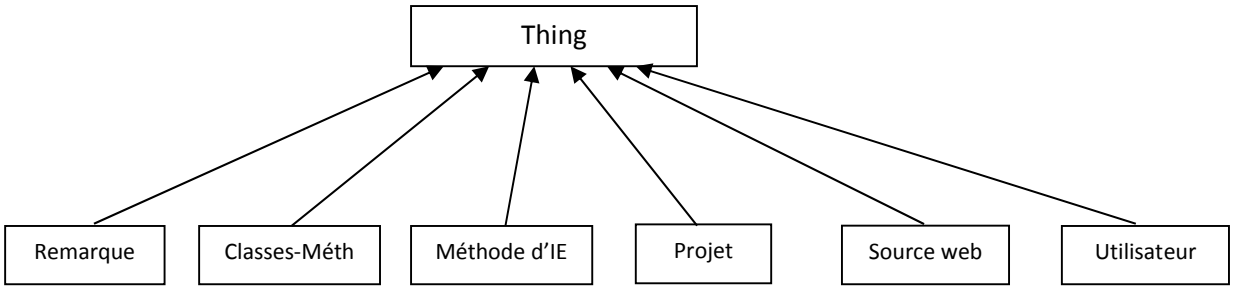


Fig 4.2. Taxonomie des concepts de l'ontologie proposée.

4.4.3 Dictionnaire des concepts

Concept	***Description***	***Parent***	***Attributs***
Méthode d'WIE	Le concept qui présente les méthodes d'WIE	Thing	T-pge, Niv-ext, MA / MVA, MOA, Nested, VF, CF, UTA, Scan-Ps, Features-U, Learning-alg, Token, User-Exp, Applica, Limitation
Classes-Méth	Le concept qui présente les classes des méthodes d'WIE (manuelle, supervisée, semi-supervisée, et non-supervisée)	Thing	/
Remarque	Cette classe regroupe les remarques signalées après l'inférence, par exemple : si aucune méthode n'est adéquate pour le niveau de manipulation d'utilisateur, alors on propose une méthode de niveau supérieur avec une remarque indiquant le niveau demandé. Un deuxième cas : si par exemple, l'extraction d'une certaine situation n'est pas possible pour la méthode adéquate inférée, alors on signale cette situation par une remarque.	Thing	/
Utilisateur	Le concept qui présente l'utilisateur.	Thing	Experience
Projet	Le concept qui présente un projet d'WIE multi-sources.	Thing	/
Source web	Le concept qui présente les sources web	Thing	Type-p, Niv-det, MA / MVA, MOA, Nested, VF, CF, UTA, Niv-données

Table 4.2. Dictionnaire des concepts.

4.4.4 Attributs des concepts

Attribut	*Description*	*Concept*	*Type*
T-pge	Type de page pris par la méthode d'extraction (texte, structurée, semi-structurée)	Méthode d'WIE	énuméré
Niv-ext	Niveau d'extraction (page, enregistrement, champ)	Méthode d'WIE	énuméré
MA / MVA	Propriété pour les champs (valeur absente/ multi-valeurs)	Méthode d'WIE	booléen
MOA	Propriété pour les champs (attribut à plusieurs valeurs)	Méthode d'WIE	booléen
Nested	Propriété pour les champs (non atomique à plusieurs valeurs)	Méthode d'WIE	booléen
VF	(Varied Format) Présentation variée pour le même champ	Méthode d'WIE	booléen
CF	(Common Format) même présentation pour plusieurs champs	Méthode d'WIE	booléen
UTA	(Untokenized Attributes) Attributs concaténés	Méthode d'WIE	booléen
Scan-Ps	NB-passages (sur la page web) pour générer l'adaptateur	Méthode d'WIE	Entier
Features-U	Technique utilisée pour la présentation de la page web (HTML et/ou Word, DOM)	Méthode d'WIE	énuméré
Learning-alg	La classe de l'algorithme utilisé pour générer l'adaptateur (Non, Ad-hoc, Pattern-Mining et/ou String-A)	Méthode d'WIE	énuméré
Token	Le niveau d'abstraction de la page web à traiter (TAG et/ou Word)	Méthode d'WIE	énuméré
User-Exp	Le niveau de manipulation d'utilisateur demandé (Programmation : motif / ontologie, labelling, post-labelling, novice)	Méthode d'WIE	énuméré
Applica	(Applicability) Possibilité d'appliquer le même adaptateur au cas de changements sur la source web (High, Medium, Low)	Méthode d'WIE	énuméré
Limitation	Contrainte sur la page pour pouvoir appliquer chaque méthode (Not restricted, Multi-R-page et/ou More-one-page)	Méthode d'WIE	énuméré
Type-p	Type de pages de la source web (texte, structurée, semi-structurée)	Source web	énuméré
Niv-detail	Niveau de détail des pages web (un/ plusieurs) enregistrement(s)	Source web	énuméré
MA / MVA-sw	Propriété pour les champs (valeur absente/ multi-valeurs)	Source web	booléen
MOA-sw	attributs multi-ordre	Source web	booléen
Nested-sw	Enregistrement composé (sous forme hiérarchique)	Source web	booléen
VF-sw	Présentation variée pour le même champ	Source web	booléen
CF-sw	même présentation pour plusieurs champs	Source web	booléen
UTA-sw	(Untokenized Attributes) absence de délimiteurs entre deux Attributs	Source web	booléen
context	S'il ya un nombre considérable de champs typés	Source web	booléen
Niv-donnees	(Niveau de données à extraire) Balaise ou Mot	Source web	énuméré
Experience	(Programmation : motif / ontologie, labelling, post-labelling, novice)	Utilisateur	énuméré
domaine	Si toutes les sources web traitent le même domaine, on l'associe au projet	Projet	String

Table 4.3. Attributs des concepts.

4.4.5 Table de relations binaires

Relation	*Description*	*Concept Source*	*Concept Cible*	*(Min/Max)*
classe-pour-prj	Pour chaque projet on l'associe la classe adéquate suivant le niveau de manipulation d'utilisateur	Projet	Classes-Méth	(1,1)
meth-pour-projet	Cette relation est établie uniquement si la méthode adéquate est « BYU »	Projet	Méthode d'WIE	(0,1)
classe-pour-src	Pour chaque source web on associe la classe adéquate : initialement la classe adéquate pour le projet est affectée aussi à la source, si aucune méthode n'est adéquate dans cette classe, alors on ajoute (change) la classe.	Source web	Classes-Méth	(1, n)
meth-pour- src	Pour chaque source web on associe la méthode d'WIE adéquate.	Source web	Méthode d'WIE	(1,1)
Remarque-pour-src	Etablissement éventuel des remarques pour les sources web (voir le dictionnaire des concepts / section 4.4.3)	Source web	Remarque	(0, n)

Table 4.4. Liste des relations binaires.

4.4.6 Règles d'inférence avec le langage SWRL

R1. projet (? p) ^ source_web(?s) ^ utilisateur(?u) ^ Experience(?u, "Prog-motif") → classe-pour-prj(?p, manuelle), classe-pour-src(?s, manuelle)

R2. projet (? p) ^ source_web(?s) ^ utilisateur(?u) ^ Experience(?u, "novice") → classe-pour-prj(?p, non-supervisee), classe-pour-src(?s, non-supervisee)

R3. projet (? p) ^ source_web(?s) ^ utilisateur(?u) ^ Experience(?u, "post-labelling") → classe-pour-prj(?p, semi-supervisee), classe-pour-src(?s, semi-supervisee)

R4. projet (? p) ^ source_web(?s) ^ utilisateur(?u) ^ Experience(?u, "labelling") → classe-pour-prj(?p, supervisee), classe-pour-src(?s, supervisee)

R5. projet (? p) ^ source_web(?s) ^ utilisateur(?u) ^ Experience(?u, "Prog-onto") → classe-pour-prj(?p, non-supervisee), classe-pour-src(?s, non-supervisee)

R6. Projet (? p), utilisateur (? u), Experience(?u, "Prog-onto"), domaine(?p, ?d) → meth-pour-projet (? p, BYU)

R7. source_web(?s) ^ classe-pour-src(?s, manuelle) ^ Type-p(?s, "semi_structuree") ^ UTA-sw(?s, false) → meth-pour-src(?s, WebOQL)

R8. source_web(?s) ^ classe-pour-src(?s, manuelle) ^ Niv-donnees(?s, "WORD") ^ Type-p(?s, "structuree") → meth-pour-src(?s, WebOQL)

R9. source_web(?s) ^ classe-pour-src(?s, manuelle) ^ Niv-donnees(?s, "TAG") ^
Type-p (? s, "structuree") → meth-pour-src(?s, W4F)

R10. source_web(?s) ^ classe-pour-src(?s, manuelle) ^ MOA-sw(?s, false) ^
Type-p(?s, "semi_structuree") ^ UTA-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R11. source_web(?s) ^ classe-pour-src(?s, manuelle) ^ MOA-sw(?s, true) ^
Type-p(?s, "semi_structuree") ^ UTA-sw(?s, true) →
Remarque-pour-src(?s, extraction_UTA_non_possible), meth-pour-src(?s, WebOQL)

R12. source_web(?s) ^ Type-p(?s, "texte") → classe-pour-src(?s, supervisee)

R13. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ Type-p(?s, "texte") →
meth-pour-src(?s, WHISK)

R14. source_web(?s) ^ meth-pour-src(?s, WHISK) ^ Nested-sw(?s, true) →
Remarque-pour-src(?s, extraction_Nested_non_possible)

R15. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, false) ^
MOA-sw(?s, false) ^ Nested-sw(?s, false) ^ Type-p(?s, "structuree") ^ UTA-sw(?s, false) ^
VF-sw(?s, false) → meth-pour-src(?s, WIEN)

R16. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, false) ^
MOA-sw(?s, false) ^ Nested-sw(?s, false) ^ Type-p(?s, "semi_structuree")^
UTA-sw(?s, false) ^ VF-sw(?s, false) → meth-pour-src(?s, WIEN)

R17. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ Nested-sw(?s, true) ^
Type-p(?s, "semi_structuree") ^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R18. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, true) ^
Type-p(?s, "structuree") ^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R19. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ Nested-sw(?s, true) ^
Type-p(?s, "structuree") ^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R20. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, true) ^
Type-p(?s, "structuree")^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R21. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, true) ^
Type-p(?s, "semi_structuree")^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R22. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ Type-p(?s, "semi_structuree") ^
UTA-sw(?s, false) ^ VF-sw(?s, true) → meth-pour-src(?s, STALKER)

R23. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, true) ^
Type-p(?s, "semi_structuree") ^ UTA-sw(?s, false) → meth-pour-src(?s, STALKER)

R24. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ Type-p(?s, "structuree") ^
UTA-sw(?s, false) ^ VF-sw(?s, true) → meth-pour-src(?s, STALKER)

R25. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, false) ^
Type-p(?s, "structuree")^ UTA-sw(?s, true) ^ VF-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R26. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, false) ^ Nested-sw(?s, true) ^
Type-p(?s, "structuree")^ UTA-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R27. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, true) ^
MOA-sw(?s, false) ^ Type-p(?s, "structuree")^ UTA-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R28. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, false) ^
Type-p(?s, "semi_structuree")^ UTA-sw(?s, true) ^ VF-sw(?s, true)→ meth-pour-src(?s, SoftMealy)

R29. source_web(?s) ^classe-pour-src(?s, supervisee) ^ MA/MVA-sw(?s, true) ^ MOA-sw(?s, false),
Type-p(?s, "semi_structuree") ^ UTA-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R30. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, false) ^ Nested-sw(?s, true) ^
Type-p(?s, "semi_structuree")^ UTA-sw(?s, true) → meth-pour-src(?s, SoftMealy)

R31. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, true) ^
Type-p(?s, "semi_structuree")^ UTA-sw(?s, true) → classe-pour-src(?s, manuelle)

R32. source_web(?s) ^ classe-pour-src(?s, supervisee) ^ MOA-sw(?s, true) ^
Type-p(?s, "structuree")^ UTA-sw(?s, true) → classe-pour-src(?s, manuelle)

R33. projet(?p) ^ source_web(?s) ^ classe-pour-prj(?p, supervisee) ^ classe-pour-src(?s, manuelle)
→ Remarque-pour-src(?s, niveau_supplementaire_requis_prg_motif)

R34.source_web(?s) ^ classe-pour-src(?s, semi-supervisee) ^ MOA-sw(?s, false) ^
Nested-sw(?s, false) ^ Niv-detail(?s, "multi-enreg")^
Type-p(?s, "structuree") → meth-pour-src(?s, IEPAD)

R35. source_web(?s) ^ classe-pour-src(?s, semi-supervisee) ^ MOA-sw(?s, false) ^
Nested-sw(?s, false) ^ Niv-detail(?s, "detaillee") ^
Type-p(?s, "structuree") → meth-pour-src(?s, OLERA)

R36. source_web(?s) ^ classe-pour-src(?s, semi-supervisee) ^ MOA-sw(?s, false) ^
Nested-sw(?s, true) ^ Type-p(?s, "structuree") → classe-pour-src(?s, non-supervisee)

R37. source_web(?s) ^ classe-pour-src(?s, semi-supervisee) ^ Type-p(?s, "semi_structuree")
→ classe-pour-src(?s, supervisee)

R38. source_web(?s) ^ classe-pour-src(?s, semi-supervisee) ^ MOA-sw(?s, true)
→ classe-pour-src(?s, supervisee)

R39. projet (?p) ^ source_web(?s) ^ classe-pour-prj(?p, semi-supervisee) ^
classe-pour-src(?s, manuelle)
→ Remarque-pour-src(?s, niveau_supplementaire_requis_prg_motif)

R40. Projet (?p) ^ source_web(?s) ^ classe-pour-prj(?p, semi-supervisee) ^
classe-pour-src(?s, supervisee) → Remarque-pour-src(?s, niveau_supplementaire_requis_labelling)

R41. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ MOA-sw(?s, false) ^
Niv-detail(?s, "multi-enreg")^ Niv-donnees(?s, "TAG") ^ Type-p(?s, "structuree") ^
UTA-sw(?s, false) → meth-pour-src(?s, DeLa)

R42. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ MOA-sw(?s, false) ^
Niv-donnees(?s, "TAG")^ Type-p(?s, "structuree")^ UTA-sw(?s, false) ^ VF-sw(?s, false) ^
Niv-detail(?s, "detaillee") → meth-pour-src(?s, RoadRunner)

R43. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, false) ^
Niv-detail(?s, "detaillee")^ VF-sw(?s, true) → classe-pour-src(?s, semi-supervisee)

R44. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, false) ^
Niv-donnees(?s, "WORD") → classe-pour-src(?s, semi-supervisee)

R45. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, false) ^
UTA-sw(?s, true) → classe-pour-src(?s, semi-supervisee)

R46. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, true) ^
Niv-detail(?s, "detaillee") ^ VF-sw(?s, true) → classe-pour-src(?s, supervisee)

R47. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^
MOA-sw(?s, true) → classe-pour-src(?s, supervisee)

R48. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, true) ^
Niv-donnees(?s, "WORD") → classe-pour-src(?s, supervisee)

R49. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^ Nested-sw(?s, true) ^
UTA-sw(?s, true) → classe-pour-src(?s, supervisee)

R50. source_web(?s) ^ classe-pour-src(?s, non-supervisee) ^
Type-p(?s, "semi_structuree") → classe-pour-src(?s, supervisee)

R51. projet(?p) ^ source_web(?s) ^ classe-pour-prj(?p, non-supervisee) ^
classe-pour-src(?s, supervisee)
→ Remarque-pour-src(?s, niveau_supplementaire_requis_labelling)

R52. projet(?p) ^ source_web(?s) ^ classe-pour-prj(?p, non-supervisee) ^
classe-pour-src(?s, manuelle)
→ Remarque-pour-src(?s, niveau_supplementaire_requis_prg_motif)

R53. projet(?p) ^ source_web(?s) ^ classe-pour-prj(?p, non-supervisee) ^
classe-pour-src(?s, semi-supervisee) →
Remarque-pour-src(?s, niveau_supplementaire_requis_post_labelling)

4.5 Présentation des modules

Dans cette section, nous détaillons l'architecture de notre système qui comporte huit modules comme illustré dans la *figure 4 .1*.

4.5.1 Aspiration

Dans cette étape nous procédons à un parcours séquentiel de l'ensemble des pages de la source web en question. Le but en est donc de récupérer ces pages, pour les segmenter, filtrer, et les mettre dans des clusters, et donc les exploiter par la suite pour tirer les paramètres de la source web.

Une source web est une collection de pages reliées entre eux avec des hyperliens. On appelle 'Aspirateur' le balayage de ces pages, à l'aide d'un programme qui les télécharge page par page, et qui découvre et ensuite explore les liens qu'elles contiennent. Au démarrage, on donne à l'aspirateur l'URL de la source web. Cette URL sera mise dans une queue, dont les URLs sont extraites un par un et visités. La queue est ré-remplie ensuite avec les nouvelles URLs découvertes dans les pages téléchargées et filtrées, et le processus continue jusqu'à visiter toutes les pages de la source web. Les étapes principales de ce module sont décrites dans l'algorithme suivant :

Algorithme 4.1 : Aspiration des Pages de la Source Web

Entrée **:** *Url* (url de la source web).
Sortie **:** *Liste-Page* (liste contenant toutes les pages du site).
Variables **:** *Queue* (File d'attente des pages à traiter).
U, *Page* : String.

```
Début
   Lire (Url) ;
   Queue ← Url ;
   Tant Que (Queue n'est pas vide) Faire
        U ← Tirer-URL (Queue) ;
        Page ← Télécharger(U) ;
        Ajouter (Liste-Page, U) ;
        Pour (tout lien contenu dans Page) Faire
           Si (lien n'est pas dans Liste-Page) Alors
                Ajouter (Queue, lien) ;
           Fin ;
        Fin ;
   Fin ;
Fin.
```

4.5.2 Prétraitement

Une étape préalable au traitement est absolument nécessaire, elle consiste à préparer les documents web pour effectuer le reste de procédures. Cela se fait par l'insertion de document HTML dans le module de prétraitement. Ce module analyse le document HTML, élimine les parties invisibles en suite le transforme en document bien formaté (XHTML), et corrige les irrégularités existantes. Alors le document XHTML produit est utilisé pour générer une représentation arborescente de la page Web initiale. Cette dernière prend la forme d'un arbre DOM en mémoire.

Définition 4.1 (**Document DOM**) : Le type Document DOM contient les représentations sous forme d'arbres DOM des documents XML. Dans la spécification DOM, un document est composé de parties appelées des nœuds [B. HABEGGER, 2004].

Définition 4.2 (**Nœud DOM**) : Un Nœud DOM est une partie d'un Document DOM. Il peut lui-même être décomposé en sous-parties qui sont aussi des Nœud DOM [B. HABEGGER, 2004].

Le traitement des parties invisibles se compose des étapes suivantes :

A. Eliminer toutes les balaises en dehors de la balaise : <BODY>
B. Eliminer les scripts :(<script>.</ script>), styles (<style>. </ style>), commentaires (<! - . ->).
C. Remplacer les espaces continus avec un seul espace.
D. Eliminer les balaises vides (les nœuds feuilles qui ne contiennent pas de texte).

4.5.3 Classification

Une page Web contient typiquement de nombreux blocks d'information. Outre les principaux éléments du contenu, elle contient généralement ces blocks : les panneaux de navigation, droits d'auteur, et des publicités … etc. Ces blocks sont dits bruyants. Nos objectifs sont : classer les pages en clusters, déterminer les régions de données, et éliminer les blocks bruyants. Nous proposons une technique pour réaliser ces tâches en se basant sur l'observation suivante :

> Dans une source Web donnée, les blocks bruyants partagent généralement des contenus et des styles de présentation communs, tandis que les principaux blocks présentent du contenu varié avec un style de présentation commun [L. Yi et al, 2003].

Partant de cette observation, nous proposons une méthode de classification hiérarchique. Cette méthode est très adaptée pour la structure arborescente des pages web (Arbre DOM), dans cette section on va présenter le principe de la classification hiérarchique, en suite on va l'adopter à notre cas.

➢ *Principe générale de classification* : La classification de données est une approche de Datamining qui vise à extraire des connaissances à partir d'un ensemble de données. Elle peut être supervisée ou non supervisée. La classification qui nous intéresse est la classification non supervisée. Elle représente la répartition de données en plusieurs classes non connues au départ. Chaque classe regroupe les données similaires. Deux types de classification non supervisée existent: la classification par partitionnement et la classification hiérarchique.

La classification par partitionnement (ou l'algorithme k-means) n'est pas intéressant dans notre cas parce que l'utilisateur doit fixer le nombre de clusters d'avance. Par contre dans la classification hiérarchique le nombre de clusters n'est pas fixé à l'avance. Il y a deux genres de méthodes de classification hiérarchiques : la classification ascendante hiérarchique (CAH) et la classification descendante hiérarchique (CDH).

✓ La CAH permet de construire une hiérarchie entière des objets sous forme d'un arbre dans un ordre ascendant. On commence en considérant chaque individu comme un cluster et on essaye de fusionner deux ou plusieurs clusters appropriés (selon une similarité) pour former un nouveau cluster. Le processus est itéré jusqu'à ce qu'une certaine condition d'arrêt soit vérifiée.

✓ Dans la CDH, on considère tous les individus comme un seul cluster au début, on divise successivement les clusters en clusters plus raffinés. Le processus marche jusqu'à ce que la condition d'arrêt soit vérifiée.

Notre méthode consiste à parcourir les arbres DOM en largeur, et à chaque itération on essaye de classer les pages web suivant leurs structures. Notre choix s'est porté sur la classification descendante hiérarchique (CDH) car cette méthode répond bien à nos besoins de classification. Nous allons détailler le principe de cette méthode dans le paragraphe qui suit :

➢ *Principe de la classification descendante hiérarchique (CDH)* : la méthode de classification descendante hiérarchique part d'un ensemble d'individus I et construit, de manière itérative, une partition de l'ensemble d'individus. Le schéma suivant représente graphiquement la classification hiérarchique descendante.

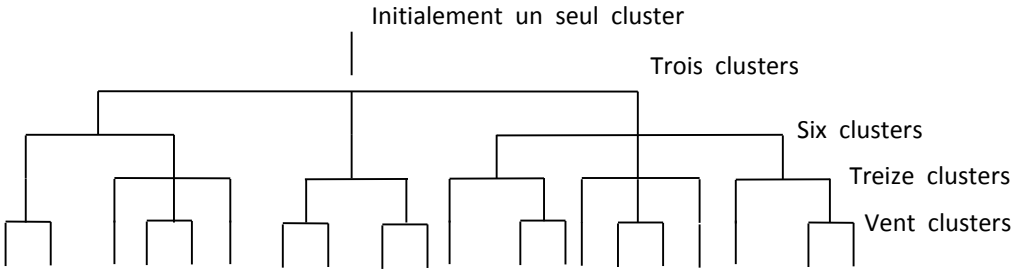


Fig 4.3. Principe de la classification descendante hiérarchique.

- *Adaptation de CDH à notre problème (classification des pages web)* : nous visons à effectuer une classification non supervisée des pages web en clusters qui ne doivent pas se chevaucher entre eux. Pour ce faire, notre démarche est de parcourir les pages web (les arbres DOM) en largeur, et pour chaque niveau on fait les deux tâches :

1) diviser les pages en clusters suivant les nœuds de niveau actuel.

2) pour chaque cluster résultat de l'étape (1), On cherche les blocks bruyants à partir du niveau actuel (le test se fait uniquement sur les nœuds du niveau actuel, on élimine les nœuds ayant un contenu commun pour toutes les pages du cluster).

Après l'étape (2) On fait une comparaison entre les clusters obtenus. Le but de cette comparaison est de détecter les caractéristiques des pages web. Pour ce faire on prend une page web (exemple) pour chaque cluster, en suite on les compare deux à deux. En se basant uniquement sur les nœuds feuilles, si deux pages ont une distance minimale (permutation ou absence des nœuds feuilles … etc.) alors on garde la remarque concernant la différence entre les clusters (permutation ou absence des nœuds feuilles … etc.). Cette remarque sera utile pour détecter les différents formats d'attributs (valeurs manquantes, les attributs multi-valeurs, attributs multi-ordonnées, attributs présentées en multi-formats, format commun d'attributs) au plus tard.

Le résultat de cette étape est un ensemble de remarques très utiles. Pour le reste de traitement on a besoin d'un seul cluster. Pour ce faire on procède à un filtrage qui nous permet de ne garder qu'un seul cluster incluant un nombre plus grand de pages. Cette stratégie de choix est basée sur l'hypothèse que la grande partie des pages web sont générées dynamiquement.

En suite, à partir du cluster retenu, on détermine la région contenant les données (RD). Cette étape est essentielle, car elle nous permet de déterminer le type de page (texte, semi-structurée ou structurée), et le niveau de détaille de la page (page multi-enregistrement ou page détaillée). Pour déterminer la RD on considère que les pages web contiennent une seule région de données. Pour commencer, on prend une page exemple, en suite, on parcoure la page en largeur. Le premier niveau qui contient plus d'un nœud est considéré comme la RD, alors leur père est pris comme la référence de la RD. Pour le reste de cette section, on va présenter l'algorithme de classification.

Le module de prétraitement est responsable de la transformation et de nettoyage des pages web, le résultat est un ensemble d'arbres DOM propres. Pour pouvoir effectuer la classification, on crée une liste (table) pour les arbres DOM. Chaque élément de la liste contient l'adresse URL de la page web, et une case pour contenir le nom de cluster auquel elle appartient la page web au cours de

classification. La liste des clusters est initialement vide. Les étapes principales de la classification sont décrites dans l'algorithme suivant :

Algorithme 4.2 : Classification des Pages de la Source Web

Entrée **:** *TPW* (Table de Pages Web).
Sortie **:** *Clusters* (liste de clusters).
Variables **:** *P* (Profondeur) ; *T* : Booléen, *Cluster-retenu* : String ;
Début
P ← 0 ; // initialisation de la profondeur
Clusters ← Ø;
Initialisation *(Clusters, TPW) ;*
Tant Que (il y a des Clusters non marqués) ***Faire***
Pour (Tout *C* non marqué ϵ *Clusters*) ***Faire***
Clustering (*C*, *P*) ;
Fin ;
Pour (Tout *C* non marqué ϵ *Clusters*) ***Faire***
Elimination-Block-B (*C*, *P*);
Fin ;
Pour (Tout *C* non marqué ϵ *Clusters*) ***Faire***
T ← Test (*C*, *P*) ;
Si (*T* = *Vrais*) ***Alors***
Marquer (*C*) ;
Fin ;
Fin ;
P ← *P* + 1 ; // incrémentation de la profondeur
Fin Tant Que;
Comparaison (*Clusters*);
Cluster-retenu ← Filtrage (*Clusters*);
Détection-RD (*Cluster-retenu*);
Fin.

Nous allons maintenant définir les primitives utilisées dans l'algorithme de classification. Comme nous l'avons déjà décrit, on a utilisé deux structures de données pour notre algorithme de classification :

1. *Clusters* : il s'agit d'une file d'attente qui contient les clusters. Chaque élément de cette file se compose de deux cases (le nom du cluster, et une case qui indique que le cluster est marqué ou non). Cette file d'attente est mise à jour à chaque itération, pour laquelle la profondeur des arbres DOM est incrémenté. La file d'attente *Clusters* est initialisée à vide.
2. *TPW* (Table de Pages Web) : après son traitement, chaque page web est représentée par un arbre DOM. l'URL de chaque page est mise comme racine de l'arbre DOM correspondant pour distinguer ces arbres. La *TPW* contient la liste des arbres DOM. Il s'agit donc d'une table où chaque ligne représente une page web (arbre DOM). Chaque ligne se compose de deux cases (une pour l'URL de la page web, et une représente le cluster auquel appartient la page web). Avant d'appeler le module de classification la *TPW* est remplis par tous les arbres. La case représentant le cluster auquel appartient la page web est laissée vide.

1. **Initialisation** *(Clusters, TPW)* : Elle consiste en deux tâches : assignation de toutes les pages (*TPW*) à un seul cluster (C1) ; initialement toutes les pages sont similaires. Le cluster crée est inséré dans la file d'attente (*Clusters*).

2. **Clustering** (*C*, *P*) : pour une valeur de *P*, on prend les nœuds à la profondeur *P*, ceci pour tous les arbres appartenant au cluster *C*. on les compare, en nombre, en symboles, et en ordre. Les arbres qui présentent les mêmes nœuds à ce niveau sont mis dans un nouveau cluster. Si plusieurs clusters sont crées pour le même niveau, alors le cluster parent est retiré de la file *Clusters*. Dans le cas contraires, le cluster parent est gardé.

3. **Elimination-Block-B** (*C*, *P*) : Pour ce faire, on prend chaque nœud à la profondeur *P*, on compare son contenu pour toutes les pages du cluster *C*, si toutes les pages ont un contenu commun pour un nœud *N*, alors ce dernier est éliminé de toutes les arbres du cluster *C*.

4. **Test** (*C*, *P*) : La fonction Test (*C*, *P*) retourne " true" si *P* est la profondeur pour tous les arbres DOM du cluster *C*, et retourne " false" dans le cas contraire. L'objectif de cette fonction est de disposer d'une condition pour arrêter le processus de classification.

5. **Marquer** (*C*) : marquer un cluster consiste à lui assigner une étiquette. Pour le reste d'itérations, ce cluster n'est plus concerné par le clustering. Les pages étiquetées par ce cluster ne sont plus concernées par le traitement.

6. **Comparaison** (*Clusters*) : dans l'algorithme 4.2, le résultat après la fin de la boucle *tant que* est un nombre considéré de clusters. Chacun de ces derniers contient les pages similaires partant de la racine jusqu'aux feuilles. Comme nous l'avons décrit, l'objectif de cette étape est tirer les caractéristiques des attributs.

   Pour ce faire, l'idée est de se concentrer uniquement sur les nœuds feuilles. Alors on prend une page exemple pour chaque cluster, en suite on compare les nœuds feuilles de ces pages. Si deux listes présentent un certain degré de similarité (différence d'ordre des nœuds, absence d'un nœud feuille … etc.) alors une remarque concernant la comparaison est prise. Cette approche a l'avantage d'offrir des heuristiques pour la détection de quelques caractéristiques concernant les clusters (valeurs manquantes, attributs multi-ordre, …etc.). Pour réaliser cette tâche de comparaison on a proposé des heuristiques, chacune d'elles traite un cas particulier.

   Les nœuds feuilles pour chaque page exemple sont présentés par un vecteur. Dans le reste de cette section on va présenter les cas possibles de comparaison :

   A. *Attributs multi-ordres (MOA)* : on compare les vecteurs deux à deux. Si deux vecteurs ont les mêmes éléments, alors une remarque ('MOA') est prise. Cette remarque sera utilisée dans le module d'extraction des paramètres.

   B. *Valeur absente (MA) / Attribut multi-valeurs (MVA)* : pour ce faire on suppose que l'absence des valeurs concerne un seul attribut. On compare les vecteurs deux à deux. si un vecteur inclus l'autre, on vérifie si la différence concerne un seul élément (maque d'une ou plusieurs occurrences d'un même élément dans le vecteur le plus petit). Si c'est le cas alors la remarque ('MA /MVA') est prise.

   C. *Format varié (VF)* : on suppose aussi que le format varié concerne un seul attribut. Alors on compare les vecteurs deux à deux, si les différences se trouvent dans le(s) même(s) position(s) et que les éléments à coté de ces positions sont identiques alors la remarque ('VF') est prise.

7. **Filtrage** (*Clusters*) : Le but de cette étape est de garder un seul cluster qui couvre une portion importante des pages web. Comme nous l'avons décrit, cette stratégie est basée sur l'hypothèse que la grande partie des pages web sont générées dynamiquement, le reste des pages ne sont pas nombreuses et ne présentent pas d'informations issues de la base de données. Pour faire le filtrage on garde le cluster ayant le plus grand nombre de pages.

8. **Détection-RD** (*Cluster-retenu*) : Pour le cluster retenu dans l'étape précédente, on essaye de déterminer la région contenant les données concernées par l'extraction. Nous rappelons que les blocks bruyants sont éliminés pendant le processus de classification. Et nous supposons que chaque page web contient une seule région de données. Alors on prend, à partir du cluster retenu, une page exemple (arbre DOM), en suite on parcoure cet arbre partant de la racine ; Le premier niveau contenant plus qu'un nœud rencontré est considéré comme début de la région de données, on fait un retour et on garde le nœud père comme région de données avec son niveau.

4.5.4 Déterminer-Type-Page

On peut remarquer l'aspect de l'information présentée par les pages web sur navigateur : des pages présentent l'information en paragraphes, d'autres la présentent en tableaux ou listes, alors d'autres partitionnent la région de données en petites zones, où chaque zone présente l'information de la même manière que les autres. D'autres pages présentent l'information en mélange de l'aspect texte et l'aspect structuré, on peut trouver des paragraphes au sein ou à coté des blocks structurés. Sauf que dans la littérature on n'a pas trouvé des travaux qui permettent de déterminer le type de la page (texte, structuré, ou semi-structuré). Pour fournir une solution à ce problème on a proposé des heuristiques pour catégoriser le type de la page, alors le processus est le suivant :

Etape 1 : On prend une page exemple pour le cluster retenu, en suite on se concentre uniquement sur la région de données.

Etape 2 : On calcule la densité de texte pour la région de données. Il s'agit d'une métrique qui nous permet de savoir le pourcentage texte/ balaise pour un nœud donné. Cette idée est inspirée du travail proposé dans [Fei Sun et al, 2011]. Dans ce dernier, les auteurs ont proposé une technique pour déterminer les blocks de données dans une page web. Ils ont basé sur l'observation que dans une page web les blocks bruyants sont bien formatés et contiennent de petites phrases. Par contre les blocks de données contiennent plus de texte et moins de balaises.

Définition 4.3 (**Densité de Texte**) [Fei Sun et al, 2011]: soit *I* une balaise (correspondant à un nœud dans l'arbre DOM) dans une page web, alors la métrique (Text Density) pour *I* (**TD***i*) est le rapport de son nombre de caractères à son nombre de balaises.

TD$i = \frac{C_i}{T_i}$, où *Ci* est le nombre de tous les caractères au dessous de *I*, *Ti* est le nombre de toutes les balaises au dessous de *I*. Si *Ti* est nul, on le remet à 1. Le **TD** est calculé pour chaque nœud, si cette métrique dépasse un seuil ***t*** alors le nœud est considéré comme block de données. Si non le nœud est considéré comme block bruyant. Pour trouver une valeur du seuil, les auteurs ont proposé la valeur **TD** du nœud **<body>** comme valeur de ***t***.

Dans notre situation, on calcule la **TD** uniquement pour le nœud qui représente la région de données, on la note par **TDRD**. La **TD** du nœud **<body>** de la même page exemple est choisi comme valeur de ***t***. sauf que pour calculer cette dernière la page exemple doit être re-téléchargée.

Etape 3 : Dans cette étape on procède aux tests pour déterminer le type de la page. Chaque test correspond à un type de page :

- ✓ *Test A :* si la **TDRD** est supérieure à ***t***, et les nœuds feuilles contiennent les balaises <p>,
 Alors le type de page est : texte.
- ✓ *Test B :* les pages structurées (Template) sont caractérisées par une **TDRD** faible, ces pages peuvent utiliser quelques balaises pour bien formater leurs contenus, et sont caractérisées aussi par l'absence des paragraphes. Pour dire que la page est bien structurée on fait trois tests :
    1. La **TDRD** doit être inférieure à ***t***.
    2. La présence des tables et/ou listes augmente la possibilité que la page soit structurée. D'après nos observations, il existe des pages qui utilisent les tables et les listes pour formater leurs contenus mais au dessous des éléments de tables et listes on trouve aussi d'autres balaises et qui peuvent êtres composées. Pour éviter cette possibilité on teste uniquement les nœuds feuilles. Si ces derniers sont des éléments de tables et/ ou listes (<td>, <tr>, <li>) alors la possibilité que la page est structurée est augmentée.
    3. Pour vérifier l'absence des paragraphes dans la région de données, on propose une fonction : *paragraphe* (*n*). où *n* est un nœud feuille. Il s'agit d'une fonction booléenne qui calcule le nombre de caractères pour chaque nœud feuille, si le nombre de caractères est supérieur à un seuil (100 caractères par exemple) alors le nœud est considéré comme paragraphe. Si un nœud est <br> on appel la fonction pour son nœud père. Cette fonction est très importante, car elle doit retourner la

valeur (Faux) pour chaque nœud feuille pour dire que la page ne contient pas de paragraphes.

Remarque : les tests (**1**) et (**3**) sont indispensables pour que la page soit bien structurée, le test (**2**) n'est pas indispensable, il confirme notre jugement.

- ✓ *Test C :* si le type de la page n'est pas (texte) et n'est pas (structurée) alors la page est considérée comme semi-structurée.

4.5.5 Déterminer-Records

Le résultat obtenu dans le module précédent constitue une étape importante pour notre processus d'extraction des paramètres concernant la source web. En effet, nous avons déterminé le type des pages de la source web (soit : texte, structurée, ou semi-structurée).

Le but derrière cette étape est de savoir si les pages contiennent un ou plusieurs enregistrements. Si la source web a un type de page (texte), alors elle n'est pas concernée par cette étape ni par les étapes qui suivent. Maintenant, nous allons définir les deux cas (page détaillée, page multi-enregistrements) avant de présenter notre solution pour déterminer ce paramètre important.

Définition 4.4 (**Page multi-enregistrements / List page**) : une page web est dite multi-enregistrements s'elle présente plusieurs objets. Ces objets sont formatés de la même manière (le même schéma est utilisé pour formater chaque objet) [Bing Liu, 2007].

Définition 4.5 (**Page détaillée/ Detail page**) : une page web est dite détaillée s'elle est destinée pour présenter un seul objet. Elle donne tous les détails de l'objet d'intérêt [Bing Liu, 2007].

Pour pouvoir déterminer automatiquement si la page est multi-enregistrements ou détaillée, nous avons proposé une fonction booléenne, **test-multi-enregistrement** ( ), qui détermine si la page est multi-enregistrements. Si cette fonction retourne (Vrai), la page est considérée multi-enregistrements. Dans le cas contraire la page est considérée détaillée. Il s'agit d'un algorithme qui détermine quelles sont les nœuds représentants les enregistrements en analysant la structure et les régularités des pages web.

Notre solution est basée sur l'observation suivante :

Pour les pages multi-enregistrements, la structure DOM offre deux points importants : les nœuds représentant les enregistrements sont formatés de la même manière (donc ils sont similaires). Ces nœuds se trouvent aussi au dessous du même nœud parant [Bing Liu, 2007].

Etapes :

Etape 1 : on prend une page exemple de cluster retenu. Et on se concentre uniquement sur la région de données.

Etape 2 : on parcoure la région de données en largeur. Pour chaque niveau on compare les nœuds ayants le même parant. Si on trouve que ces nœuds sont similaires et que chaque nœud n'est pas une feuille.

Alors **test-multi-enregistrement** ( ) ← Vrai ;
Marquer-un-nœud comme enregistrement ;

Etape 3 : si non (le niveau des feuilles est atteint sans détecter des nœuds similaires ayants le même parant) alors **test-multi-enregistrement** ( ) ← Faux ;

4.5.6 **Extraction-param** (Extraction des paramètres)

Il s'agit ici d'une étape finale avant le lancement du processus d'inférence de la méthode adéquate. Une grande partie des paramètres (caractéristiques des clusters) à été extrait dans les étapes précédentes. Dans cette étape on va rassembler les paramètres extraits et proposer des solutions pour obtenir le reste des paramètres. On va présenter les caractéristiques de la source web comme elles sont citées dans l'ontologie :

- *Type-p* : il s'agit du type de la page (texte, structurée, ou semi-structurée), ce paramètre prend le résultat obtenu par le module (**Déterminer-Type-Page**).
- *Niv-detail* : il s'agit du nombre d'enregistrements présentés dans une page web (une page détaillée ou multi-enregistrements). On affecte à ce paramètre le résultat obtenu dans le module (**Déterminer-Records**).
- Pour les caractéristiques *MA / MVA, MOA, VF* : On vérifie s'il ya des remarques retenues dans l'étape **Comparaison**(*Clusters*) du module **Classification**. Pour chaque remarque signalée on met le paramètre correspond à (Vrai). Dans le cas contraire la caractéristique est mise à (Faux).
- *CF* (Common Format) : ce paramètre prend la valeur (Vrai) si plusieurs attributs sont formatés de la même manière. Pour détecter de telle situation, il ya deux cas :

  Cas 1 : si la page est multi-enregistrements, alors on prend le nœud marqué comme enregistrement, en suite, on teste ses nœuds fils, si deux nœuds sont similaires alors *CF* aura la valeur (Vrai). Dans le cas contraire *CF* aura la valeur (Faux).

Cas 2 : si la page est détaillée, alors on prend le nœud marqué comme région de données, en suite, on teste ses nœuds fils, si deux nœuds sont similaires alors *CF* aura la valeur (Vrai). Dans le cas contraire *CF* aura la valeur (Faux).

- *Nested* : ce paramètre prend la valeur (Vrais) si le contenu des pages web est organisé de manière hiérarchique. Pour détecter de telle situation deux cas se présentent :

  Cas A (Si la page est multi-enregistrements): on prend le nœud marqué dans l'étape (Déterminer-Records), en suite on parcoure ce nœud en largeur, s'on trouve que les nœuds similaires ayants toujours le même parent alors on dit que le contenu est hiérarchique et on met *Nested* ← Vrai ;
  Si on : Nested ← Faux ;

  Cas B (Si la page est détaillée): on fait le même test, sauf que le nœud traité dans ce cas Est le nœud représentant la région de données.

- *UTA* (Attributs concaténés ou absence de délimiteurs entre deux Attributs): nous n'avons pas trouvé de méthodes pour détecter de telle situation. Donc la valeur de ce paramètre est par défaut : Faux, c à d : il n'existe pas d'attribut concaténés.

- Context (S'il ya plusieurs champs typés) : même chose comme le précédent.

- Niv-donnees (Niveau de données à extraire : Balaise ou Mot) : On dit que le niveau de données à extraire est **Balaise** si : seulement les balaises sont utilisées pour former les règles d'extraction. Dans le cas où l'extraction concerne une partie du texte dans la même balaise alors on dit que le niveau d'extraction est **Mot**.

  Pour pouvoir déterminer le niveau de données à extraire on propose l'heuristique suivant : on prend une page exemple (arbre DOM) du cluster retenu dans l'étape *Filtrage*, en suite on parcoure tous les nœuds feuilles. Pour chaque nœud on compare son contenu pour toutes les pages du cluster, si on trouve qu'une partie du contenu est commune alors on dit que le niv-ext est **Mot**. Si tous les nœuds feuilles n'ont pas une partie commune pour toutes les pages du cluster alors le niv-ext est **Balaise**.

4.6. Projet d'extraction multi-sources

Après avoir détaillé les modules de notre méthode proposée, nous présentons dans cette section le projet d'WIE multi sources. Nous commençons par décrire le schéma : (*Projet_NOM.XML* : qui représente la structure d'un projet d'WIE multi sources). En suite nous

présentons le déroulement d'un projet d'WIE multi sources : depuis la création du projet, et en passant par l'aspiration et l'inférence, jusqu'à l'extraction d'information et la création de l'entrepôt de données :

```
<?xml version="1.0"?>
<Projet_WIE>
  <nom_projet>  nom du projet   </nom_ projet>
  <Domaine>   domaine du projet </Domaine>
  <Sources_web>
    <SW>
      <Nom_SW>   source_web_1   </Nom_SW >
      <URL>      URL_1          </URL >
      <Methode>   méthode adéquate   </Methode>
    </SW>
     <SW>
      <Nom_SW>   source_web_2   </Nom_SW >
      < URL >    URL_2          </ URL >
      <Methode>   méthode adéquate   </Methode>
    </SW>
    ……
  </Sources_web>
</Projet_WIE>
```

Fig 4.4. Structure du fichier *Projet_NOM.XML*.

Les étapes de déroulement du projet sont :

Etape 1 : Création du projet, on donne ici le nom du projet, son domaine s'il existe, et la liste des sources web. Le fichier *Projet_NOM.XML* est crée pour contenir ces informations.

Etape 2 : pour chaque source on fait :

Etape 2.1 : L'aspiration.

Etape 2.2 : La classification des pages en clusters. Le résultat est l'ensemble des clusters, pour chacun : les bruyants sont éliminés, en suite un seul cluster est retenu et la région de données est détectée.

Etape 2.3 : les caractéristiques pour les pages sont extraites.

Etape 2.4 : on charge l'ontologie, et on instancie le le projet et la source web avec ses Caractéristiques.

Etape 2.5 : on déclenche le processus d'inférence. Le résultat de cette étape est la méthode d'extraction adéquate pour la source web en question. Ce résultat est gardé dans le fichier *Projet_NOM.XML* (le nœud `<Methode>` est rempli).

Etape 3 : après avoir inféré la méthode adéquate pour toutes les sources web, on lance l'extraction pour chaque source avec la méthode adéquate. Il s'agit ici d'un appel de la méthode d'extraction.

Etape 4 : le résultat de l'étape précédente est l'ensemble d'informations extraites à partir des sources web. Ces informations sont gardées dans un entrepôt XML, on l'appel : *ENT_NOM.XML*. Le schéma de ce dernier dépond des informations extraites de différentes sources web.

4.7. Conclusion

Nous avons conçu tout au long de ce chapitre, une nouvelle approche d'extraction d'information à partir du web, proposée comme solution au problème de l'extraction multi-sources, causé par l'hétérogénéité des sources web. Après avoir présenté une comparaison intéressante entre les méthodes d'extraction, nous avons approfondi les principales phases de notre méthode. Elle consiste, à partir du tableau de comparaison, à créer une ontologie pour inférer une méthode d'extraction adéquate pour chaque source web.

L'inférence est basée sur trois axes : l'expérience d'utilisateur, les caractéristiques des méthodes d'extraction, et caractéristiques des sources web. L'obtention des caractéristiques des sources web automatiquement nécessite une analyse des pages de sources web. Pour que cette analyse soit efficace, nous avons proposé une technique intelligente qui consiste à classer les pages de la même source web en clusters, et en suite, comparer les clusters obtenus pour tirer les caractéristiques de la source web. Ainsi, nous avons proposé des heuristiques pour réaliser cette tâche de comparaison. La méthode inférée est appliquée sur les pages de la source web pour extraire l'information. Cette démarche est considérée complète, car elle commence par la création du projet d'WIE multi-source jusqu'à l'obtention de l'entrepôt de données.

CHAPITRE 5

EXPERIMENTATION

SOMMAIRE

5.1. Introduction

Nous présentons dans ce dernier chapitre, l'ensemble des expérimentations que nous avons mené durant notre travail. L'objectif de ces expérimentations était double : d'abord, montrer la pertinence et la faisabilité de notre approche dans sa globalité pour l'extraction d'information à partir du web, puis de réaliser l'extraction d'information pour alimenter un entrepôt médical. A cet effet nous avons, en premier lieu, implémenté les modules de notre méthode d'extraction basée sur l'inférence, partant de la création d'un projet d'WIE jusqu' à l'inférence d'une méthode adéquate pour chaque source web. En deuxième lieu, et pour réaliser la tâche d'extraction, nous avons implémenté la méthode STALKER.

En ce qui concerne le domaine médical, nous avons choisi d'extraire -à partir du web- des informations liées aux médicaments. Pour ce faire, nous avons choisi les deux sources web : *www.pharmacielafayette.com* et *lasante.net* : les deux sources présentent des services de vente en ligne des médicaments. Le but de l'extraction est de rassembler les informations liées aux produits en vente afin de construire un entrepôt XML naïve de médicaments. Pour le reste de ce chapitre, nous commençons par présenter les outils utilisés pour les implémentations. En suite, nous présentons brièvement les fonctionnalités des deux logiciels réalisés : l'WIE basé sur l'inférence, et la méthode STALKER. Nous finalisons par détailler les résultats obtenus, et une discussion.

5.2. Outils utilisés

5.2.1. Protégé 2000 : Protégé est un éditeur d'ontologies distribué en open source par l'université en informatique médicale de Stanford. Il s'agit d'un outil pour développer des systèmes basés sur les connaissances (Ontologies). Des applications développées avec Protégé sont utilisées dans la résolution des problèmes et la prise de décision dans un domaine particulier. L'outil Protégé possède une interface utilisateur graphique (GUI) lui permettant de manipuler aisément tous les éléments d'une ontologie : classe, méta-classe, propriété, instance,…etc. Il permet de créer ou d'importer des ontologies écrites dans les différents langages d'ontologies tel que : RDF-Schéma, OWL, DAML, OIL, …etc. Nous avons utilisé la version 4.3.0 pour développer notre ontologie d'application.

5.2.2. Java : est un langage orienté objet similaire à C++ (d'un point de vue syntaxique) qui est utilisé pour produire des programmes pouvant être exécutés sur une machine virtuelle pratiquement disponible sur tous les types de machine. Il offre la capacité de créer des applications multitâches. Pour notre implémentation, nous avons utilisé la version JDK 1.7.0

5.2.3. Eclipse : est un environnement de développement intégré (IDE) dont le but est de fournir une plateforme modulaire pour le développement d'applications. Il est écrit en Java, ce qui lui permet de s'exécuter sans difficulté sur plusieurs plateformes notamment Windows, Linux ou Mac OS. Pour notre cas, nous avons utilisé la version : Luna Service Release 2 (4.4.2).

5.2.4. Jena : est un Framework écrit en Java, dont l'objectif est de fournir un environnement facilitant le développement d'applications dédiées au web sémantique. Il s'agit d'un API à importer dans un projet java. Jena permet de manipuler, dans une application java, les ontologies aux formats : RDF, RDFS et OWL, et fournit en plus un moteur d'inférences permettant des raisonnements sur les ontologies. La version que nous avons utilisée dans notre implémentation est développée par le laboratoire de Hewlett-Packard (API Jena 2.5.5).

5.2.5. Pellet : est un moteur d'inférence (raisonneur ou reasonner en anglais) basé sur OWL. Il peut être utilisé en conjonction, dans une application java, avec les bibliothèques JENA et OWL API. Pellet permet de raisonner ou d'inférer sur une ontologie. Ce qui permet de déduire de nouvelles connaissances dans une ontologie. Nous l'avons utilisé pour inférer la méthode adéquate pour une source web. La version que nous avons utilisée est 2.3.1.

5.2.6. RegExp : Les expressions régulières (dites aussi « expressions rationnelles ») sont issues des recherches en mathématiques dans le domaine des automates. Les abréviations reconnues sont « RegExp » et RegEx ». Il s'agit d'une chaine de caractères qui décrit un ensemble de chaines de caractères. Une RegExp s'apparente à une expression mathématique, car on y trouve des opérateurs, des valeurs et des variables. Les RegExp permettent de se lancer à la recherche de motifs décrits par la combinaison d'opérateurs et de valeurs. Les RegExp ont l'avantage d'offrir une analyse plus fin des textes. Ce qui nous a permis un traitement efficace des pages web. Les classes qui manipulent les RegExp sont dans le paquetage java.util.regex du JDK.

5.2.7. JDom : (acronyme de l'anglais Java Document Object Model), est une bibliothèque open source pour la manipulation des fichiers XML en Java. Elle intègre DOM et SAX, et supporte XPath et XSLT. Nous l'avons utilisée pour la manipulation des fichiers associés aux projets d'extraction d'information, comme les fichiers de configuration des projets, et les fichiers contenant les informations extraites à partir du web. La version utilisée est 2.0.5.

5.2.8. JTidy : Il s'agit d'une API pour java, qui permet de corriger et de valider le code HTML d'une page web. Nous l'avons utilisée pour corriger et transformer les pages HTML en XML. Ceci est pour les analyser et tirer les caractéristiques concernant les sources web.

5.3. Présentation des implémentations réalisées

Dans cette section nous allons présenter brièvement les deux implémentations réalisées. Ces dernières sont développées dans un projet java en deux paquetages :

- ✓ *Projet_EI* : contient les classes qui implémentent notre méthode d'extraction multi-source basée sur l'inférence.
- ✓ *STALKER* : contient les classes qui implémentent la méthode STALKER.

5.3.1. Méthode d'extraction multi-source basée sur l'inférence

Cette application permet à l'utilisateur de manipuler des projets d'extraction multi-source. De plus, l'utilisateur a la possibilité de lancer, à partir de l'interface de cette application, directement la méthode adéquate pour une source web, ceci a pour but de lancer la tâche d'extraction, en appliquant la méthode inférée pour la source sélectionnée qui est la méthode STALKER pour notre scénario. Dans ce qui suite, nous allons montrer les deux fichiers primordiaux au fonctionnement de cette application avant de présenter les étapes de manipulation des projets d'extraction multi-source. Ainsi que les fichiers XML manipulés.

A. Le fichier *ontologie_WIE.owl* : il contient l'ontologie publiée avec Protégé. Initialement tous les concepts, sauf le concept de méthode d'extraction, ne sont pas instanciés. Ce fichier est chargé et utilisé par notre programme. Ainsi, à partir de ce dernier, on instance les autres concepts avant de lancer le processus d'inférence, qui sont : le concept d'utilisateur, celui du projet, et celui de la source web.

B. Le fichier *Projets.xml* : il contient la liste de noms des projets d'WIE multi-source créés.

- ➢ Créer un projet d'WIE multi-source : pour cette tâche, l'utilisateur fournit le nom du projet, le domaine (optionnel), et l'expérience d'utilisateur (le niveau de manipulation : Novice, Post-étiquetage, étiquetage, programmation des motifs, ou programmation des ontologies).

  Une fois l'utilisateur valide ses choix, le nom du projet est ajouté au fichier *Projets.xml,* ainsi qu'un fichier XML portant le nom du projet sera créé (par exemple : *Entrepot medical.xml*).

```
<?xml version="1.0" encoding="UTF-8"?>
<projets>
  <projet>Entrepot medical</projet>
  <projet>projet extraction 2</projet>
</projets>
```

Fig 5.1. Le fichier (*Projets.xml*)

```
<?xml version="1.0" encoding="UTF-8"?>
<projet_WIE>
 <nom_projet>Entrepot medical</nom_projet>
 <domaine>médical</domaine>
 <experience_ut>labelling</experience_ut>
 <Sources_web>
 </Sources_web>
</projet_WIE>
```

Fig 5.2. Le fichier (*Entrepot medical.xml*)
Initialement (aucune source web n'est crée)

- Créer une source web : pour un projet donné, on peut créer des sources web. pour ce faire, l'utilisateur fournit le nom de la source web à créer, l'url principale de la source, et une liste de liens de les télécharger et les analyser. Ceci afin de dégager les caractéristiques concernant cette source web, qui sont (type de page, niveau de détail, niveau de données, existence des : valeurs absentes ou les attributs multi-valeurs, attributs multi-ordres, attributs composées (Nested), attributs au format varié, format commun d'attributs, attributs concaténées (Untokenized).

  Une fois l'utilisateur valide ses choix, un nouveau nœud (<source_w>) sera ajouté sous le nœud (<Sources_web>) dans le fichier du projet (la figure suivante présente un exemple : le fichier *Entrepot medical.xml* après l'ajout du site pharmacielafayette).

```
<Sources_web>
 <source_w>
  <nom_source> pharmacielafayette </nom_source>
  <URL_source> www.pharmacielafayette.com </URL_source>
  <URLs_EXP>
 <url_exp> www.pharmacielafayette.com/cosmetologie-soins-du-visage-c- 20_21_45.html</url_exp>
 <url_exp> http://www.pharmacielafayette.com/solaires-protection-solaire-c-20_36_57.html</url_exp>
  </URLs_EXP>
  <caracteristiques />
  <meth_inf />
 </source_w>
</Sources_web>
```

Fig 5.3. Le nœud <Sources_web> du fichier *Entrepot medical.xml* après création de la première source web.

- Lancer l'inférence pour une source web sélectionnée : pour cette étape, nous chargeons –dans notre programme- l'ontologie à partir du fichier : *ontologie_WIE.owl*, en suite on instancie le projet, l'utilisateur, et la source web sélectionnée. Le résultat de cette tâche est la méthode adéquate pour la source web en question. Ceci est affiché directement à l'utilisateur.
- Lancer l'extraction pour une source web sélectionnée : une fois la méthode inférée est enregistrée comme méthode adéquate pour la source. L'utilisateur peut ensuite appliquer cette méthode sur la source en question (soit pour notre scénario la méthode STALKER).

5.3.2. Implémentation de la méthode STALKER

Rappelons (dans la section 2.8.2.2) que STALKER est un système d'apprentissage supervisé des règles d'extraction qui traite une page Web mise sous forme d'arbre et extrait l'information de manière hiérarchique. L'induction d'adaptateur est fondée sur un catalogue (arbres-ECT) qui décrit la structure hiérarchique et logique des données, construit à la main pour chaque ensemble de pages Web (site Web). Les éléments feuilles de la hiérarchie représentent des valeurs d'attributs. Son avantage est d'optimiser le processus d'extraction dans des documents hiérarchiques. Les détails d'implémentation de cette méthode sont présentés dans l'annexe (*Annexe A*) de ce mémoire.

Le résultat d'extraction à partir d'une source web « s1 » est enregistré dans un fichier XML nommé : *infos_extraites_a_partir_s1.xml*. La structure du contenu de ce dernier dépond de la structure des enregistrements exprimée dans le fichier de présentation de la source web (arbres-ECT), comme sera illustré dans l'annexe.

5.4. Expérimentation

5.4.1. Sources web utilisées

www.pharmacielafayette.com : Il s'agit d'un site web d'une pharmacie de vente en ligne des médicaments. Les pages de ce site web présentent des listes de médicaments suivant un but d'utilisation : pour chaque médicament, la page présente son nom et parfois sa référence.

lasante.net : même chose que la source web précédente. Les pages de ce site web présentent des listes de médicaments, pour chacun la page présente son nom, son prix, et sa forme.

Remarque : une capture de page exemple, ainsi que l'arbre-ECT sont présentés pour chacune des deux sources web dans l'annexe A.

5.4.2. Résultats et discussion

Après appliqué la méthode STALKER pour les deux sources web, nous avons obtenu un fichier XML contenant les informations extraites à partir de chacune des sources. Les figures suivantes montrent une capture des fichiers crées :

Fig 5.4. Capture du fichier *infos_extraites_a_partir_pharmacielafayette.xml*.

Fig 5.5. Capture du fichier *infos_extraites_a_partir_la_sante.xml*.

Résultats obtenus à partir de www.pharmacielafayette.com : nous avons lancé l'extraction pour 300 pages web à partir de cette source, les résultats obtenus sont :

- ✓ Le nombre total de pages web traitées : **300**
- ✓ Le nombre total de pages web à partir desquelles l'extraction d'informations effectuée : **284**
- ✓ Le nombre total d'enregistrements corrects extraits : **7716**
- ✓ Le nombre total d'enregistrements incorrects extraits : **54**
- ✓ Le nombre total d'enregistrements corrects non extraits : **106**

Résultats obtenus à partir de lasante.net : nous avons lancé l'extraction pour 416 pages web à partir de cette source, les résultats obtenus sont :

- ✓ Le nombre total de pages web traitées : **416**
- ✓ Le nombre total de pages web à partir desquelles l'extraction d'informations effectuée : **367**
- ✓ Le nombre total d'enregistrements corrects extraits : **4602**
- ✓ Le nombre total d'enregistrements incorrects extraits : **0**
- ✓ Le nombre total d'enregistrements corrects non extraits : **21**

Pour tester la méthode développée de façon pertinente, nous avons utilisé les trois mesures de performance d'une tâche d'extraction d'information qui sont :

- **Le Rappel (R)** : Il correspond au pourcentage des informations correctes extraites, au total d'informations correctes.

$$\mathbf{R} = \mathbf{EXT}_{CORRECTES} \ / \ (\mathbf{EXT}_{CORRECTES} + \mathbf{Non\text{-}EXT}_{CORRECTES})$$

Le rappel tend vers 1 (resp. 0) s'il y a beaucoup (resp. peu) d'enregistrements corrects extraits. Cet indicateur permet d'évaluer dans quelle mesure les enregistrements présentés par une source web sont extraits.

- **La Précision (P)** : Elle correspond au pourcentage des informations correctes extraites, au total d'informations correctes.

$$\mathbf{P} = \mathbf{EXT}_{CORRECTES} \ / \ (\mathbf{EXT}_{CORRECTES} + \mathbf{EXT}_{INCORRECTES})$$

La précision tend vers 1 (resp. 0) s'il y a -parmi les enregistrements extraits- beaucoup (resp. peu) d'enregistrements corrects. Cet indicateur permet d'évaluer dans quelle mesure les enregistrements extraits via une source web sont corrects.

Les résultats obtenus sont présenté dans le tableau 5.1.

Source Web	$EXT_{CORRECTS}$	$Non\text{-}EXT_{CORRECTS}$	$EXT_{INCORRECTS}$	Rappel %	Précision %
www.pharmacielafayette.com	7716	106	54	98.64 %	99.30 %
lasante.net	4602	21	0	99.54 %	100 %

Table 5.1. Evaluation d'extraction

Enfin ces résultats sont confirmés par la F-Mesure qui combine la précision et le rappel et leur pondération comme suite :

$$F_1 = \ 2 * PR \ / \ (P + R) \ , \quad \text{Donc :}$$

F_1 pour *www.pharmacielafayette.com* est : 98.96 %

F_1 pour *lasante.net* est : 99.76 %

Le résultat obtenu montre un pourcentage très élevé de rappel et de précision, ainsi l'application de la méthode STALKER pour les deux sources web rend l'extraction d'informations efficace. Ce résultat s'explique par la forte adéquation entre ces sources web et la méthode STALKER. Ceci grâce au système d'inférence qu'a effectué le bon choix de la méthode d'extraction.

5.5. Conclusion

Dans ce chapitre nous avons présenté nos expérimentations visant à valider notre démarche pour l'extraction d'information à partir du web. Le but était, sur un premier axe, d'étudier et de discuter l'apport de notre méthode pour choisir une méthode d'extraction adéquate, et qui assure un degré élevé de rappel et de précision d'extraction. Plus précisément nous avons essayé de répondre à la question : 'à quel point l'utilisation de notre démarche peut être bénéfique pour obtenir des résultats pertinents lors d'une tâche d'extraction d'information à partir du web ?'. Nous avons développé un outil qui met en œuvre une ontologie d'application pour inférer une méthode adéquate d'extraction pour une source web donnée. Cet outil permet d'assister l'utilisateur pour le choix d'une méthode d'extraction afin de l'appliquer sur une source web.

Sur le deuxième axe relatif au processus d'extraction d'information à partir du web pour construire un entrepôt de données, nous avons effectuée l'extraction sur deux sources web pour extraire des informations relatives aux médicaments. Cet essaye nous a permis d'extraire, pour les deux sources, 12318 enregistrements à partir de 716 pages web traitées.

Conclusion Générale

Le recueil de données web est devenu courant dans de nombreux domaines, notamment dans le domaine médical pour l'étude, par exemple, d'une pathologie donnée. La principale motivation de ce travail était de réaliser ce processus d'extraction, à l'aide de techniques regroupées sous le nom de méthode d'extraction.

Dans ce mémoire, nous nous sommes intéressés en particulier à l'extraction multi-sources d'information à partir du web. Nos travaux se concentrent sur la proposition d'une démarche basée sur l'inférence d'une méthode adéquate d'extraction pour chaque source web, dans un projet d'extraction multi-sources.

Après avoir effectué une revue bibliographique sur l'extraction d'information, nous avons démontré l'importance et la difficulté de ce domaine. Cette difficulté vient du fait que les pages web sont constituées d'un mélange de données et de consignes de présentation. Cependant l'étude des méthodes d'extraction, nous a permis de dégagé les deux points suivants:

- ✓ Les méthodes d'extraction sont comparées selon le degré d'automatisation de la méthode (le degré de l'intervention d'utilisateur pour générer l'adaptateur). En se basant sur ce facteur, quatre catégories sont définies : les méthodes manuelles, supervisées, semi-supervisées, et non-supervisées.
- ✓ Chacune des méthodes d'extraction fonctionne sous un ensemble de contraintes : par ex. quelques méthodes ne traitent que les pages web multi-enregistrements, d'autres méthodes ne traitent que le niveau de balaises dans les pages web…etc. Ainsi, ces méthodes se distinguent pour le traitement les contraintes liées aux formats attributs : par ex. attributs multi-ordre, valeurs absentes…etc.

Partant de ces deux principaux constats, nous avons proposé une nouvelle approche, basée sur un système d'inférence, pour l'extraction multi-sources d'information. Le système d'inférence proposé exploite les connaissances contenues dans une ontologie d'application. Cette dernière regroupe les connaissances selon trois axes : les caractéristiques des méthodes d'extraction,

le niveau de manipulation d'utilisateur, et les caractéristiques de la source web pour laquelle on veut suggérer une méthode adéquate d'extraction. Les principales étapes de notre démarche sont :

(1) La définition d'une ontologie intégrant les méthodes d'extraction. S'effectue une fois par le développeur du système d'inférence.
(2) La création d'un projet d'extraction contenant plusieurs sources web.
(3) Lancement du processus d'inférence pour chaque source web. Le résultat de cette étape est que notre outil suggère une méthode d'extraction. Ce qui permet, à l'utilisateur, de lancer la méthode suggérée pour commencer l'extraction.

Nous avons également proposé un outil validant cette méthode supportant toutes les étapes d'extraction. Cet outil prend en entrée un ensemble de sources web référencées par l'utilisateur, et fournit une méthode adéquate pour chacune des sources web. En plus, nous avons implémenté, comme un scénario, la méthode STALKER pour réaliser la tâche d'extraction.

Nos travaux ont été évalués dans le chapitre cinq, dans le cadre d'alimenter un entrepôt médical à partir du web. Pour ce faire, nous avons choisi d'extraire des informations liées aux médicaments pour construire un entrepôt XML, ainsi l'extraction a été effectuée à partir de deux sources web. Les résultats obtenus ont montré un pourcentage très élevé de rappel et de précision d'extraction, ces résultats ont été causés par la forte adéquation entre la source web et la méthode d'extraction choisie.

Perspectives : Nous avons identifié plusieurs perspectives pour améliorer et compléter le travail présenté dans ce mémoire :

- Tout d'abord, il serait intéressant d'implémenter un ensemble de méthodes d'extraction. Ceci permet, expérimentalement, de réaliser l'extraction multi-source avec plus d'une seule méthode (STALKER). Ce qui permettrait de quantifier l'apport en termes d'efficacité de l'approche que nous avons proposée.
- Dans ce travail, nous avons proposé une approche d'extraction multi-sources d'information à partir du web. Pour pouvoir obtenir un entrepôt final à partir du web, il serait intéressant d'adapter notre solution dans une architecture d'intégration matérialisée, en mettant en évidence les différentes composantes d'un système d'intégration : la transformation des données au niveau structurel et sémantique, l'intégration des données, et le stockage des données intégrées dans le système cible.

Bibliographies

[T.R. Gruber, 1993] T.R. Gruber: «*A Translation Approach to Portable Ontology Specifications*». Knowledge Acquisition, Volume 5, Issue 2, pp 199-220 (1993).

[C. Parent et S. Spaccapietra, 1996] C. Parent, S. Spaccapietra : « *Intégration de bases de données : Panorama des problèmes et des approches*. » . Ingénierie des systèmes d'information, Volume 4, N 3 (1996).

[J. Hammer et al, 1997] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, A. Crespo: *« Extracting Semistructured Information from the Web*». In Proceedings of the Workshop on Management of Semistructured Data, (1997).

[N. Kushmerick, 1997] N. Kushmerick: *« Wrapper Induction for Information Extraction*». Thèse pour obtenir le grade de docteur de l'université de Washington (1997).

[C.N. Hsu et M.T. DUNG, 1998] C.N. Hsu, M.T. DUNGS: *« GENERATING FINITESTATE TRANSDUCERS FOR SEMI-STRUCTURED DATA EXTRACTION FROM THE WEB*». Information Systems, Volume 23, Issue 8, pp 521-538 (1998).

[G.O. Arocena et A.O. Mendelzon, 1998] G.O. Arocena, A.O. Mendelzon: *« WebOQL: Restructuring Documents, Databases and Webs*». Proceedings of the 14th International Conference on Data Engineering (ICDE), pp 24-33 (1998).

[C.H. GOH et al, 1999] C.H. GOH, S. BRESSAN, S. MADNICK, M. SIEGEL: « *Context Interchange: New Features and Formalisms for the Intelligent Integration of Information*». ACM Transactions on Information Systems (TOIS), Volume 17, Issue 3, pp 270-293 (1999).

[D.W. Embley et al, 1999] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, R.D. Smith: «*Conceptual-model-based data extraction from multiple-record Web pages* ». Data & Knowledge Engineering, Volume 31, Issue 3, pp 227-251 (1999).

[I. Muslea et al, 1999] I. Muslea, S. Minton, C. Knoblock: *« A Hierarchical Approach to Wrapper Induction*». Proceedings of the third annual conference on Autonomous Agents, pp 190-197 (1999).

[L. Eikvil, 1999] L. Eikvil: « *Information Extraction from World Wide Web - A Survey* - ». Norwegian Computing Center, P.B. 114 Blindern, N-0314 Oslo (1999).

[M. N. Garofalakis et al, 1999] M. N. Garofalakis, R. Rastogi, S. Seshadri, K. Shim: « *Data Mining and the Web: Past, Present and Future* ». WIDM '99 Proceedings of the 2nd international workshop on Web information and data management, pp 43-47 (1999).

[S. Soderland, 1999] S. Soderland : « *Learning Information Extraction Rules for Semi-Structured and Free Text* ». Machine Learning, Volume 34, Issue 1-3, pp 233-272 (1999).

[D. Beneventano et al, 2000] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, M. Vincini: «*Information integration: The MOMIS project demonstration*». Proceedings of the 26th VLDB Conference, pp 611-614 (2000).

[A. Sahuguet et F. Azavant, 2001] A. Sahuguet, F. Azavant: *« Building Intelligent Web Applications Using Lightweight Wrappers*». Data & Knowledge Engineering, Volume 36, Issue 3, pp 283–316 (2001).

[C.H. Chang et S.C. Lui, 2001] C.H. Chang, S.C. Lui: « *IEPAD: Information Extraction Based on Pattern Discovery* ». Proceedings of the 10th international conference on World Wide Web, pp 681-688 (2001).

[H. David et al, 2001] H. David, M. Heikki, S. Padhraic: « *Principles of Data Mining* ». MIT Press (2001).

[H. Snoussi et al, 2001] H. Snoussi, L. Magnin, J.Y. Nie : « *Heterogeneous Web Data Extraction using Ontology* ». Proc. Agent-Oriented Information Systems, pp 99-110 (2001).

[H. Wache et al, 2001] H. Wache, T. Vogele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hubner: « *Ontology-based integration of information - a survey of existing approaches*». Proceedings of IJCAI-01Workshop on Ontologies and Information Sharing, pp 108–117 (2001).

[M. Tsukada et al, 2001] M. Tsukada, T. Washio, H. Motoda: « *Automatic Web-Page Classification by Using Machine Learning Methods* ». Springer: Web Intelligence: Research and Development, pp 303-313 (2001).

[V. Crescenzi et al, 2001] V. Crescenzi, G. Mecca, P. Merialdo: « *RoadRunner: Towards Automatic Data Extraction from Large Web Sites* ». Proceedings of the 27th VLDB Conference, pp 109-118 (2001).

[A.H.F. Laender et al, 2002] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, J. S. Teixeira: « *A Brief Survey of Web Data Extraction Tools* ». ACM SIGMOD Record, Vol. 31, Issue 2, pp 84–93 (2002).

[A.H.F. Laender et al, 2002]A.H.F. Laender, B.R. Neto, A.S. da Silva: *« DEByE: Data Extraction By Exapmle*». Data & Knowledge Engineering, Volume 40, Issue 2, pp 121–154 (2002).

[M. Lenzerini, 2002] M. Lenzerini: «*Data integration: a theoretical perspective* ». PODS '02 Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pp 233 - 246 (2002).

[C.H. Chang et al, 2003] C.H. Chang, C.N. Hsu, S.C. Lui: «*Automatic information extraction from semi-structured Web pages by pattern discovery* ». Decision Support Systems, Volume 35, Issue 1, pp 129–147 (2003).

[J. Wang et F.H. Lochovsky, 2003] J. Wang, F.H. Lochovsky: «*Data extraction and label assignment for web databases* ». Proceedings of the 12th International Conference on World Wide Web, pp 187-196 (2003).

[L. Yi et al, 2003] L. Yi, B. Liu, X. Li: « *Eliminating Noisy Information in Web Pages for Data Mining*». KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp 296-305 (2003).

[S.W. Liddle et al, 2003] S.W. Liddle, K.A. Hewett, D.W. Embley: « *An Integrated Ontology Development Environment for Data Extraction* ». Brigham Young University (2003).

[B. HABEGGER, 2004] B. HABEGGER: «*Extraction d'informations à partir du Web*». Thèse pour obtenir le grade de docteur de l'université de nantes (2004).

[B. Habegger et M. Quafafou, 2004] B. Habegger, M. Quafafou: « *Web Services for Information Extraction from the Web*». Proceedings of the IEEE International Conference on Web Services, pp 279 – 286 (2004).

[B. Liu et K. C. C. Chang, 2004] Bing Liu, Kevin Chen-Chuan Chang : « *Editorial: Special Issue on Web Content Mining* ». ACM SIGKDD Explorations Newsletter, Volume 6 Issue 2, pp 1-4 (2004).

[C.H. Chang et S.C. Kuo, 2004] C.H. Chang, S.C. Kuo: «*OLERA: A Semisupervised Approach for Web Data Extraction with Visual Support* ». IEEE Intelligent Systems, Volume 19, Issue 06, pp 56-64 (2004).

[H. He et al, 2004] H. He, W. Meng, C. Yu, Z. Wu: «*Automatic integration of Web search interfaces with WISE-Integrator*». The International Journal on Very Large Data Bases VLDB Volume 13, Issue 3, pp 256-273 (2004).

[J.R Meurisse, 2004] J.R. Meurisse : « *Extraction de données de sites web : Méthodologie, outils et étude de cas* ». Mémoire présenté en vue de l'obtention du grade de licencié en informatique de l'université de Notre-Dame de la Paix, Namur (2004).

[J. Wang et al, 2004] J. Wang, J. Wen, F. Lochovsky, W.Y. Ma: «*Instance-based Schema Matching for Web Databases by Domain-specific Query Probing*». Proceedings of the 30th VLDB Conference, Volume 30, pp 408-419 (2004).

[M.S. Hacid et C. Reynaud, 2004] M.S. Hacid, C. Reynaud: «*L'intégration de sources de données*». www.irit.fr/journal-i3 (2004).

[R. Song et al, 2004] R. Song, H. Liu, J. Wen, W.Y. Ma: « *Learning Block Importance Models for Web Pages* ». WWW '04 Proceedings of the 13th international conference on World Wide Web, PP 203-211 (2004).

[A. Wessman et al, 2005] A. Wessman, S.W. Liddle, D.W. Embley: « *A Generalized Framework for an Ontology-Based Data-Extraction System* ». Brigham Young University (2005).

[J. Srivastava et al, 2005] J. Srivastava, P. Desikan, V. Kumar: « *Web Mining – Concepts, Applications and Research Directions* ».Springer Berlin Heidelberg : Foundations and Advances in Data Mining, pp 275-307 (2005).

[C.H. Chang et al, 2006] C.H. Chang, M. Kayed, M.R. Girgis, K. Shaalan: « *A Survey of Web Information Extraction Systems*». IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, Volume 18, Issue 10, pp 1411 – 1428 (2006).

[D.N. Xuan, 2006] D.N. Xuan : «*Intégration de bases de données hétérogènes par articulation à priori d'ontologies: Applications aux catalogues de composants industriels*». Thèse pour obtenir le grade de docteur de l'Université de Poitiers (2006).

[D.T. NGUYEN, 2006] D.T. NGUYEN: « *Extraction d'information à partir de documents Web multilingues : une approche d'analyses structurelles* ». Thèse pour obtenir le grade de docteur de l'université de CAEN/BASSE-NORMANDIE (2006).

[S.Patwardhan et E.Riloff, 2006] S. Patwardhan, E. Riloff: « *Learning Domain-Specific Information Extraction Patterns from the Web* ». IEBeyondDoc '06 Proceedings of the Workshop on Information Extraction Beyond The Document, pp 66-73 (2006).

[B. Liu, 2007] B. Liu: « *Web Data Mining* ». Springer Berlin Heidelberg, ISBN-10 3-540-37881-2 (2007).

[M. Ceci et D. Malerba, 2007] M. Ceci, D. Malerba: « *Classifying web documents in a hierarchy of categories: a comprehensive study* ». Springer: Journal of Intelligent Information Systems, Volume 28, Issue 1, pp 37-78 (2007).

[A. BOUSSIS, 2008] A. BOUSSIS: «*INTÉGRATION DE SOURCES DE DONNÉES Á BASE ONTOLOGIQUE DANS UN ENVIRONNEMENT P2P* ». Mémoire pour obtenir le diplôme de Magister de l'Institut National d'Informatique (INI), Algérie (2008).

[J.J. Lee et al, 2009] J.J. Lee, J.H. Lee, J. Ha, S. Lee: « *Novel Web Page Classification Techniques in Contextual Advertising* ». WIDM '09 Proceedings of the eleventh international workshop on Web information and data management, pp 39-47 (2009).

[S. BELLAOUAR, 2009] S. BELLAOUAR: « *Construction et utilisation d'un thésaurus pour la recherche d'information sur le web* ». THÈSE Pour l'obtention du diplôme de Magister en informatique de l'université de Ouargla (2009).

[S. KHOURI, 2009] S. KHOURI: «*Modélisation conceptuelle à base ontologique d'un entrepôt de données*». Mémoire pour obtenir le diplôme de Magister de l'Institut National d'Informatique (INI), Algérie (2009).

[S. Tan et al, 2009] S. Tan, C. Xu, Y. Jiang: « *Web Data Extraction System Based on Label Library* ». Sixth International Conference on Fuzzy Systems and Knowledge Discovery, pp 450–454 (2009).

[S. Zhang et P. Shi, 2009] S. Zhang, P. Shi: « *An Efficient Wrapper for Web Data Extraction and its Application*». 4th International Conference on Computer Science & Education, pp 1245- 1250 (2009).

[X. QI et B.D. DAVISON, 2009] M.X. QI, B.D. DAVISON: « *Web Page Classification: Features and Algorithms* ». ACM Computing Surveys (CSUR), Volume 41 Issue 2, Article No. 12 (2009).

[C. Gueydan, 2010] C. Gueydan: «*XeuTL: un outil ETL pour l'intégration de données*». Mémoire en vue d'obtenir LE DIPLÔME D'INGENIEUR C.N.A.M. en INFORMATIQUE, CENTRE D'ENSEIGNEMENT DE GRENOBLE (2010).

[L. LIU et al, 2010] L. LIU, J. SHI, X. LIU: « *Web Information Extraction Algorithm based on Ontology and DOM Tree* ». 2010 International Conference on Computational Intelligence and Software Engineering (CiSE), pp 1-4 (2010).

[R. Etemadi et N. Moghaddam, 2010] R. Etemadi, N. Moghaddam: « *An Approach in Web Content Mining for Clustering Web Pages* ». 2010 Fifth International Conference on Digital Information Management (ICDIM), pp 279-284 (2010).

[S. BOUARROUDJ, 2010] S. BOUARROUDJ: « *Raisonnement sur une ontologie enrichie par des règles SWRL pour la recherche sémantique d'images annotées*». THÈSE Pour l'obtention du diplôme de Magister en informatique de l'université de SKIKDA (2010).

[W. Liu et al, 2010] W. Liu, X. Meng, W. Meng: « *ViDE: A Vision-Based Approach for Deep Web Data Extraction* ». IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, Volume 22, Issue 3, pp 447– 460 (2010).

[W. Rui et W. Nianbin, 2010] W. Rui, W. Nianbin: «*Ontology-Based Deep Web Data Interface Schemas Integration Method*». 2nd International Conference on e-Business and Information System Security (EBISS), pp 1-4 (2010).

[F. Sun et al, 2011] F. Sun, D. Song, L. Liao : « *DOM Based Content Extraction via Text Density* ». Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pp 245-254 (2011).

[J. BOUHASSOUN et L. BOUHASSOUN, 2011] J. BOUHASSOUN, L. BOUHASSOUN: «*Interrogation de sources de données médicales à base de Services Web DaaS* ». Mémoire présenté en vue de l'obtention du grade de MASTER en informatique de l'université d'Abou Bakr Belkaid– Tlemcen, Algérie (2011).

[L. Fatima, 2011] L. Fatima: «*Une approche Hybride d'intégration de sources de données hétérogènes dans les datawarehouses*». THESE pour obtenir le diplôme de DOCTORAT en SCIENCES de l'Université Mentouri de Constantine, Algérie (2011).

[S. Alcic et S. Conrad, 2011] S. Alcic, S. Conrad: « *Page Segmentation by Web Content Clustering* ». WIMS '11 Proceedings of the International Conference on Web Intelligence, Mining and Semantics, Article No. 24 (2011).

[K. S. Kim et al, 2012] K. S. Kim, K. Y. Kim, K. H. Lee, T. K. Kim, et W. S. Cho : « *Design and Implementation of Web Crawler Based on Dynamic Web Collection Cycle* ». 2012 International Conference on Information Networking (ICOIN), pp 562 - 566 (2012).

[A. Billard et al, 2013] A. Billard, A. Rouby, T. Tournier: « *Scraping & Crawling, Collecter ET exploiter les données du Web* ». INSA, Lyon (2013).

[A. J. Shaikh et V. L. Kolhe, 2013] A. J. Shaikh, V. L. Kolhe: « *Framework for Web Content Mining Using Semantic Search and Natural Language Queries* ». 2013 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp 1 - 5 (2013).

[G.M. Upadhyay et K. Dhingra, 2013] G.M. Upadhyay, K. Dhingra: « *Web Content Mining: Its Techniques and Uses* ». International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 11, pp 610-613 (2013).

[L.J. Cui et al, 2013] L.J. Cui, H. He, et H.W. Xuan : « *Analysis and Implementation of an Ajax-enabled Web Crawler* ». International Journal of Future Generation Communication and Networking ,Vol 6, No 2, pp 139-146 (2013).

[R. Malarvizhi et K. Saraswathi, 2013] R. Malarvizhi, K. Saraswathi : « *Web Content Mining Techniques Tools & Algorithms–A Comprehensive Study* ». International Journal of Computer Trends and Technology (IJCTT), volume 4 Issue 8, pp 2940- 2945 (2013).

[S. Balan et P. Ponmuthuramalingam, 2013] S. Balan, P. Ponmuthuramalingam : « *A Study of Various Techniques of Web Content Mining Research Issues and Tools* ». International Journal of Innovative Research & Studies (IJIRS), pp 508-517(2013).

[S. Gupta et K.K. Bhatia, 2013] S. Gupta, K.K. Bhatia: « *Domain Identification and Classification of Web Pages Using Artificial Neural Network* ». Springer: Advances in Computing, Communication, and Control, pp 215–226 (2013).

[Y. Shen et al, 2013] Y. Shen, S. Shi, H. Wang, W. Wei, C. Yuan, Y. Huang: *« Parallel Approach and Platform for Large-scale Web Data Extraction*». 2013 International Conference on Advanced Cloud and Big Data (CBD), pp 192– 196 (2013).

[E. Ferrara et al, 2014] E. Ferrara, P.D. Meo, G. Fiumara, R. Baumgartner: « *Web Data Extraction, Applications and Techniques: A Survey* ». Knowledge-Based Systems, Volume 70, pp 301–323 (2014).

[J. Ananthi, 2014] J. Ananthi: « *A Survey Web Content Mining Methods and Applications for Information Extraction from Online Shopping Sites* ». (IJCSIT) International Journal of Computer Science and Information Technologies, pp 4091-4094 (2014).

ANNEXE A

IMPMLEMENTATION DE LA METHODE STALKER

Dans cette annexe on va présenter l'implémentation de la méthode STALKER (une méthode d'extraction d'informations à partir du web basée sur l'apprentissage supervisé).

A.1. La méthode STALKER en quelques lignes

- ✓ STALKER est une méthode basée sur l'apprentissage supervisé : dans cette méthode l'utilisateur fournit au système un ensemble de pages exemples avec un étiquetage des informations à extraire. En se basant sur ces pages, le but derrière l'apprentissage est d'apprendre les délimiteurs qui précède/suivre les informations étiquetées.
- ✓ STALKER représente les pages web sous forme hiérarchique : la représentation des documents HTML sous forme hiérarchique permet de couvrir les documents ayant une structure tabulaire et aussi les documents ayant une structure hiérarchique. La représentation hiérarchique des documents HTML se fait via les arbres ECT (Embaded Catalog Tree). Dans cet arbre il y a trois types de nœuds (liste, tuple, feuille).
- ✓ STALKER se base sur une analyse lexicale (abstraction) des documents HTML : cette abstraction permet de classer les mots contenus dans une page HTML en trois catégories (balaise, ponctuation, mot) ceci permet de réduire le nombre de symboles dans page HTML et d'ajouter aussi des jetons (un jeton est un symbole qui représente une classe de symboles)
- ✓ L'extraction se fait dans des niveaux séparés : soit le nœud n1 dans l'arbre ECT et soit n2 son père : l'extraction des occurrences pour le nœud n1 se fait à partir des occurrences de nœud n2.
- ✓ STALKER ignore l'ordre d'attributs dans un tuple : la structure hiérarchique des documents HTML offre cette possibilité, on cherche une occurrence d'un nœud dans l'occurrence de son nœud père sans tenir en compte la dernière occurrence extraite ou la dernière position marquée.
- ✓ STALKER ignore l'absence d'attributs dans un tuple : même explication que la précédente.

A.2. Architecture de l'implémentation

L'architecture générale de notre implémentation est présentée ci-dessous :

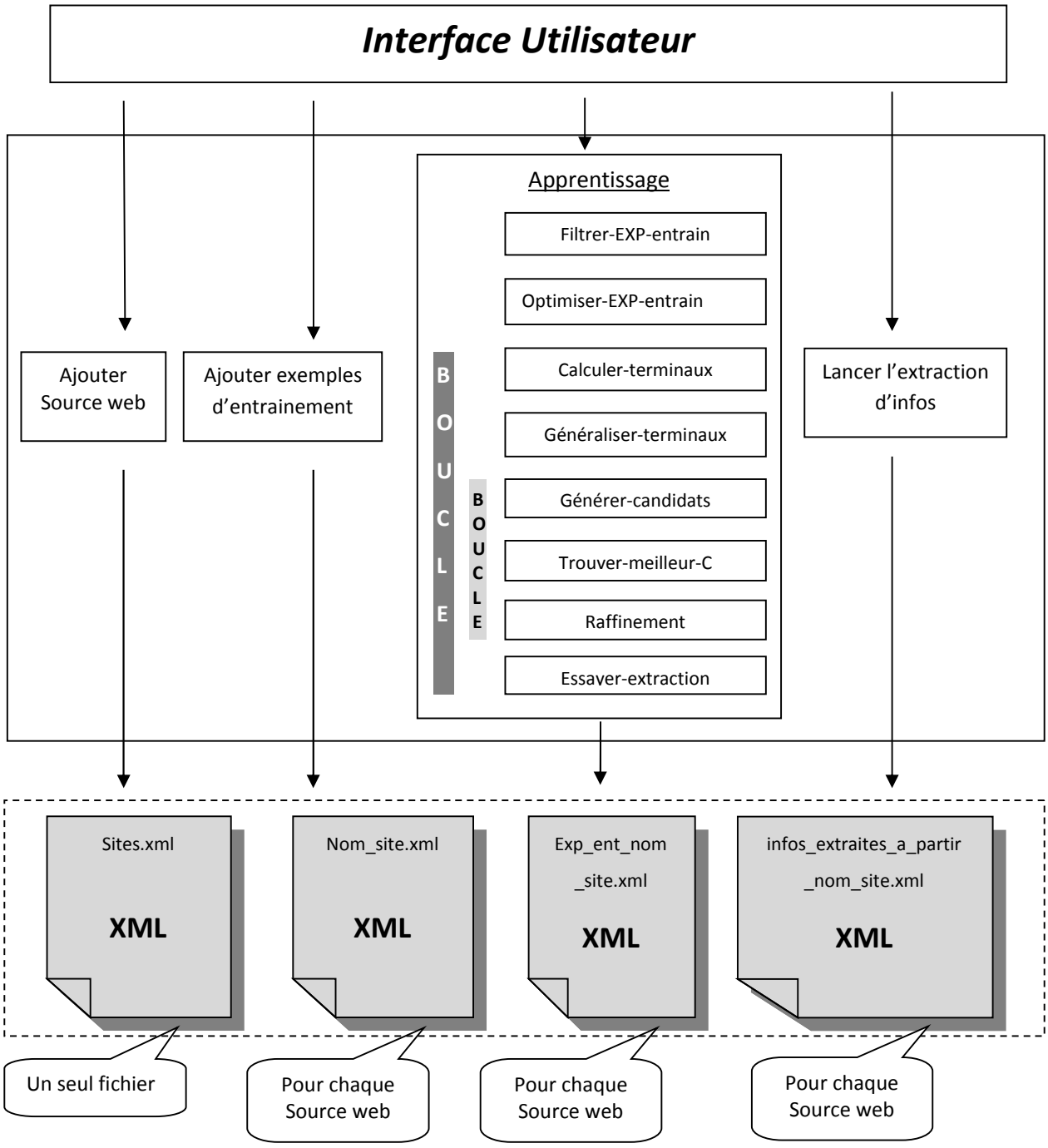


Fig A.1. Architecture générale de l'implémentation de la méthode STALKER

A.3. Présentation des modules de l'architecture

- Les documents XML manipulés :

  Le fichier : *sites.xml* : contient les noms de toutes les sources web crées.

  Le fichier *nom_site.xml* : contient le nom de site web et les nœuds de l'arbre ECT associé au site, chaque nœud de l'arbre ECT est caractérisé par : nom, type, père, niveau. En plus de ces informations, on associe pour chaque nœud –après la fin du processus des règles d'extraction- un ensemble de délimiteurs qui construisent les règles d'extraction gauches et droites. Ce fichier est créé pour chaque source.

  Le fichier *exp_ent_nom_site.xml* : est créé pour chaque source web ajoutée, ce fichier contient :

  A. Les pages web chargées : utilisées pour l'étiquetage d'exemples d'entrainement.
  B. Les exemples d'entrainement : chaque exemple (occurrence) est caractérisé par : la position de début, la position de fin, le numéro de page, la position de début de l'occurrence de son père, la position de fin de son père, un attribut (nommé couvre) indiquant que l'occurrence est couvre ou non, et le nom de nœud auquel est associé cet exemple d'entrainement.

- Module ajouter source web : permet d'ajouter de nouvelles sources web à la bibliothèque de source web. En appelant ce module, les opérations suivantes s'effectuent :

  A. Ajouter le nom de la nouvelle source au fichier *sites.xml*
  B. Créer le fichier *nom_site.xml*, en le remplissant par la structure hiérarchique de la nouvelle source (les nœuds de l'arbre ECT).
  C. Créer le fichier *exp_ent_nom_site.xml*, sans le remplir.

- Module ajouter exemples d'entrainement : permet de saisir les exemples d'entrainement pour les nœuds d'une source déjà crée. En appelant ce module, les opérations suivantes s'effectuent :

  A. Ajouter de nouvelles pages au fichier exp*_ent_nom_site.xml*. (une page web est une chaine de caractères).
  B. Ajouter de nouveaux exemples d'entrainement (occurrences) au fichier *exp_ent_nom_site.xml*.

     Remarque : les attributs : position de début, position de fin, numéro de page, nom de nœud : sont saisies à la main. Les autres attributs sont initialisés :

     La position du début de l'occurrence de père : initialisé à -1.

     La position du début de l'occurrence de père : initialisé à -1.

     L'attribut (couvre) : initialisé à 0.

❖ Module apprentissage : c'est le plus important parmi les modules, voici son architecture détaillée :

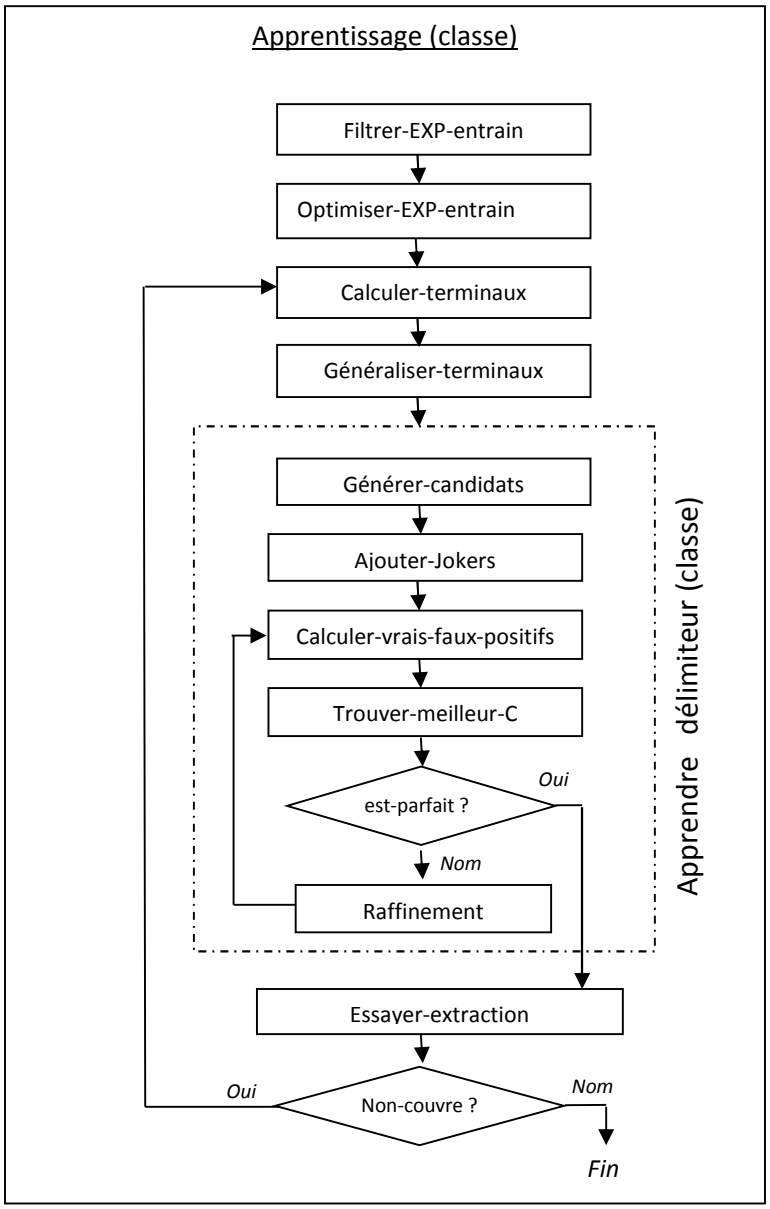


Fig A.2. Organigramme du module apprentissage (la méthode STALKER).

1. Filtrer-EXP-entrain : permet de filtrer les exemples d'entrainement d'un nœud spécifique.
2. Optimiser-EXP-entrain : pour chaque occurrence, modifier les valeurs : pd_p (position de début de l'occurrence père) et pf_p (position de fin de l'occurrence père), ceci en se basant sur l'arbre ECT et le numéro de page de chaque nœud.
3. Calculer-terminaux : pour chaque occurrence on analyse le morceau HTML se trouvant entre pd_p et pd (ou entre pf de son prédécesseur et pd) dans le code de la page qu'elle contient.

Ensuite, on procède à classer les symboles trouvés en trois catégories : balaise html, ponctuation, et mot.

4. Généraliser-terminaux : on ajoute, pour chaque catégorie trouvée dans l'étape 3, un symbole représentant.
5. Générer-candidats : on prend le symbole qui se trouve juste avant ou après l'exemple étiqueté. Le résultat de cette étape est un tableau de symboles candidats, avec :
   Candidats est une classe {Integer vp, fp, nb_jokers, long_m, nb_scp; String [] symboles ;}
6. Ajouter-Jokers : pour chaque classe de candidats (balaise, ponctuation, mot), ajouter un symbole représentant.
7. Calculer-vrais-faux-positif : cette procédure fait un essaye d'extraction pour chaque symbole candidat, en comptant le nombre d'occurrences justes extraites, et le nombre d'occurrences fausses extraites.
8. Trouver-meilleur-C : parmi les candidats générés dans l'étape 7, on sélectionne le meilleur candidat. Ce dernier doit vérifier les conditions suivantes :
   Maximum de vrais positifs.
   Minimum de faux positifs.
   Minimum de nombre de jokers.
   Et le Maximum de longueur.
9. Est –parfait ? : tester le candidat trouvé dans l'étape précédente s'il est parfait, un candidat est parfait s'il présente un nombre nul de faux positifs.
10. Raffinement : si le candidat trouvé n'est pas parfait, on lui rajoute les terminaux calculés auparavant, le but derrière cette étape est de former des candidats composés. A la fin de cette étape on reboucle à l'étape 7.
11. Essayer-extraction : si le candidat trouvé est parfait, on l'applique sur les exemples non encore couvre. Le résultat de cette étape est que les exemples extraits sont marqués comme couvre.

A la fin de cette étape on test s'il y a encore d'exemples d'entrainement non encore couvre, si cet ensemble n'est pas vide on reboucle à l'étape 3. Dans le cas contraire on sort. L'ensemble de délimiteurs retenus sont gardés comme règles d'extraction pour le nœud en question.

Remarque : l'organigramme expliqué précédemment présente le processus d'apprentissage pour la règle d'extraction gauche d'un nœud donné. Afin d'apprendre la règle d'extraction droite on suive le même organigramme, sauf que les terminaux sont calculés à partir des morceaux se trouvant après la position de fin des exemples d'entrainement.

A.4. Expérimentation

Nous avons appliqué la méthode STALKER sur les deux sources web : www.pharmacielafayette.com, et lasante.net. Dans cette section nous allons présenter l'expérimentation réalisée :

A.4.1. www.pharmacielafayette.com

Il s'agit d'un site web pour une pharmacie de vente en ligne des médicaments. La figure ci-dessous présente une capture d'une page web à partir du site:

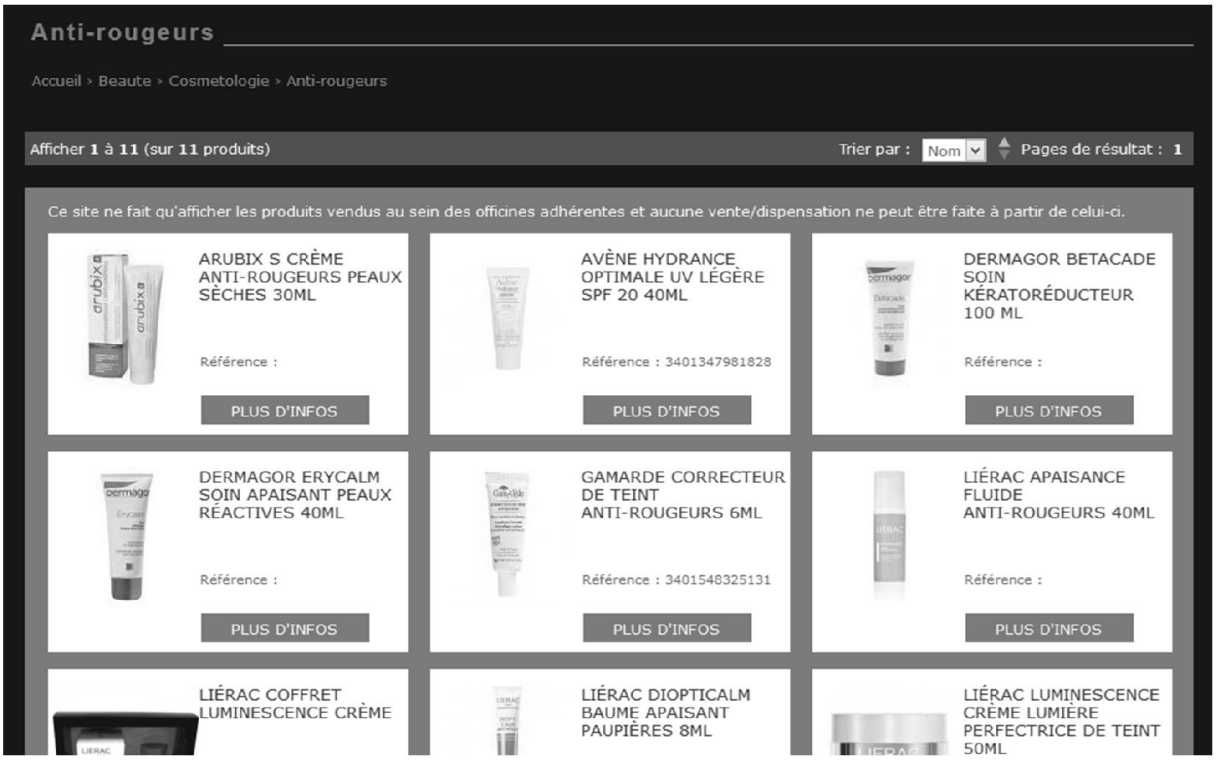


Fig A.3. Capture d'une page à partir du site web : www.pharmacielafayette.com.

L'arbre ***ECT*** qu'on a crée pour cette source est le suivant :

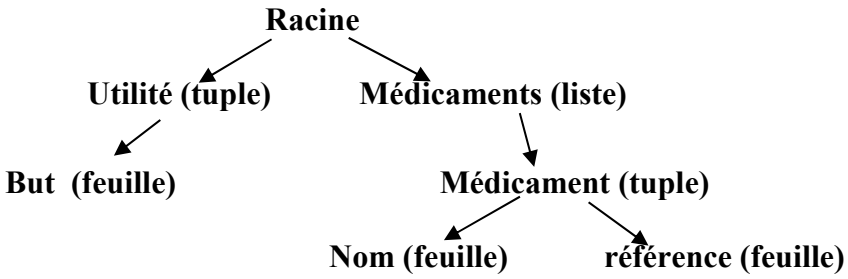


Fig A.4. L'arbre ***ECT*** pour la source www.pharmacielafayette.com.

Le résultat de la création de la source précédente-avec ECT comme montré- est la création du fichier suivant :

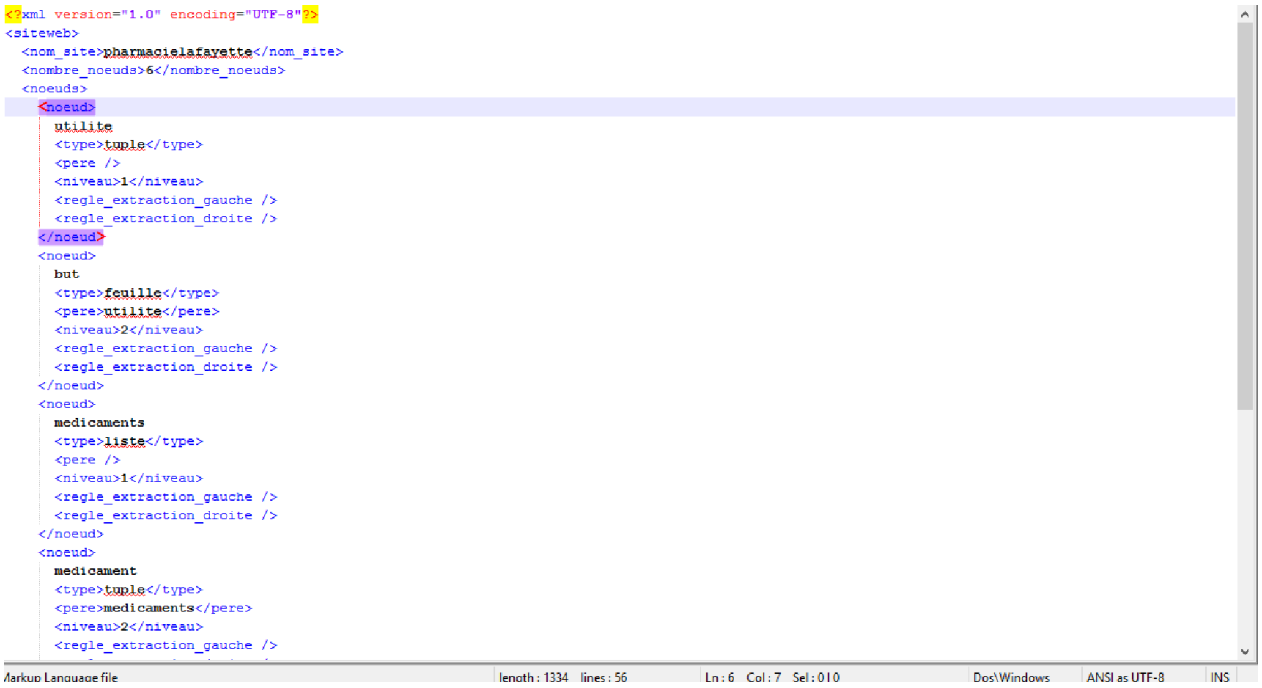


Fig A.5. Une partie du fichier *lafayette.xml*

Après finir le processus d'apprentissage, les nœuds <regle_extraction_gauche /> et <regle_extraction_droite /> sont remplis, comme le montre la figure suivante :

Fig A.6. Le nœud *utilite* après l'apprentissage (fichier *lafayette.xml*).

La figure suivante présente une capture d'écran pour le fichier qui contient les exemples d'entrainement


```xml
<?xml version="1.0" encoding="UTF-8"?>
<siteweb>
  <nom_site>pharmacielafayette</nom_site>
  <nombre_pages>4</nombre_pages>
  <nombre_exemples>29</nombre_exemples>
  <pages>
    <page>&lt;!DOCTYPE html&gt;&lt;html&gt;&lt;head&gt; &lt;meta charset="utf-8"&gt; &lt;!--[if IE]&gt;&lt;meta http-equiv="X-UA-Compatible" content=
    <page>&lt;!DOCTYPE html&gt;&lt;html&gt;&lt;head&gt; &lt;meta charset="utf-8"&gt; &lt;!--[if IE]&gt;&lt;meta http-equiv="X-UA-Compatible" content=
    <page>&lt;!DOCTYPE html&gt;&lt;html&gt;&lt;head&gt; &lt;meta charset="utf-8"&gt; &lt;!--[if IE]&gt;&lt;meta http-equiv="X-UA-Compatible" content=
    <page>&lt;!DOCTYPE html&gt;&lt;html&gt;&lt;head&gt; &lt;meta charset="utf-8"&gt; &lt;!--[if IE]&gt;&lt;meta http-equiv="X-UA-Compatible" content=
  </pages>
  <exps_ent>
    <occurence>
      <pd>52683</pd>
      <pf>52759</pf>
      <pg>0</pg>
      <pd_p>-1</pd_p>
      <pf_p>-1</pf_p>
      <couvre>0</couvre>
      <noeud>utilite</noeud>
    </occurence>
    <occurence>
      <pd>52682</pd>
      <pf>52757</pf>
      <pg>1</pg>
      <pd_p>-1</pd_p>
      <pf_p>-1</pf_p>
      <couvre>0</couvre>
      <noeud>utilite</noeud>
    </occurence>
    <occurence>
      <pd>42424</pd>
      <pf>42526</pf>
      <pg>2</pg>
```


Fig A.7. Une partie du fichier *exp_ent_pharmacielafayette.xml*.

Une fois les règles d'extraction sont apprises pour tous les nœuds de l'arbre ECT, l'utilisateur peut lancer la tâche d'extraction. Le résultat de cette dernière est la création d'un fichier XML contenant les informations extraites. La figure suivante montre une partie de ce fichier :


```xml
  <medicament>
    <nom>Listerine stay white bain de bouche 250ml</nom>
    <reference>3401397539130</reference>
  </medicament>
  <medicament>
    <nom>Listerine stay white bain de bouche 500ml</nom>
    <reference>3401397539369</reference>
  </medicament>
  <medicament>
    <nom>Listerine total care 250ml</nom>
    <reference>3401598127686</reference>
  </medicament>
  <medicament>
    <nom>Listerine total care 500ml</nom>
    <reference>3401598127518</reference>
  </medicament>
  <medicament>
    <nom>Listerine total care Ã©mail 250ml</nom>
    <reference>3401321231321</reference>
  </medicament>
  <medicament>
    <nom>Listerine total care Ã©mail 500ml</nom>
    <reference>3401321231260</reference>
  </medicament>
  <medicament>
    <nom>Listerine total care zero 500 ml</nom>
    <reference>3401320532597</reference>
  </medicament>
  <medicament>
    <nom>Listerine zero bain de bouche 250ml</nom>
    <reference>3401599833814</reference>
  </medicament>
```


```
cup Language file    length : 1056269   lines : 31700    Ln : 18295   Col : 13   Sel : 0 | 0    Dos\Windows    ANSI as UTF-8    INS
```

Fig A.8. Capture du fichier *infos_extraites_a_partir_pharmacielafayette.xml*.

A.4.2. lasante.net

Il s'agit d'un site web pour une pharmacie de vente en ligne des médicaments. La figure ci-dessous présente une capture d'une page web à partir du site:

Fig A.9. Capture d'une page à partir du site web : lasante.net

L'arbre ***ECT*** qu'on a créé pour cette source est le suivant :

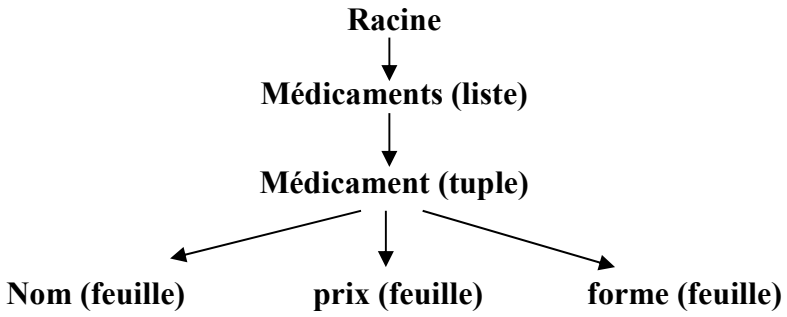


Fig A.10. L'arbre ***ECT*** pour la source lasante.net

Remarque : Les étapes d'apprentissage et d'extraction sont les même que le site précédent. Dans la section (5.4.2) Nous avons présenté les résultats d'expérimentation pour les deux sources web.