Transformation d'informations structurées en documents XML guidée par une ontologie

Mémoire de DEA I^3

Information-Interaction-Intelligence

Responsables de stage :

Hélène GAGLIARDI
Ollivier HAEMMERLÉ
Nathalie PERNELLE
Marie-Christine ROUSSET

Équipe Intelligence Artificielle et Systèmes d'Inférence (IASI)
L.R.I-Université Paris XI

Présenté par : **Fatiha SAÏS** Email : **sais@lri.fr**

03 septembre 2004

Remerciements

Je tiens à remercier mes encadrants Hélène Gagliardi, Ollivier Haemmerlé, Nathalie Pernelle et Marie-Christine Rousset pour leur accueil bienveillant, leur sympathie, et pour m'avoir donné les moyens et l'assistance nécéssaire à la réalisation du stage. Leur expérience et leurs conseils m'ont été précieux, tant pour la compréhension du domaine que pour apprendre à présenter les connaissances acquises et les travaux réalisés.

Remerciement aux équipes BIA/INRA et Gemo/INRIA pour leur collaboration constructive et pour les séminaires et les diffusions de connaissances qui y furent dispensées.

Enfin merci à toute l'équipe IASI (Brigitte, Chantal, Djalil, François, Gloria, Hassen, Laurent, Nedjem, Philippe A., Philippe C. et Véronique) pour l'atmosphère de travail acharné masquée dans une ambiance détendue et propice à la reflexion.

Table des matières

1 Introduction

1.1 Contexte du stage : Exploitation des données du Web

La popularité et le développement du Web ont contribué à l'expansion de la quantité d'informations disponibles. Ceci a rendu les méthodes traditionnelles de recherche d'informations, telles que la navigation et la recherche d'information par mots clés, de plus en plus inappropriées pour la recherche d'informations précises sur le Web et inadaptées aux traitement automatique de ce gisement d'informations.

Plusieurs organismes de recherche et de nombreux industriels souhaitent aujourd'hui exploiter cette immense quantité de données disponibles. Ils s'intéressent à ces informations, dans la volonté de compléter des bases de données déjà existantes et de rendre les réponses aux requêtes des utilisateurs les plus *pertinentes* et les plus *précises* possible.

Pour réaliser cet objectif, l'intérêt s'est porté sur les applications permettant l'interrogation de données dans des sources hétérogènes et distribuées (e.g. collections de pages Web, données semi-structurées, données relationnelles, etc.). Ces diverses sources sont exploitées dans des systèmes d'intégration de données. Pour la conception de ces derniers, il existe deux approches d'intégration de données. La première est l'approche *médiateur* où les données sont virtuelles (i.e. les données ne sont pas rapatriées) et la deuxième est l'approche *entrepôt de données* où les données sont rapatriées et *matérialisées* dans un schéma fédérateur (e.g. une base de données relationnelle, une base de données semi-structurées, etc.).

Dans le cas, où les données matérialisées dans un entrepôt traitent du même domaine on parle *d'entrepôts thématiques de données*. Pour alimenter un entrepôt thématique, les données doivent donc être non seulement compatibles avec le schéma de données stockées mais aussi pertinentes pour le domaine d'intérêt. Pour ce faire, un vocabulaire commun pour décrire les données et les requêtes des utilisateurs est utilisé. Ce vocabulaire est en général représenté par une ontologie du domaine.

Il convient de rappeler la définition d'une ontologie fondée sur une première proposition de T. Gruber [1] qui est la suivante : *"une ontologie est une spécification explicite et formelle d'une conceptualisation commune"*. Dans le cadre de cette définition, le terme *conceptualisation* renvoie à un modèle abstrait d'un phénomène fondé sur l'identification de concepts significatifs, le terme *explicite* signifie que les concepts utilisés et leur contraintes d'utilisation sont définis de façon explicite, l'adjectif *formelle* précise que les connaissances représentées dans l'ontologie peuvent être manipulables par un ordinateur et, enfin, le terme *commune* renvoie à l'idée qu'une ontologie rend compte d'un savoir consensuel (elle n'est pas l'objet d'un individu mais elle est reconnue par un groupe).

L'approche *entrepôts de données* est au centre du projet **e.dot** [2] *(Entrepôt de Données Ouvert sur la Toile)* dont l'objectif est le développement de technologies permettant de construire et de maintenir des entrepôts thématiques de qualité le plus automatiquement possible. Les données stockées traitent du domaine *du risque alimentaire*, domaine représenté par l'ontologie construite dans le cadre du projet Sym'Previus [3]. Le projet e.dot repose principalement sur l'utilisation de XML comme langage de représentation des données. C'est dans le contexte de ce projet que s'inscrit le sujet de ce stage de DEA.

1.2 Problématique et objectifs du stage

En vue d'une exploitation automatique, les informations provenant du Web sont tout d'abord collectées (*crawling*), filtrées, extraites et enfin transformées en des formats appropriés. Pour la dernière étape de cette chaîne de traitements, nous proposons, dans le cadre de ce stage, une étude sur la façon d'exploiter une ontologie existante, pour transformer des informations structurées sous forme de tableaux en documents XML enrichis par les termes d'une ontologie du domaine. Autrement dit, la plupart des tags et de valeurs possibles de ces documents XML, sont des termes construits à partir de l'ontologie. Pour représenter ce type de document d'une manière flexible, nous avons spécifié la nature et l'organisation de leurs éléments, dans une *Défintion de Type de Document (DTD)* que l'on appelle ***SML (Semantic Markup Language)***.

L'expérimentation de l'approche que nous proposons a pour cadre la transformation d'informations structurées dans des tableaux (traitant du domaine du *risque alimentaire*) en documents SML, dans le contexte du projet e.dot.

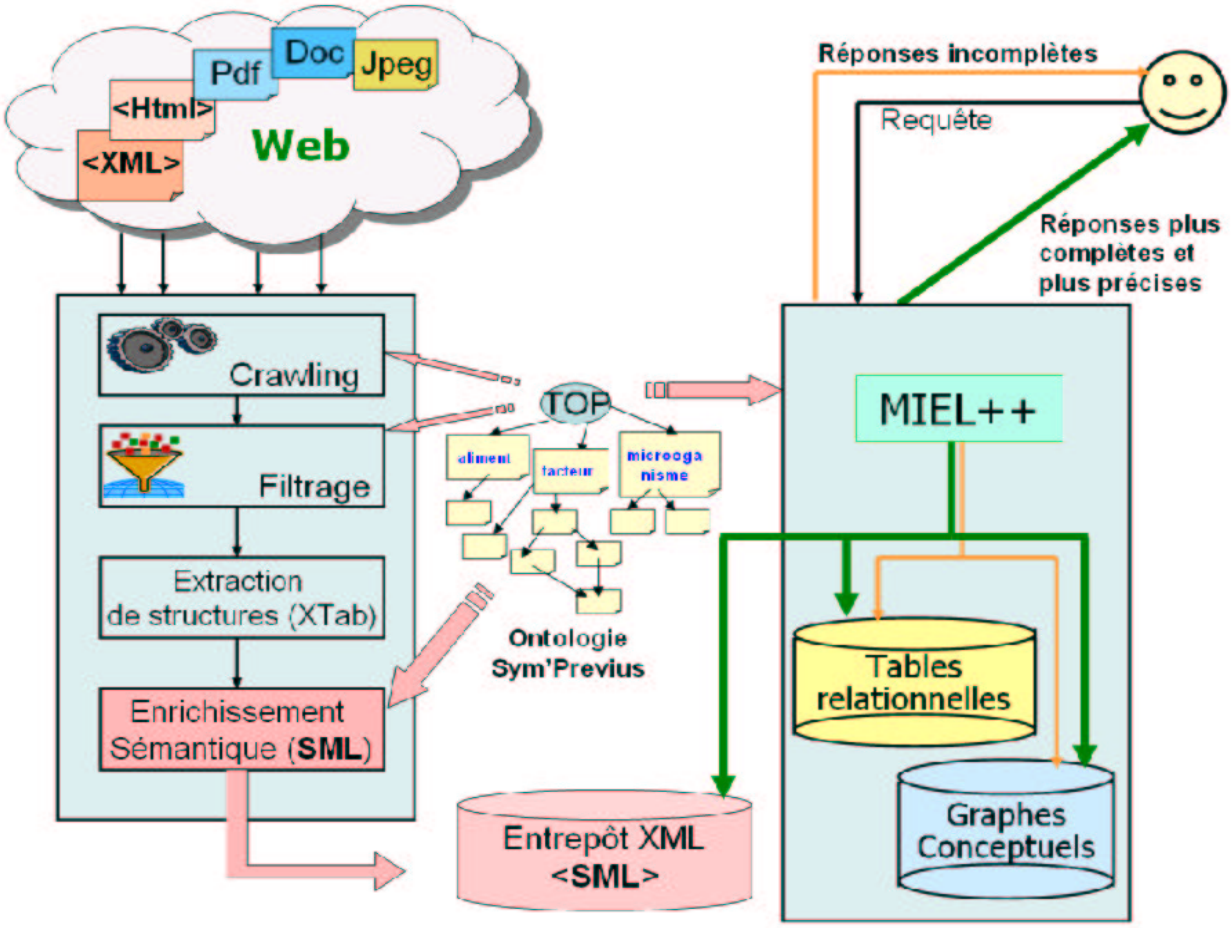


FIG. 1.1 – Une architecture fonctionnelle du projet e.dot

Dans le cadre du projet e.dot, la chaîne de traitement des informations du Web, montrée dans la figure 1.1, est la suivante : dans un premier temps, nous collectons (*crawling*) les documents potentiellement pertinents par rapport au domaine visé (*ontologie Sym'Previus*) et de natures hétérogènes (HTML, Pdf, etc.). Nous les soumettons ensuite à un *filtrage* pour recueillir les documents pertinents. Ces derniers sont soumis à la phase *d'extraction d'informations structurées* et de transformation au format XML (format **XTab** : représentation générique en XML d'un tableau sous forme d'un

ensemble de cases organisées en lignes *[voir annexe DTD XTab B.1]*) et enfin nous *enrichissons sémantiquement* ces informations[1] structurées (documents XTab) par la connaissance du domaine du risque alimentaire représentée par l'ontologie *Sym'Previus*. L'enrichissement sémantique des documents XTab et la génération de documents SML a pour but de rapprocher la sémantique des données des tableaux de celle du domaine visé. Les données sont donc plus compréhensibles et exploitables automatiquement par différentes applications (e.g. intégration de données).

Dans le cadre du projet e.dot, les documents SML sont interrogés par des requêtes pré-définies en utilisant les termes de l'ontologie du risque alimentaire. Ceci est réalisé par le moteur d'interrogation *MIEL++*. Ainsi, l'interrogation est effectuée simultanément sur l'ensemble des sources de données déjà existantes (bases de données relationnelle et de graphes conceptuels) en combinaison avec les données provenant du Web **(i.e. base de documents SML).**

1.3 Plan du rapport

Le rapport est organisé comme suit : dans le chapitre 2, nous présentons un état de l'art des travaux les plus significatifs ayant abordé l'intégration de données et l'extraction d'informations. Dans le chapitre 3, nous décrivons le cadre applicatif dans lequel se place le sujet du stage. Nous présentons dans le chapitre 4 les contributions apportées par le présent travail. Nous mettons l'accent sur le prototype implémenté dans le chapitre 5. Dans le chapitre 6 nous concluons et nous situons notre approche par rapport aux travaux antérieurs avant de donner quelques perspectives.

[1] Il faut remarquer que les informations provenant des fournisseurs de contenus (sites bibliographiques, documents Excel) peuvent être soumis aux mêmes traitements.

2 Etat de l'art

2.1 Intégration des sources de données hétérogènes

L'expansion du nombre de sources d'information disponibles sur le Web a fait de l'intégration de données un domaine de recherche de plus en plus important. Les sources d'informations sont autonomes, réparties et hétérogènes. Pour optimiser l'exploitation des informations, les systèmes d'intégration de données offrent à l'utilisateur une vue uniforme et une interrogation transparente des contenus de ces différentes sources.

Dans la littérature, il existe deux approches principales pour la mise en œuvre des systèmes d'intégration qui sont l'approche ***médiateur*** ou virtuelle et l'approche ***entrepôt de données*** ou matérialisée.

Dans l'approche ***médiateur***, l'intégration de données est fondée sur la définition d'un schéma global unifiant les schémas hétérogènes des sources à intégrer et sur la description homogène, uniforme et abstraite, du contenu des sources par des vues. Une première proposition d'architecture médiateur, en trois couches (application, médiateurs et sources), a été proposée dans [4].

Dans ce cadre, le schéma global fournit un vocabulaire unique pour l'expression des requêtes des utilisateurs et pour la description des contenus des sources. Pour ce faire, on dispose des relations du schéma global et des relations du schéma local. Il existe deux approches pour mettre en correspondance les relations du schéma global et celles du schéma local :

- **Global As View (GAV)** : les relations du schéma global sont exprimées en fonction des relations du schéma local (sources) ;
- **Local As View (LAV)** : les relations du schéma local (sources) sont exprimées en fonction des relations du schéma global. Le contenu des sources est décrit par un ensemble de vues sur les relations globales.

Pour répondre à la requête de l'utilisateur, le médiateur exploite les vues abstraites qui décrivent le contenu des sources. La tâche du médiateur consiste, en effet, à reformuler les requêtes qui lui sont posées en termes du schéma global en des plans de requêtes exprimées en termes de vues sur les sources de données. Ce processus de re-formulation de la requête est appelé processus de "**réécriture de requêtes en termes de vues**" ; c'est une tâche très complexe dans le cas de l'approche **LAV**, comme l'a montré Goasdoue F. [5]

Dans l'approche ***entrepôt de données***, les données des sources distribuées sont collectées et stockées dans une base de données réelle appelée ***"entrepôt"***. C'est pour cette raison que l'approche *entrepôt* est appelée "*matérialisée*". L'intégration y est fondée sur un schéma global fournissant une vue intégrée des sources. Une fois que le schéma de l'entrepôt est conçu, les données sont extraites à partir des sources, transformées au format de représentation des données de l'entrepôt, elles sont éventuellement filtrées pour ne garder que les données pertinentes et enfin stockées. L'interrogation s'appuie sur les techniques d'interrogation du domaine des bases de données.

Les difficultés rencontrées dans cette approche consistent en la définition du schéma global unifiant les contenus des sources et la mise à jour des données déjà stockées. Une description de l'approche *entrepôts de données* est donnée dans le travail de Chaudhuri et Dayal [6].

Dans le cadre du projet e.dot nous suivons une approche entrepôt de données.

2.2 XML et les bases de données semi-structurées

La popularité du Web et l'essor de XML ont contribué à l'émergence des bases de données semi-structurées et des bases de documents. Les modèles classiques de bases de données relationnelles

ou objets supportent difficilement les données semi-structurées qui sont complexes, hétérogènes, distribuées, parfois incomplètes.

Le langage XML, l'acronyme de *eXtended Markup Language*, est un format textuel qui permet de créer des documents contenant des données semi-structurées et permet de concevoir des définitions de types de données (DTD). Une DTD est avant tout un document comprenant l'ensemble des règles qui doivent être respectées par des documents XML, c'est pourquoi que l'on parle de documents XML "*valides*" et "*bien formés*".

Le langage XML joue aujourd'hui un rôle de plus en plus important dans l'échange d'informations sur le Web. Avec le développement des nouveaux standards dérivés de XML (e.g. XSchema, XQuery), ce dernier est devenu un langage de représentation des données très utilisable dans les systèmes d'informations.

2.3 Outils d'extraction d'informations à partir de documents

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* (EI).

D'une manière plus formelle, une tâche d'extraction d'information est définie par ses entrées (documents textes, pages Web, etc.) et par la nature des informations à extraire : des relations entre attributs ou des régularités dans les documents. Il s'agit en général d'une relation entre attributs définie en extension sous forme de k-uplets (enregistrements), où k est le nombre d'attributs. Une méthode d'extraction est appelée *"single-slot"* (resp. *"multiple-slot"*) quand k vaut 1 (resp. $k > 1$). Une page contenant un seul (resp. plusieurs) enregistrement d'intérêt, est appelée page à enregistrement unique *"Single-Record page"* (resp. page à enregistrements multiples *"Multiple-Record page"*).

Le programme permettant d'effectuer la tâche d'extraction d'informations est appelé extracteur, ou plus couramment ***wrapper***. Un *wrapper* est une procédure fondée sur le principe du *"pattern-matching"* qui, étant donné un ensemble de règles d'extraction, effectue une recherche de *patterns* (motifs) dans des documents. Une manière simple d'effectuer cette opération consiste à décrire manuellement les règles d'extraction pour un ensemble de pages en entrée. Ces méthodes nécessitent une bonne expertise en programmation (Java, Perl, etc.). De plus, elles sont inefficaces en temps et sujettes à un taux d'erreurs important (à l'exception de l'outil Lixto [7] qui permet de générer des *wrappers* d'une manière interactive, en assistant l'utilisateur à la création de programmes d'extraction très *expressifs*.)

De nouvelles approches permettent aujourd'hui d'extraire de l'information, ou plus exactement de générer des *wrappers*, en intégrant les techniques d'apprentissage (approches semi-automatiques) ou en exploitant les régularités dans les documents (approches automatiques).

Nous présentons dans la section suivante des travaux de recherche autour des problèmes de conception d'outils d'extraction d'informations (automatique et semi-automatique) à partir de documents.

2.3.1 Extraction d'information fondée sur un apprentissage supervisé

La procédure de construction de *wrappers* avec apprentissage supervisé appelée ***l'Induction de wrappers***, prend en entrée un ensemble d'exemples de pages étiquetées et fournit en sortie un ensemble de règles d'extraction. Pour induire de nouvelles règles d'extraction, la procédure compare les chaînes de cractères précédant et suivant les attributs de l'enregistrement à extraire. Elle apprend donc les délimiteurs et les motifs d'extraction pour chaque attribut. L'étiquetage des exemples d'apprentissage est fait par un humain via un éditeur de texte ou une interface graphique.

Les caractéristiques d'un "bon" système d'induction de *wrappers* sont :

1. La capacité de traiter à la fois des textes libres et des pages Web ;
2. l'expressivité (capacité de traiter de nouvelles pages Web avec des contenus hétérogènes) ;

3. l'efficacité (en terme de nombre d'exemples d'apprentissage et en terme de temps de calcul).

Le premier système d'induction de wrappers développé par l'Université de *Washington* appelé **WIEN [8]** permet de traiter à la fois les pages Web et les textes libres et définit six classes de wrappers pour exprimer les structures des sites Web. La classe la plus importante exploite les délimiteurs gauches et droits des attributs à extraire. En combinant les six classes, WIEN arrive à traiter 70% des sites Web.

Les systèmes appelés **STALKER** [9] et **SoftMealy** [10] sont plus expressifs que le système WIEN. Ils ont intégré des règles disjonctives pour pallier le problème de variation dans les pages Web. STALKER est utilisé dans plusieurs travaux dans le domaine de l'extraction d'information, comme **TheaterLoc[11]** et **Georgios et al. [12]**

Plus précisément, le système STALKER traite une page Web mise sous forme d'arbre et extrait l'information de manière hiérarchique. L'induction de *wrappers* dans STALKER est fondée sur :

1. un catalogue 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.

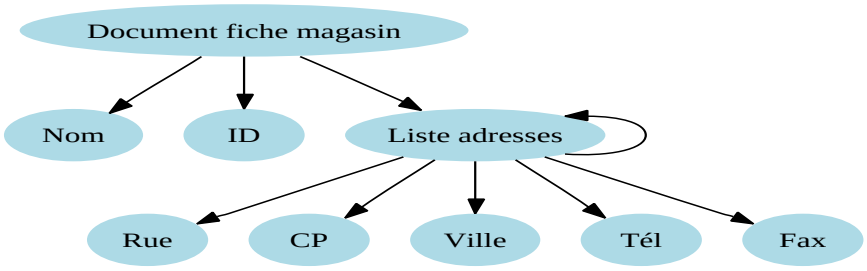


FIG. 2.1 – Catalogue d'une page contenant une fiche d'un magasin

   Une telle représentation permet à STALKER d'extraire des données à partir des pages Web avec un fort degré de structuration comme les tableaux ;

2. un ensemble de règles d'extraction contenant deux listes ordonnées d'étiquettes. La première liste correspond à la séquence de délimiteurs droits et la deuxième correspond à la séquence de délimiteurs gauches de la règle.
   Exemple de règle : CP := *sautA*( ) | *sautA*(1-) | *sautA*(#), avec *sautA* une fonction de déplacement dans le document.

Cet ensemble de règles est ensuite généralisé de manière à pourvoir couvrir le maximum de pages, ceci en utilisant le catalogue comme guide. 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.

Le système appelé **LP2 [13]** est également fondé sur l'induction de *wrappers*. Il utilise des *techniques du traitement du langage naturel* pour extraire des informations à partir de documents en texte libre. Pour ce faire, il combine des techniques d'apprentissage et de traitement du langage naturel pour l'extraction des informations.

Après avoir défini une taille de contexte, les règles permettant d'insérer automatiquement des tags SGML sont apprises sur un échantillon de corpus étiqueté manuellement. Les règles sont ensuite généralisées grâce à des informations linguistiques (catégorie lexicale, lemmatisation, etc.) ou grâce à des informations de mises en forme (casse, etc.). Cette approche a fourni de bons résultats sur un corpus de 485 mails comportant des annonces de séminaires et sur un corpus d'offres d'emplois.

Exemple : Dans le cas du corpus contenant des annonces de séminaires, voici un exemple d'enregistrement à extraire : (nom de l'orateur, heure de début, heure de fin, lieu)

2.3.2 Extraction automatique d'information

Les systèmes que nous allons décrire ont une caractéristique commune : l'absence d'intervention humaine dans le processus d'extraction. Nous présentons, sans être exhaustifs, deux catégories d'outils : ceux fondés sur la régularité dans la manière de construire les pages et ceux exploitant une ontologie comportant la connaissance du domaine d'intérêt.

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

Dans le contenu d'une page, nous distinguons deux parties une partie fixe (i.e. des informations liées à la structure et aux caractéristiques de la mise en forme) et une partie variable (i.e. les données représentées dans la page).

Le schéma régulier *"template"* d'une page correspond au à la partie fixe de la page qui est plus ou moins communne à toutes les pages prises en compte en entrée.

Nous citons deux systèmes adoptant cette approche et permettant d'extraire des informations, d'une manière complètement automatique, par déduction de template.

RoadRunner [14] : Initialement, un premier *template* est généré à partir de la première page en entrée. Il est ensuite modifié incrémentalement, en comparant les caractéristiques des nouvelles pages HTML mises sous forme arborescente. À partir du template obtenu, une grammaire capable de reconnaître les attributs à extraire est générée. Enfin, tout texte différent des attributs contenus dans le template est extrait comme donnée.

ExALG [15] : Tout comme RoadRunner [14], ExALG est fondé sur la création d'un *template* pour l'extraction de données structurées à partir de pages Web. À la différence de RoadRunner, ExALG permet d'exprimer la présence de parties optionnelles et alternatives dans les pages, ceci dans le but de couvrir le maximum de modèles de pages possibles.

Exemple de Template :
Pour un ensemble de pages contenant les coordonnées d'un groupe de personnes, le template pourrait être le suivant :
{*<Name :*
, (Email :*
) ?, (Organisation :*
) ?)>*}, ou bien ;
{*<Name :*
,(Email :
) ?,(Organisation :*
) ?)>,<Fonction :
,* },
(*) : aucune occurrence ou plusieurs occurrences et (?) instance optionnelle : zéro ou une occurrence au plus.

Dans **ExALG**, le problème d'extraction considéré est le choix du meilleur template dans un ensemble de templates candidats.

Nous tenons à citer un autre système d'extraction automatique d'informations, appelé **IEPAD[16]**, fondé sur la construction d'un template commun des pages comme dans RoadRunner et dans ExALG. Alors que les informations extraites par les systèmes RoadRunner et ExALG sont structurées, IEPAD cherche à extraire des blocs d'informations intéressantes présentées de manière contiguë dans les pages. 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.

Dans le système OLERA[17], on a amélioration le système IEPAD en intégrant une interface graphique pour choisir les motifs d'extraction. OLERA est donc semi-automatique mais capable d'extraire des blocs d'informations à partir de n'importe quelle page Web.

Les approches automatiques proposées dans RoadRunner, ExALG et IEPAD, ont montré de bons résultats dans le cas d'un ensemble de pages homogènes, qui sont en général des pages générées

dynamiquement à partir de bases de données (e.g. des sites de vente en ligne). Cependant, des faiblesses apparaissent dès qu'il s'agit de pages hétérogènes.

L'essor de XML a également contribué au développement de travaux de recherche, concernant l'extraction de structures à partir d'un corpus hétérogène de données structurées. Dans ce domaine on vise à découvrir des structures régulières dans les données. Cette régularité peut se traduire de différentes manières : par exemple, par des structures fréquentes ou par une distribution particulière des structures dans les données du corpus. Par exemple **A. Termier, M.C. Rousset et M. Segab [18]** proposent une méthode de découverte de motifs d'arbres fréquents fermés dans une base de données d'arbres hétérogènes (e.i. algorithme DRYADE) [19].

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

Certaines approches exploitent 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[20]** développé à l'université *"Brigham Young"*. Dans ce système, 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.

À partir de l'ontologie, l'outil peut concevoir et alimenter automatiquement une base de données relationnelle par des informations reconnues et extraites à partir des documents donnés en entrée. Ces documents sont supposés traiter du domaine et contenir de multiples enregistrements (*"Multiple-Record"*). L'ontologie est construite manuellement par un expert du domaine visé. Quand l'ontologie est assez représentative des caractéristiques des données, la tâche d'extraction est très efficace et flexible. Ceci est reflété par le fait que l'outil assure de bons résultats pour des pages hétérogènes traitant du domaine considéré.

Des expérimentation sur 120 pages Web contenant des annonces de décès ont donné 90% de succès.

Dans cet ensemble de pages, voici un exemple de données extraites pour alimenter la table : **DeceasedPerson** (DeceasedPerson : integer, DeceasedName : varchar(80), Age : varchar(4), DeathDate : varchar(16), BirthDate : varchar(16))

(1001,"Brian Fielding Frost",'41", "March 7, 1998","August 4, 1956")

Les systèmes présentés précédemment, ont tous montré d'assez bons résulats. Cependant, pour chaque catégorie de systèmes, nous retenons des points forts mais également des limitations pour le traitement automatique de tout type de documents. Les outils manuels (e.g. Lixto) ont besoin d'un temps important pour l'écriture du *wrapper* et son adaptation pour la prise en compte de nouvelles pages ; les outils avec apprentissage dépendent de la qualité des exemples et nécessitent l'intervention d'un humain et les outils automatiques ne sont adaptés qu'à un ensemble particulier de documents.

Ces obstacles, rencontrés par les outils d'extraction actuels, montrent la difficulté et la complexité de la tâche d'extraction d'informations à partir de documents textes et de pages Web.

Les outils qui exploitent les connaissances représentées dans une **ontologie** du domaine permettent une extraction d'informations plus automatique et plus flexible.

3 Présentation du cadre applicatif

Le cadre applicatif du stage s'inscrit dans le contexte du projet e.dot. Le projet e.dot est soutenu par le Ministère de l'industrie, il a été labellisé RNTL en mai 2002 et il réunit trois équipes de recherche (IASI/Gemo, INRIA-Futurs/Gemo et BIA/INRA) et une start-up (Xyleme). Son objectif est de développer des entrepôts de données dans des domaines spécifiques en intégrant de manière automatique des informations découvertes sur le Web, avec des données privées obtenues de fournisseurs de contenus.

Il s'agit de permettre une véritable exploitation de l'immense gisement d'informations qu'est le Web. Le projet est fondé sur une utilisation intensive de XML et de nouveaux standards développés autour de ce modèle : XQuery pour l'interrogation, XSchema pour la définition de types, SOAP/WSDL ou XML-RMI pour la définition de services Web. Une des caractéristiques innovantes du projet e.dot est la généricité de l'approche.

Pour la spécification du système, la validation des approches et des développements, le projet e.dot s'appuie sur la construction et l'exploitation d'un entrepôt de données spécialisé dans le domaine du risque alimentaire. Ce domaine particulier a été choisi car il s'agit d'un bon exemple pour montrer les limites des constructions manuelles d'entrepôts, qui sont dues au fait de :

- la dispersion de l'information sur une multitude de serveurs (nombreux laboratoires) ;
- l'évolution rapide des données déjà répertoriées ;
- la nécessité de découvrir rapidement des informations nouvelles apparaissant sur le Web.

De nombreux autres domaines ayant des caractéristiques semblables auraient pu être choisis comme domaine d'application : les dépêches d'agences de presse, les publications de bilans d'entreprises, la biologie, etc.

Le projet e.dot est décomposé en quatre lots de travail :

- Lot 1 : Spécification d'un entrepôt de données pour le risque alimentaire.
- Lot 2 : Acquisition des données pertinentes du Web.
- Lot 3 : Organisation et structuration de l'entrepôt.
- Lot 4 : Validation auprès des utilisateurs.

Dans le cadre de ce stage de DEA notre travail, concernant l'enrichissement sémantique des informations structurées, il fait donc partie du lot 3 du projet. Nous nous appuyons sur les documents XML fournis par le service d'acquisition du lot 2.

Pour séparer la phase d'alimentation de l'entrepôt e.dot de son exploitation, l'architecture est divisée en un entrepôt de travail et en un entrepôt final. Les services d'acquisition et d'enrichissement partageront l'espace de travail de l'entrepôt et l'interrogation se fera sur les données stockées dans l'entrepôt final.

3.1 La base de données Sym'Previus

Le projet Sym'Previus [3] soutenu par le Ministère de la Recherche et le Ministère de l'Agriculture a commencé à la fin de l'année 1999. Son objectif est la mise en œuvre d'un outil d'aide à l'expertise en microbiologie prévisionnelle, tenant compte des conditions de croissance des microorganismes mais aussi des paramètres liés à la nature des aliments ainsi qu'aux procédés de transformation.

Son originalité tient du fait qu'il s'appuie d'une part sur des modèles statistiques, et d'autre part sur une base de données contenant des résultats expérimentaux, provenant d'environ 700 publications scientifiques choisies comme les plus pertinentes. À l'issue de ce projet, une base de données relationnelle, une base de graphes conceptuels et un moteur permettant leur interrogation simultanée (MIEL) [21], ont été conçus. De plus, une ontologie décrivant les connaissances du domaine du risque alimentaire a été mise en œuvre. Nous présentons dans les deux sections suivantes l'ontologie et le moteur d'interrogation (MIEL).

3.1.1 L'ontologie du domaine du risque alimentaire

Cette **ontologie** a été conçue dans la volonté d'avoir un modèle de données commun représentant les propriétés sémantiques des données à exploiter. Cette ontologie est composée de six parties :

1. une taxonomie est un couple $(T,\preceq)$ où T est une terminologie, i.e. un ensemble fini et non-vide de noms, ou termes pertinents (du domaine du risque alimentaire), et $\preceq$ est une relation reflexive et transitive sur T appelée subsomption, ou spécialisation. La taxonomie contient 428 termes (e.g. aliment, microorganisme, facteur, viande, etc.)
   Exemple $\boldsymbol{terme_1} \preceq \boldsymbol{terme_2}$ **:** *salade verte* $\preceq$ *salade.*
   Les différents termes de la taxonomie décrivent le vocabulaire utilisé pour spécifier le schéma des bases relationnelle et graphes conceptuels.
2. un ensemble de termes organisés par des relations "sémantiquement proches".
3. un ensemble de traductions (Français–Anglais) pour un certain nombre de termes ;
4. un schéma relationnel R contenant les signatures des relations r entre des termes de la taxonomie. On notera *attrs(r)* l'ensemble d'attributs de la signature de r dans R. Il contient actuellement une trentaine de relations.
   Exemple : *alimentTempératureMicroorganime(aliment, température, microorganisme)*
5. un ensemble de contraintes sur les domaines de certains attributs et termes
   Exemple : pH : [1..14] pour dénoter que la valeur de pH est nécessairement comprise entre 1 et 14 ;
6. un support nécessaire à l'utilisation du modèle des graphes conceptuels [22] et [23].

L'ontologie est représentée dans le langage XML ; la DTD, un extrait sont donnés en *Annexe* A et une représentation graphique de l'ontologie dans ***Annexe* A.1**).

3.1.2 Le Moteur d'Interrogation ELargie MIEL

Le moteur d'interrogation (MIEL) est un système d'interrogation unifiée. Il exploite simultanément deux bases de données distinctes :

- une base de données relationnelle dans laquelle est stockée l'information structurée et
- une base de connaissances de graphes conceptuels dans laquelle est stockée l'information faiblement structurée.

L'interrogation se fait par le biais d'une interface graphique dans laquelle l'utilisateur exprime sa requête en spécifiant les valeurs qu'il recherche pour un certain nombre d'attributs.

Le système MIEL est en cours d'extension pour la prise en compte des données provenant du Web et stockées dans l'entrepôt de données qui sera construit dans le cadre du projet e.dot. Le système MIEL étendu est appelé **MIEL++**. Il sera capable d'interroger simultanément les bases de données relationnelles, graphes conceptuels et XML (documents SML) (voir la figure 1.1).

3.2 Acquisition et pré-traitement de documents

3.2.1 Crawling et filtrage de documents du Web

Récupération de données du Web : La chaîne d'acquisition des données du Web, dans le cadre du projet e.dot, s'est focalisée dans un premier temps sur la découverte et la réception de documents *HTML* pertinents. Par documents pertinents, on entend tout document contenant des informations centrées sur la thématique du risque alimentaire, et en particulier des résultats expérimentaux concernant l'impact de germes sur certaines catégories d'aliments. Certains types de mesures doivent être identifiés comme par exemple l'activité de l'eau (aw), la température, la valeur de pH, la teneur en acides gras, etc.

La chaîne d'acquisition se décompose en deux parties, fonctionnellement distinctes :

1. Un module de ***"crawling"*** des documents HTML du Web : ce module parcourt automatiquement le Web, de lien en lien, et rapatrie chaque nouveau document ***contenant une liste de termes de l'ontologie du domaine.***
2. Un module de configuration des abonnements comme premier élément de filtrage des documents réceptionnés du Web.

Pour prospecter le Web à la recherche de documents *PDF*, un module de *crawl* spécifique, s'appuyant sur le moteur de recherche *Google*, a été mis en œuvre. Ce module parcourt le Web à la recherche de documents potentiellement intéressants, c'est-à-dire une publication scientifique au format PDF, incluant au moins un tableau contenant des données expérimentales et éventuellement des références bibliographiques, un résumé, une introduction, un paragraphe "Matériels et méthodes" et une conclusion.

Filtrage et validation : Les résultats fournis lors de la phase d'acquisition de données du Web ne sont pas directement utilisables tels quels : une grande partie des documents ne sont pas pertinents par rapport au domaine. Deux modules, *EdotFilter* et *PDFFilter*, ont été développés pour améliorer le filtrage des documents récupérés selon deux axes :

1. *EdotFilter* est un outil de filtrage fondé sur des techniques issues du domaine de l'intelligence artificielle, permettant une caractérisation multicritères des pages et un filtrage des pages HTML contenant des tableaux. La requête de filtrage est disponible à l'adresse [1].
2. *PDFFilter* est un module axé sur la découverte et la récupération des documents *PDF* du Web.

Pour s'assurer de la pertinence des documents, ils sont simplement filtrés, en se fondant sur la présence ou l'absence de mots clés spécifiques. L'utilisation de ce système permet notamment l'élimination des documents ne traitant pas du domaine visé, en l'occurrence le risque alimentaire.

3.2.2 Extraction et représentation générique d'information structurée (XTab)

Il est important que les documents trouvés sur le Web en vue d'alimenter un entrepôt de données traitant du risque alimentaire contiennent des tables. La restriction à ce type de documents est justifée par le fait qu'une table contient en général beaucoup d'informations synthétiques, relativement faciles à exploiter automatiquement par rapport aux texte brut.

Pour réaliser le peuplement de l'entrepôt, les tableaux sont extraits et mis sous format XML.

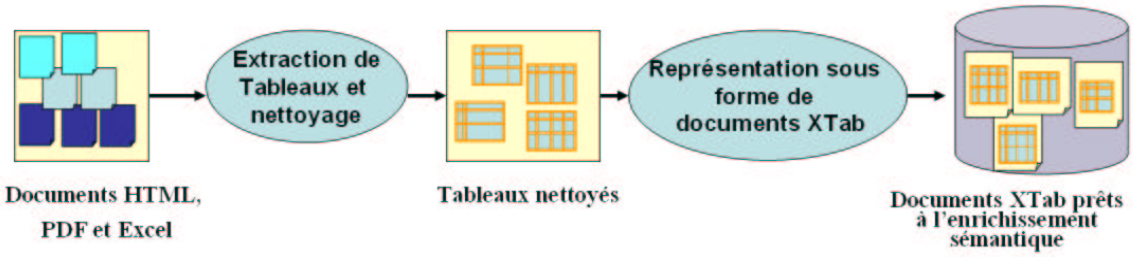


FIG. 3.1 - Le module Any2XTab

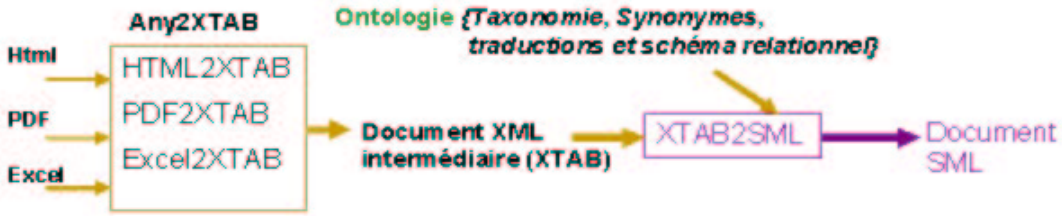
Afin de fournir une entrée unifiée au module d'enrichissement sémantique, une DTD (voir annexe DTD Xtab) a été définie. Les documents XML génériques, résultant de l'extraction de tableaux à partir de document HTML, PDF, Excel, devront se conformer à cette DTD, qui permet de représenter de manière très simple une structure de tableau. Cette structure ne comporte aucun élément de mise en forme et surtout représente les tableaux comme des matrices simples même si les tableaux d'entrée html, Excel ou pdf peuvent avoir une structure beaucoup plus complexes.

Cette phase d'extraction et de conversion consiste à remplir les différents champs des documents XML se conformant au format XTab. Les informations extraites sont essentiellement le titre du tableau, les titres des colonnes, ainsi que le nombre de colonnes et les valeurs à mettre dans les cases du tableau. Ce traitement est effectué par le module Any2XTab (voir FIG. 4).

[1] http ://www.lri.fr/ mezaour/edot/DefaultFilteringQuery/FilteringQuery2.xml.

4 Enrichissement sémantique d'informations structurées

Une fois que les tableaux sont extraits à partir de documents de formats hétérogènes (HTML, PDF et Excel) et convertis au format XTab, nous procédons à leur enrichissement par la sémantique du domaine, représentée dans l'ontologie (taxonomie, synonymes, traductions et schéma relationnel). Ce processus est réalisé par le module que l'on a nommé Any2SML, montré ci-après.



- FIG.4 Le module Any2SML -

4.1 La tâche d'enrichissement sémantique de tableaux

L'objectif de l'enrichissement sémantique est d'exploiter une ontologie du domaine pour identifier, de façon plus ou moins précise, les données apparaissant dans un tableau. Il s'agit bien sûr de traiter le contenu de chaque case (i.e. d'identifier les termes de l'ontologie auxquels la valeur de la case se réfère). De plus, sachant qu'un tableau est créé initialement pour représenter des relations entre différentes entités, nous souhaitons retrouver puis extraire une ou plusieurs relations sémantiques entre les données d'une même ligne. Ces relations sont identifiées grâce à la liste des relations décrites dans l'ontologie avec leur signature. Pour enrichir les documents XML (format XTab) donnés en entrée, il s'agit donc ici de réécrire chaque ligne du tableau en fonction des relations et des termes présents dans l'ontologie.

De plus, nous nous interdisons de rejeter des données que nous n'avons pas pu identifier avec certitude afin de ne pas perdre d'informations susceptibles d'intéresser l'utilisateur. Aussi, nous avons défini une série d'indicateurs associés à chaque information (relation ou attribut) qui seront exploités par le moteur d'interrogation MIEL++.

4.2 Description du format SML de documents

Un document SML ***(Semantic Makup Language)*** est un document XML dans lequel nous représentons un tableau XTab après son enrichissement par la sémantique du domaine considéré. Autrement dit, la plupart des valeurs et des tags possibles du document SML sont des termes construits à partir de l'ontologie (e.g. l'ontologie *Sym'Previus*).

Pour représenter un tableau dans un document SML, nous gardons la structure en ligne du tableau de la même façon que pour les documents XTab. Cependant, les lignes ne sont plus représentées par un ensemble de cases mais par un ensemble de relations identifiées comme contenues dans les lignes du tableau. Ceci est traduit par l'imbrication des tags du documents SML qui correspond à la signature de cet ensemble de relations[1].

Les documents SML se conforment à une Définition de Types de Données (DTD) SML (*Annexe* C), que nous avons définie et pouvant être générée automatiquement à partir de l'ontologie.

[1]La signature d'une relation dans le schéma relationnel est : (nom relation,(attribut$_1$, attribut$_2$, ..., attribut$_k$))

Voici un exemple de tableau à transformer en SML :

Produit	Valeur de pH
Blancs d'œufs	7.96
Champignon	5.00
Pommes	3.60

TAB. 4.1 – *pH approximatif des produits alimentaires*

Dans ce tableau, nous pouvons constater l'existence d'une relation entre la valeur de pH et celles des aliments contenus dans le tableau.

"Les pommes sont des aliments dont le pH vaut 3.60 (pH acide) et les blancs d'œufs sont des aliments dont le pH vaut 7.96 (pH basique)"

Une représentation simplifiée des informations (relations sémantique) extraites du tableau 4.1, dans un document SML :

<tableau> <titre-tableau> le titre du tableau </titre-tableau>
<titre-col> colonne 1 </titre-col> ...
<contenu>
<ligneRel> <**alimentPh**>
<aliment><finalVal>**œufs**</finalVal></aliment>
<ph> 7.96 </ph> </**alimentPh**> </ligneRel> ...
</contenu>
<origine> perso.club-internet.fr/fderad/acidobasicite.htm </origine>
<info-supplementaire>...</info-supplementaire>
</tableau>

- Représentation d'un tableau en SML-

Pour concevoir un document SML, nous utilisons le langage de représentation de données XML. Un document SML est donc composé d'un ensemble de balises XML ou – *tags* – organisées hiérarchiquement. Dans le document SML ci-dessus nous trouvons les éléments suivants :

tableau : l'élément racine du document SML qui englobe les éléments titres, contenu, origine et info-supplémentaires ;

Nous présentons essentiellement l'élément principal du tableau qui est :
contenu : le contenu du tableau est structuré sous forme d'un ensemble de lignes ***ligneRel*** qui contient un ensemble de relations sémantiques identifiées dans le tableau (e.g. ***alimentPh***). Une relation est représentée par l'ensemble de ses attributs décrits dans sa signature dans le schéma relationnel (e.g. *aliment*, *ph*). Chaque attribut représente l'ensemble de valeurs d'une colonne du tableau (i.e. ensemble de cases), pour chacune d'entre elles nous recherchons un *terme* de l'ontologie ayant une sémantique proche de la valeur. Si un tel terme est trouvé nous le représentons dans un *tag finalVal*. Par exemple pour la valeur *Blancs d'œufs* nous avons trouvé le terme *œufs* comme correspondant.

Nous ajoutons à ce *contenu* des informations concernant le tableau qui seront utilisées lors de l'exploitation du document SML. Par exemple les titres des colonnes et du tableau, l'adresse d'origine du tableau *URI* et d'autres informations supplémentaires.

4.3 Représentation de documents XTab en SML (XTab2SML)

Maintenant que nous avons définie la structure générale des documents SML, nous décrivons dans la suite les différentes tâches du processus d'enrichissement sémantique des tableaux. Ce processus s'effectue en deux phases : tout d'abord l'identification des relations représentées dans les

lignes du tableau et ensuite la recherche des correspondances (*proximités sémantiques*) entre les valeurs du tableau et les termes de l'ontologie, pour l'instantiation des relations.

Mais avant de décrire le processus d'enrichissement en détail nous donnons une definition formelle des tableaux que nous prenons en entrée (i.e XTab).

Représentation formelle des tableaux XTab

La forme tabulaire d'un ensemble de données permet d'avoir une représentation synthétique des informations et une facilité dans l'identification des relations sémantiques existant entre les données.

Définition 1 (un tableau) : Nous considérons dans un tableau son titre, les titres des colonnes et l'ensemble de ses valeurs. Un tableau est donc de la forme : *Tab =(tabTit, titles, Vals)* où :
Tab : un tableau contenu dans un document XTab,
tabTit : titre du tableau,
titles : $\{tit(c_1), tit(c_2), \ldots, tit(c_n) \mid tit(c_i)$ *est le titre de la colonne* $c_i\}$

Nous modélisons les données d'un tableau sous forme d'un ensemble de valeurs organisées en colonnes :
*Vals(*Tab*)*=$\{ c_1, \ldots, c_n\}$, avec :
c_i=$\{v_{i,1}, v_{i,2},\ldots,v_{i,k}\}$, *où* $v_{i,j}$ est une valeur de la colonne i et de la ligne j, k est le nombre de lignes et n est le nombre de colonnes du tableau *Tab*.

Pour rechercher les correspondances entre les données du tableau et celles de l'ontologie, nous effectuons un pré-traitement sur les données. Celui-ci consiste à représenter les valeurs du tableau et les termes de l'ontologie sous forme d'un ensemble de mots noté M($\mathcal{CH}$). Ce dernier est construit à partir des successions de caractères séparées par des caractères séparateurs (ex : ',', ';', ' :', '.', etc.). On supprime ensuite de cet ensemble de mots les mots vides (voir définition 2). Pour limiter les différences entre les valeurs et les termes, nous considérons la forme lemmatisée (voir définition 3) des mots ajoutés à l'ensemble M.

Définition 2 (mot vide) :

Un *mot vide* désigne tout élément d'un texte n'ayant pas de référence concrète ou notionnelle dans la réalité, et dont le rôle est d'assurer la liaison ou la cohérence entre les principales parties d'un texte. Il s'agit des articles, des conjonctions, des adjectifs non qualificatifs, des prépositions et de la ponctuation.

Définition 3 (lemmatisation) :

La lemmatisation est une procédure permettant de ramener un mot portant des marques de flexion (par exemple, la forme conjuguée d'un verbe) à sa forme de référence (dite *lemme*). Autrement dit, il s'agit, dans le cadre d'une analyse de texte par ordinateur ou d'un traitement automatique de la langue de manière générale, de reconnaître un mot par rapport à sa forme de base, et cela quelle que soit la forme sous laquelle il apparaît dans un texte.

Exemple : Lemmatisation(sèches) = sec.

4.3.1 Catégorisation

Pour réaliser la tâche d'enrichissement sémantique, différentes difficultés propres aux tableaux ont été identifiées, dont certaines ont un impact sur la structure du résultat SML. La première difficulté concerne ***le nombre de relations*** trouvées dans une ligne du tableau. Mais également dans le cas le plus simple une ligne du tableau se réécrit en une seule relation de l'ontologie. En effet

il peut arriver que l'on trouve plusieurs relations dans une ligne. La deuxième difficulté est liée à ***la structure de la valeur d'une case***. Nous avons répertorié trois types de structures en fonction du nombre et du type des valeurs :

1. Une seule valeur par case ;
2. Plusieurs valeurs de même type par case (e.g. liste de produits) ;
3. Plusieurs valeurs de types différents par case (e.g. liste de produits, liste valeurs de pH).

4.3.2 Identification des relations sémantiques dans les données

L'identification des relations représentées dans un tableau consiste en l'extraction des relations sémantiques existant entre les données. Cette extraction est effectuée par une mise en correspondance avec les relations décrites dans le schéma relationnel de l'ontologie. Pour mettre en œuvre cette tâche d'extraction de relations, nous calculons a priori un ***schéma TabSch*** qui permet de rapprocher la sémantique des données du tableau de celle du domaine. Nous décrivons ci-dessous les différents éléments de ce schéma.

Calcul du schéma d'un tableau :

Pour calculer le schéma d'un tableau, nous attribuons à chaque colonne un *terme-attribut* qui correspond à la catégorie de ses valeurs. Si aucun *terme-attribut* ne correspond à un ensemble de valeurs d'une colonne, nous recherchons un éventuel *terme-attribut* qui soit contenu dans le titre de la colonne. Dans le cas contraire, la catégorie de cette colonne est considérée comme non identifiée et donc un *attribut générique* (défini plus précisément plus bas) lui est assigné

terme-attribut : Terme de la taxonomie apparaissant au moins une fois comme attribut dans la signature d'une relation du schéma relationnel de l'ontologie. Nous notons *attrsSchRel* l'ensemble des termes-attributs. **Exemple de signature d'une relation :**

<relation>produitMicroorganismeConcentration
<attribut> Aliment </attribut>
<attribut> Microorganisme </attribut>
<attribut>Concentration </attribut>
</relation>

FIG. 4.1 – *Représentation d'une relation en XML*

Les *termes-attributs* de la relation "produitMicroorganismeConcentration" sont { Aliment, Microorganisme, Concentration }

Définition 4 (Catégorie d'une colonne pour un support donné) :
Un *terme-attribut* t est dit correspondant à la ***catégorie***[2] d'une colonne C, si et seulement si les deux conditions suivantes sont satisfaites :

- si il existe un sous-ensemble $\mathcal{P}$ des valeurs de la colonne C, correspondant à un sous ensemble de termes généralisés par t dans l'ontologie par rapport à la relation $\preceq$; et
- si un support *supp* (pourcentage) est satisfait par le sous-ensemble $\mathcal{P}$. Un support *supp* est satisfait par $\mathcal{P}$, si le rapport entre le le nombre de valeurs de l'ensemble $\mathcal{P}$ et le nombre de valeurs de la colonne C est supérieur ou égal à *supp*.

[2]Il peut y avoir plusieurs termes possibles pour une colonne, dans le cas d'une ontologie sous forme de treillis. Mais dans la version actuelle de l'implémentation nous nous arrêtons au premier plus-spécifique *terme-attribut* trouvé.

Nous décrivons ci-dessous la prédicat, nommé *cat* et qui vérifie qu'un terme t est la catégorie d'une colonne C pour un support *supp*.

$$(cat(t, C, supp) = vrai) \Leftrightarrow (\exists\ P \subseteq C\ tq \forall\ v\ \in\ P\ v^3 \preceq t\ et\ tel\ que\ \frac{|\mathcal{P}|}{|C|} \geq supp\ et\ t \in attrsSchRel)$$

Si pour une colonne aucune catégorie n'a pu être identifiée, alors nous recherchons un ventuel *terme-attribut* qui soit inclus dans son titre.

Exemple 1 :

Produit	**Item**	**lipide**	**calorie**
poulet	250 g	20 g	270 kcal
merlan	100 g	25 g	200 kcal

TAB. 4.2 – *Compositions nutritionnelles des aliments*

Si, pour une colonne, nous n'avons identifé aucune catégorie, alors une catégorie "vide" lui est assignée, notée ***"attribut"*** et que l'on appelle par *attribut générique*. Par exemple pour la deuxième colonne du tableau (TAB. 4.2) on ne peut trouver aucun *terme-attribut* qui puisse lui correspondre dans l'ontologie du risque alimentaire. Les valeurs de cette colonne seront donc instanciées dans toutes les relations identifées, et y seront représentées par l'attribut générique "*attribut*".

Définition 5 (le schéma d'un tableau) Le schéma d'un tableau consiste en un ensemble fini de couples *(colonne, catégorie)* où chaque couple représente pour une colonne la catégorie (un *terme-attribut* dans *attrsSchRel*) lui correspondant, si celle-ci a pu être identifiée.

$$TabSch = \{(c, t) \mid cat(t, c, supp) = vrai\}$$

Extraction des relations contenues dans le schéma du tableau (TabSch) :

Une relation r de R du schéma relationnel est dite représentée dans le tableau si l'ensemble de ses attributs sont complètement ou partiellement inclus dans l'ensemble des *termes-attributs* du schéma du tableau, calculé auparavant. Nous avons deux cas de figures en fonction de l'appartenance des attributs dans le schéma du tableau :

1. Complète :

$$InTab(tabSch) = \{r \mid \forall\ a\ \in\ attrs(r)\ ,\ \exists\ (c, t) \in TabSch\ tel\ que\ t \preceq a\}$$

2. Partielle :
   Nous nous permettons de représenter les relations partiellement contenues dans le tableau, dans le but de garder toute information pouvant être extraite. En effet des attributs peuvent être absents, d'une part parce qu'ils peuvent être des valeurs constantes et donc non exprimés dans le tableau, d'autre part les experts ne formulent pas la même quantité d'arguments par relation sémantique[4]. Une relation est donc partiellement représentée dans le tableau, si tous ses attributs n'y sont pas contenus et qu'il existe au moins deux attributs inclus dans le schéma du tableau. Au moins deux, parce que nous considérons la relation binaire comme la relation minimale pouvant être instanciée.

$$PartInTab(tabSch) = \{r \mid \exists\ a_1\ \in\ attrs(r),\ \ \exists\ a_2\ \in\ attrs(r)\ tels\ que$$

[3] $v \preceq t$: Il ne s'agit pas forcement de la valeur v elle même mais également d'un terme trouvé dans l'ontologie grace à un mapping d'égalité *(mappEq)* ou grace à un mapping d'inclusion *(mappInc1 et mappInc2) (voir la définition des mappings dans la section 4.2.2)*

[4] (ex : Nom de l'expérimentateur, temps, ...) cela pose le problème de la discrimination entre les attributs obligatoires et optionnels d'une relation sémantique.

$$\exists\ (c,t) \in TabSch,\ \exists\ (c',t') \in TabSch\ tq\ t\ \preceq a_1\ ,t'\ \preceq a_2,\ et\ c \neq c'\}$$

Dans le cas où les deux ensembles *InTab* et *PartInTab* sont vides, une relation **générique** notée "***relation***" est créé. Nous avons choisi ceci afin de pouvoir interroger les informations du tableau même si aucune relation nà été identifiée.

Après avoir extrait les relations, nous les instantions par les valeurs du tableau. Nous avons fait le choix d'instancier toute relation extraite car nous ne disposons pas de critères nous permettant de discriminer les relations candidates.

Supposons que l'ontologie contienne les relations suivantes : {*produitLipide, produitCalorie, produitPh*, ...} et que les relations *produitLipide* et *produitCalorie* aient été normalisées, par les experts du domaine, aus 100g de produit.

Dans le tableau de **l'exemple 1**, nous extrayons les relations suivantes :
- produitLipide : instancie les valeurs des 1ère et 2ème colonnes.
- produitCalorie : instancie les valeurs des 1ère et 4ème colonnes.

À chacune de ces relations, nous ajoutons un attribut générique qui contiendra les valeurs de la deuxième colonne, dont la catégorie n'a pas été identifée.

En l'absence de l'attribut générique, la première ligne du tableau peut être interprétée par : "***100g*** *(normalisé par les experts) de poulet correspond à 270 kcal*", alors que si on cosidère l'attribut générique, l'interprétation serait : "***250g*** *de poulet correspond à 270kcal*". En effet, une information absente dans une relation pourrait conduire à une mauvaise interprétation de la relation sémantique entre les données.
Cet exemple montre le grand intérêt de l'ajout des attributs génériques aux relations représentées dans le documents SML résultat.

Nous remarquons qu'un *terme-attribut* du schéma du tableau *TabSch* peut être couvert par plusieurs relations. Par exemple : "*aliment*" correspondant à la 1ère colonne est couvert par les deux relations. D'après les descriptions et l'exemple présenté précedemment, nous pouvons voir que notre problème **d'identification des relations** ne consiste pas à trouver le *plus petit ensemble* de relations couvrant *totalement* l'ensemble des termes-attributs du schéma du tableau. Donc notre problème ne se ramène pas au problème **NP-Complet *Set Cover*** (ou *Vertex Cover*)[**?**].

4.3.3 Correspondance entre les données et les termes de l'ontologie

Pour instancier les relations indentifiées avec les valeurs du tableau, nous recherchons tout d'abord pour chaque valeur le ou les termes de l'ontologie qui lui sont proches. Nous procédons par comparaison syntaxique de chaque valeur avec les termes de l'ontologie, ce que nous nommons **"opérateurs de mapping"** ou opérateurs de correspondance. Pour définir les "opérateurs de mapping", nous considérons les valeurs et les termes comme des ensembles de mots.

Une opération élémentaire commune à tous les opérateurs de mapping est celle qui teste l'égalité de deux mots

Les opérateurs de mappings :

1. **Égalité entre deux ensembles de mots (une valeur et un terme) :** Deux ensembles de mots M(v) et M(t) sont dits égaux, si pour tout mot de M(v) il existe un mot dans M(t) qui lui est égal et que la réciproque est vraie.

$$mappEq(v) = \{t \mid \forall\, m\ \in\ M(v),\ \exists\, m'\ \in M(t),\ (m = m')$$

$$et \forall\, m' \in M(t),\ \exists\, m \in M(v),\ (m = m')\}$$

Avec '=' : l'opérateur de l'égalité lexicographique.

Exemple : mappEq(lait entier et lait surcé) = lait entier sucré
Nous pouvons trouver plusieurs termes égaux à une seule valeur, mais nous ne gardons que le premier trouvé.

2. **Inclusion entre deux ensembles de mots (une valeur et un terme) :** Pour vérifier l'inclusion, ou plus exactement le "recouvrement", entre une valeur d'un tableau et un terme de l'ontologie, nous testons l'inclusion entre leurs ensembles de mots respectifs. Tout d'abord nous définissons le "recouvrement" entre deux ensembles de mots. (**NB.** dans la suite nous appelons *inclusion* d'ensembles de tels *recouvrements*). Soit le prédicat suivant, nommé *Inc*, qui vérifie l'inclusion entre une valeur et un terme.

$$Inc(t, v) \Leftrightarrow (\forall\, m \in M(v) \Rightarrow m \in M(t))$$

Exemple : *Inc*(poudre de lait, lait entier en poudre)= vrai

car : {poudre, lait} est inclus dans {lait, entier, poudre}
Rappelons que nous ne considérons pas les mots vides : {de, en}

Nous utilisons le test d'inclusion de deux manières :

- **Inclusion valeur → terme :** Lors du test d'inclusion d'une valeur dans un terme, nous considérons une heuristique qui consiste à ne retenir que les termes incluant la valeur et ayant un nombre minimum de mots et non comparables (i.e. ne sont pas inclus les uns dans les autres).

  $$mappInc_1(v) = \{t \mid Inc(v, t) \wedge [not\ \exists\, t',\ (inc(v, t') \wedge inc(t', t))]\}$$

  Nous justifions cette décision par le fait que plus le terme incluant la valeur est grand plus l'approximation sémantique s'affaiblit.

  Exemple : $mappInc_1$(Viande) = {viande hachée, viandes sèches, viande de poisson mariné}
  Nous pouvons trouver aussi *viande hachée de bœuf* comme terme incluant la valeur (viande). Il n'est pas retenu car il existe un terme déjà retenu et inclus dans celui-ci *viande hachée*. On considère en effet que *viande hachée* est sémantiquement plus proche de *viande* que *viande hachée de bœuf.*

- **Inclusion terme → valeur :** Cette fois-ci nous retenons les termes les plus grands et non comparables (i.e. qui ne sont pas inclus les uns dans les autres).

  $$mappInc_2(v) = \{t \mid Inc(t, v) \wedge [not\ \exists\, t',\ (inc(t', v) \wedge inc(t, t'))]\}$$

  Nous justifions cette décision par le fait que plus le terme inclus dans la valeur est grand, plus l'approximation sémantique augmente.

  Exemple : $mappInc_2$(lait entier et écrémé en poudre sucré) = {lait entier en poudre, lait écrémé en poudre, lait sucré}

  Nous pouvons trouver aussi {*lait entier, poudre de lait*} comme terme inclus dans la valeur (*lait entier et écrémé en poudre sucré*), mais nous ne les retenons pas car il existe un terme de taille plus grande déjà retenu auquel ils sont comparables : {lait entier en poudre} (i.e. ils y sont inclus).

On considère en effet que *lait entier en poudre* est plus proche sémantiquement de *lait entier et écrémé en poudre sucré* que les termes {*lait entier, poudre de lait*}.

3. **Intersection entre deux ensemble de mots (une valeur et un terme) :** De la même manière que pour l'inclusion, nous définissons tout d'abord le prédicat qui vérifie l'intersection entre une valeur et un terme ou plus exactement entre deux ensembles de mots.

$$Inter(v,t) \Leftrightarrow \exists\, m \in M(v)\, et\, m \in M(t)$$

Exemple : Inter(salades batavia, salade verte) = vrai.

L'opérateur de mapping qui renvoie les termes de l'ontologie ayant une intersection avec la valeur, prend en compte une heuristique de discrimination entre les termes candidats. Cette heuristique consite à ne retenir que les termes pour lesquels les ensembles d'intersection sont incomparables (i.e. ne sont pas inclus les uns dans les autres)

$$mappInter(v) = \{t \mid Inter(v,t) \wedge (not\, \exists\, t',\ Inc(interSet(v,t), interSet(v,t'))\}$$

L'ensemble de mots qui constituent l'intersection entre une valeur et un terme est calculé de la façon suivante :

$$InterSet(v,t) = \{m \mid m \in M(t) \wedge m \in M(v)\}$$

Exemple :
V = "Biscuits fourrés à la confiture de fraise ou au chocolat"
mappInter(V) = {biscottes et biscuits au chocolat, compotes et confitures de fraise, petits pains fourrés} avec :
Les ensembles d'intersection :
I_1 =[*InterSet*(V, biscottes et biscuits au chocolat)={biscuit, chocolat}]
I_2 =[*InterSet*(V, compotes et confitures de fraise)={confiture, fraise}]
I_3 =[*InterSet*(V, petits pains fourrés)={fourrés}]

Dans cet exemple les termes retenus ont bien des ensembles d'intersection (I_1, I_2, I_3) incomparables.

Nous avons trouvé {confiture d'abricot, chocolat noir} comme termes candidats. Mais en applicant l'heuristique, nous ne retenons pas ces termes, car *l'InterSet* I'_1 = {confiture} (resp. I'_2= {chocolat}) est inclus dans I_2 (resp. I_1)

Le rapprochement sémantique d'une valeur du tableau vers une ontologie

La sémantique d'une valeur du tableau, dans l'ontologie d'un domaine, est contenue dans l'ensemble des termes qui lui sont sémantiquement proches. Pour calculer cet ensemble, nous recherchons les termes de l'ontologie correspondant à la valeur du tableau par l'intermédiaire des "opérateurs de mapping".

Pour résumer, l'enrichissement sémantique d'une valeur est donné soit par le terme qui lui est égal, soit par l'ensemble des termes trouvés par des tests d'inclusion dans les deux sens, ou obtenu par l'ensemble des termes trouvés par des tests d'intersection. Le **'ou'** est exclusif, donc les termes enrichissant une valeur résultent de l'un des trois traitements égalité, inclusion ou intersection. L'enrichissement d'une valeur par la sémantique d'un domaine d'intérêt est décrit par le formalisme suivant :

Si ($|mappEq(\mathrm{v})| \neq \emptyset$) *alors*
mapp(v)=*mappEq*(v)
Sinon Si ($|mappInc_1(\mathrm{v})) \bigcup mappInc_2(\mathrm{v})| \neq \emptyset$) *alors*
mapp(v)= $mappInc_1(\mathrm{v})) \bigcup mappInc_2(\mathrm{v})$
Sinon Si ($|mappInter(\mathrm{v})| \neq \emptyset$) *alors*
mapp(v)=*mappInter*(v)
Sinon **mapp**(v)=NULL

4.4 L'algorithme Xtab2SML

Description de l'algorithme Xtab2SML : Comme nous l'avons décrit dans les sections (CF.4.1 et 4.2), l'algorithme permettant l'enrichissement sémantique des tableaux en document SML prend en entrée un tableau stocké dans un document *XTab* (CF. la DTD XTab *Annexe B.1* et un extrait d'un document XTab *Annexe B.2*) et une ontologie du domaine considéré représentée dans le langage XML (CF. DTD ontologie *Annexe A.2* et un extrait *Annexe A.3*). Nous présentons ci-après l'algorithme global d'enrichissement sémantique, nommé ***XTab2SML***.

Algorithm 1 `Algorithme Xtab2SML : enrichissement sémantique de tableaux`

Inputs : Ontologie *Onto : (taxonomie, synonymes, tarductions, schema-relationnel)*
Ducument *Xtab : (tabTit, titles, Vals, infoSupp)*
Support *supp* : un pourcentage à utiliser pour la recherche des catégories des colonnes.

Output : Document *docSml* : un document XML conforme à la DTD SML

1: *docSml* ← document XML vide
2: *TabSch* ← construireSchemaTab(*Xtab.titles, Xtab.Vals, Xtab, Onto, S*)
3: *Rels* ← identificationRelationsTabSch(*TabSch, onto.schema-relationnel*)
4: **If** *Rels* = ∅ **Then**
5: *relGen* ← creationRelationGenerique(TabSch)
6: *R* ← instanciationRelationDansTableau(*relGen*, Xtab.Vals, TabSch)
7: ajouterRelation(*R,docSml*)
8: **Else If** *Rels* ≠ ∅ **Then**
9: **Foreach** *relation* $R_i = (nomRel, (attr_1, \ldots, attr_n)) \in Rels$ **Do**
10: *R* ← instanciationRelationDansTableau(R_i, *Xtab.Vals, TabSch*)
11: ajouterRelation(*R,docSml*)
12: **End For**
13: **End If**
14: ajouterInfoSupplementaire(*Xtab.infoSupp, docSml*)
15: **Return** *docSml*

Nous décrivons ci-dessous les différentes routines utilisées par l'algorithme :

1. Construction du schéma du tableau (*construireSchemaTab*) : cette routine permet de déduire un schéma à partir du tableau. Rappelons qu'un schéma d'un tableau est l'ensemble des catégories (*termes-attributs*) affectés à l'ensemble des colonnes (CF. Définition 5).
2. Identification des relations (*identificationRelationTabSch*) : dans cette routine, en se servant du schéma du tableau calculé dans une étape précédente, nous extrayons l'ensemble des relations représentées dans schéma du tableau.

Algorithm 2 `Fonction construireSchemaTab`

Inputs : *titles* : la liste des titres des colonnes du tablau,
Vals : l'ensemble de valeurs du tableau organisées en colonnes,
Onto : l'ontologie du domaine (Ex : Sym'Previus),
supp : un pourcentage que les valeurs d'une colonne doivent satisfaire.

Output : *tabSch* : le schéma du tableau $\{(C_1,\ Cat_1),\ (C_2,\ Cat_2),\ ...,\ (C_n,\ Cat_n)\}$

1: *attrsSchRel* ← extraireAttributsRelation(*Onto.schema-relationnel*)
2: **Foreach** colonne $C_i = [v_1,\ v_2,...,\ v_m] \in$ Vals **Do**
3: j←0 et *catTrouve* ← Faux
4: *catTemp* ← ""
5: **While** (*catTrouve* = faux et $j <$ *attrsSchRel.taille* -1) **Do**
6: *SubTree* ← construireSubTree(*attrsSchRel[j])*
7: **If** ((colonneSatisfaitSupport(C_i, *SubTree, S*)) = Vrai) **Then**
8: *catTrouve* ← Vrai
9: **Else If** (Inc(*titles[i]*, *attrsSchRel[j]*) **Then**
10: *catTemp* ← *attrsSchRel[j]*
11: **End If**
12: j← j+1
13: **End While**
14: **If** (catTrouve = Vrai) **Then**
15: tabSch[C_i]← attrsSchRel[j-1]
16: **Else If** (catTrouve = Faux) **Then**
17: **If** (catTemp = "") **Then**
18: tabSch[C_i]← "attribut"
19: **Else If** (catTemp ≠ "") **Then**
20: tabSch[C_i]← catTemp
21: **End If**
22: **End If**
23: **End For**
24: **Return** *tabSch*

Algorithm 3 `Fonction identificationRelationTabSch`

Inputs : *tabSch* : le schéma du tableau
Onto.schema-Relationnel : le schéma relationnel de l'ontologie

Output : *Rels* : la liste des relations identifiées dans le schéma. Elles sont de la forme :
$R_i = \{(attr_1,numCol),\ ...,\ (attr_n,numCol)\}$

1: **Foreach** relation R =(nom, ($attr_1$,..., $attr_2$))
∈ *Onto.schema-relationnel* **Do**
2: **If** (InTab(R, TabSch)) **Then**
3: ajouterRelation(R, *Rels*)
4: **End If**
5: **End For**
6: **Return** *Rels*

3. Instantiation des relations identifiées (*instanciationRelationDansTableau*)[5] : dans cette routine nous recherchons pour chaque valeur d'une case l'ensemble des termes de l'ontologie lui

[5]La relation instanciée résultat correspond à un ensemble ordonné suivant l'ordre d'apparition des attributs de la relation et leurs colonnes respectives

correspondant. Si aucun terme ne lui correspond, nous gardons uniquement la valeur d'origine du tableau et nous mettons l'indicateur *indOnto* à la valeur *notFound*.

Algorithm 4 `Fonction instanciationRelationDansTableau`

Inputs :
R = {(attr$_1$,numCol), ..., (attr$_k$,numCol)} : une relation à instancier,
avec k le nombre d'attributs de la relation R (l'arité de R)
Vals : ensemble des valeurs du tableau (la matrice (lignes, colonnes))
TabSch : le schéma du tableau pré-calculé

Output :
relInst : La relation R instanciée par les valeurs du tableau, qui est de la forme suivante :
{(attr$_1$,{origineVal$_{(1,numCol)}$, finalVal$_{(1,numCol)}$})
,...,(attr$_k$,{origineVal$_{(k,numCol)}$, finalVal$_{(k,numCol)}$})}

1: **For** (i=0 ; i < R.nbAttrs ; i++) **Do**
2: valCol ← getValues(TabSch[R$_i$])
3: **For** (j=0 ; j < valCol.nbVals ; j++) **Do**
4: finalVal ← ∅
5: origine←valCol[j]
6: valsRelInst← ∅
7: ensTerme ← appliquerOperateurMappingEq(valCol[j],R, onto)
8: **If** (ensTerme = ∅) **Then**
9: ensTerme ← appliquerOperateurMappingInc1(valCol[j],R, onto)
10: ensTerme ← ensTerme ⋃ appliquerOperateurMappingInc2(valCol[j],R, onto)
11: **If** (ensTerme = ∅) **Then**
12: ensTerme ← appliquerOperateurMappingInter(valCol[j],R, onto)
13: **If** (ensTerme ≠ ∅) **Then**
14: finalVal←ensTerme
15: *valsRelInst* ← *valsRelInst*⋃ {(attr$_i$,{ origineVal, finalVal})}
16: **Else If** ensTerme = ∅ **Then**
17: *valsRelInst* ← *valsRelInst*⋃ {(attr$_i${origineVal, NULL})}
18: **End If**
19: **End If**
20: **End If**
21: **End For**
22: *relInst* ← *relInst* ⋃ {valsRelInst}
23: **End For**
24: **Return** *relInst*

Estimation de la complexité de l'algorithme XTab2SML : Pour estimer cette complexité nous avons besoin de la taille des données en entrée :

- **L'ontologie du domaine :** {*nbrel* : nombre de relations du schéma relationnel, *nbAttrs* : taille de l'ensemble *attrsSchRel*, $\Sigma\delta$: nombre de relations de spécialisation entre les termes de l'ontologie, n : nombre de termes de l'ontologie}
- **Le tableau XTab :** {*nbLg* : nombre de lignes, *nbCols* : nombre de colonnes}

Pour l'algorithme global XTab2SML, les instructions des lignes (1, 5, 7, 11 et 14) se font en un temps constant. Les instructions des lignes (2, 3, 6 et 10) ne sont pas élémentaires, nous estimons la complexité de chacune ci-dessous.

1. ***construireSchemaTab* à la ligne 2 (CF. Algorithm 2)** : la boucle Foreach de la ligne 2 est executée en $\mathcal{O}$(nbCol). À l'intérieur de cette boucle, la boucle While de la ligne 5 s'execute en $\Sigma\delta$. Dans cette boucle, la fonction à la ligne 6 est executée en $\mathcal{O}(n+\Sigma\delta)$ (e.g. un parcours en largeur d'un graphe) et le test de la ligne 7 est effectué en un temps $\mathcal{O}(n+\Sigma\delta)$. Les lignes (8, 9, 10 et 11) en un temps constant et de la ligne 14 jusqu'à la ligne 22, en un temps constant.

Donc cet algorithme a la complexité :
$\mathcal{O}$(nbCol)+$\Sigma\delta$ * 2*$\mathcal{O}$(n+$\Sigma\delta$)= $\mathcal{O}$($\Sigma\delta$ * $\mathcal{O}$(n+$\Sigma\delta$)).

2. ***identificationRelationTabSch* à la ligne 3 (CF. Algorithm 3) :** la boucle Foreach de la ligne 1 est executée en $\mathcal{O}$(nbRel) et le test de la ligne 2 est effectué en $\mathcal{O}$(nbAttrs(R)) (c'est un temps négligeable par rapport à $\mathcal{O}$(nbRel)). Donc la complexité de cet algorithme est de : $\mathcal{O}$(nbRel).
3. ***instanciationRelationDansTableau* aux lignes 6 et 10 (cf Algorithm 4) :** la boucle For de la ligne 1 s'excute en $\mathcal{O}$(nbAttrs). À l'intérieur de cette boucle, la boucle For de la ligne 3 est effectuée en $\mathcal{O}$(nbLg). Les instructions à l'intérieur de cette boucle sont effectuées en 2*$\mathcal{O}$(n+$\Sigma\delta$). Donc cette algorithme a la compléxité :
$\mathcal{O}$(nbAttrs)*$\mathcal{O}$(nbLg)*2*$\mathcal{O}$(n+$\Sigma\delta$) = $\mathcal{O}$(nbAttrs)*$\mathcal{O}$(nbLg)*$\mathcal{O}$(n+$\Sigma\delta$)

À partir de ces estimations de compléxité des différentes routines utilisées, la complexité de l'algorithme ***XTab2SML*** est la suivante, sachant que la boucle Foreach à la ligne 9 de l'algorithme 1 est executée en $\mathcal{O}(nbRel)$:
Complexite(algorithm 2) + *Complexite*(algorithm 3) + *nbRel* Complexite*(algorithm 4)=
$\mathcal{O}(\Sigma\delta$ * $\mathcal{O}(n+\Sigma\delta))$ + $\mathcal{O}(nbRel)$ + nbRel * ($\mathcal{O}(nbAttrs)$*$\mathcal{O}(nbLg)$+$\mathcal{O}(n+\Sigma\delta)$

$$\text{Complexit}\acute{e}(XTab2SML) = \mathcal{O}(nbRel) * (\mathcal{O}(nbAttrs)*\mathcal{O}(nbLg)*\mathcal{O}(n+\Sigma\delta))$$

4.5 Trace des traitements et du contexte

Nous avons défini une série d'indicateurs permettant d'évaluer la pertinence du tableau par rapport au domaine d'intérêt et la validité des informations. Ces indicateurs permettent également d'indiquer la présence de renseignements complémentaires en provenance du document original. Avant de décrire ces indicateurs, voici un exemple de tableau :

Exemple 1 : Dans ce tableau, nous identifions la relation ***alimentTempératureMicroorganisme***, qui représente la relation sémantique entre les données des première, deuxième et quatrième colonnes.

Produit	**température**	**Item**	**Germe**
Pomme	20 degrés	20%	Candida
Compote de prunes	35 degrés	40%	Levure
Champignons de Paris	15 degrés	38%	Escherichia coli

TAB. 4.3 – *Contamination des aliments*

Les huit indicateurs définis sont les suivants :

1. **additionalAttr** : Un attribut XML ajouté aux tags *ligneRel* du document SML. Son ensemble de définition est : Def(additionalAttr)={yes,no}.
   Il permet d'informer de la présence d'un ou plusieurs attributs apparaissant dans les colonnes du tableau et non reconnus dans le schéma relationnel de l'ontologie. En d'autres termes, la catégorie de cette colonne n'appartient à aucune signature de relation du schéma relationnel de l'ontologie. Dans l'exemple 1, nous avons la troisième colonne pour laquelle aucune catégorie de données n'a été trouvée. Dans ce cas, nous ajoutons un attribut générique *attribut* à la relation *alimentTempératureMicroorganisme*.
2. **indProc** : un attribut XML ajouté aux tags *attribut* d'une relation du document SML (e.g. aliment, microorganisme,). Son ensemble de définition est : Def(indProc) = {yes, no}. Il indique si un traitement a été nécessaire pour la recherche d'un terme correspondant à la valeur d'origine du tableau. Cet indicateur vaut *yes* quand ***indOnto*** = {*inclusion, intersection*} et quand il s'agit d'une liste de valeurs segmentée.

3. **indOnto** : un attribut XML ajouté aux tags attribut d'une relation du document SML, son ensemble de définition est : Def(indOnto) = {complete, inclusion, intersection, notFound}.

   Il permet d'indiquer si un traitement a été effectué lors de la recherche d'un terme correspondant à la valeur d'origine du tableau. Dans l'exemple 1, dans les troisième et quatrième lignes, les valeurs de produit n'existent pas telles quelles dans l'ontologie Sym'Previus. Nous trouvons le terme "*compote et confiture*" comme terme correspondant à *"compote de prunes"* après un test d'intersection. Le terme *"Champignon"* comme terme correspondant à la valeur *"Champignon de Paris"* après un test d'inclusion. L'indicateur indOnto vaut respectivement, pour ces deux valeurs, ***intersection*** et ***inclusion***.

4. **indCat** : un attribut XML ajouté aux tags attribut d'une relation du document SML. Son ensemble de définition est : Def(indCat) = {aliment, microorganisme, facteur, ph,..., notFound}. Il indique le nom de la catégorie à laquelle les termes correspondant à la valeur d'origine appartiennent. Cet indicateur est indispensable pour pallier le problème des colonnes dont les types de données sont hétérogènes. Supposons que la colonne *produit* de l'exemple 1 contienne la valeur *chlore* et que nous trouvions *"concentration de chlore"* comme terme correspondant. Dans ce cas, l'indicateur *indCat* vaudra *Facteur* et non pas *aliment*, comme pour le reste des valeurs.

5. **indTrans** : un attribut ajouté aux attributs d'une relation du document SML. Son ensemble de définition est : Def(indTrans) = {interne, externe, mixte, notRequired}. Pour prendre en compte des tableaux écrits en différentes langues, nous devons les traduire en Français car le contenu de l'ontologie est en Français. Quand il s'agit de tableaux en Anglais, nous exploiterons en priorité la partie traductions de l'ontologie ; si cette étape échoue, nous utiliserons un traducteur externe. Cet indicateur représente le type de traducteur utilisé pour retrouver un terme correspondant à la valeur d'origine du tableau. Cet indicateur vaut :
   - *interne* : si la partie traduction de l'ontologie a été utilisée,
   - *externe* : si un traducteur externe, par exemple "Systran", a été utilisé,
   - *mixte* : quand les deux traducteurs ont été utilisés,
   - *notRequired* : dans le cas des valeurs numériques, aucune traduction n'est nécessaire.

   Exemple 2 :

Product	**Water activity (Aw)**
Fresh meat	0.99
Dried fruit	0.6
Apple	0.97

TAB. 4.4 – *The typical water activity of some foodstuffs*

Les valeurs du tableau :
V = fresh meat : Dans la partie traduction de l'ontologie nous trouvons le terme viandes fraîches.

V = Dried fruit : Dans la partie traduction de l'ontologie, aucun terme ne correspond donc nous utilisons un traducteur externe et il nous renvoie la valeur Fruit sec.

V = Apple : Dans la partie traduction de l'ontologie, aucun terme ne correspond donc nous utilisons un traducteur externe et il nous renvoie la valeur "Apple". Nous avons une mauvaise traduction, due à une d'interprétation par le traducteur (Systran), car il a considéré la marque de matériel informatique Apple!!! Systran renvoie pomme pour la valeur apple (le 'a' en minuscule).

Nous voyons ici l'intérêt de cet indicateur de type de traduction, qui influence le degré de confiance que MIEL++ affecte à l'enrichissement sémantique.

Il renseigne du type de traducteur utilisé pour retrouver un terme correspondant à la valeur d'origine du tableau

6. **indMatch** : un attribut XML ajouté aux tags des attributs génériques d'une relation du document SML (les tags "*attribut*"). Son ensemble de définition est : Def(indMatch) = {aliment, microorganisme, facteur, ph,...}.

   Il indique le nom de la catégorie de l'ontologie correspondant à la valeur des attributs de cette colonne. Sinon, indMatch prend la valeur *none*. Dans l'exemple 1, si la catégorie de la troisième colonne est identifiée, c'est à dire que "Item" existe dans l'ontologie, *indMatch* vaut "item".

7. **indWordsIntersection** : Cet attribut est ajouté aux tags identifiés dans l'ontologie des relations du document SML (aliment, microorganisme,). Son ensemble de définition est Def(indWordsIntersection) = [0..nombre de mots du plus grand terme]. Il permet d'indiquer, lors d'une transformation de type "*intersection*" (attribut *indOnto*), le nombre de mots constituant l'intersection. Cet attribut est optionnel, il apparaît uniquement quand l'indicateur indOnto vaut intersection et est exploité par le moteur d'interrogation MIEL++. Dans l'exemple 1, pour la valeur "*compote de prunes*" le terme "*compote et fruit*" a été trouvé avec un test d'intersection. Seul le mot "*compote*" est commun aux deux valeurs donc *indWordsIntersection* vaut 1.

8. **indNumCol** : Un attribut XML ajouté aux attributs d'une relation du document SML. Son ensemble de définition Def(indNumCol) = [1..N]. Il indique le numéro de la colonne du tableau XTab que l'attribut de la relation instancie. Cet indicateur est omis dans les documents SML donnés comme exemples.

De plus, nous gardons la valeur d'origine du tableau pour garder trace du contexte des données et pour permettre au moteur d'interrogation de proposer à l'utilisateur les valeurs d'origine du tableau, en cas d'insatisfaction.

4.6 Description d'un tableau dans un document SML

Nous décrivons dans cette section la structure du document SML en commençant par les éléments les plus généraux et en terminant par les plus spécifiques. De plus, nous mettons l'accent sur les indicateurs de traitements et du contexte au fur et à mesure de la description.

Structure SML en fonction du nombre et de la nature des relations

Comme nous l'avons décrit brièvement ci-dessus, un tableau est structuré sous forme d'un ensemble de lignes. Une ligne peut contenir une ou plusieurs relations.

Une seule relation par ligne :

<ligneRel> <alimentPh> ... </alimentPh></ligneRel>
<ligneRel> <alimentPh> ... </alimentPh> </ligneRel>
...

FIG. 4.2 – *Représentation d'une seule relation en SML*

Plusieurs relations par ligne :

Aucune relation identifiée (CF. 4.4) :
Dans ce cas, aucune relation du schéma relationnel n'est contenue dans le schéma du tableau **TabSch**(définition 5). Une relation générique ***"relation"*** est donc créée et ajoutée aux ligneRel du document SML. Cette relation contient un ensemble d'attributs génériques ***"attribut"*** (définis plus bas), correspondant à chaque colonne du tableau.

```
<ligneRel> <alimentPh> ...</alimentPh>
<alimentPh> ... </alimentPh>
<alimentlipide> ... </alimentlipide>
</ligneRel>
...
```

FIG. 4.3 – *Représentation de plusieurs relations en SML*

```
<ligneRel additionalAttr="yes"> <relation>
<attribut... > ... </attribut> ...
</relation> </ligneRel>
...
```

FIG. 4.4 – *Représentation d'une relation générique en SML*

Contenu d'une relation : La structure d'une relation dans un document SML est conforme à sa signature dans le schéma relationnel de l'ontologie (aux attribut génériques près). Chaque attribut de la signature d'une relation est représenté par un tag XML qui contient la valeur du tableau, enrichie par les termes de l'ontologie.

Dans le cas d'un tableau ayant une ou plusieurs colonnes pour lesquelles la catégorie n'a pas été identifée, nous ajoutons à toutes les relations du tableau des attributs génériques, représentés par les tags XML ***"attribut"***. Pour garder cette information, un indicateur de traitement représenté par un attribut XML ***"additionalAttr"***, initialisé à la valeur *"yes"*, est ajouté à chaque ligne relationnelle "*ligneRel*".

À chaque attribut de la relation sont ajoutés les indicateurs de traitements *indOnto, indCat, indTrans, indProc* (décrits dans la section 5.3.4).

Une relation d'un tableau sans attributs génériques (CF. 4.5) :

```
<ligneRel additionalAttr="no">
<alimentPh><aliment ...> ... </aliment>
<ph ...>... </ph>
</alimentPh>
...
```

FIG. 4.5 – *Représentation simplifiée d'une relation sans attributs génériques*

Une relation d'un tableau avec un attribut générique (CF. 4.6) : En plus des indicateurs de traitements ajoutés habituellement à tous les attributs, nous ajoutons un indicateur *indMatch* (voir section 5.3.4) à tous les attributs génériques ajoutés aux relations.

```
<ligneRel additionalAttr="yes">
<alimentPh><aliment ...> ... </aliment>
<ph ...>... </ph>
<attribut indOnto="inclusion" indCat="acide" indTrans="none" ind-
Proc="yes" indMatch="acide">
<finalVal> acide </finalVal>
<origineVal> Acide gras polyinsaturés </origineVal>
</attribut> </alimentPh>
...
```

FIG. 4.6 – *Représentation simplifiée d'une relation avec des attributs génériques*

Repésentation SML des valeurs du tableau enrichies par des terme de l'ontologie

Les valeurs des cases du tableau sont contenues dans les attributs des relations du document SML. Nous avons ajouté un niveau de structuration à chaque attribut de relation, qui consiste en les tags ***"finalVal"*** et ***"origineVal"***.

Dans le tag *finalVal* nous stockons les termes de l'ontologie, s'ils existent, résultant de l'application des opérateurs de mapping entre les valeurs et les termes. Nous avons vu qu'il peut y avoir plusieurs tags *finalVal* (plusieurs termes sémantiquement proches) pour une seule valeur du tableau.

Dans le tag *origineVal* nous stockons, dans tous les cas, la valeur d'origine du tableau pour garder trace du contexte. De plus, le moteur d'interrogation pourrait exploiter ces valeurs d'origines (en particulier quand *indProc =yes*) en les proposant à l'utilisateur comme réponses supplémentaires.

Nous allons présenter ci-après la structure SML de ces valeurs, en fonction des opérateurs de mapping appliqués et en fonction de la structure de la valeur (valeur simple ou liste).

Structure du document SML en fonction du traitement effectué

1. **Égalité** : Dans ce cas, l'indicateur *indOnto* vaut "complete" et l'indicateur *indProc* vaut "no".

```
<aliment indOnto="complete" indCat="aliment" indTrans="none"
indProc="no">
<finalVal>pommes</finalVal>
<origineVal>pommes</origineVal>
</aliment>
```

FIG. 4.7 - *Représentation SML d'une valeur déjà existante dans l'ontologie*

2. **Inclusion** : Dans ce cas, l'indicateur *indOnto* vaut "inclusion", l'indicateur *indProc* vaut "*yes*" et il peut y avoir plusieurs termes correspondant à une même valeur.

```
<aliment indOnto="inclusion" indCat="aliment" indTrans="none"
indProc="yes">
<finalVal>compote de fruits</finalVal>
<finalVal>fruits exotiques</finalVal>
<origineVal>compote de fruits exotiques</origineVal>
</aliment>
```

FIG. 4.8 - *Une valeur enrichie par des termes après test d'inclusion*

3. **Intersection** : Dans ce cas, l'indicateur *indOnto* vaut "intersection", l'indicateur *indProc* vaut "*yes*" et il peut y avoir plusieurs termes correspondant à une même valeur.

```
<aliment indOnto="intersection" indCat="aliment" indTrans="none"
indProc="yes">
<finalVal>raisin secs</finalVal>
<finalVal>pain aux raisins</finalVal>
<finalVal>vin blanc</finalVal>
<origineVal>raisin blanc</origineVal>
</aliment>
```

FIG. 4.9 - *Une valeur enrichie par des termes après test d'intersection*

4. **Aucun terme n'a été trouvé** : Dans ce cas, l'indicateur *indOnto* vaut "notFound", l'indicateur *indProc* vaut "no" et nous ajoutons un tag *finalVal* vide.

```
<ph indOnto="notFound" indCat="ph" indTrans="none"
indProc="no">
</finalVal>
<origineVal>6.56</origineVal>
</ph>
```

FIG. 4.10 - *Une valeur n'ayant aucun terme proche sémantiquement*

Cas de liste de valeurs dans une case

Jusque là, nous n'avons considéré qu'une seule valeur par cellule. Une case du tableau peut contenir une liste de valeurs, de même type ou de type différents et nous n'aborderons, dans cette partie, que les valeurs du même type. Pour prendre en compte ce cas de listes de valeurs et éviter les ambiguïtés, nous encapsulons l'attribut aliment dans un tag ***listeAliments***, quelque soit la structure de la valeur.

```
<ligneRel additionalAttr="no">
<alimentPh>
<listeAliments> <aliment ...> ... </aliment> </listeAliments>
<ph indOnto="complete" indCat="aliment" indTrans="none"
indProc="no"> ... </ph> </alimentPh> ...
</ligneRel>
```

FIG. 4.11 - *La représentation SML d'une relation avec prise en compte de liste de valeurs* -

Nous gardons tout de même la valeur d'origine, dans les deux cas (avec ou sans segmentation), dans un tag XML ***origineValListe***.

Cas de segmentation :

Quand il s'agit de liste, avec des séparateurs de valeurs distingués, nous segmentons la liste en plusieurs valeurs. Ceci est obtenu, en extrayant toute valeur suivie d'un ensemble de caractères considérés comme séparateurs. Nous représentons cette configuration en SML, en ajoutant au tag ***listeAliments*** autant de tags *aliment* qu'il y a de valeurs segmentées. Dans ce cas nous initialisons l'indicateur ***indProc*** à *yes*.

```
<ligneRel><alimentPh>
<listeAliments> <aliment ...>
<finalVal>viandes fraîches</finalVal>
<origineVal>Viandes fraîches</origineVal>
</aliment><aliment ...> ... </aliment>
<origineValListe> Viandes fraîches et poisson</origineValListe>
</listeAliment>
<ph ...>... </ph>
</alimentPh> <ligneRel>
```

FIG. 4.12 - *Liste de valeur avec segmentation* -

Cas de non segmentation :

Cependant, la segmentation des listes peut être parfois ambiguë, donc une segmentation ne serait pas pertinente. Dans ce cas, nous considérons la liste comme une seule valeur et nous recherchons les termes de l'ontologie qui lui sont proches. Nous représentons les termes trouvés dans des tags XML ***finalValListe*** auxquels nous ajoutons les indicateurs de traitement (indOnto, indCat, indTrans, indProc).

```
<ligneRel><alimentPh>
<listeAliments>
<finalValListe indOnto="inclusion" indCat="aliment" ind-
Trans="none" indProc="yes">viandes fraîches</finalValListe>
<finalValListe indOnto="inclusion" indCat="aliment" ind-
Trans="none" indProc="yes">poisson</finalValListe>
<origineValListe>Viandes fraîches et poisson</origineValListe>
</listeAliments>
<ph ...> ... </ph></alimentPh></ligneRel>
```

FIG. 4.13- *Liste de valeurs sans segmentation* -

4.7 Interrogation des documents SML par MIEL++

Les documents SML peuvent être stockés dans un entrepôt et interrogés à l'aide d'un moteur d'interrogation qui exploite les éléments de contexte visualisables (titres, lignes, valeurs originales, indicateurs de traitement et la présence de mappings possibles (relations ou termes)) en cas d'ambiguïté. Dans le cadre du projet e.dot, les documents SML sont exploités par le moteur d'interrogation MIEL++ de la manière suivante : ils sont stockés sur un serveur XZS (Xylème Zone Server). Ces documents ne sont pas stockés ensemble mais séparés en trois "clusters" différents correspondant chacun à un ensemble de requêtes pré-définies que l'on appele ***vue*** du logiciel. La vue *effet/facteur*, la vue *charge microbienne* et la vue divers. En fonction du contenu des documents SML, leur répartition est effectuée dans trois "clusters " (CF. *Annexe E.1*).

Exemple : la vue charge microbienne En vue d'une interrogation de la vue SML charge microbienne, 5 relations sémantiques, décrites dans le schéma relationnel de l'ontologie, sont exploitées :

alimentPh (aliment, ph, attribut*) - **alimentAw** (aliment, aw, attribut*) - **alimentEtat** (aliment, état, attribut*)- **alimentStructure** (aliment, structure, attribut*) - **alimentMicroorganismeConcentration**(aliment, Microorganisme, concentration, attribut*)

Remarque : Un document SML doit être stocké dans cette vue s'il possède la relation sémantique intitulée ***"alimentMicroorganismeConcentration"***.

Une requête *XQuery* a été définie pour interroger les relations *alimentPh* et *alimentMicroorganismeConcentration*. La spécification des valeurs d'attributs de sélection est la suivante :

{aliment ="chou", pH =5, microorganisme="Listeria M."}.

Les indicateurs de traitements *additionnalAttr* et *indMatch* sont pris en compte pour montrer à l'utilisateur d'éventuelles informations supplémentaires[6] (*additionnalAttr = yes*) et de montrer leur catégorie si elle a été identifiée (*indMatch* $\neq$ *attribut*). La requête est donnée en *Annexe E.3*. Le résultat de cette requête peut avoir la forme suivante :

```
<tableau>
<titre> le titre du tableau </titre>
<ligne><ligneRel>...</ligneRel>
<donneeSupp> 25 DegC </donneeSupp>
<categorie>température</categorie>
<tableau>
```

FIG. 4.14 - *Une structure possible du résultat de cette requête* -

Pour la requête XQuery de le vue *Aliment/effet du facteur analysé* est également donnée en Annexe E.2. (voir la forme arborescente des deux requêtes en Annexe FIG. E.3 et FIG. E.2).

Vue divers

Dans un second temps, une 3ème vue plus flexible pour l'utilisateur doit être définie. Dans cette vue, l'utilisateur pourra effectuer une interrogation sur l'ensemble des attributs qui ne sont pas présents dans une des deux autres vues (par exemple : lipide, calorie,).

Remarque : Tous les documents SML ne répondant pas aux critères des 2 autres vues sont stockés dans cette vue *divers*.

[6] dans cette requête nous montrons à l'utilisateur le premier attribut supplémentaire contenu dans la relation.

5 Développement et expérimentations

5.1 Implémentation

Comme le schéma de la *FIG. 4 (du chapitre 4)* le montre, l'application XTab2SML prend en entrée d'une part l'ontologie du domaine en format XML et d'autre part le document XTab à transformer. Elle renvoie un document SML résultant de l'enrichissement du document XTab par la sémantique du domaine stockée dans l'ontologie. Pour mettre en œuvre les idées présentées dans ce rapport, nous avons implémenté un prototype d'application écrit dans le langage **Java**. Ce dernier s'est révélé particulièrement adapté à cette réalisation. Comme principal atout du langage Java sa portabilité, qui assure l'execution de tout programme développé en Java sur n'importe quelle plate-forme (*MS-Windows, UNIX, MAC-OS,* etc.).

Le modèle DOM[1] (*Document Object Model*) a été utilisé pour avoir un accès plus aisé à des documents XML via un ensemble d'objets classés en arborescence. Cette arborescence est en fait une représentation équivalente au fichier XML, mais sa manipulation via Java en est grandement simplifiée. Ainsi les fichiers XML représentant l'ontologie et le tableau XTab sont convertis en arbre DOM, en utilisant l'API de la librairie JAXP [2]. Cette dernière nous a également permis de générer le document SML résultat. Après avoir représenté l'ontologie et le document XTab sous une forme d'arbres DOM. Ces derniers, sont ensuite compilés dans une représentation interne sous la forme d'une *"Table de Hachage"*. Ce choix est fait dans le but d'optimiser la recherche des données à manipuler. En effet l'utilisation de la structure de données *"Table de Hachage"* permet d'avoir un accès direct aux données alors qu'un accès dans une représentation arborescente nécessite un parcours de l'arbre (dans le pire des cas).

5.2 Expérimentation

Pour évaluer notre application, nous avons effectué des expérimentations selon le degré de difficulté des différents cas de figure. Nous les présentons brièvement ci-dessous en fonction du nombre de relations représentées par ligne, la structure des cases et la présence ou l'absence des attributs génériques :

1. Un tableau contenant une seule relation par ligne et une seule valeur par case : Dans cet exemple nous avons identifié une seule relation qui est *alimentAw(aliment, aw)* (voir le document SML généré *Annexe D.1*).
2. Un tableau contenant plusieurs relations par lignes et un attribut générique : Nous avons identifié quatre relations par ligne, auxquelles nous avons ajouté un attribut générique qui correspond à la cinquième colonne (il n'existe aucun *terme-attribut* qui correspond à alcool) (voir le document SML généré *Annexe D.2*).
3. Un tableau contenant une relation générique : Le schéma du tableau ne contient que des attribut génériques alors qu'aucune relation du schéma relationnel de l'ontologie n'est représentée. Donc une relation générique *relation* est créée (voir le document SML généré *Annexe D.3*).
4. Un tableau avec plusieurs valeurs par case : Dans ce tableau une case contient une liste de valeurs séparées par des virgules. Nous avons segmenté cette liste en autant d'attributs *aliment* que de valeurs et nous avons mis l'attribut *indProc* à la valeur *yes* (voir le document SML généré *Annexe D.4)* .

[1] Comme tout ce qui concerne le Web, la spécification de DOM [24] est gérée par le consortium *W3C (World Wide Web Consortium)*

[2] JAXP : Java API for XML Parsing

6 Conclusion

6.1 Résumé des contributions

Nous avons étudié dans ce travail les problèmes liés à l'enrichissement sémantique des informations structurées provenant du Web. Cette étude a abouti à la définition d'une DTD *Annexe C*, dans laquelle nous avons spécifié la nature et l'organisation des éléments apparaissant dans une instance de document SML. Cette DTD peut être générée automatiquement à partir de l'ontologie du domaine, ce qui garantit, entre autre, la généricité de notre approche. Dans les documents SML, l'imbrication des tags est compatible avec les signatures des relations du schéma relationnel de l'ontologie. En d'autres termes, chaque ligne du tableau d'origine est représentée par l'ensemble des relations sémantiques identifées dans le tableau. De plus, nous enrichissons la plupart des valeurs du tableau d'origine par des termes, recherchés dans l'ontologie du domaine et ayant une sémantique proche de celle de la valeur considérée. Ainsi, en cas d'ambiguïté , il est possible d'associer plusieurs termes à une valeur et d'associer plusieurs relations pour un même ensemble de valeurs. Comme nous nous sommes placés dans le cadre d'un enrichissement automatique de tableaux hétérogènes, guidé uniquement par l'ontologie, la DTD que nous avons définie permet d'avoir une représentation flexible des informations extraites.

Il est important de signaler que nous gardons la trace des traitements que nous effectuons sur les valeurs (e.g. test d'inclusion, segmentation, type de traductions) et des informations supplémentaires que l'on a extraites du document traité (e.g. le contexte du tableau d'origine, la catégorie des colonnes, le numéro de la colonne). Ces informations permettent de moduler le degré d'adéquation et de présenter des éléments de contexte à l'utilisateur, en cas d'incertitude.

Pour mettre en œuvre ce processus d'enrichissement, nous avons établi un algorithme (***XTab2SML***) permettant d'exploiter une ontologie du domaine pour transformer les informations stockées dans des documents XML au format *XTab* en documents XML au format *SML*.

L'approche que nous avons proposé est en cours d'expérimentation, dans le domaine du risque alimentaire, par le biais du prototype que nous avons développé en langage Java. Dans l'état actuel du prototype, nous arrivons à transformer en SML des documents XTab écrits en Français et contenant des tableaux avec une seule valeur par case. Nous avons établi une ébauche pour le traitement des tableaux avec plusieurs valeurs de même type par case. Notre application actuelle est capable d'appréhender des tableaux dont les lignes représentent une ou plusieurs relations.

Pour l'interrogation des documents SML, nous avons écrit dans le language *XQuery*, avec la collaboration de l'equipe BIA/INRA, des exemples de requêtes. Ces dernières concernent les deux vues de MIEL, prédéfinies (effet/facteur et charge microbienne) (voir le rapport intermédiaire d'e.dot [2] et exploitent quelques indicateurs de traitements ajoutés aux documents SML. Certaines de ces requêtes ont été déjà expérimentées par le moteur d'interrogation MIEL++ sur des documents SML que nous avons générés.

6.2 Discussion

L'idée de concevoir des outils d'extraction *automatique* d'information à déjà été abordée par un certain nombre de travaux : on trouve les deux systèmes RoadRunner [14] et ExALG [15] fondés sur la structure hiérarchique des documents HTML et qui supposent que les pages en entrée sont générées dynamiquement à partir d'une base de données. Ces systèmes sont inadaptés au traitement des documents HTML ne satisfaisant pas cette contrainte. Par rapport à ces deux systèmes, notre approche extrait des informations d'une manière complètement automatique et à partir de

documents de natures hétérogènes. Actuellement, nous traitons (par le module Any2SML) des informations contenues dans des documents (HTML, PDF et Excel). Le même traitement pourrait s'etendre pour appréhender d'autres formats de documents (e.g. DOC, JPEG, SVG ...).

Le système BYU est qualifié par *la résilience et l'adaptatibilité* dans [25]. On entend par *la résilience et l'adaptatibilité* la capacité au 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 la majorité des approches fondées sur l'induction de wrappers ([8], [10] et [9]) et pour les systèmes fondés sur la structure et sur les caractéristiques de mise en forme des document HTML comme RoadRunner, EXALG et IEPAD [16]. Le système BYU [20], fondé sur une ontologie du domaine, est capable d'extraire des informations à partir de documents HTML mais également à partir de documents textes. Toutefois, il n'est efficace que si l'ontologie est assez représentative du domaine visé.

Dans l'approche proposée par BYU, le schéma de représentation des données (schéma des tables relationnelles) est construit à partir de l'ontologie du domaine. Par rapport à leur approche, nous assurons que le schéma de représentation soit contenu dans l'ontologie du domaine (i.e. l'imbrication des tags du document SML est compatible avec les signatures des relations du schéma relationnel de l'ontologie), mais, de plus, nous *enrichissons sémantiquement* les données elles mêmes par les termes de l'ontologie. Les données que nous stockons sont donc des entités compréhensibles et interprétables d'une manière automatique par les applications.

Une des qualités de notre approche qu'il est important de mettre en évidence, est la capacité à conserver des éléments du contexte original (e.g. reconstruire le tableau XTab original) et toutes les interprétations potentielles et les indicateurs de traitements effectués.

Une autre innovation de notre approche, par rapport aux travaux cités précedemment, est le format de représentation de données que nous avons adopté qu'est le *format XML* (document SML). Ce choix de XML constitue une caractéristique potentiellement importante, vu la popularité et l'avenir prometteur de ce langage comme format d'échange et de représentation de données sur le Web.

6.3 Directions futures

Pour compléter le travail qui a été mis en œuvre jusqu'à présent, plusieurs améliorations méritent d'être réalisées. Nous en citons quelques unes ci-après, suivant trois axes :

Approfondissement et optimisation de la tâche de transformation XTab2SML : pour améliorer notre approche, plusieurs points nécessitent d'être traités :
- étude plus avancée des cas des listes de valeurs de même type et des listes de valeurs de types différents dans une case ;
- recherche d'outils linguistiques (e.g. WordNet : les synonymes, la catégorie grammaticale, etc.) et des techniques du *Traitement du Langage Naturel* pour optimiser la recherche de correspondances entre les valeurs du tableau et les termes de l'ontologie ;
- exploitation de la partie *contraintes sur les domaines d'attributs* de l'ontologie pour améliorer la recherche des catégories des colonnes ;
- étude du cas d'un ensemble de relations organisées hiérarchiquement. Par exemple dans le domaine du risque alimentaire avoir une relation *alimentFacteurMicroorganisme* qui généralise les relations : *alimentTempératureMicroorganisme, alimentChlorationMicroorganisme*, etc.

Implémentation pour assurer le bon fonctionnement de l'application et réaliser toutes les tâches définies dans l'étude du problème, plusieurs points restent à développer :
- intégration dans l'implémentation actuelle de la prise en compte des relations partiellement contenues dans les tableaux ;

- publication de l'application sous forme d'un service Web ;
- mise en interaction du service Web XTab2SML avec les différents services du projet e.dot (i.e. services Web *Any2XTab*, *GetOntology*[1] et *MIEL++*) ;
- exploitation des indicateurs de traitements pour filtrer les tableaux. En effet un document jugé comme pertinent par rapport au domaine visé par le module EDotFilter, peut contenir des tableaux n'ayant aucun rapport avec la thématique de l'entrepôt de données ;

Expérimentation pour une meilleure évaluation de notre approche, deux types d'expérimentations doivent être effectués :

- expérimentation du service XTab2SML sur un ensemble plus important de documents et sur des documents hétérogènes écrits en plusieurs langues ;
- expérimentation du service Web XTab2SML dans le cadre d'une thématique autre que le risque alimentaire, pour évaluer les carctéristiques de généricité et *d'adaptabilité* de notre approche.

[1] un service Web permettant aux autres services Web exploitant l'ontologie d'accéder aux parties de l'ontologie utilisées.

A L'ontologie Sym'Previus

A.1 Une représentation graphique du contenu de l'ontologie

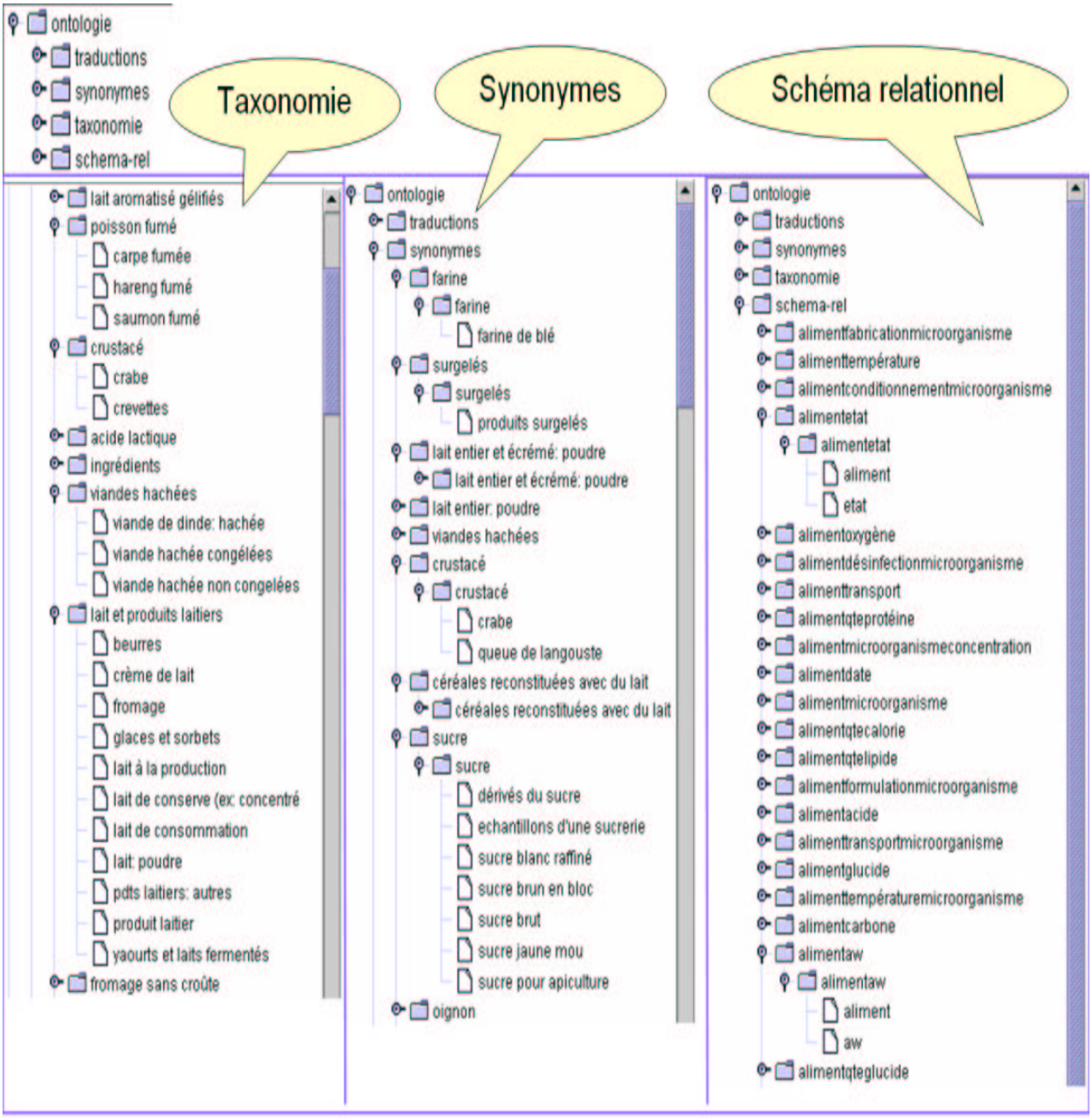


FIG. A.1 – Un extrait graphique de l'ontologie Sym'Previus

A.2 La DTD de l'ontologie

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT ontologie (taxonomie, synonymes, traductions, schema-rel, contraintes-domaines, schema-graphes)>
<!ELEMENT taxonomie (specialisation)+>
<!ELEMENT specialisation (terme, sous-termes)>
<!ELEMENT terme(#PCDATA)>
<!ELEMENT sous-termes (terme)*>
<!ELEMENT synonymes(synonyme)+>
<!ELEMENT synonyme (terme, synset)+>
<!ELEMENT synset (syn)*>
<!ELEMENT syn (#PCDATA)>
<!ELEMENT traductions (francais, anglais)*>
<!ELEMENT francais (#PCDATA)>
<!ELEMENT anglais (#PCDATA)>
<!ELEMENT schema-rel (signature)*>
<!ELEMENT signature (relation, attribut*)>
<!ELEMENT relation (#PCDATA)>
<!ELEMENT attribut (#PCDATA)>
<!ELEMENT contraintes-domaines (contrainte)*>
<!ELEMENT contrainte (attribut, (inter — enum))>
<!ELEMENT inter (born-inf, born-sup)>
<!ELEMENT born-inf (#PCDATA)>
<!ELEMENT born-sup (#PCDATA)>
<!ELEMENT enum (terme)*>
<!ELEMENT schema-graphes (concepts, relations, valeurs)>
<!ELEMENT concepts (concept)*>
<!ELEMENT relations (relation)*>
<!ELEMENT valeurs (valeur)*>
<!ELEMENT concept (#PCDATA)>
<!ELEMENT valeur (#PCDATA)>
```

A.3 Extrait de l'ontologie

```
<?xml version="1.0" encoding="UTF-8" ?>
<ontologie>
<taxonomie>
<specialisation>
<terme> Top </terme>
<sous-termes>
<terme> Aliment </terme>
<terme> Microorganisme </terme>
<terme> Facteur </terme>
</sous-termes>
</specialisation>
<specialisation>
<terme> Aliment </terme>
<sous-terme>
<terme> Denrées alimentaires </terme>
<terme> Environnement et divers </terme>
<terme> Milieu de culture </terme>
<terme> Nouveau produit à intégrer </terme>
</sous-terme> ....
</taxonomie>
/********** Fin de la partie taxonomie et début de la partie synonymes de l'ontologie **********/
<synonymes>
<synonyme>
<terme> Agneau : carcasse </terme>
<synset>
```

<syn> carcasse d'agneau </syn>
</synset>
</synonyme>
<synonyme>
<terme> Aliments pour animaux </terme>
<synset>
<syn> Aliments pour animaux a base de blé </syn>
<syn> Conserve de viande pour chien </syn>
<syn> paté </syn>
</synset>
</synonyme> ...
*/********** Fin de la partie synonymes et début de la partie traductions de l'ontologie **********/*
</synonymes>
<traductions>
<francais>Abats</francais>
<anglais>offal</anglais>
<francais>Air</francais>
<anglais>Air</anglais>
<francais>Aliment hospitalier</francais>
<anglais>Hospitable food</anglais>
<francais>Aliments pour animaux</francais>
<anglais>Animal Food Product</anglais>
....
</traductions>
*/********** Fin de la partie traductions de l'ontologie et début de la partie schéma relationnel **********/*
<schema-rel>
<signature>
<relation>AlimentPh</relation>
<attribut>aliment</attribut>
<attribut>Ph</attribut>
</signature>
<signature>
<relation>AlimentAw</relation>
<attribut>aliment</attribut>
<attribut>Aw</attribut>
</signature>
<signature>
<relation>AlimentMicroorganisme</relation>
<attribut>aliment</attribut>
<attribut>microorganisme</attribut>
</signature>
</schema-rel>
</ontologie>
</ontologie>

B Document XTab

B.1 La DTD XTab

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- ********** le contenu du tableau ********** -->
<!ELEMENT tableau (titre, nb-col, contenu,origine,meta-data ?)>
<!ELEMENT titre (titre-tableau ?, titre-col+)>
<!ELEMENT nb-col ANY>
<!ELEMENT contenu (ligne+)>
<!ELEMENT titre-tableau ANY>
<!ELEMENT titre-col ANY>
<!ELEMENT ligne (case+)>
<!ELEMENT case ANY>
<!-- ********** informations supplémentaires ********** -->
<!ELEMENT origine (URI, information ?, plan ?)>
<!ELEMENT URI (#PCDATA)>
<!ELEMENT information (titre-publi ?, auteur ?, journal ?, annee ?)>
<!ELEMENT titre-publi ANY>
<!ELEMENT auteur (nom-auteur*)>
<!ELEMENT journal ANY>
<!ELEMENT annee ANY>
<!ELEMENT nom-auteur ANY>
<!ELEMENT plan (resume ?, introduction ?, mat-methode ?, resultat ?, discussion ?, conclusion ?, remerciement ?, reference ?, autre ?)>
<!ELEMENT resume ANY>
<!ELEMENT introduction ANY>
<!ELEMENT mat-methode ANY>
<!ELEMENT resulat ANY>
<!ELEMENT discussion ANY>
<!ELEMENT conclusion ANY>
<!ELEMENT remerciement ANY>
<!ELEMENT reference ANY>
<!ELEMENT autre ANY>
<!-- ********** -->
<!ENTITY % metaData SYSTEM './metaData.dtd'>
%metaData;
```

Le contenu du fichier des méta-données liées au domaine du risque alimentaire :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!- la DTD de l element meta-data du document XTAB ->
<!ELEMENT meta-data (aliment ?, facteur ?, micoorganisme ?)>
<!ELEMENT aliment ((nom, probabilite)+)>
<!ELEMENT facteur ((nom, probabilite)+)>
<!ELEMENT microorganisme ((nom, probabilite)+)>
<!ELEMENT nom ANY>
<!ELEMENT probabilite ANY>
```

B.2 Exemple de document XTab

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<tableau><titre>
<titre-tableau>The typical water activity of some foodstuffs</titre-tableau>
<titre-col>Type of aliment</titre-col>
```

<titre-col>Water Activity aw</titre-col></titre>
<nb-col>2</nb-col>
<contenu>
<ligne>
<case>Viandes fraîches et poisson</case>
<case>0.99</case>
</ligne>
<ligne>
<case>Pain</case>
<case>0.95</case>
</ligne>
<ligne>
<case>cheddar aggé</case>
<case>0.85</case>
</ligne>
<ligne>
<case>gelé et confiture</case>
<case>0.8</case>
</ligne>
<ligne>
<case>Pudding de prune</case>
<case>0.8</case>
</ligne>
<ligne>
<case>Fruit sec</case>
<case>0.6</case>
</ligne>
<ligne>
<case>Biscuits</case>
<case>0.3</case>
</ligne>
</contenu>
<URI>http ://www.foodscience.afisc.csiro.au/waterfs.htm </URI>
</tableau>

C La DTD SML (Semantic Markup Language)

```
<?xml version="1.0" encoding="UTF-8" ?>

<!ELEMENT tableau (id, titre, nb-col, contenu)>
<!ELEMENT id ANY>
<!ELEMENT titre (titre-tableau ?, titre-col+)>
<!ELEMENT nb-col ANY>
<!ELEMENT titre-tableau ANY>
<!ELEMENT titre-col ANY>

<!ELEMENT contenu ( ligneRel+)>

<!ATTLIST contenu tabValide (yes | no) #REQUIRED "no">

<!ELEMENT ligneRel ( alimentPh ?, alimentAw ?,alimentEtat ?,alimentStructure ?, alimentMicroorganismeConcentration ?, alimentFacteurMicroorganisme ?, alimentFacteur ?, alimentMicroorganisme ?, alimentCalorie ?, alimentProteine ?, alimentGlucide ?, alimentLipide ?, alimentAcide ?,alimentVitamine ?, alimentQantitéCalorie ?, alimentQantitéProteine ?, alimentQantitéGlucide ?, alimentQantitéLipide ?, alimentQantitéAcide ?, alimentCarbone ?, alimentOxygne ?, alimentAzote ?, relation ?)>

<!ATTLIST lingeRel additionalAttr (yes | no) #REQUIRED >

<!ELEMENT alimentPh (aliment | listeAliments,pH,attribut*)>
<!ELEMENT alimentAw (aliment | listeAliments,Aw,attribut*)>
<!ELEMENT alimentEtat (aliment | listeAliments,etat,attribut*)>
<!ELEMENT alimentStructure (aliment | listeAliments,structure,attribut*)>
<!ELEMENT alimentMicroorganismeConcentration (aliment | listeAliments,microorganisme, concentration,attribut*)>
<!ELEMENT alimentFacteurMicroorganisme (aliment | listeAliments, facteur, microorganisme, attribut*)>
<!ELEMENT alimentFacteur (aliment | listeAliments, facteur, attribut*)>
<!ELEMENT alimentMicroorganisme (aliment | listeAliments,microorganisme, attribut*)>
<!ELEMENT alimentCalorie (aliment | listeAliments,calorie,attribut*)>
<!ELEMENT alimentProteine (aliment | listeAliments,proteine,attribut*)>
<!ELEMENT alimentGlucide (aliment | listeAliments,glucide,attribut*)>
<!ELEMENT alimentLipide (aliment | listeAliments,lipide,attribut*)>
<!ELEMENT alimentAcide (aliment | listeAliments,acide,attribut*)>
<!ELEMENT alimentVitamine (aliment | listeAliments,vitamine,attribut*)>
<!ELEMENT alimentQantitéCalorie (aliment | listeAliments,quantité, calorie,attribut*)>
<!ELEMENT alimentQantitéProteine(aliment | listeAliments,quantité, proteine,attribut*)>
<!ELEMENT alimentQantitéGlucide(aliment | listeAliments,quantité, glucide,attribut*)>
<!ELEMENT alimentQantitéLipide(aliment | listeAliments,quantité, lipide,attribut*)>
<!ELEMENT alimentQantitéAcide(aliment | listeAliments,quantité, acide,attribut*)>
<!ELEMENT alimentCarbone (aliment | listeAliments,carbone,attribut*)>
<!ELEMENT alimentOxygne (aliment | listeAliments,oxygene,attribut*)>
<!ELEMENT alimentAzote (aliment | listeAliments,azote,attribut*)>
<!ELEMENT relation (attribut*)>

<!ELEMENT listeAliments (aliment+)>
<!ELEMENT aliment (finalVal, origineVal)>
<!ELEMENT ph (finalVal, origineVal)>
<!ELEMENT aw (finalVal, origineVal)>
<!ELEMENT microorganisme (finalVal, origineVal)>
<!ELEMENT calorie (finalVal, origineVal)>
```

<!ELEMENT proteine (finalVal, origineVal)>
<!ELEMENT glucide (finalVal, origineVal)>
<!ELEMENT lipide (finalVal, origineVal)>
<!ELEMENT acide (finalVal, origineVal)>
<!ELEMENT quantité (finalVal, origineVal)>
<!ELEMENT carbone (finalVal, origineVal)>
<!ELEMENT oxygne (finalVal, origineVal)>
<!ELEMENT azote (finalVal, origineVal)>
<!ELEMENT etat (finalVal, origineVal)>

<!ELEMENT facteur (nomFacteur, finalVal, origineVal)>
<!ELEMENT nomFacteur (finalVal, origineVal)>

<!ELEMENT finalVal (#PCDATA)>
<!ELEMENT origineVal (#PCDATA)>

<!ELEMENT attribut (finalVal, origineVal)>

<!ATTLIST attribut indProc(yes | no) #REQUIRED
indOnto (complete | inclusion | intersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indMatch CDATA #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST aliment indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST ph indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST aw indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST micoorganisme indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST calorie indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST proteine indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST glucide indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST lipide indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST acide indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST quantité indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST carbone indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST oxygène indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST azote indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST état indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED

indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST facteur indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

<!ATTLIST nomFacteur indProc (yes | no) #REQUIRED
indOnto (complete | inclusion | inersection | notFound) #REQUIRED
indCat CDATA #REQUIRED
indTrans (interne | externe | mixte | notRequired) #REQUIRED
indNumCol CDATA #REQUIRED
indWordsIntersection CDATA #IMPLIED>

D Expérimentation du prototype *XTab2SML*

D.1 Tableau avec une seule relation et une seule valeur par case

<?xml version="1.0" encoding="UTF-8" ?> <tableau>
<titre>
<titre-tableau>The typical water activity of some foodstuffs</titre-tableau>
<titre-col>type of aliment</titre-col>
<titre-col>water activity aw</titre-col>
</titre>
<nb-col>2</nb-col>
<contenu>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="inclusion" indProc="yes" indTrans="none">
<finalVal>viande</finalVal>
<finalVal>poisson</finalVal>
<origineVal>Viande fraîche et poisson</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.99</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>Pain</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.95</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="inclusion" indProc="yes" indTrans="none">
<finalVal>cheddar</finalVal>
<origineVal>cheddar agé</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.85</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="intersection" indProc="yes" indTrans="none">
<finalVal>confitures et compotes</finalVal>

<origineVal>Gelées et confitures</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.8</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
<finalVal>Pudding de prunes</finalVal>
<origineVal>Pudding de prunes</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.8</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="inclusion" indProc="yes" indTrans="none">
<finalVal>fruits et légumes secs ou séchés</finalVal>
<origineVal>Fruits secs</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.6</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="inclusion"
indProc="yes" indTrans="none">
<finalVal>biscottes et biscuits</finalVal>
<origineVal>Biscuits</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.3</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentaw>
<aliment indCat="aliment" indOnto="inclusion"
indProc="yes" indTrans="none">
<finalVal>lait en poudre stérile et reconstitué</finalVal>
<finalVal>lait en poudre stérile et reconstitué</finalVal>
<origineVal>Poudre de lait</origineVal>
</aliment>
<aw indCat="notFound" indOnto="notFound" indProc="no" indTrans="none">
</finalVal>
<origineVal>0.2</origineVal>
</aw>
</alimentaw>
</ligneRel>
<ligneRel additionnalAttr="no">

<alimentaw>
<aliment indCat=”notFound” indOnto=”notFound” indProc=”no” indTrans=”none”>
<finalVal>Café instantné</finalVal>
<origineVal>Café instantné</origineVal>
</aliment>
<aw indCat=”notFound” indOnto=”notFound” indProc=”no” indTrans=”none”>
</finalVal>
<origineVal>0.2</origineVal>
</aw>
</alimentaw>
</ligneRel>
</contenu>
</tableau>

D.2 Tableau avec plusieurs relations contenant des attributs génériques

<?xml version=”1.0” encoding=”UTF-8” ?>
<tableau>
<titre>
<titre-tableau>Don’t have title</titre-tableau>
<titre-col>item</titre-col>
<titre-col>proteine</titre-col>
<titre-col>glucide</titre-col>
<titre-col>lipide</titre-col>
<titre-col>alcool</titre-col>
<titre-col>calorie</titre-col>
</titre>
<nb-col>6</nb-col>
<contenu>
<ligneRel additionnalAttr=”yes”>
<alimentproteine>
<aliment indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>compote rhubarbe</finalVal>
<origineVal>compote rhubarbe</origineVal>
</aliment>
<proteine indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0,4 g</finalVal>
<origineVal>0,4 g</origineVal>
</proteine>
<attribut indOnto=”notFound” indMatch=”attribut” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentproteine>
<alimentglucide>
<aliment indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>compote rhubarbe</finalVal>
<origineVal>compote rhubarbe</origineVal>
</aliment>
<glucide indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>31,6 g</finalVal>
<origineVal>31,6 g</origineVal>
</glucide>
<attribut indOnto=”notFound” indMatch=”attribut” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentglucide>

<alimentlipide>
<aliment indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>compote rhubarbe</finalVal>
<origineVal>compote rhubarbe</origineVal>
</aliment>
<lipide indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0,1 g</finalVal>
<origineVal>0,1 g</origineVal>
</lipide>
<attribut indOnto="notFound" indMatch="attribut" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentlipide>
<alimentcalorie>
<aliment indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>compote rhubarbe</finalVal>
<origineVal>compote rhubarbe</origineVal>
</aliment>
<calorie indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>129 Kcal </finalVal>
<origineVal>129 Kcal </origineVal>
</calorie>
<attribut indOnto="notFound" indMatch="attribut" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentcalorie>
</ligneRel>
<ligneRel additionnalAttr="yes">
<alimentproteine>
<aliment indOnto="inclusion" indCat="aliment" indTrans="none" indProc="yes">
<finalVal>fruits</finalVal>
<finalVal>salade</finalVal>
<origineVal>salade de fruits</origineVal>
</aliment>
<proteine indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0,3 g</finalVal>
<origineVal>0,3 g</origineVal>
</proteine>
<attribut indOnto="notFound" indMatch="attribut" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentproteine>
<alimentglucide>
<aliment indOnto="inclusion" indCat="aliment" indTrans="none" indProc="yes">
<finalVal>fruits</finalVal>
<finalVal>salade</finalVal>
<origineVal>salade de fruits</origineVal>
</aliment>
<glucide indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal>25 g</finalVal>
<origineVal>25 g</origineVal>
</glucide>
<attribut indOnto="notFound" indMatch="attribut" indCat="notFound" indTrans="none" indProc="no">
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentglucide>

<alimentlipide>
<aliment indOnto=”inclusion” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>fruits</finalVal>
<finalVal>salade</finalVal>
<origineVal>salade de fruits</origineVal>
</aliment>
<lipide indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0,1 g</finalVal>
<origineVal>0,1 g</origineVal>
</lipide>
<attribut indOnto=”notFound” indMatch=”attribut” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentlipide>
<alimentcalorie>
<aliment indOnto=”inclusion” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>fruits</finalVal>
<finalVal>salade</finalVal>
<origineVal>salade de fruits</origineVal>
</aliment>
<calorie indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>102 Kcal </finalVal>
<origineVal>102 Kcal </origineVal>
</calorie>
<attribut indOnto=”notFound” indMatch=”attribut” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal>0 g</finalVal>
<origineVal>0 g</origineVal>
</attribut>
</alimentcalorie>
</ligneRel>
... </contenu>
</tableau>

D.3 Tableau avec une relation générique

< ?xml version=”1.0” encoding=”UTF-8” ?>
<tableau>
<titre> <titre-tableau/>
<titre-col>Teneur en graisse (%)</titre-col>
<titre-col>L’atmosphère</titre-col>
<titre-col>La température 11ÂC.W0 de stockage</titre-col>
<titre-col>La température 11ÂC.D de stockage</titre-col>
</titre> <nb-col>4</nb-col>
<contenu>
<ligneRel additionalAttr=”yes”>
<relation>
<attribut indCat=”notFound” indOnto=”notFound”
indProc=”no” indTrans=”none” indMatch=”attribut”>
<finalVal/>
<origineVal/>
</attribut>
<attribut indCat=”aliment” indOnto=”complete” indProc=”no” indTrans=”none” indMatch=”attribut”>
<finalVal>air </finalVal>
<origineVal>Air</origineVal>
</attribut>
<attribut indCat=”notFound” indOnto=”notFound” indProc=”no” indTrans=”none” indMatch=”température”>
<finalVal/>

<origineVal>1.25 </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>
<origineVal>0.10 </origineVal>
</attribut>
</relation>
</ligneRel>
<ligneRel additionalAttr="yes">
<relation>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="attribut">
<finalVal/>
<origineVal>17.8 </origineVal>
</attribut>
<attribut indCat="facteur" indOnto="inclusion" indProc="yes" indTrans="none" indMatch="attribut">
<finalVal>co2 </finalVal>
<origineVal>15% CO2 </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>
<origineVal>0.97 </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>
<origineVal>0.07 </origineVal>
</attribut>
</relation>
</ligneRel>
<ligneRel additionalAttr="yes">
<relation>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="attribut">
<finalVal/>
<origineVal/>
</attribut>
<attribut indCat="facteur" indOnto="inclusion" indProc="yes" indTrans="none" indMatch="attribut">
<finalVal>co2 </finalVal>
<origineVal>30% CO2 </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>
<origineVal>0.50 </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>
<origineVal>0.02 </origineVal>
</attribut>
</relation>
</ligneRel>
<ligneRel additionalAttr="yes">
<relation>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="attribut">
<finalVal/>
<origineVal/>
</attribut>
<attribut indCat="aliment" indOnto="complete" indProc="no" indTrans="none" indMatch="attribut">
<finalVal>air </finalVal>
<origineVal>Air </origineVal>
</attribut>
<attribut indCat="notFound" indOnto="notFound" indProc="no" indTrans="none" indMatch="température">
<finalVal/>

<origineVal>1.24 </origineVal>
</attribut>
<attribut indCat=”notFound” indOnto=”notFound” indProc=”no” indTrans=”none” indMatch=”température”>
<finalVal/>
<origineVal>0.10 </origineVal>
</attribut>
</relation>
</ligneRel>
... </contenu>
</tableau>

D.4 Tableau avec une liste de valeurs par case

<tableau>
<titre>
<titre-tableau>Tableau approximatif du PH des aliments</titre-tableau>
<titre-col>produit</titre-col>
<titre-col>Valeur de PH</titre-col>
</titre>
<nb-col>2</nb-col>
<contenu>
<ligneRel additionnalAttr=”no”>
<alimentph>
<**listeAliments**>
<aliment indOnto=”complete” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>fromage</finalVal>
<origineVal>fromage</origineVal>
</aliment>
<aliment indOnto=”complete” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>viande</finalVal>
<origineVal>viande</origineVal>
</aliment>
<aliment indOnto=”complete” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>poisson</finalVal>
<origineVal>poisson</origineVal>
</aliment>
<aliment indOnto=”complete” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>carottes</finalVal>
<origineVal>carottes</origineVal>
</aliment>
<aliment indOnto=”inclusion” indCat=”aliment” indTrans=”none” indProc=”yes”>
<finalVal>pommes de terre pelées et fraîches</finalVal>
<finalVal>pommes de terre, pelées, crues, sous-vide</finalVal>
<origineVal>pommes</origineVal>
</aliment>
<origineValListe>fromage, viande, poisson, carottes, pommes <origineValListe>
</**listeAliments**>
<ph indOnto=”notFound” indCat=”notFound” indTrans=”none” indProc=”no”>
<finalVal/>
<origineVal>0.6</origineVal>
</ph>
</alimentph>
</ligneRel>
<ligneRel additionnalAttr=”no”>
<alimentph>
<aliment indOnto=”complete” indCat=”aliment” indTrans=”none” indProc=”no”>
<finalVal>riz</finalVal>
<origineVal>riz</origineVal>

</aliment>
<ph indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal/>
<origineVal>0.9</origineVal>
</ph>
</alimentph>
</ligneRel>
<ligneRel additionnalAttr="no">
<alimentph>
<aliment indOnto="complete" **indCat="microorganisme"** indTrans="none" indProc="no">
<finalVal>levures</finalVal>
<origineVal>levures</origineVal>
</aliment>
<ph indOnto="notFound" indCat="notFound" indTrans="none" indProc="no">
<finalVal/>
<origineVal>0.4</origineVal>
</ph>
</alimentph>
</ligneRel>
</contenu>
</tableau>

E Requêtes d'interrogation de MIEL++

E.1 Représentation des trois vues de MIEL++

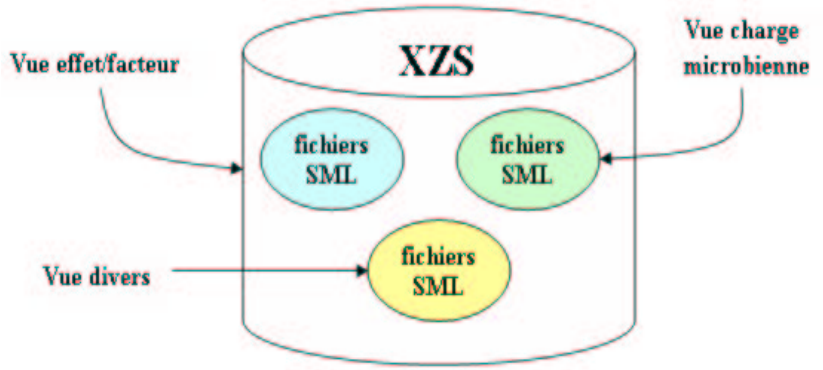


FIG. E.1 – Une représentation des trois vues de MIEL++

E.2 Requête sur la vue MIEL++ *Aliment/effet du facteur analysé*

L'utilisateur saisit : **aliment : Chou, ph : 5, facteur : Température.**
La requête correspondante se représente sous forme d'arbre de la manière suivante (CF. FIG.E.2) :

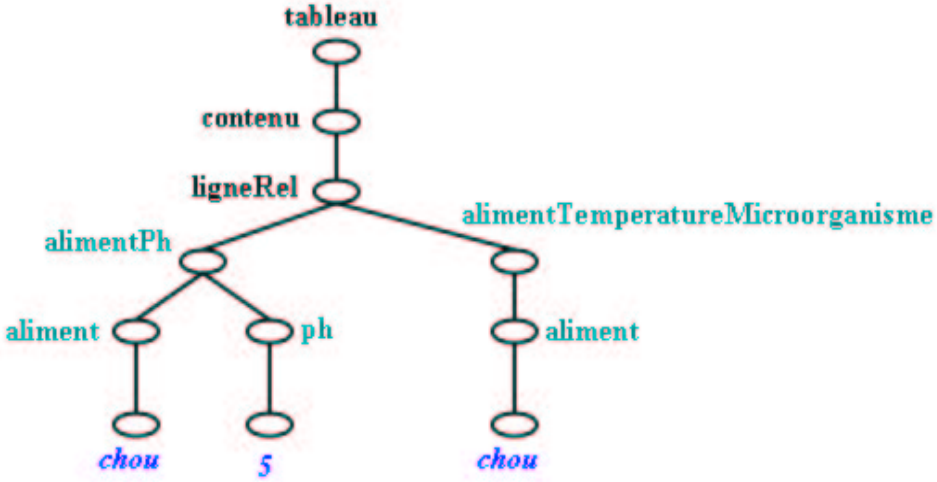


FIG. E.2 – Représentation arborescente de la requête *Aliment/effet du facteur analysé*

E.3 Requête sur la vue MIEL++ *charge microbienne*

L'utilisateur saisit : **Aliment : chou, Ph : 5, Microorganisme : Listeria M.**
La requête correspondante se représente sous forme d'arbre de la manière suivante (CF. FIG.E.4) :

```
Let $aliment:=chou
Let $microorganisme:= Listeria Monocytogenes
Let $ph:=5
For $contenu IN tableau
    Return
        <tableau>
        <titre>$contenu/ancestor()//titre-tableau</titre>
        {
        For $ligne IN $contenu/ligneRel
         Let $alimentDonnee := $ligne /AlimentMicroorganismeConcentration
         /listeAliments/aliment/finalVal
         Let
         $microorganismeDonnee:=$ligne/AlimentMicroorganismeConcentration/
         microorganisme/finalVal
         Let $phDonnee:=$ligne/AlimentPh/ph/finalVal

        Where(min, ad($aliment,alimentDonnee),
        ad($microorganisme,$microorganismeDonnee),
        ad($ph,$phDonnee))> 0)

        Return <ligne>
                    $ligne
                     <ad>
                        min (ad($aliment,alimentDonnee),ad($ph,$phDonnee))
                     </ad>
                </ligne>
        }
        </tableau>
```

FIG. E.3 – La requête XQuery *Aliment/effet du facteur analysé*

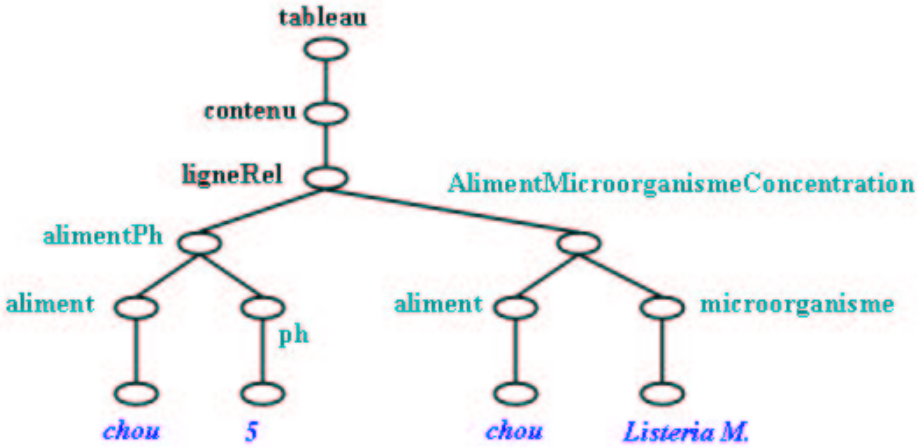


FIG. E.4 – Représentation arborescente de la requête *Charge microbienne*

```
Let $aliment:=chou
Let $microorganisme:=Listeria M.
For $contenu IN tableau
    Return
        <tableau>
        <titre>$contenu/ancestor()//titre-tableau</titre>
        {
            For $ligne IN $contenu/ligneRel
            if ($ligne/@additionalAttr:="yes") then
           if($ligne//attribut/@match !="attribut") then
                    Let categorie:= ($ligne//attribut/@match
                    Let catIdent="yes"
               else
                    Let catIdent="no"

                 Let $donneeGen:=$ligne//attribut
            Let typeRes:="generique"
   else
              Let $donneeGen:=()
              Let typeRes:="notGenerique"

 Let $alimentDonnee:=$ligne/AlimentMicroorganismeConcentration/aliment
 Let
$microorganismeDonnee:=$ligne/AlimentMicroorganismeConcentration/microor
ganisme

 Where(min(ad($aliment,$alimentDonnee),
 ad($microorganisme,$microorganismeDonnee))> 0

 Return
            <ligne> $ligne  </ligne>
            if ($typeRes :="generique") then
                 <donneeSupp>
                $donneeGen
                 </donneeSupp>
            if ($catIdent := "yes") then
                <categorie>
                $categorie
                </categorie>
       }
        </tableau>
```

FIG. E.5 – la requête XQuery *Charge microbienne*

Bibliographie

[1] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6) :907–928, 1995.

[2] www.telecom.gouv.fr/rntl/projet/posters-pdf/rntl-poster-e-dot.pdf, www-rocq.inria.fr/gemo/.

[3] http ://www.symprevius.net.

[4] Gio Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3) :38–49, 1992.

[5] Goasdoue F. *Réécriture de requêtes en termes de vues dans CARIN et intégration d'informations.* PhD thesis, Université de Paris XI, Orsay, novembre 2001.

[6] Surajit Chaudhuri and Umeshwar Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1) :65–74, 1997.

[7] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 119–128. Morgan Kaufmann Publishers Inc., 2001.

[8] Nicholas Kushmerick. Wrapper induction : efficiency and expressiveness. *Artif. Intell.*, 118(1-2) :15–68, 2000.

[9] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1-2) :93–114, 2001.

[10] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst.*, 23(9) :521–538, 1998.

[11] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steven Minton, Andrew Philpot, and Cyrus Shahabi. The theaterloc virtual application. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 980–987. AAAI Press / The MIT Press, 2000.

[12] Georgios P. Constantine D. S. Georgios S. and Michalis H. Mining web sites using wrapper induction, named entities ans post-processing. Athens, Greece, 2003. Proceedings of the International Workshop Tutorial on Adpative Text Extraction and Mining.

[13] Fabio Ciravegna, Alexiei Dingli, Yorick Wilks, and Daniela Petrelli. Adaptive information extraction for document annotation in amilcare. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 451–451. ACM Press, 2002.

[14] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Automatic web information extraction in the roadrunner system. In *Revised Papers from the HUMACS, DASWIS, ECOMO, and DAMA on ER 2001 Workshops*, pages 264–277. Springer-Verlag, 2002.

[15] Arvind Arasu, Hector Garcia-Molina, and Stanford University. Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM Press, 2003.

[16] Chia-Hui Chang and Shao-Chen Lui. Iepad : information extraction based on pattern discovery. In *Proceedings of the tenth international conference on World Wide Web*, pages 681–688. ACM Press, 2001.

[17] Chia-Hui Chang and Shih-Chien Kuo. Olera : A semi-supervised approach for web data extraction with visual support. In *IEEE Intelligent Systems*, 2004.

[18] M-C. Rousset A. Termier and M. Sebag. Treefinder : a first step towards xml data mining. In *International Conderence of Data Mining (ICDM)*, 2002.

[19] M-C. Rousset A. Termier and M. Sebag. a new approach for discovering closed frequent trees in heterogeneous tree dadabases. In *International Conderence of Data Mining (ICDM)*, 2004.

[20] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng.*, 31(3) :227–251, 1999.

[21] Patrice Buche, Juliette Dibie-Barthélemy, Ollivier Haemmerlé, and Mounir Houhou. Towards flexible querying of xml imprecise data in a dataware house opened on the web. In *Flexible Query Answering Systems (FQAS)*. Springer Verlag, june 2004.

[22] Sowa J. F. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[23] Sowa J. F. *Knowledge Representation : Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., 1999.

[24] http ://www.w3c.org/dom.

[25] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec.*, 31(2) :84–93, 2002.