SIRIUS : Un Moteur de Recherche d'Informations Contextuelles dans les Bases de Documents XML.

G.Ménier, P.F Marteau, E. Le Saux, J. Revault
Equipage, VALORIA
Université de Bretagne Sud
{prenom.nom}@univ-ubs.fr

Résumé : Sirius est un moteur de recherche exploitant la structuration XML de documents hétérogènes. L'organisation des données met en avant une indexation adaptée à la prise en compte de contextes différents, définis à la fois par le contenu informationnel du document et l'organisation hiérarchique de celui-ci. Le mécanisme de recherche est basé sur une mise en correspondance d'arbres de contextes qui exploite une distance d'édition de type Levenstein associée à une heuristique de fusion d'information.

Mots clé :
XML, recherche documentaire, fusion de données, optimisation, distance d'édition, TALN

1. Introduction
L'émergence du standard XML est très significative de l'importance accordée à la normalisation des mécanismes de représentation de l'information. Dans cette optique, la réalisation de moteurs de recherches capables d'associer les méthodes classiques d'indexation à des méthodes de fusion de données devient absolument cruciale. Les bases documentaires seront en effet, à terme, non seulement composées de documents de sources et de natures très différentes, mais encore et surtout devront gérer l'information suivant des critères d'appréciation et d'indexation très hétérogènes. Des systèmes de bases de données XML sont développés pour répondre à cette demande émergente [Zhao, 99], [Xhive], etc. Afin de permettre une mise en œuvre rapide et une évaluation de différentes stratégies de prise en compte du contexte lors des recherches d'information, nous développons un moteur adapté, SIRIUS, dont les principes sont décrits dans ce document.

2. Organisation générale
L'organisation générale du système est donnée dans la figure 1. Lors de la phase d'indexation, la structure des documents XML est extraite pour construire les listes inverses. Les contextes XML sont référencés et codés en vue d'une analyse lors de la recherche d'information. Les requêtes sont exprimées dans un langage spécialisé et enrichies lors de l'analyse par un mécanisme d'enrichissement terminologique et linguistique. La partie 'structurelle' de la requête est ensuite comparée aux structures référencées à l'aide d'un mécanisme de calcul de distances inspiré des techniques de calcul de distances d'édition. Une matrice de coût adaptée permet d'évaluer une mise en correspondance d'éléments et attributs XML à l'aide d'informations éventuellement également fournies par le système d'enrichissement terminologique.

3. Principes d'indexation
Un document XML peut être représenté par un arbre, dont les feuilles contiennent les informations textuelles (TEXT et CDATA) et dont les nœuds sont associés à des éléments XML valués par des attributs. Cet arbre est extrait d'un document XML à l'aide d'un « parseur » qui s'appuie sur un modèle de document (DOM Document Object Model) [Xerces]. Chaque mot du texte appartient donc à une feuille particulière de l'arbre DOM et

hérite du chemin des nœuds de la racine vers la feuille, ou encore d'une séquence ordonnée d'éléments XML (associée aux attributs correspondants).

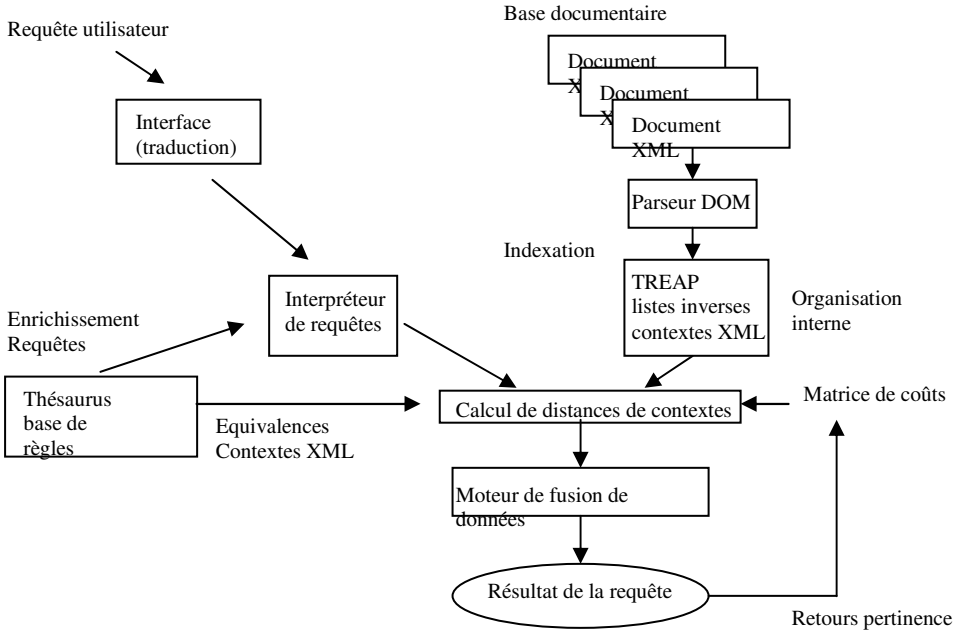

Figure 1 : Organisation générale de SIRIUS

Dans la suite de ce document, nous appellerons donc 'contexte XML' la séquence des noms d'éléments XML associés à leurs attributs. Lors d'une indexation totale des documents, le moteur répertorie les contextes XML différents rencontrés et associe les mots indexés à (principalement) trois éléments d'informations :

- un lien vers l'URI du document : ce lien permet de retrouver le document indexé.

- un indexe permettant de situer le mot dans le texte : cet indexe permet de retrouver rapidement une région du document contenant le texte et de proposer à l'utilisateur un extrait significatif autour du mot ou de l'élément indexé.

- un lien vers un contexte répertorié : ce lien permet, indépendamment de la nature de l'information indexée, de garder une trace sur la position dans l'arbre XML sans avoir à « re-parser » le document complet. La réalisation des listes inverses, résultantes du processus d'indexation, fait appel à une structure d'arbre de recherche binaire aléatoire nommée TREAP [Seidel, 96].

4. Langage de requêtes

Afin de permettre une prise en compte optimale des informations de contexte XML, nous avons développé un langage de requêtes dédié. L'interface de saisie transforme la requête de l'utilisateur vers le langage interne de SIRIUS : ce mécanisme permet d'affranchir le moteur de recherche de l'interface de saisie. La gestion des opérations ensemblistes se fait à l'aide d'opérateurs booléens classiques, des heuristiques permettant d'introduire des notions de présences incertaines des unités linguistiques dans les éléments XML ou des opérateurs de séquence ou de proximité. Ces opérateurs n-aires sont exploités par un mini interpréteur dédié post-fixé qui permet d'analyser les requêtes, par exemple :

(and (or critère$_1$ critère$_2$ … critère$_n$) …)

La définition des critères peut se restreindre aux termes indexés, par exemple :

(and FRANCE (or BELGIQUE SUISSE))

dans ce cas, le contexte XML n'est pas pris en compte et le système se comporte comme un système d'indexation en texte intégral (*full text)* [Loupy et al., 98a], [Salton 89, Salton et al.94], [VanRijsbergen, 1979] . Les contextes XML sont représentés de la manière suivante :

/ élément$_1$ / élément$_2$ / élément$_3$ /…/ élément$_n$ /

pour un chemin complet partant de la racine. Le mot réservé '*' correspond à une partie manquante. Par exemple :

/ * / élément$_i$ / * / élément$_j$ / * /

représente un contexte XML contenant (dans l'ordre) les éléments 'élément$_i$' et 'élément$_j$'.
Si les attributs ne sont pas mentionnés, ils sont tout simplement ignorés. Dans le cas contraire, il est possible de définir des opérations de comparaison. Par exemple :

/* / livre (or (== isbn 34567) (>= isbn 6000)] /

définit un contexte XML de 'dernier' élément 'livre' possédant un attribut 'isbn' de valeur 34567 ou supérieure à 6000 (les opérateurs de comparaison numérique tentent automatiquement une conversion numérique et sinon, utilisent l'ordre alphabétique).

Un certain nombre d'opérateurs spécialisés permet d'évaluer des propriétés spéciales sur les attributs, par exemple : (! attribut) permet de tester l'existence d'un attribut, la variable $ATTR donne le nombre d'attributs de l'élément, etc.

A titre d'exemple, la requête de recherche des documents contenant une balise <AUTEUR NOM='wells'> ou le terme 'martien' dans un élément quelconque <CHAPITRE> s'exprime de la manière suivante :

(or /*/auteur(== nom wells)/
/*/chapitre/martien)

Notez l'absence de '/' après 'martien' indiquant la présence d'un terme du document (et non pas du contexte). Nous nommerons 'critère de recherche XML' la chaîne associant la définition d'un contexte XML (éventuellement incomplet, avec ou sans attributs et opérations sur les attributs) avec un terme (éventuellement absent).

5. Enrichissements terminologique et linguistique de la requête

La prise en compte des connaissances linguistiques et terminologiques dans les modèles de contenu documentaire peut s'effectuer à différents niveaux d'analyse, en particulier grâce à l'utilisation de connaissances d'ordre lexical, syntaxique, sémantique ou encore terminologique. Nous examinons ici les principes de prise en compte de ces connaissances dans SIRIUS.

5.1 Etiquetage grammatical et lemmatisation

L'utilisation d'un outil d'étiquetage grammatical dans un système de traitement de l'information textuelle conduit à des gains de pertinence importants, [Hull et al., 96], principalement parce que la connaissance d'un mot et de sa catégorie grammaticale permet de connaître, de manière quasi-systématique, son lemme, c'est-à-dire sa racine [El Bèze, 90]

[Krovetz, 93]. Ce rattachement du mot à sa racine permet de tenir compte des variations morphologiques et par là même de résoudre certains types d'ambiguïtés qui peuvent par exemple exister entre substantifs et verbes (‘avions’, ‘portes’ par exemple). La mise en rapport des termes engendrés à partir d'une même racine, par différentes flexions grammaticales, permet d'éviter une diminution importante du taux de silence des recherches, au prix d'une légère baisse de la précision [Krovetz, 93], [Riloff, 95]. Chaque lemme peut alors simplement être considéré comme une unité d'information élémentaire dans le modèle proposé. Le degré d'appartenance de ce type d'unité linguistique à un élément textuel est une variable binaire (un lemme est présent ou absent d'une unité documentaire), ce qui se justifie dans la mesure où les taux de bon étiquetage des étiqueteurs grammaticaux sont élevés.

5.2 Analyse contextuelle dans les éléments textuels

Un élément textuel ne peut pas être systématiquement réduit à un ensemble de termes indépendants pourvus de propriétés. Il peut s'avérer avantageux par exemple de considérer le contenu d'un document sous la forme d'une suite plus ou moins ordonnée d'unités linguistiques, car souvent l'enchaînement des termes caractérise la sémantique associée au contenu. Trois méthodes sont envisagées pour exploiter le contexte d’occurrence des termes :

5.2.1 Les affinités lexicales

La simple proximité des termes est en soit une information discriminante, car si deux termes se trouvent à une faible distance l'un de l'autre dans une phrase ou dans un paragraphe, il est raisonnable de considérer qu'ils partagent une certaine affinité qui dépend le cas échéant du domaine couvert. La notion d'*affinité lexicale* définie par [Maarek, 91] considère que deux termes composent une affinité lexicale si la distance qui les sépare est inférieure à un seuil. Cette notion est intéressante car elle permet, à moindre frais, d'introduire des composantes de discrimination d'ordre supérieur (n>2 termes successifs peuvent être considérés). Un fenêtre de 5 mots pleins avant et après le mot courant semble être acceptable pour capter une majorité d'affinités intéressantes pour l'Anglais et le Français. Les affinités lexicales constituent des unités d'informations associées à des degrés d'appartenance binaires, ou flous selon que l'on considère un critère en tout ou rien ou une fonction de forme gaussienne s'appliquant sur la distance séparant les termes : si deux termes sont très rapprochés dans une unité documentaire, le degré d'appartenance de l'affinité qu'ils réalisent tendra vers "1" pour cette unité documentaire, si au contraire, ces deux termes sont éloignés, le degré d'appartenance de l'affinité tendra vers 0.

5.2.2 Les patrons syntaxiques

Les patrons syntaxiques sont destinés à la prise en compte des expressions ou des mots composés qui peuvent s'avérer être très discriminants. Par exemple, les formes du type *Nom de Nom, Nom à Nom* et *Nom Nom* ("plaques de plâtre", "voiture à cheval", "capital risque") ou encore *Adjectif Nom* ("grande arche") autorisent l'extraction d'un grand nombre d'expressions relativement sélectives. Ces expressions peuvent jouer le rôle d'unités d'informations pour lesquelles le degré d'appartenance à une unité documentaire est une variable booléenne.

5.2.3 Exploitation de connaissances sémantiques

Pour tenir compte des proximités sémantiques qui rapprochent certains termes, il est nécessaire de faire appel à des ressources sémantiques qui relèvent soit de la langue générale, soit de domaine de spécialité [Voorhees, 1994]. Parmi les ressources existantes, WordNet[1] [Miller et al.1993a] constitue par exemple pour l'anglais une ressource de langue générale qui intègre entre autres les liens de synonymie, d'hyperonymie (relation de généralisation :

[1] La version 1.6 de WordNet est disponible gratuitement sur http://www.cosgi.princeton.edu/~wn

véhicule est un hyperonyme de *bicyclette*) et d'hyponymie (relation de spécialisation : *bicyclette* est un hyponyme de *véhicule*). Le projet EuroWordNet [CHUM, 1998] vise le développement de ressources similaires pour couvrir certaines langues européennes, dont le Français.

La prise en compte de connaissances d'experts, qui relèvent plutôt d'un domaine de spécialité, permet également d'augmenter sensiblement les performances des systèmes de recherche d'information [Anand et al., 1995]. Ces connaissances d'experts peuvent être décrites de façon relativement simple sous la forme de thesaurus de spécialité ayant une structure similaire à celle exploitée par les réseaux sémantiques de langue générale évoqués précédemment. Ainsi, un expert en instrumentation, pourrait être amené à définir un thesaurus spécifique pour modéliser son domaine de spécialité, comme illustré en Figure 2 :

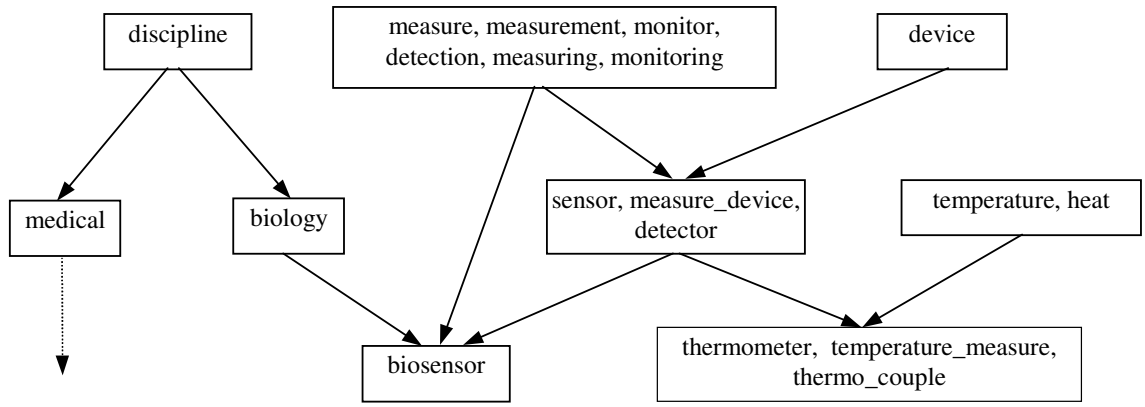

Figure 2 : Exemple de connaissances expertes modélisées sous la forme d'un thesaurus. Les cadres contiennent des termes considérés par l'expert comme équivalents ou synonymes, et les flèches désignent les spécialisations des concepts introduits par les cadres. Ainsi, la requête " thermometer" pointera vers des textes contenant :"Thermometer ", "temperature measurement ", "temperature sensor ", " thermo-couple", etc.

Le degré d'appartenance de ce type d'unités linguistiques à un élément textuel est une variable non binaire assimilable à des heuristiques qui caractérisent les liens syntaxiques ou sémantiques existants entre les unités linguistiques. Ces heuristiques peuvent être ajustées pour satisfaire les attentes des utilisateurs par apprentissage incrémental par exemple. L'enrichissement permet de générer des requêtes complexes en exploitant un langage d'interrogation basé sur des opérateurs préfixés qui prend en compte les heuristiques précédentes.

6. Principes de recherche et de fusion

Après enrichissement et interprétation de la requête, le moteur de recherche évalue les références adéquates.

6.1 Principe général

Nous passons sur les mécanismes de gestion des opérations ensemblistes, pour présenter ici le principe de mise en correspondance des contextes XML. Chaque élément XML rencontré au cours de l'indexation des documents est géré dans une table de symboles. A chaque intitulé d'élément XML (indépendamment des attributs) correspond une valeur numérique et une seule qui sert de clé dans une table de hachage. Le critère de recherche XML est tout d'abord codé par des indices correspondants aux intitulés des éléments XML pour former une chaîne. Par exemple, le critère de recherche XML ' /livre /introduction/ contexte/ ' pourrait être codé

en ‘ 12 ; 32 ; 75‘. Dans un premier temps, les attributs et les opérations d’évaluation sur les attributs sont ignorés.

6.2 Recherche exacte

L’auteur de la requête peut mentionner le mode de recherche : dans le cas d’une recherche exacte, seuls les critères exactement concordants avec la structure des documents sont retenus. Une recherche dans les contextes XML référencés est tout d’abord effectuée par comparaison successive de la chaîne codée du critère de requête avec les chaînes-contexte XML codées référencées.

Dans l’hypothèse d’une base documentaire respectant un ensemble de DTD fini, on peut à juste titre supposer que le nombre de contextes (hors attributs) différents est relativement limité, en tout cas, d’un ordre de grandeur largement inférieur à celui des éléments indicés en texte intégral. Pour cette raison, nous effectuons une comparaison et recherche de manière linéaire sur l’ensemble des contextes XML référencés.

Dans le cas de l’évolution à moyen ou long terme d’une base documentaire hétérogène, il est légitime de prévoir l’insertion de nouveaux documents éventuellement structurés de manière différente. Il peut être intéressant d’essayer, dans la mesure du possible, d’éviter d’une part de re-formater l’ensemble des documents existants, et d’autre part de ré-indexer toute la base à chaque modification ou évolution marginale du format des données. Dans cette optique, nous proposons l’utilisation d’une table de correspondance lors de la comparaison des chaînes-contextes : cette table peut contenir des équivalences d’éléments ou portions de contextes XML. La traduction (ancien format -> nouveau format) ne se manifeste donc pas par une modification de la base complète, mais de manière interne, lors de la recherche / indexation incrémentale de la base au fur et à mesure des ajouts de documents. Ce mécanisme ne peut bien évidemment pas éviter une restructuration éventuellement complète de la base, mais peut permettre une certaine marge dans l’adaptation *a posteriori* des structures internes.

Quand le critère concerne également un terme du document (un mot), une recherche sur ce mot dans la structure TREAP permet d’obtenir la liste des documents. Un simple filtrage sur la valeur du contexte XML du mot permet de conserver un ensemble de mots candidats. Ce n’est qu’à ce stade que les évaluateurs sur les attributs sont utilisés pour écarter des solutions.

6.3 Recherche approchée

Dans certains cas, il peut être intéressant de gérer de manière approchée les contextes XML composant les critères de recherche, comme par exemple, accepter un document dont la structure est très proche du critère de contexte XML. Dans ces cas, il est nécessaire de mettre en œuvre une comparaison d’arbres de contexte qui peut être coûteuse en temps de calcul.

Nous proposons d’utiliser le principe de distance d’édition [Levenstein 66] à poids modifiés en utilisant une version adaptée de l’algorithme de Wagner et Fisher [Wagner et Fisher, 74]. Cette technique permet en effet de limiter la combinatoire de la comparaison approchée de chaînes de caractères. Dans le cas qui nous intéresse, les chaînes à comparer sont précisément les codages des contextes XML présentés dans la partie précédente.

La complexité de l’algorithme est en O(n) si n est la longueur de la plus longue chaîne à comparer. Comme il est rare d’avoir des contextes extrêmement imbriqués (plus d’une 20 aine de niveaux), cet algorithme est beaucoup plus efficace qu’une mise en correspondance

dynamique classique : il remplace un parcours d'arbre exhaustif par l'utilisation d'une matrice de coûts calculée de manière incrémentale par heuristique.

Une distance d'édition est un coût mesurant les opérations à effectuer pour passer d'une chaîne (par exemple la chaîne de la requête) à une autre chaîne (la chaîne contexte XML référencée). Ce coût permet de trier les contextes XML référencés, un coût proche de zéro indiquant un contexte très proche de la requête et un coût élevé, une différence notable de structure. L'algorithme permet de calculer un ensemble minimal d'opérations pour passer d'une chaîne à une autre. Ces opérations sont les suivantes :

- la suppression : on supprime un nœud dans l'arbre (ou encore un élément XML dans le chemin)
- l'insertion : on rajoute un élément XML
- la substitution : on remplace un élément XML par un autre

L'exemple de la figure 3 illustre le calcul de la distance entre un contexte de requête et la structure d'un document : quand l'élément est le même (exemple : racine), le coût est généré par une différence sur les attributs.

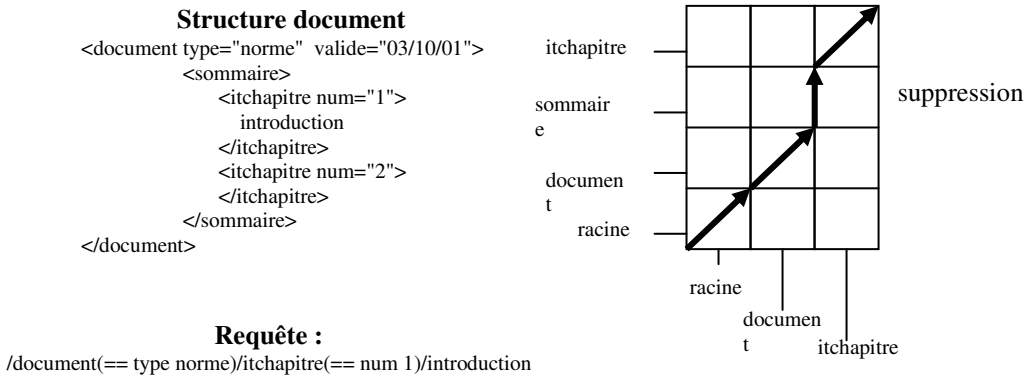

Figure 3 : Exemple de calcul de distance

L'algorithme général de mise en correspondance de chaînes est donnée en figure 4 :

Distance de deux contextes XML ($CXML_1$,$CXML_2$) :

$CXML_1$ de longueur n_1, et $CXML_2$ de longueur n_2, max = MAX(n_1,n_2)
Csupp(i) : coût de suppression du nœud (i) de $CXML_1$
Cins(i) : coût d'insertion du nœud (i) de $CXML_1$
Cremp(i,j) : coût de remplacement du nœud (i) de $CXML_1$ par le nœud (j) de $CXML_2$

```
D(0,0) <-0
Pour i variant de 1 à max faire
        D(i,0)<- D(i-1,0) + Csupp(0)
Refaire
Pour j variant de 1 à max faire
        D(0,j) <- D(0,j-1) + Cins(0)
Refaire
Pour i de 1 à n1
        Pour j de 1 à n2 faire
                M1 <- D(i-1,j-1) + Crempl(i,j)
                M2 <- D(i-1,j   ) + Csupp(i)
                M3 <- D(i,j-1   ) + Cins(j)
                D(i, j) <- min(m1,m2,m3)
        Refaire
Refaire
Distance d'édition <- D(n1,n2)
```

Figure 4 : Algorithme de recherche

Le calcul correspond à un chemin de coût minimal dans la matrice D (représentée dans la figure 3) entre D(0,0) et D(n_1,n_2). Les coûts de la substitution sont calculés en fonction de la table de correspondance et d'équivalence entre les éléments et en fonction de l'adéquation des opérations d'évaluation des attributs.

L'algorithme de calcul de coût permet ici d'évaluer plusieurs stratégies différentes pour la prise en compte des valeurs des attributs. Il est possible d'introduire un coût de substitution lié au moteur d'enrichissement linguistique : contrairement à la recherche 'exacte' qui concerne une adéquation parfaite, on peut ici donner une valeur de distance a priori sur les éléments du chemin. Ces mécanismes permettent de proposer une recherche étendue sur la structure du document et pas seulement sur les critères d'appartenance d'un terme ou mot à un sous-ensemble de documents. L'application essentielle de cette caractéristique n'est pas limitée à une simple évaluation d'une stratégie de recherche approchée : elle permet aussi de concevoir un système de recherche documentaire s'adaptant à l'utilisateur par spécialisation/apprentissage de ces matrices de coût. L'intégration de ces distances, se fait ensuite à l'aide d'un schéma d'optimisation multicritère [Ho & al. 92], [Franke 92] inspirée des diagrammes de *PARETO*. Cette technique permet d'éviter les effets de bords des opérateurs plus classiques en fusion. Ce choix est un choix préliminaire en cours d'évaluation.

7. Conclusion et perspectives

Nous avons développé une plate forme d'évaluation pour différentes stratégies de fusion d'information en recherche documentaire. Ce travail marque une réalisation préalable à un travail de fond sur les méthodes de fusion et de recherche documentaire. Le moteur d'indexation et de recherche SIRIUS développé exploite les données semi-structurées et non structurées (textes) contenues dans les documents XML. Un langage d'interrogation contextuel est spécifiquement proposé pour tenir compte de la localisation des éléments d'information jugés pertinents, vis-à-vis de la requête spécifiée, dans la structure documentaire. Ce langage s'appuie sur des principes d'alignement de structure arborescente capable d'exploiter des heuristiques a priori, apprises, ou optimales vis-à-vis de critères de performance souhaités. Le moteur SIRIUS intègre également des algorithmes de traitement automatique des langues naturelles associés à des notions de proximité lexicale, terminologique ou sémantique entre les termes contenus dans les éléments textuels des documents ou dans les « tags et attributs associés» XML eux mêmes. Hormis la prise en compte de données textuelles et de données structurées sous la forme de couples <ATTRIBUT/VALEUR>, l'exploitation par le moteur SIRIUS d'éléments d'information se présentant sous forme de séquences ou de séries temporelles fait l'objet de travaux complémentaires.

Bibliographie

[Anand et al., 1995] Sarabjot S. Anand, David A. Bell & John G. Hughes*; The role of domain knowledge in data mining*; in Proceedings of the 1995 international conference on information and knowledge management (CIKM'95); pp. 37-43; 1995.

[CHUM, 98] Editors-in-Chief : N. Ide & D. Greenstein; Guest Editor: P. Vossen; Double Special Issue on EuroWordNet; Computers and the Humanities, Volume 32, Nos. 2-3; 1998.

[El-bèze, 93] M. El Bèze, *Les modèles probabilistes de langage* Habilitation à diriger des recherches, LIPN, Paris 1993.

[Franke et Mandler 92] J. Franke et E. Mandler, *a comparison of two approaches for combining the votes of cooperating classifiers*, 11th ICPR, Vol. 2, p.611-614, 1992.

[Ho et al, 92] Tin Kam Ho, J.T. Hull, S.N. *Srihari, Combining of decisions by multiple classifiers*, Structured document image analysis, Springer Verlag, 1992.
[Krovetz, 1993] Robert Krovetz; Viewing Morphology as an Inference Process; in Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; pp. 191-203; 1993.
[Levenstein, 66] A. Levenstein, *Binary codes capable of correcting deletions, insertions and reversals*, Sov. Phys. Dohl., Vol. 10, p.707-710, 1966.
[Loupy et al., 98a] C. de Loupy, P.F. Marteau & Marc El-Bèze; *Navigating in Unstructured Textual Knowledge Bases*; in Proceedings of Nîmes'98 - La Lettre de l'IA; pp. 82-85; May 1998.
[Maarek, 91] Y. S. Maarek; *Software Library Construction From an IR Perspective*; SIGIR forum, Fall 1991, 25:2; pp. 8-18; 1991.
[Miller et al., 93a] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross & K. Miller; *Introduction to WordNet: An On-Line Lexical Database*; http://www.cosgi.princeton.edu/~wn; August 1993.
[Riloff, 95] Ellen Riloff; Little words can make big difference for text classification; in Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; pp. 130-136; Seattle, Washington, USA; July 9-13, 1995; (1995:023).
[Robertson & Sparck-Jones, 76] Robertson, S.E. and K., Sparck-Jones. *Relevance weighting of search terms*. Journal of the American Society for Information Science, 27: pp. 129-146, 1976
[Salton, 89] G.Salton, " *Automatic Text Processing* ", Reading, Mass. Addison-Wesley, 1989.
[Salton, 94] Gerard Salton, James Allan, Chris Buckley, " *Automatic Structuring and Retrieval of Large Text Files* ", Communications of the ACM, Vol.37, n°.2, 1994.
[Seidel, 96] Raimund G. Seidel and Cecilia R. Aragon, "Randomized Search Trees," Algorithmica, 16:464-497 (1996). Also in 30th Annual Symposium on Foundations of Computer Science, pages 540-545, Research Triangle Park, North Carolina, 30 October-1 November 1989. IEEE.
[VanRijsbergen, 1979] C. J. Van Rijsbergen, " *Information Retrieval* ", Buttherwords, London, 1979 (2nd edition).
[Voorhees, 93]: Ellen M. Voorhees; *Using WordNet to Disambiguate Word Sense for Text Retrieval*; ACM-SIGIR'93; pp. 171-180; Pittsburg, PA, USA; June 1993.
[Voorhees, 94]: Ellen M. Voorhees; Query expansion using lexical-semantic relations; in Proceedings of the seventeenth annual international ACM-SIGIR conference on Reearch and development in information retrieval (SIGIR'94); pp. 61-69; 1994; (1994:039).
[Wagner et Fisher, 74] *The string to string correction problem*, Journal of the Association for Computing Machinery, Vol 12., p168-173, 1974
[Xerces] *The Java XML Xerces Parser*, http://xml.apache.org/xerces-j/
[Xhive] The Xhive XML data base system http://www.xhive.com/
[Zhao, 99] Ben Y. Zhao, Anthony Joseph*, XSet: A Lightweight XML Search Engine for Internet Applications*, http://www.cs.berkeley.edu/~ravenben/xset/