République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri de Tizi-Ouzou

Faculté du Génie Electrique et d'Informatique

Département Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en Informatique

***Option** : Systèmes Informatiques*

Thème :

Indexation sémantique de documents textuels

Dirigé par

M^me^ AMIROUCHE Fatiha

Réalisé par

M^elle^ ALILECHE Lydia

Promotion 2012

Remerciements

Mes vifs remerciements vont à m^{me} Amirouche Fatiha pour avoir accepté de m'encadrer, pour ses orientations, ses conseils et tout le temps qu'elle m'a consacrée. Puisse-elle trouver en ces quelques lignes l'expression de ma profonde gratitude.

Je tiens aussi à remercier les membres du jury, à savoir, M^{r} Redaoui , M^{r} Hammache et M^{r} Sadou pour avoir accepté de juger ce travail.

A mes parents

A tous ceux que j'aime

Table de Matières

Chapitre II : Indexation Sémantique

Chapitre III : Approche d'Indexation Sémantique

Chapitre IV : Résultats et Expérimentations

Table des Figures

Introduction générale

Bien avant l'avènement de l'informatique et des technologies de communication, la recherche d'information (RI) était déjà présente dans la vie quotidienne. Elle était réalisée en se contentant d'utiliser des moyens empiriques comme l'utilisation de répertoires ou annuaires d'index manuels ainsi que des annotations et résumés pour accéder aux documents contenant l'information recherchée.

Avec l'essor du web dans les années 1990 et avec l'explosion des ressources d'information disponibles et leur hétérogénéité, la RI a été propulsée au premier plan. Plusieurs moteurs de recherche sont ainsi apparus permettant l'accès à l'information à grande échelle (Yahoo[1], Altavista[2]) et des systèmes de recherche d'information (SRI) ont été crées. L'objectif SRI est donc d'une part de permettre aux utilisateurs d'exprimer facilement leurs besoins d'information au moyen des requêtes, et d'autre part de leur fournir les documents potentiellement pertinents par rapport aux besoins d'information qui ont été transmis au système.

Cette explosion du web a néanmoins fait que la RI se retrouve confrontée à un problème: collections gigantesques, diversifiées et surabondance de l'information. Ainsi, la tâche principale de la RI qui est de retrouver l'information pertinente à une requête devient difficile à accomplir, non plus par un manque d'informations mais par sa surabondance qui rend ardu le filtrage des documents pertinents.

Dans le but d'améliorer la qualité des SRI, plusieurs approches d'indexation et d'appariement ont étés proposées. On est ainsi passé des approches dites « classiques » à des approches « sémantiques ». Les approches classiques se basent sur des formalismes mots-clés. Un document (ou requête) est représenté par une liste de mots clés auxquels sont associés des poids selon leur fréquence d'apparition dans le document. ce type d'approche présente un inconvénient majeur, celui de ne pas tenir compte du sens des mots, ce qui est en soit un handicap quand on traite des documents textuels vu la richesse de langue. L'indexation sémantique tente de remédier à ce problème en intégrant le sens des mots dans la représentation des documents et requêtes .

---

[1] http://www.yahoo.fr

[2] http://www.Altavista.com

L'objectif de ce travail de master consiste en deux étapes : d'abord implémenter une approche d'indexation sémantique de documents textuels, ensuite, intégrer le module qui en résulte à la plateforme de RI Terrier.

Pour ce faire, nous commencerons par présenter dans le premier chapitre, les notions de base de la RI et les modèles de RI les plus connus.

Nous enchainerons dans le deuxième chapitre avec un état de l'art sur l'indexation sémantique. Nous présenterons les principaux travaux en relation, les méthodes de désambiguïsation utilisées avant de nous consacrer à l'introduction du domaine du discours dans le processus de désambiguïsation.

Dans le troisième chapitre, nous présenterons notre approche basée sur l'utilisation du domaine du discours (Domain of Speech) comme moyen de désambiguïsation des mots polysémiques. Notre désambiguïsation utilisera le dictionnaire *WordNet , WordNet Domains* et l'étiqueteur syntaxique *Stafford POS Tag* comme ressources externes. Nous allons donc concevoir notre module d'indexation sémantique et décrire son intégration à la plateforme de RI Terrier 3.5.

Les résultats et expérimentations de notre approche dans la plateforme Sem-Terrier résultante de l'intégration de notre module à Terrier seront présentés dans le quatrième chapitre, avant de conclure et de présenter quelques perspectives pour des recherches ultérieures.

Chapitre I

La Recherche d'Information

I.1 Introduction :

La recherche d'information (RI) (ou « Information Retirieval » en Anglais), est définie par Gérard Salton [Salton & al., 83a] comme étant "*un domaine de recherche qui s'intéresse à la structure, à l'analyse, à l'organisation, au stockage, à la recherche et à la découverte de d'information*". La RI est une discipline de recherche qui intègre des modèles et des techniques dont le but est de faciliter l'accès à l'information pour un utilisateur ayant un quelconque besoin, formulé sous forme de requête et adressée au « Système de Recherche d'Information » (SRI). Le SRI se charge de comparer un ensemble de documents avec la requête et de restituer un sous ensemble de documents susceptibles de répondre au besoin informationnel de l'utilisateur.

Dans ce chapitre, nous présentons une vue générale de la RI, ses concepts de base, les modèles de RI les plus connus, ainsi que la problématique inhérente à la RI classique.

I.2 Concepts de base de la RI :

I.2.1 Définition d'un SRI :

Un système de recherche d'information est un système informatique qui permet de retrouver, à partir d'une collection de *documents*, les *documents* susceptibles d'être *pertinents* à un besoin en information d'un utilisateur exprimé sous forme de *requête*. Cette définition met en évidence trois notions clés que nous allons expliciter dans ce qui suit: *document*, *requête*, *pertinence*.

Document : désigne toute unité qui peut présenter une réponse à une requête donnée. Un document peut être un morceau de texte, une page Web, une image, une séquence vidéo, etc. L'ensemble des documents sur lequel porte une recherche forme la collection de documents.

Requête : La requête constitue l'expression du besoin en information de l'utilisateur, généralement sous forme d'un ensemble de mots clés.

Pertinence : La pertinence est une notion fondamentale en RI. Elle peut être définie comme le degré de correspondance entre un document et une requête. Cette mesure vue coté utilisateur définit la pertinence utilisateur , vue coté système, elle définit la pertinence système.

I.2.2 Mise en œuvre d'un SRI :

L'objectif de tout SRI est la mise en relation des informations contenues dans la collection d'une part, et les besoins de l'utilisateur d'autre part. Un tel système est mis en œuvre à travers le processus en U de la RI, qui est illustré dans la figure suivante. Ce processus consiste en deux principales phases : l'indexation et l'interrogation [Daoud, 09].

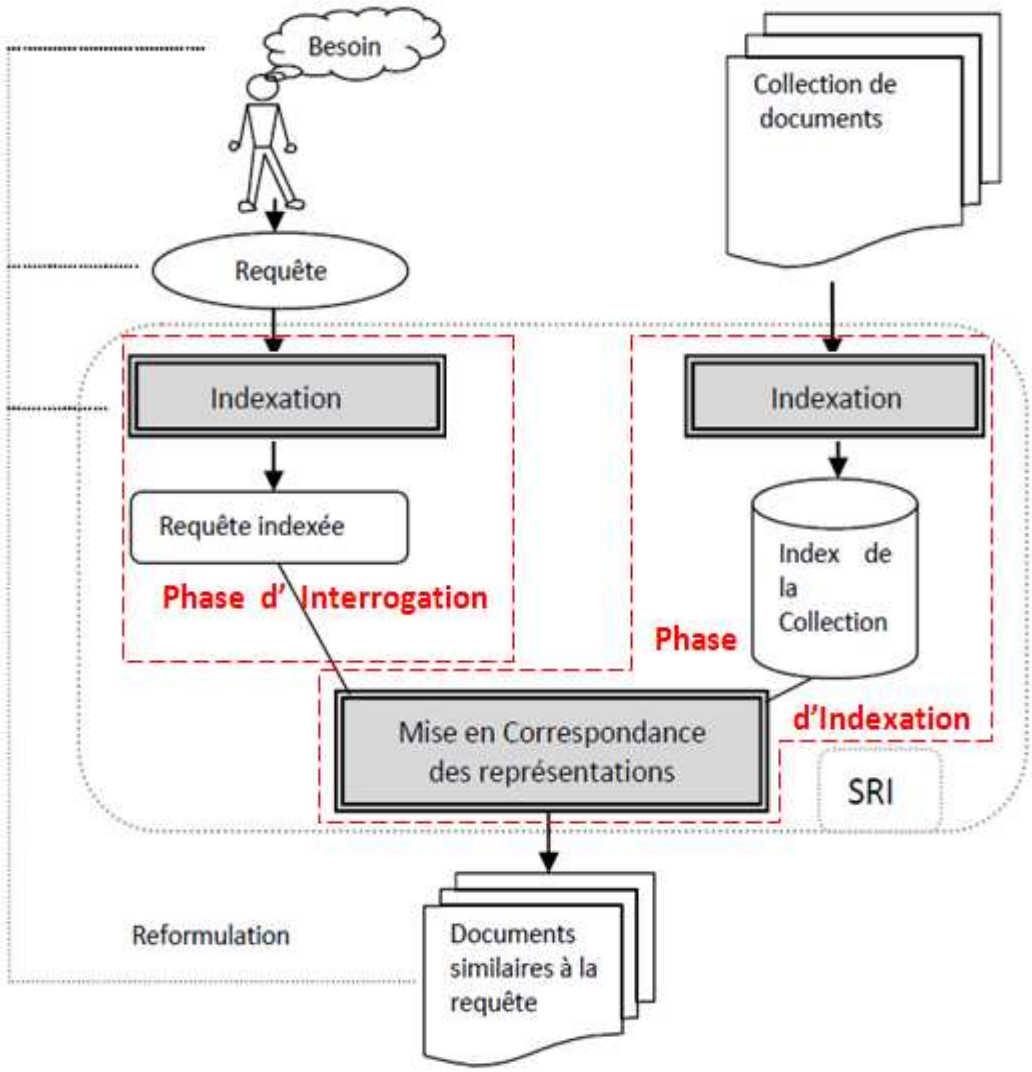


Figure 1.1 Processus en U de la RI [Champclaux, 09]

1. L'indexation : consiste à extraire et représenter le contenu de la collection de documents de manière interne sous forme d'index. Cette structure d'index permet de retrouver rapidement les documents contenant les mots clés de la requête. La construction des index peut être assez longue en fonction du nombre de documents de la collection ainsi que de la taille des documents.

2. L'interrogation : se fait lors de l'interaction avec le SRI d'un utilisateur final ayant un besoin d'information, une fois les documents représentés sous forme interne d'index.

Pour chaque requête utilisateur, le système indexe la requête pour en extraire la représentation interne. Il calcule la pertinence de chaque document de l'index de la collection vis à vis de la requête utilisateur indexée, selon une mesure de correspondance définie dans le modèle de RI utilisé. Le SRI retourne la liste des résultats à l'utilisateur classés par ordre croissant de degré de pertinence, si possible.

Certains systèmes peuvent permettre le raffinement et l'amélioration de la requête par reformulation.

1.3 Indexation :

L'indexation consiste en un ensemble de techniques visant à analyser les documents (lors de la phase d'indexation) et les requêtes (lors de la phase d'interrogation) et en extraire les mots clés (mot ou groupe de mots) caractérisant leur contenu informationnel. Les mots-clés descriptifs du contenu sémantique d'un document sont dits termes d'indexation, L'ensemble de tous les termes d'indexation constitue le langage d'indexation (ou Vocabulaire d'indexation) [Amirouche, 08]. En complément, à chaque couple (terme d'indexation, document) est associé un poids qui représente l'importance du terme dans un document [Ralalason, 10]. Lorsqu'une requête est soumise au SRI, les termes qu'elle contient sont mis en correspondance avec les termes d'indexation extraits des documents pour en déduire les documents à restituer à l'utilisateur.

1.4 Vocabulaire d'Indexation :

Lors de la conception d'un SRI, la question cruciale du vocabulaire d'indexation ou langage d'indexation [Cleveland & al., 00] arrive en premier. Ce langage peut être libre ou contrôlé. [Abichahine, 11].

1.4.1 Vocabulaire Libre :

Construit à partir des termes en langue naturelle, souvent issus du texte original, et permettant de décrire son contenu [Harter, 86] (on parle d'indexation par extraction ou *extraction indexing* en anglais). On trouve ce type d'indexation par vocabulaire libre dans les moteurs de recherche de type GOOGLE.

1.4.2 Vocabulaire Contrôlé :

Construit à partir des termes extraits d'un thésaurus. Le thésaurus est une liste de descripteurs (mots-clés) normalisés et reliés entre eux par des relations sémantiques [Boucham, 09].

Ces relations, au nombre de trois [Roussey & al., 01]:

- La relation d'équivalence regroupe les termes jugés équivalents (synonymes ou termes très proches sémantiquement)
- La relation hiérarchique construit une hiérarchie entre les termes d'indexation, du général au particulier ou d'un tout à ses parties.
- La relation d'association lie des termes d'indexation ayant des connotations (basées sur la cooccurrence des termes).

Selon [Boucham, 09] l'organisation du thésaurus permet de trouver le terme d'indexation le plus approprié pour représenter un concept. Par exemple, l'utilisateur d'un système de recherche d'information utilise un terme de son vocabulaire comme entrée dans le thésaurus et, en suivant différentes relations, trouve le terme d'indexation reconnu par le système pour composer sa requête.

1.5 Les Approches d'Indexation :

En RI l'indexation peut être manuelle, automatique ou semi-automatique [Salton & al., 88].

1.5.1 Indexation Manuelle :

Basée sur un vocabulaire contrôlé, elle est réalisée par un expert documentaliste qui analyse l'intégralité du document et choisit une liste des termes qu'il juge pertinents dans la description sémantique de ce document [Daoud, 09]. Bien évidement une telle indexation se base uniquement sur les connaissances personnelles de cet expert, et est donc sujette à sa subjectivité. Cette méthode de travail est non seulement très coûteuse en temps, mais aussi inapplicable sur un gros volume de données. Cependant, elle a l'avantage d'être plus précise dans les résultats [Ren & al., 99], car les spécialistes d'un domaine choisissent de meilleurs termes pour indexer les documents.

1.5.2 Indexation Automatique :

Utilise un vocabulaire libre issu des documents [Ralalason, 10]. Cette indexation consiste en une analyse automatisée basée sur des algorithmes [Daoud, 09] chargés d'extraire les termes représentatifs du contenu du document. Cette indexation ne fait intervenir aucun expert ce qui pallie au problème de subjectivité et de temps d'indexation. Elle est donc adaptée aux collections de documents volumineuses.

Il existe plusieurs possibilités concernant le choix des termes: on peut en effet opter pour des mots seuls ou des groupes de mots. Bien que l'idée de choisir un groupe de mots comme représentant de concept semble bonne, l'expérience a montré que l'utilisation de représentations complexes améliorait de façon marginale le processus de recherche [Croft, 95]. De nombreuses expérimentations ont utilisé des mots seuls extraits des documents et de la requête pour la représentation du contenu, et ont montré de bons résultats [Baeza & al., 92].

L'indexation automatique est une notion très importante en RI, nous la détaillons d'avantage dans la suite de ce chapitre.

1.5.3 Indexation Semi-automatique (ou Supervisée) :

Les indexes sont automatiquement extraits des textes, puis sont soumis aux spécialistes du domaine qui les valident en utilisant un thésaurus [Maniez & al., 91]

1.6 L'indexation Automatique :

L'indexation automatique repose sur un ensemble des méthodes automatisées d'analyse des documents. Elle est basée sur plusieurs étapes : l'Analyse lexicale, la sélection, la Radicalisation , la Pondération des termes. Ces étapes sont illustrées dans cette figure :

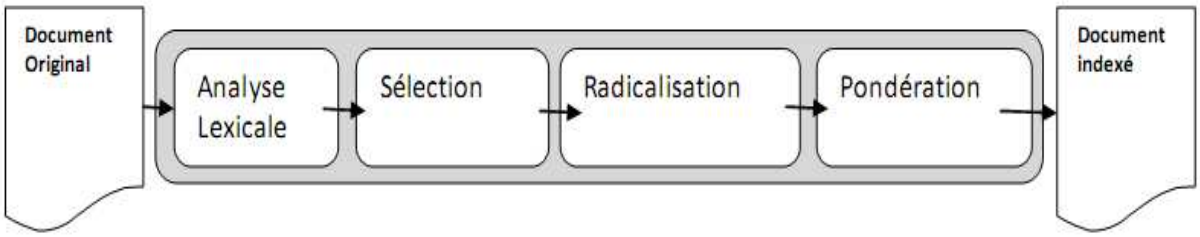


Figure 1.2 Traitements effectués lors de l'indexation [Champclaux, 09]

1.6.1 Etapes d'une indexation automatique :

a. Analyse lexicale (ou tokenisation)

C'est l'étape qui permet de transformer un document textuel en un ensemble de termes (ou lexèmes). Un terme est un groupe de caractères constituant un mot significatif [Baeza & al., 92]
Pendant cette phase, les espaces, la ponctuation, la casse, et la mise en page sont supprimés[Champclaux, 09].

b. Sélection

Afin de ne garder que les termes importants, il est nécessaire d'éliminer les mots vides de sens c'est-à-dire ne reflétant pas le contenu informationnel des documents. Les mots vides peuvent être des pronoms personnels, des articles, des mots de liaison, des prépositions ou des mots athématiques (les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple : être, avoir, contenir, posséder, appartenir, consister …). A cet effet, il existe plusieurs techniques:

- l'utilisation d'un anti-dictionnaire ou liste de mots vides (aussi appelée Stoplist en anglais). Ainsi, quand un mot est rencontré dans un texte à indexer, s'il apparaît dans l'anti-dictionnaire, il n'est pas considéré comme terme d'index [Salton & al., 88].

- L'élimination des mots dépassant une certaine fréquence d'occurrences ou ayant une fréquence d'occurrence quasi nulle [Luhn, 58].

c. Radicalisation :

Appelée aussi lemmatisation ou racinisation [Porter, 80], cette étape consiste à éliminer les différentes variations morphologiques d'un mot en extrayant le radical du mot (lemme ou racine). On peut citer pour exemple : éduquer, éducation, éducateur, éducatif, proviennent tous d'un même racine. Il est donc inutile d'indexer tous ces mots puisqu'un seul suffirait à représenter le concept véhiculé.

Il existe plusieurs stratégies de radicalisation. On peut citer les algorithmes de suppression des affixes (préfixes et suffixes), parmi lesquels figure l'algorithme de Porter [Porter, 80]. L'objectif de cet algorithme est de ramener un mot de langue la Anglaise à un radical en supprimant sa terminaison. Pour cela il applique successivement plusieurs règles de transformation visant à supprimer le pluriel, les participes passés puis les différentes dérivations telles que « able », « ness », « tly ».

d. La pondération :

C'est une étape très importante dans une indexation automatique. Elle permet d'assigner aux termes leurs degré d'importance dans les documents. Ce degré (ou poids) étant une valeur numérique calculée selon des méthodes statistiques.
Un terme peut être expressif s'il apparait suffisamment fréquemment pour être statistiquement important sans toutefois excéder une certaine limite qui le classerait dans la catégorie des mots outils (vides) [Hlaoua, 07]. Il existe un grand nombre de formules de pondération dont la plus connue est ***Tf*Idf ,*** qui est basée sur deux facteurs [Robertson & al., 97] [Singhal & al., 97] [Sparck, 79] : fréquence de terme (*Tf*) et fréquence documentaire inverse (*Idf*). Ces deux facteurs sont définis dans ce qui suit :

Term Frequency : *($tf_{i,j.}$)* C'est une pondération locale qui calcule la fréquence (nombre d'occurrences) du terme i dans un document j

Inverse of Document Frequency : (*idf)* C'est une pondération globale qui calcule la fréquence du terme i dans un ensemble de documents. Elle peut être mesurée comme suit :

$$\boldsymbol{Idf_i = \frac{1}{df_i}}$$

Tel que df_i représente la fréquence documentaire (ou le nombre documents de la collection qui contiennent le terme t_i)

Cette fréquence peut être calculée autrement par :

$$\boldsymbol{Idf_i = log\left(N/n_i\right)}$$

Tel que : **N** : le nombre de documents dans la collection.

$\mathbf{n_i}$: le nombre de documents contenant le terme i.

La formule de pondération ***Tf*Idf*** consiste à multiplier les deux mesures $\boldsymbol{Tf_{i,j}}$ et $\boldsymbol{Idf_i}$. La mesure obtenue est $\boldsymbol{w_i}$ qui représente le poids du terme i dans le document j

$$\boldsymbol{w_i = tf_{ij} * idf_i = tf_{ij} * \frac{1}{df_i} = \frac{tf_{ij}}{df_i}}$$

A la fin de cette étape le document indexé est généré.

Dans une indexation classique on fait appel à l'analyse lexicale mais suivant les fonctionnalités voulues, cette analyse peut s'avérer insuffisante et nécessiter une analyse syntaxique qui se charge de repérer les groupes de mots ou des mots composés [Fagan, 87] [Salton, 88], voire une analyse sémantique qui s'intéresse à reconnaitre les sens des mots, les mots synonymes et les relations sémantiques entre les mots dans un document [Amirouche, 08]. Dans de tels systèmes les descripteurs seront des mots-clés (concepts) associés à des entrées dans un vocabulaire contrôlé, thésaurus [Salton & al., 68] [Sparck Jones, 86], ontologie (exemple PenMan, Cyc [Sowa, 84] et WordNet [Miller, 95] etc.)

L'utilisation des ontologies offre de grandes perspectives pour l'intégration de la sémantique à la recherche d'information. Ceci fera l'objet d'une étude plus détaillée dans le chapitre suivant notamment l'ontologie WordNet.

1.6.2 Illustration des étapes d'indexation :

[Champclaux, 09] a appliqué dans ses travaux le processus d'indexation automatique au document ayant pour titre « 18 Editions of the Dewey Decimal Classifications » et pour auteur « Comaromi, J.P. ». voici les résultats de chaque étape :

- **Document original :**

The present study is a history of the DEWEY Decimal Classification. The first edition of the DDC was published in 1876, the eighteenth edition in 1971, and future editions will continue to appear as needed.

- **Après analyse lexicale :**

the present study is a history of the dewey decimal classification the first edition of the ddc was published in 1876 the eighteenth edition in 1971 and future editions will continue to appear as needed

- **Après suppression des mots vides :**

present study history dewey decimal classification edition ddc published eighteenth edition future editions continue needed

- **Après radicalisation avec l'algorithme Porter :**

present studi histori dewey decim classif edit ddc publish eighteenth edit futur edit continu need

- **Le résultat de l'indexation : l'index**

Le résultat d'une indexation donne un ensemble de termes et leurs pondérations pour chaque document comme suit :

d_j ,…(t_i, a_{ij})…

Avec *t* le terme d'indice i dans le vocabulaire et *a*ij son poids dans le document *d*j.

Avec cette structure, il est facile de trouver les termes inclus dans un document. Cependant, étant donné une requête contenant quelques termes, il est plus intéressant de retrouver les documents correspondant à chacun de ces termes. Pour cela un fichier inverse est construit avec la structure suivante :

t_i ,…(d_j, a_{ij})…

L'entrée de l'index correspondant au document de l'exemple avec une pondération ***tf*idf*** est:

d_1 {(edit , 0.090) ; (dewey, 0.25); (decim, 0.125); (classif, 0.019); (present, 0.003); (studi, 0.002); (histori, 0.039); (publish, 0.008); (ddc, 0.4); (eighteenth, 1.0); (futur, 0.010); (continu, 0.014); (need, 0.022)}

1.7 Taxonomie des Modèles de RI :

Après l'indexation qui permet de déterminer les termes représentatifs des documents et requêtes, le calcul du degré de pertinence des documents pour les requêtes se fait par l'intermédiaire d'une mesure de pertinence implémentée dans le modèle de RI.

Le rôle d'un modèle de RI est fondamental dans un SRI. Il consiste en deux points importants: d'abord la création d'une représentation formelle pour un document ou pour une requête basée sur leurs termes d'index. Ensuite la définition d'une méthode de

comparaison entre une représentation de document et une représentation de requête, afin de déterminer leur degré de correspondance (appariement ou pertinence).

Bien qu'il existe un grand nombre de modèles de RI, ces modèles ont tous en commun le vocabulaire d'indexation basé sur le formalisme « mots clés » et diffèrent principalement par le modèle d'appariement requête-document.

Les modèles de RI peuvent être classifiés en 3 grandes catégories :

Les modèles booléens *:* ils sont basés sur la théorie des ensembles. Ce sont les plus simples et les premiers à avoir été mis en place.

Les modèles vectoriels*:* ces modèles sont algébriques. Ils définissent la pertinence d'un document vis-à-vis d'une requête par des mesures de distance entre leur représentation dans un espace vectoriel.

Les modèles probabilistes*:* ces modèles sont basés sur la théorie des probabilités et estiment des probabilités de pertinence d'un document en fonction de la requête.

La figure suivante présente la taxonomie des modèles de RI existants :

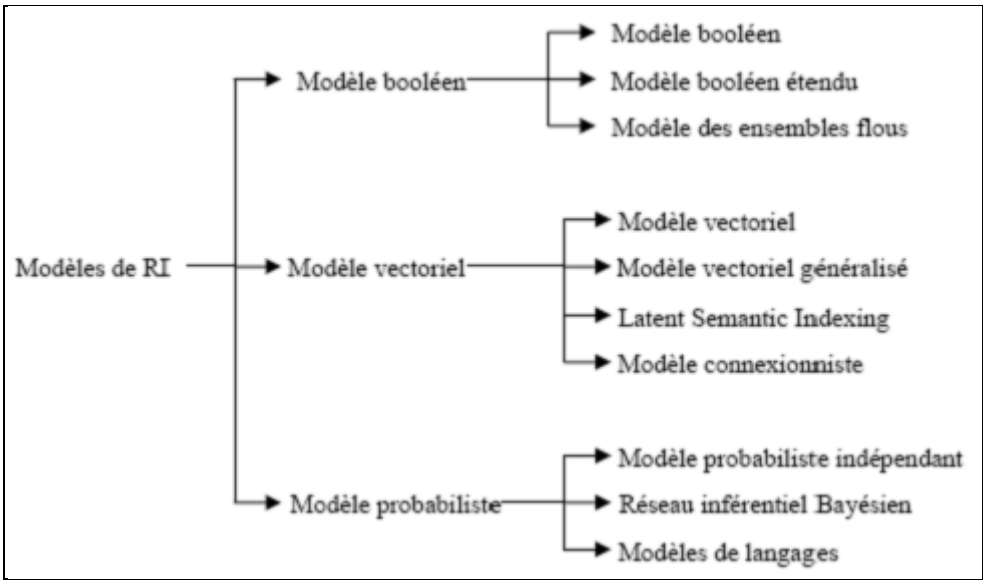


Figure 1.3 Taxonomie des modèles en RI [Baeza-Yates & al., 99]

Dans ce qui suit nous présentons les principaux modèles de chaque catégorie.

1.7.1 Modèles Booléens :

a. Modèle booléen de base

C'est le premier modèle de RI. Il est basé sur la théorie des ensembles [Salton, 71b] . Chaque document *d* est représenté par une conjonction logique de n termes t_i (non pondérés) qui constitue l'index du document.

$$d = t_1 \wedge t_2 \wedge t_3 \ldots \wedge t_n$$

Une requête utilisateur est formulée à l'aide d'une expression booléenne des mots clés. Un exemple de requête *or* est donné par :

q=(t_1 AND t_2) OR (NOT (t_3 AND t_4))

ou *q=($t_1 \wedge t_2$) $\vee$ $\neg$($t_3 \wedge t_4$)*

La fonction de calcul de la similarité, ici notée *RSV (q, d)* (Retrieval Status Value) définit l'appariement entre une requête q et un document d de la façon suivante :

RSV (d, t_i)=1 si *ti∈d,* 0 sinon.

RSV (d, t_i, AND t_j)=1 si *(t_i∈d)* et *(t_j∈d) ,0* sinon.

RSV (d, t_i, OR t_j)=1 si *(t_i∈d)* ou *(t_j ∈d) ,0* sinon.

RSV (d, NOT t_i)=1 si *RSV (d, t_i)=0 ,0* sinon.

Cet appariement est simple mais présente quelques inconvénients. En effet, étant donné le critère binaire (soit 0, soit 1) de ce modèle, un document est soit « pertinent », soit « non pertinent ». On se retrouve dans l'incapacité de classer le documents selon leur degré de pertinence car tous les documents retournés sont tous pertinents "de la même façon". Ceci induit la non efficacité du critère binaire par rapport à la pondération des termes qui améliore les résultats [Rallalason, 10]. De plus, la difficulté d'expression des requêtes issues du langage naturel ou relativement longues, sous forme booléenne, réduit l'effectif des personnes aptes à utiliser ce modèle et fait qu'il soit réservé aux experts [Mallak, 11].

b. Modèle booléen étendu

C'est une extension du modèle booléen de base. Il fut Introduit par SALTON [Salton & al., 83]. Il vise l'intégration des poids d'indexation dans l'expression de la requête et du document, ce qui permet ainsi d'ordonner les documents retrouvés par le SRI.

Considérons un ensemble de termes *t_1, …, t_n,* et soit *d_{ij}* le poids du terme ti dans le document *dj tel que*

dj= (d_{1j}, …, d_{nj}), avec *: 1≤ i≤ n* et *0≤ d_{ij}≤ 1*.

Soient *p* est une constante telle que *0≤ p ≤ ∞*

q_{ik} le poids du terme *t_i* dans la requête *q_k*

La similarité entre le document d_j et une requête q_k décrite sous une forme conjonctive ou disjonctive est donnée comme suit :

Operateur OR:

$$RSV(dj, qk) = \left(\frac{\sum_{i=1}^{n} q_{ik}^{p} d_{ij}^{p}}{\sum_{i=1}^{n} q_{ik}^{p}} \right)^{\frac{1}{p}}$$

Operateur AND:

$$RSV(dj, qk) = 1 - \left(\frac{\sum_{i=1}^{n} q_{ik}^{p} \left(1 - d_{ij}^{p}\right)}{\sum_{i=1}^{n} q_{ik}^{p}} \right)^{\frac{1}{p}}$$

Dans ce modèle, lorsque p=1, il n'y a aucune distinction entre les deux connecteurs AND et OR. Dans ce cas on se ramène au modèle booléen.

Cependant d'après [Ponte, 98], il n'y a aucune méthode formelle proposée pour la détermination de la valeur du paramètre *p.*

L'avantage de ce modèle c'est qu'il permet de classer les résultats en plaçant les documents jugés les plus similaires à la requête en premier. Ce modèle présente néanmoins des inconvénients, à savoir:

1. Les opérateurs ne sont pas associatifs, contrairement au modèle booléen simple
2. Ce modèle est complexe et difficile à appréhender pour les utilisateurs. Il nécessite un module expert de formulation de requête.

c. Modèle booléen basé sur les ensembles flous [Zadeh, 65]

Dans ce modèle, inspiré des ensembles flou, chaque terme a un degré d'appartenance à un document. Ce degré correspond au poids du terme dans le document. Une requête est toujours représentée par une expression booléenne classique, tandis que, l'évaluation des opérateurs logiques □ et □ est remplacé par les fonctions *min* et *max*.

Ces évaluations ont été proposées à la fin des années 1970 et au début des années 1980. Maintenant, ces extensions sont devenues standard: la plupart des systèmes booléens utilisent un de ces modèles étendus.

Dans le modèle booléen flou introduit par Ogawa, Morita, et Kobayashi en 1991, l'appartenance d'un élément à un ensemble est pondérée par un scalaire $\mu \in [0,1]$:

- Si $\mu = 0$ alors l'élément n'appartient pas a l'ensemble.
- Si $\mu = 1$ alors l'élément appartient totalement a l'ensemble.

[Ogawa & al., 91] ont construit une matrice de corrélation terme-terme par analyse des documents du corpus, permettant de calculer la similarité de 2 termes. La similarité de Jaccard est utilisée pour la création de la matrice, où n_1, n_2 et $n_{1,2}$ sont respectivement les nombres de documents contenant le terme t1, t2 et (t1 et t2) :

$$\boldsymbol{sim(t_1, t_2) = \frac{n_{1,2}}{n_1 + n_2 + n_{1,2}}}$$

L'objectif du calcul est d'estimer l'appartenance d'un document à une classe représentée par le terme. Celle-ci dépend de la corrélation entre les termes du document et t. Ainsi, le degré d'appartenance d'un document d à l'ensemble flou associe à un terme t est :

$$\boldsymbol{\mu_{t,d} = 1 - \prod_{t_i \in d} \left(1 - sim(t, t_i)\right)}$$

Si t est présent dans le document alors l'appartenance est totale, si d contient un terme très similaire à t alors l'appartenance est proche de 1. Dans l'optique de la recherche d'information, nous calculons le degré d'appartenance d'un document d a une requête q. Nous résolvons la FND de $Q = \{Q_1 \ OR \ Q_2 \ \dots \ OR \ Q_n\}$ avec chaque Q_i composé des éléments $(q_{i,1}, q_{i,2} \dots q_{i,k})$.

$$\boldsymbol{\mu_{Q,d} = 1 - \prod_{Q_i \in Q} \left(1 - \mu_{Q,d}\right)}$$

$$\boldsymbol{\mu_{Q,d} = \prod_{q_{i,j} \in Q_i} \mu_{i,j}}$$

Ainsi $\mu_{Q,d}$ représente le score de similarité pour le modèle booléen flou. Mais la similarité entre les termes peut être calculée d'une autre manière que par coefficient de Jaccard.

1.7.2 Modèles vectoriels :

Modèle Vectoriel de Base

Proposé par Salton dans le système SMART (*Salton's Magical Automatic Retriever of Text*) [Salton, 70]. Ce modèle repose sur les bases mathématiques des espaces vectoriels.

L'espace vectoriel (à n dimensions) est défini par l'ensemble de termes d'index. Soit l'espace vectoriel suivant : $E = \{t_1, \ldots, t_n\}$. Un document ou une requête est représenté comme un vecteur de poids. Chaque poids dans le vecteur désigne l'importance d'un terme correspondant dans ce document (ou dans la requête). Formellement, un document d_i est représenté par un vecteur de dimension n,

$d_i = (w_{i1}, w_{i2}, \ldots, w_{in})$ pour $i=1,2,\ldots,m$.

Où w_{ij} est le poids du terme t_j dans le document d_i ; $w_{ij} \in [0,1]$,

m est le nombre de documents dans la collection.

n est le nombre de termes d'indexation.

Une requête q est aussi représentée par un vecteur de dimension n de mots clés défini dans le même espace vectoriel que le document.

$q= (w_{q1}, w_{q2}, \ldots, w_{qn})$

Où w_{qj} est le poids du terme t_j dans la requête q. ce poids peut être affecté par l'utilisateur

La pertinence d'un document vis-à-vis d'une requête est définie par des mesures de similarité dans l'espace vectoriel. Le mécanisme de recherche consiste à retrouver les vecteurs documents qui sont les plus proches du vecteur requête.

Il existe plusieurs fonctions permettant de mesurer la similarité entre deux vecteurs :

Le produit scalaire: $$\boldsymbol{Sim}\,(\boldsymbol{d_i}, \boldsymbol{q}) = \sum_{j=1}^{n} \boldsymbol{w_{qi}} * \boldsymbol{w_{ij}}$$

La mesure du cosinus: $$\boldsymbol{Sim}\,(\boldsymbol{d_i}, \boldsymbol{q}) = \frac{\sum_{j=1}^{n} \boldsymbol{w_{qi}} * \boldsymbol{w_{ij}}}{\left(\sum_{j=1}^{n} \boldsymbol{w_{qi}^2}\right)^{1/2} * \left(\sum_{j=1}^{n} \boldsymbol{w_{ij}^2}\right)^{1/2}}$$

La mesure de Dice: $$\boldsymbol{Sim\ (d_i, q) = \frac{2 * \sum_{j=1}^{n} w_{ij} * w_{qi}}{\sum_{j=1}^{n} w_{qi}^2 + \sum_{j=1}^{n} w_{ij}^2}}$$

Le coefficient de superposition: $$\boldsymbol{Sim\ (d_i, q) = \frac{\sum_{j=1}^{n} w_{ij} * w_{qi}}{\sum_{j=1}^{n} w_{qi}^2 + \sum_{j=1}^{n} w_{ij}^2 - \sum_{j=1}^{n} w_{ij} * w_{qi}}}$$

La mesure de Jacard: $$\boldsymbol{Sim\ (d_i, q) = \frac{\sum_{j=1}^{n} w_{ij} * w_{qi}}{min_i\left(\sum_{j=1}^{n} w_{qi}^2, \sum_{j=1}^{n} w_{ij}^2\right)}}$$

Ce modèle présente l'avantage de la simplicité de conception et à l'inverse du modèle booléen, les résultats peuvent satisfaire la requête partiellement et donc être ordonnés par ordre de pertinence décroissante. Cependant, ce premier modèle vectoriel ne considère pas les éventuels liens qui peuvent exister entre les termes. Le modèle vectoriel généralisé (Generalized Vector Space Model) [Wong & al., 85] viendra pallier cet inconvénient.

1.7.3 Modèles probabilistes :

a. Modèle Probabiliste de Base

Le premier modèle probabiliste a été proposé par Maron et Kuhns [Maron & al., 60] au début des années 1960. C'est un modèle mathématique fondé sur la théorie des probabilités [Robertson & al., 76][Salton & al., 83c] [Maron & al., 60]. Son principe de base consiste à retrouver des documents qui ont une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents.

En évaluant la pertinence d'un document d pour une requête Q, un document d est sélectionné si la probabilité que le document d soit pertinent, notée $p(d,\ Q)$, est supérieure à la probabilité que d soit non pertinent pour une requête q, notée $p(\bar{d},\ Q)$. Le score d'appariement entre le document d et la requête Q est noté RSV(d, Q). Plus ce score est élevé pour un document, plus ce document est pertinent à la requête. Ce score est donné par [Robertson, 94b]:

$$\boldsymbol{RSV\ (d, Q) = \frac{p(d,\ Q)}{p(\bar{d},\ Q)}}$$

Plusieurs solutions ont été proposées pour représenter le document *d* et pour estimer les paramètres du modèle. Parmi elles citons *BIR* (*Binary Independance Retrieval*) qui suppose l'indépendance des termes [Robertson & al., 76b].

On considère dans ce modèle que chaque document est représenté par un vecteur $\boldsymbol{d(w_1,\ldots w_k)}$ dont les composantes sont les pondérations des index.

Où

$\boldsymbol{w_i = 1}$ si terme est présent dans le document et 0 sinon.

Après quelques transformations, on arrive à la formule classique proposée par Robertson et

Sparck Jones en 1976 :

$$\boldsymbol{RSV(d,Q) = \sum_{i=1}^{k} x_i * w_i = \sum_{i=1}^{k} x_i * log \frac{r_i/(n-r_i)}{(R_i-r_i)/(N-R_i-n+r_i)}}$$

Avec

xi = 0 ou 1, représente l'absence ou la présence du terme t_i de la requête *Q* dans le document *d*

w_i représente le poids affecté à t_i dans le document *d* qui est estimé en fonction de N, R_i, r_i et n.

N est égal au nombre total des documents dans la collection

Ri est égal au nombre total des documents contenant le terme *ti*

r_i *est* égal au nombre de document pertinent contenant le terme *ti*

n est égale au nombre de documents pertinents.

Pour éviter les 0, un lissage de cette fonction est proposée par Robertson-Sparck Jones qui est souvent utilisé dans des approches probabilistes en RI. Elle consiste à déterminer le poids d'un terme t_i :

$$w_i = \log \frac{(r_i + 0.5)/(n - r_i + 0.5)}{(R_i - r_i + 0.5)/(N - R_i - n + r_i + 0.5)}$$

Ce modèle classe les documents retrouvés selon l'ordre de pertinence. Cependant il fait face à un inconvénient majeur qui est la difficulté de calcul des probabilités conditionnelles voire l'impossibilité d'estimer ses paramètres si des collections d'entrainement ne sont pas disponibles. Ce modèle a donné lieu à de nombreuses extensions. Il est à l'origine du système Okapi développé par [Robertson & al., 94].

Les Réseaux inférentiels Bayésiens sont une autre extension du modèle probabiliste. En effet ils reprennent les éléments de l'approche probabiliste en tenant compte de la dépendance des termes d'indexation.

b. Modèle de Langage

Le modèle de langage est basé sur l'hypothèse suivante : lorsque un utilisateur a un besoin en information, il formule une requête adressée au SRI en pensant à un ou plusieurs documents qu'il souhaite retrouver. La requête est alors inférée par l'utilisateur à partir de ces documents. Ce modèle considère alors que la pertinence d'un document pour une requête est en rapport avec la probabilité que la requête puisse être générée par le document [Ponte & al., 98] [Boughanem & al., 04].

La probabilité que la requête soit inférée par le document $P(Q/d)$ est estimée par :

$$\boldsymbol{P(Q/d) = \prod_{i=1}^{n} P(t_i/d)}$$

Où *n* est le nombre de termes dans la requête

t_i est un terme de la requête pour *i=1,2,…,n*

L'estimation maximale de vraisemblance (maximum likelihood estimation) permet d'estimer $P(t_i/d)$ tel que:

$$\boldsymbol{P(t_i/d) = \frac{tf(t_i/d)}{\sum_t tf(t/d)}}$$

ou *tf* (t_i/d) est la fréquence du terme t_i dans le document d.

Afin de pallier le problème des termes de la requête absents des documents, (ceci conduirait systématiquement à $P(Q/d) = 0$), différentes technique de lissage ont été développées comme : le lissage de Laplace, le lissage de Good-Turing ou le lissage de Backoff [Boughanem & al., 04]. Elles consistent a assigner des probabilités non nulles aux termes, qui n'apparaissent pas dans les documents.

1.8 Reformulation de requête :

La reformulation de requêtes est un processus ayant pour objectif de générer une nouvelle requête plus adéquate que celle initialement formulée par l'utilisateur, en rajoutant de nouveaux termes et/ou supprimant des termes inutiles [Zemirli, 08]. La nouvelle requête est sensée mieux répondre au besoin informationnel de l'utilisateur.

Nous distinguons principalement deux catégories de reformulation de requête : *La reformulation automatique et la reformulation interactive :*

La reformulation automatique

C'est l'une des premières techniques ayant produit des améliorations notables dans ce domaine. Son principe est d'ajouter à la requête q initiale des termes sémantiquement proches issus d'un thesaurus [Voorhees, 94] [Brajnik & al., 96], ou d'une ontologie linguistique (telle que WordNet [Miller, 95]).

La reformulation interactive de requête

Appelée aussi relevance feedback ou retour de pertinence [Rocchio, 71], [Robertson, 90], [Hains & al., 93], [Harman, 92a] [Boughanem & al., 97], [Boughanem & al., 99] ,ce mécanisme suppose que la requête de l'utilisateur fournit un ensemble de documents que l'utilisateur évalue. Une nouvelle requête est ensuite générée à partir de ces jugements en ajoutant aux termes de la requête initiale des termes extraits de documents jugés pertinents par l'utilisateur. Notons que la nouvelle requête augmente le poids des termes importants. Elle est obtenue à partir de la requête initiale en appliquant un algorithme spécifique de réinjection de pertinence comme l'algorithme de Rocchio [Salton & al., 83][Salton, 89].

1.9 Evaluation d'un SRI :

L'évaluation constitue une étape importante lors de la mise en œuvre d'un modèle de recherche d'information puisqu'elle permet de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et enfin de fournir des éléments de comparaison entre modèles. On peut évaluer un système selon le critère quantitatif (combien de documents peuvent être indexés, quel est le temps de réponse maximum à une requête?) ou selon le critère qualitatif (quelle est la pertinence des réponses ?). Le plus important pour un système est de ne retourner à l'utilisateur que les documents pertinent. A cet effet, [Kent & al., 55] furent les premiers à proposer les critères de pertinence et les mesures de Rappel et de Précision pour l'évaluation des SRI qui sont définis comme suit :

Le ***Rappel*** est défini par rapport entre le nombre de documents pertinents retrouvés et le nombre de documents pertinents de la requête. Le rappel est la mesure duale du *Silence* **(*silence = 1 - Rappel*)**.

La ***Précision*** est le rapport entre le nombre de documents pertinents retrouvés sur le nombre de documents total retournés par le moteur de recherche pour une requête donnée. La précision est la notion opposée au *Bruit* **(*bruit = 1-Précision*).**

Formellement, soient :

A : le nombre de documents retournés par un système.

R : le nombre de documents pertinents dans la collection.

Ra : le nombre de documents pertinents renvoyés par le système.

$$Rappel = {}^{Ra}/_{R}$$

$$Précision = {}^{Ra}/_{A}$$

Un système est dit *précis* si peu de documents inutiles sont proposés par le système, ce qui signifie que le taux de *précision* est élevé. En pratique, le plus souvent on obtient un taux de rappel et de précision aux alentours de 30%.

1.10 Conclusion :

Au cours de ce chapitre nous avons donné une vue d'ensemble sur la RI et les SRI. Nous avons présenté les concepts de base et étudié les modèles les plus connus.

Les premières générations de SRI n'étant basés que sur des formalismes mots-clés étaient insensibles aux différences de sens entre les mots. la correspondance entre termes était limitée à une correspondance lexicale. Ceci se répercute sur les résultats retournés qui ne pouvaient tenir compte du sens accordé par l'utilisateur à sa requête.

La recherche d'information tente sans cesse d'explorer de nouveaux horizons affin d'améliorer le résultat de ses recherches. Plusieurs travaux ont tenté d'intégrer le sens des mots dans la représentation de documents en utilisant des techniques de désambiguïsation du sens. Ceci a donné naissance à la RI Sémantique qui fera l'objet du chapitre suivant.

Chapitre II

Indexation Sémantique

II.1 INTRODUCTION :

Dans le but d'améliorer la qualité des SRI, les chercheurs ont proposé d'utiliser une indexation autre que l'indexation classique, appelée *indexation sémantique.* L'indexation sémantique s'intéresse principalement à la représentation des documents et requêtes par les sens des mots qu'ils contiennent plutôt que par les mots eux-mêmes.

L'objet du présent chapitre est de présenter les concepts de base de l'indexation sémantique, ainsi que les approches mises en œuvre dans ce cadre. Mais avant, nous posons la problématique de l'indexation classique.

II.2 PROBLEMATIQUE :

Classiquement, l'indexation permet de représenter le contenu informationnel des documents et des requêtes par des termes d'indexation issus de leur contenu. La similarité requête-document se limite à une correspondance lexicale entre les termes d'indexation

issus de la requête et ceux issus des documents. Ces dernières années, beaucoup de travaux ont souligné l'insuffisance de cette représentation basée sur des mots simples [Robertson & al., 97] [Woods, 97] [Khan, 00] [Guarino & al., 99]. En effet, se basant en général sur la fréquence d'occurrence des mots dans un texte, cette indexation ne peut prendre en charge la complexité de la langue naturelle telle que *l'Ambigüité* des mots et leur *Disparité* [Amirouche, 08] qui sont les deux inconvénients majeurs de cette indexation.

L'*Ambigüité des mots* de la langue naturelle est due aux différents sens attribués à un mot selon son contexte. Ainsi deux mots peuvent s'écrire exactement de la même marnière sans pour autant avoir le même sens. [Krovetz, 97][Krovetz & al., 92] distinguent deux types d'ambigüité : l'ambiguïté syntaxique et l'ambiguïté sémantique.

l'Ambiguïté Syntaxique : due à l'agencement des mots dans la phrase. Par exemple, le mot anglais « Book » peut être un « nom » avec pour sens « Livre », ou être un « verbe » s'il est précédé de « to » et avoir comme sens donc « le fait de lire » ou même « le fait d'effectuer une réservation» par exemple dans la phrase *to book a cruise*.

L'Ambiguïté Sémantique (ou Lexicale): est engendrée par la pluralité des sens que peut posséder un mot de la langue naturelle. Selon [Krovetz, 97], ces sens peuvent être liés (polysémie) ou pas (l'homonymie). pour illustrer ce type d'ambigüité, on peut prendre l'exemple d'une requête portant sur l'outil informatique « mouse » effectuée sur le moteur de recherche *google*[3]. Certains résultats obtenus traitent de l'animal « mouse » et non du matériel informatique.

[3] http://www.google.fr

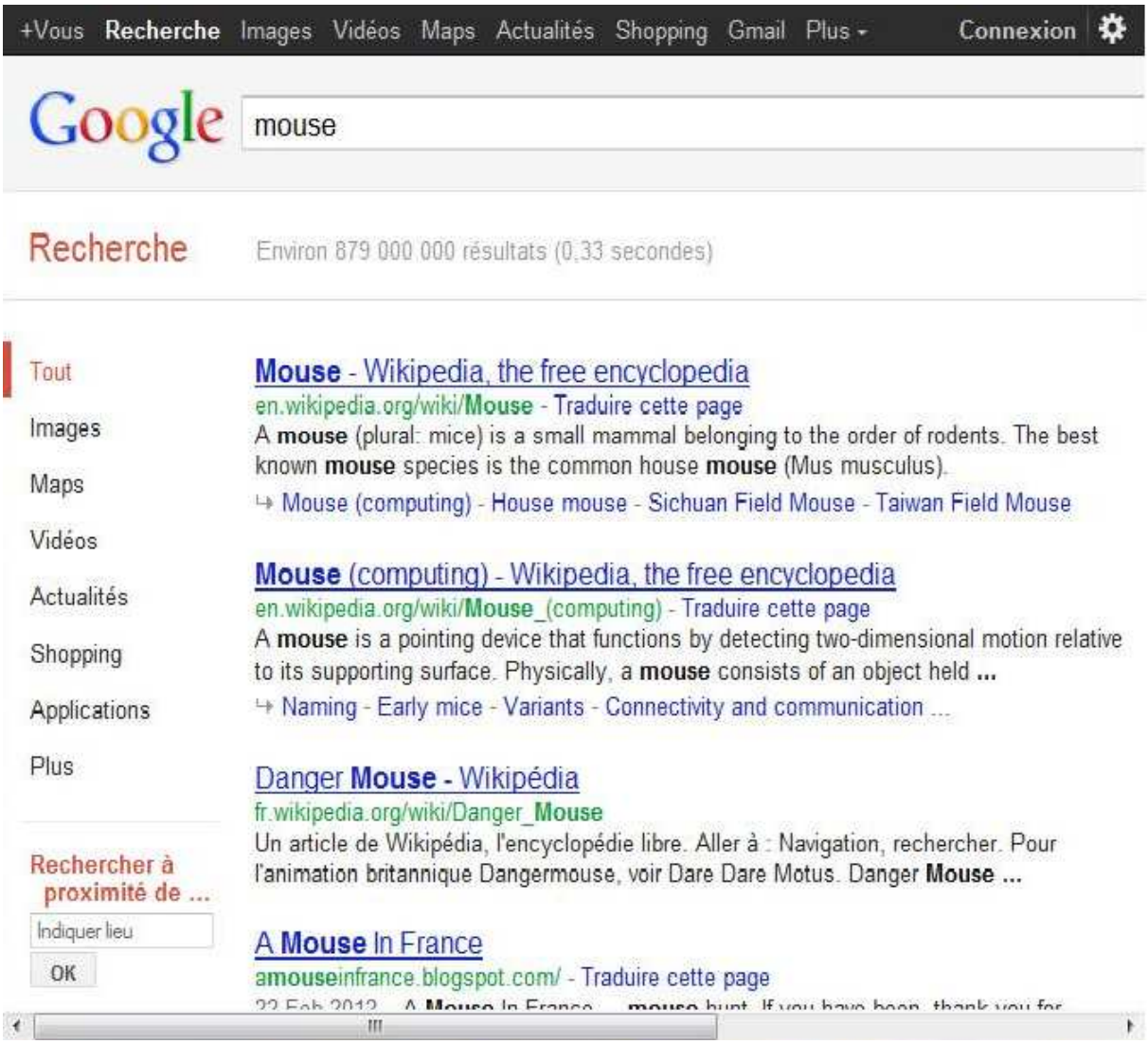


Figure 2.1 exemple de requête « mouse » sur Google

L'ambigüité des mots se répercute sur la qualité de notre SRI en retournant des documents non pertinents. En effet, les documents contenants les mots recherchés seront retournés même s'ils n'ont pas le même sens avec les termes de la requête.

La Disparité des mots : appelée aussi Word Mismatch, est un problème engendré par richesse du vocabulaire d'une langue naturelle. En effet, il existe des mots certes différents d'un point de vue lexical mais équivalents d'un point de vue sémantique (synonymie). Ceci affecte notre SRI en engendrant le problème inverse de l'ambigüité des mots, à savoir, de ne pas retourner des documents pertinents, tout simplement car même si les termes d'une requête et d'un document correspondent sémantiquement, ils ne correspondent pas lexicalement. On peut citer pour exemple une requête portant sur « une bicyclette » même si des documents pertinentes portants sur la notion de « vélo » existent dans la collection, ils ne seront pas retournés à cause de la non correspondance lexicale entre les deux termes « vélo » et « bicyclette ».

Pour pallier aux problèmes de l'indexation classique, plusieurs solutions ont été proposées comme l'indexation par des l'expansion de la requête à l'aide de mots synonymes d'un thésaurus [Salton & al., 83a]. Ces technique permettent de contourner partiellement le problème de la synonymie, l'amélioration qu'elles réalisent sur le rappel se fait en général au détriment de la précision et le problème de la polysémie subsiste toujours [Baziz, 05].

Une autre solution propose d'explorer l'exploitation de la sémantique des textes dans la représentation de l'information. Plusieurs travaux tentant d'incorporer l'information sémantique dans le processus de RI sont alors apparus. Au sein de ces travaux, deux grandes approches peuvent être distinguées [Mihalcea & al., 00] [Biemann, 05] : l'indexation sémantique et l'indexation conceptuelle.

- ✓ L' indexation sémantique est basée sur le sens des termes [Mihalcea, 04], et utilise des techniques de désambigüisation des mots, WSD (Word Sense Disambiguation), pour indexer les documents et les requêtes. En effet, cette technique consiste d'abord à *retrouver le sens correct de chaque mot dans le document* (ou la requête), ensuite le (la) *représenter*.

  Pour retrouver le sens correct d'un mot ambiguë, on se base sur son contexte. Le contexte d'un mot ambigu fait référence à ses voisins, prédécesseurs et successeurs dans la phrase ou le paragraphe dans un document à indexer. La représentation peut être soit basée sur les sens (les sens sont utilisés comme termes d'indexation) ou être une représentation combinée mots-clés/sens (les termes d'indexation sont des couples [mot-clé, sens associé]) .

  Il existe deux principales approches de désambiguïsation du sens: les approches endogènes et les approches exogènes [Audibert, 03] auxquelles sont associées respectivement deux approches d'indexation : les approches basées sur le corpus et approches basées sur les ressources externes (Bases de connaissance).

- ✓ L'indexation conceptuelle quant à elle consiste à indexer un document, non plus avec les termes des documents, mais avec les concepts d'une base de connaissances. La base de connaissances peut être un dictionnaire, une ontologie, ou même un thésaurus). L'indexation conceptuelle a pour vocation de s'affranchir de la vision purement fréquentielle ou statistique de l'indexation classique [Gibbs, 87]. L'objectif de l'indexation conceptuelle est d'extraire les concepts et de les choisir comme descripteurs du document.

  Il est important de noter que l'indexation sémantique et l'indexation conceptuelle ne s'excluent pas mutuellement. Les méthodes d'indexation conceptuelle utilisent celles de l'indexation sémantique pour lever les ambiguïtés avant de choisir les meilleurs concepts décrivant le document [Abi Chahine, 11].

II.3 LES DIFFERENTES RESSOURCES SEMANTIQUES :

Selon le dictionnaire de l'académie française, "Le concept regroupe les objets qu'il définit en une même catégorie appelée « classe » ". De façon général, le terme concept est souvent utilisé comme se référant à toute notion, de l'idée au lexème, en passant par l'entité et la catégorie.

Les concepts sont le plus souvent prédéfinis et pré-ordonnés dans des structures conceptuelles telles les ontologies. Ces structures sont souvent construites manuellement par des spécialistes du domaine qu'elles couvrent. Un concept, qui correspond à un nœud de cette structure tel que définis dans le thesaurus médical MeSH et l'ontologie de domaine WordNet. Dans WordNet un nœud est appelé aussi Sysnet.

II.3.1 DICTIONNAIRE :

Le dictionnaire est la base de connaissances la moins expressive. Il consiste en une collection de termes (un seul mot ou une séquence de mots) n'ayant aucun lien entre eux. En RI nous parlons de dictionnaires informatisés ou MDR. L'un des dictionnaires les plus connus pour la langue anglaise est le OALD (Oxford Advanced Learner's Dictionnary of Current English).

II.3.2 RESEAUX SEMANTIQUES :

En RI, un réseau sémantique peut être décrit comme toute représentation reliant des nœuds avec des arcs, où les nœuds sont des concepts et les arcs représentent différents types de relations entre concepts" [Lee et al., 93][Woods, 97].

Voici un exemple de : « Alexis » est un « Homme » , « Alexis » possède une « BMW »

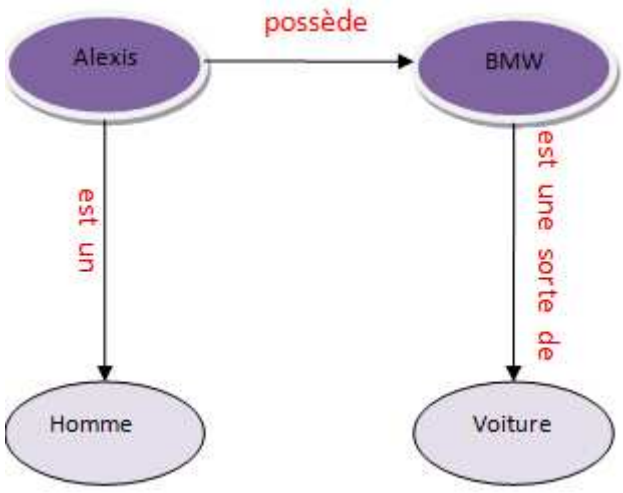


Figure 2.2 exemple de Réseau sémantique

II.3.3 TAXINOMIE :

La taxinomie est un réseau sémantique où l'unique relation est une relation hiérarchique, transitive et non réflexive. Souvent, la relation hiérarchique « est_un » (en anglais, «is_a ») est utilisée et on parle de classification. Les concepts sont appelés des taxons. Un exemple classique de taxinomie est celle qui décrit les organismes vivants. L'avantage de l'inférence dans les taxinomies est mis à mal par le faible pouvoir d'expressivité de cette représentation.

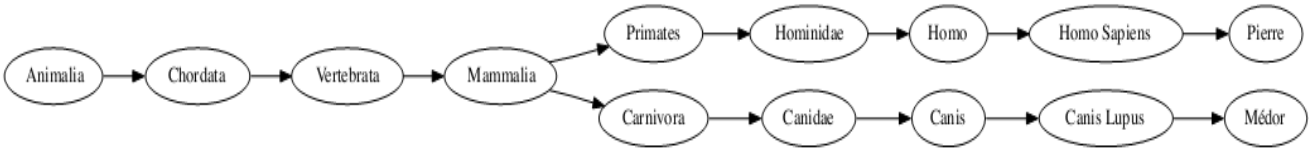


Figure 2.3 Classification Biologique tirée de Wikipédia

II.3.4 THESAURUS :

Un thésaurus, selon [Nongdo, 08], est "*une liste structurée de concepts, destinés à représenter de manière univoque le contenu des documents et des questions dans un corpus donné*" Les thésaurus contiennent une liste de termes vedettes utilisables comme termes d'indexation et l'ensemble des termes reliés, par une relation de synonymie, de généricité/spécificité ou une relation " est lié à ". Un thésaurus peut être construit soit manuellement soit de manière automatique. Il existe des thésaurus spécialisés tel que UMLS qui est spécialisé dans le domaine médical, MeSH.

II.3.5 ONTOLOGIE :

Selon Guarino, [Guarino, 99] : « l'ontologie est le terme utilisé pour désigner une compréhension partagée d'un domaine donné ».

De façon générale, le mot "ontologie" est utilisé abusivement pour renvoyer à des structures lexicales et sémantiques variées. Par exemple, les dictionnaires, les thesaurus de la communauté de l'informatique linguistique ; WordNet, l'UMLS, le MeSH, etc.

II .4 L'INDEXATION CONCEPTUELLE :

L'indexation conceptuelle utilise des concepts issus d'une base de connaissance -d'une ontologie dans la majorité des cas- pour indexer les documents [Aussenac & al., 04] [Guarino & al., 99]. Selon [Kompaoré, 08] cette approche regroupe les termes ayant des caractéristiques communes dans les documents, et considère les regroupements comme des unités de sens ou concepts. Les résultats obtenus par [Woods et al., 98] montrent une

amélioration des mesures de rappel et de précision par rapport à un SRI classique, lorsqu'une ressource de type ontologie est utilisée.

Dans [Woods, 97], Woods propose une approche d'indexation conceptuelle se référant à la construction de taxonomies conceptuelles à partir des textes. Cette méthode se déroule en deux étapes : d'abord l'extraction des concepts, ensuite leur structuration en hiérarchie.

Le système est capable de transformer les syntagmes pour qu'ils correspondent à des concepts du réseau sémantique. Par exemple, dans la phrase «la voiture est repeinte en noir», le syntagme «repeinte en noir» va être associé au concept présent dans le réseau «changement de couleur» et sera utilisé pour l'indexation. Il s'agit bien d'indexation conceptuelle, car ce ne sont plus les termes désambiguïsés qui indexent le document et la requête, mais bien les concepts du réseau.

Woods a testé son approche sur de petites collections de texte (dont les pages du manuel UNIX composé de 1819 fichiers et occupant une taille d'environ 10MB). En comparant ses résultats aux résultats des SRI classiques, Woods constaté une amélioration du rappel de l'ordre de 0.3%.

II .5 L'INDEXATION SEMANTIQUE BASEE SUR LA DESAMBIGUÏSATION :

Selon [Amirouche, 08] l'indexation sémantique s'intéresse à la représentation des documents et requêtes par les sens des mots qu'ils contiennent. Cette indexation peut nécessiter au préalable une étape de désambigüisation des sens des mots appelée WSD (Word Sense Desambiguiation) .

Selon [Pustejovsky, 95], *un mot peut avoir une infinité de sens selon le contexte*. Ceci est dû aux ambigüités de la langue naturelle qui se répercutent sur la pertinence des documents retournés par le processus d'indexation. Ainsi des documents pertinents peuvent ne pas être retournés et des documents non pertinents retournés car l'aspect sémantique n'est pas pris en charge par le processus d'indexation classique basée mot-clé.

[Krovetz, 93], ont initié l'un des premiers travaux portants sur l'évaluation de l'apport de connaissances sémantiques dans un système de recherche d'information. Plusieurs travaux s'en sont suivis [Voorhees, 93] [Sanderson, 94] [Krovetz, 97] [Gonzalo & al., 98] [Katz & al., 98] avec des avis plus ou moins divergents sur l'apport de cette technique. Ainsi [Sanderson, 94] [Krovetz, 97] [Gonzalo & al., 98] ont montré que l'impact de l'ambiguïté des sens sur l'efficacité de la recherche n'était pas dramatique, mais qu'une désambiguïsation précise (précision de plus de 90% selon [Sanderson, 94], de 60% selon [Gonzalo & al., 99]) des mots améliorerait probablement l'efficacité de la recherche lorsque peu de mots de la requête apparaissent dans le document.

Pour Voorhees [Voorhees, 93] et [Katz & al., 98], l'utilisation du sens des mots avec un désambiguïseur automatique n'améliore pas sensiblement les résultats de la recherche.

Les auteurs dans [Moldovan & al., 00] , trouvent une amélioration de 16% pour le rappel et de 4% pour la précision lorsqu'ils combinent l'indexation basée sur la désambigüisation des mots par le lexique WordNet et l' indexation basée sur les mots clés. Ces résultats montrent une amélioration non négligeable qu'il faut exploiter.

II .6 LES APPROCHES D'INDEXATION SEMANTIQUE :

Selon que l'indexation soit basée sur des méthodes de désambiguïsation *endogène* ou *exogène*, nous pouvons distinguer deux types d'approches d'indexation sémantique

II .6 .1 LES APPROCHES D'INDEXATION SEMANTIQUE BASEE SUR LA DESAMBIGÜISATION ENDOGENE :

Elles se basent sur l'utilisation des corpus pour construire les connaissances nécessaires à la désambiguïsation. Les mots d'index sont ensuite identifiés dans la collection à indexer, puis désambiguïsés. . Le principe général consiste à désambiguïser un mot polysémique à l'aide de calculs statistiques sur les instances de ce mot dans le corpus et les différents contextes dans lequel il est employé. Les textes de la collection (et des requêtes) sont indexés en utilisant les sens ainsi retrouvés. L'approche de Schütz et Pedersen [Schütz & al., 95] est un exemple typique de ce genre d'approches.

II .6 .2 LES APPROCHES D'INDEXATION SEMANTIQUE BASEE SUR LA DESAMBIGUÏSATION EXOGENE :

Ce genre d'approches se base sur des ressources externes comme moyen de désambiguïsation du sens et non sur un apprentissage sur le corpus.

Lors de cette indexation, après extractions des termes descripteurs d'un document, ces termes sont désambigüisés en utilisant une ressource sémantique externe. Le plus souvent on fait référence à des ontologies pour retrouver et désambigüiser les différents sens associés à chaque terme. Ensuite, des scores sont associés aux différents sens retrouvés sur la base de :

- La distance sémantique de ce sens aux différents sens associés aux autres termes dans le document (contexte global) [Baziz & al., 04]

- Le degré de recouvrement entre d'une part, le contexte local de ce mot et d'autre part le voisinage [Voorhees, 93] de ce sens ou la définition de ce sens (ensemble de synonymes) [Katz & al., 98]dans la ressource linguistique utilisée.

Le sens retenu est celui qui maximise le score. Une fois les termes d'indexation désambiguïsés, la représentation des textes indexés se fait soit à partir des sens ou concepts seuls, soit à partir d'une combinaison des mots-clés et sens corrects associés.

Dans ce qui suit nous allons présenter les principaux travaux relatifs à ces deux types d'approches.

II .6. 3 APPROCHES D'INDEXATION SEMANTIQUE :

II .6. 3. 1 APPROCHE DE SCHÜTZ & PEDERSEN :

Schütz & Pedersen ont proposé une désambiguïsation basée seulement sur le corpus. Ainsi pour chaque mot à désambiguïser, on examine le contexte de chaque occurrence de ce mot dans le corpus. Les contextes similaires sont regroupés. Chacun de ces contextes similaires représente un sens individuel pour ce mot que Schütz & Pedersen désignent par usage de mot (word uses). Affin de n'identifier que les sens fréquents d'un mot, un contexte similaire n'est identifié comme étant un sens que s'il apparaît plus d'une cinquantaine de fois dans le corpus, éliminant ainsi les sens les moins fréquents.

Pour désambiguïser une occurrence d'un mot, la technique consiste à classer les usages possibles d'un mot selon un score basé sur le recouvrement entre son contexte et les contextes d'usage. Une fois les mots désambiguïsés, la construction de leur index s'est faite de trois manières différentes. Dans le premier cas, une occurrence d'un mot est représentée simplement par le mot (le cas basique). Dans le second cas, par l'usage de mot le mieux classé. Et enfin, par une combinaison du mot et des n premiers usages de mots les mieux classés.

Leurs test se sont effectués sur une collection relativement petite, qui est la collection TREC-1 catégorie B. Ils n'ont utilisé à cet effet que 25 requêtes en raison de la complexité du calcul lors du regroupement des contextes similaires. Les résultats obtenus sont de l'ordre de 14 % de gain dans la précision. Le meilleur résultat correspond à la dernière représentation quand le nombre des premiers usages de mots utilisés est trois (3).

Selon [Baziz, 05], ce gain de précision élevé serait peut être dû au choix de la collection de test. En effet, vu que cette collection utilise des requêtes longues de l'ordre de 100 mots, que l'outil de désambiguïsation serait peut être capable d'identifier automatiquement les usages des mots des requêtes.

II .6. 3. 2 APPROCHE DE BAZIZ :

Baziz et al. [Baziz & al., 04 ; 05] dans une approche dite DocCore, proposent une technique d'indexation sémantique des documents à base de concepts et de relations entre concepts.

Les termes d'indexation sont d'abord extraits du document (indexation classique), puis ils sont projetés sur l'ontologie linguistique WordNet[4] afin d'identifier les concepts (ou sens) correspondants dans l'ontologie. Si un terme d'indexation apparie plus d'un concept dans WordNet alors ce terme est ambigu, il faut le désambiguïser. L'approche de désambiguïsation proposée est basée sur le principe que, parmi les différents sens possibles (dits concepts candidats) d'un terme donné, le plus adéquat est celui qui a le plus de liens avec les autres concepts du même document. Formellement, l'approche consiste à affecter un score à chaque concept candidat d'un terme d'indexation donné. Le score d'un concept candidat est obtenu en sommant les valeurs de similarité qu'il a avec les autres concepts candidats (correspondant aux différents sens des autres termes du document). Le concept candidat ayant le plus haut score est alors retenu comme sens adéquat du terme d'indexation associé. Finalement, le document est représenté comme un réseau de concepts et de liens entre concepts.
L'approche d'indexation sémantique ainsi proposée a été évaluée d'une part dans le cadre de la collection de test MuchMore [Buitelaar & al., 04], d'autre part dans le cadre de la campagne CLEF 2004. Dans les deux cas, un SRI basé sur le modèle connexionniste est utilisé [Boughanem & al., 92]. Les résultats rapportés montrent que l'utilisation des sens (concepts de WordNet) seuls pour représenter les documents ne permet pas d'améliorer les résultats comparativement à la méthode classique basée sur les mots clés. Cependant, la combinaison de l'indexation classique et de l'indexation sémantique apporte une nette amélioration de la précision.

II .6. 3. 3 APPROCHE DE VOORHEES :

Voorhees [Voorhees, 93] a construit un outil de désambiguïsation basé sur WordNet. Elle est partie du principe *qu'un groupe de mots utilisés dans un certain contexte a un sens précis (non ambigu) même si les mots qui le composent sont isolément ambigus.*

L'approche d'indexation de Voorhees, se base sur le contexte local. la phrase représente le contexte local de chacun de mots. les textes à indexer sont donc analysés phrase par phrase. Pour chaque mot non vide rencontré dans la phrase, on recherche dans WordNet le (ou les) synset(s) qui lui correspondent (un mot ambigu correspond à plusieurs synsets dans Wordnet).

[4] voir annexe

Pour déterminer le synset (sens) adéquat pour un mot ambigu dans une phrase, chaque synset de ce mot est classé en se basant sur le nombre de mots co-occurrents entre un voisinage (Voorhees l'a appelé hood) de ce synset et le contexte local du mot ambigu correspondant. Le synset le mieux classé est alors considéré comme sens adéquat du mot ambigu.

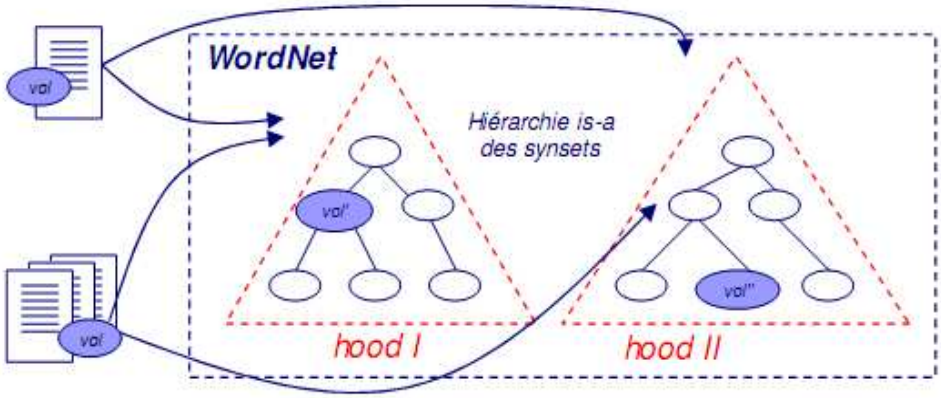


Figure 2.4 schéma de l'utilisation des hood dans WordNet

Voorhees [Voorhees, 93] a utilisé les relations is-a (hyponymie/hyperonymie) de WordNet pour faire la désambiguïsation des mots. Elle définit le voisinage d'une synset comme suit :

"*Pour définir le voisinage d'un synset s , considérons l'ensemble des synsets et les relations d'Hyperonymie et d'Hyponymie dans WordNet comme les sommets et les arcs dirigés d'un graphe. Le voisinage d'un synset donné s , serait alors le plus large sous-graphe connexe qui :*

- *contient s*
- *contient seulement les descendants d'un ancêtre de s*
- *ne contient aucun synset ayant un descendant qui inclut une autre instance d'un membre (c-à-d, un mot) de s* ."

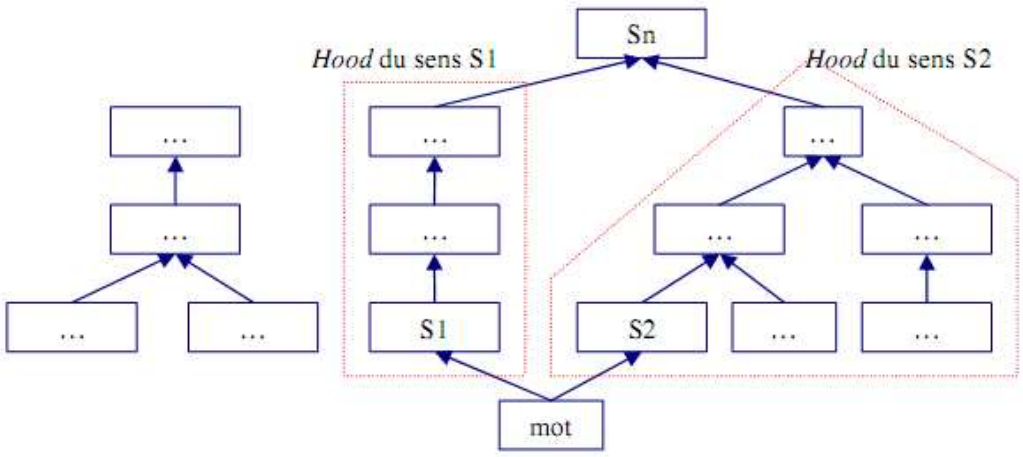


Figure 2.5 Exemple d'un hood (voisinage de mot) selon Voorhess.

Dans l'exemple illustré dans la figure précédente, pour les sens S1 et S2, La racine de la hiérarchie que les deux sens partagent n'appartient pas au hood à cause de la règle (3). La hiérarchie à gauche n'appartient pas à aucun des hoods pour la règle (2).

Voorhees a expérimenté cette approche sur les collections de test CACM [Salton & al, 83a], CISI, Cranfield 1400, MEDLINE, et Time. Elle a constaté que l'étiquetage avec les sens tel qu'il est réalisé n'est pas exact et cause une dégradation des performances du système pour la plupart des requêtes. De plus, pour un certain nombre de requêtes courtes, elle a trouvé que l'outil de désambiguïsation était incapable de déterminer de façon exacte le sens attendu des mots dans les requêtes. Elle suppose que l'exploitation des deux relations de WordNet n'est pas suffisante pour retrouver les sens corrects des mots.

II .6. 3. 4 APPROCHE DE KATZ ET AL. :

Dans une approche similaire à celle de Voorhees, Katz et al. [Katz & al., 98] analysent les textes à indexer mot par mot. Chaque mot non vide rencontré est projeté sur WordNet dans l'objectif d'identifier le (ou les) synset(s) correspondant(s). Si un mot apparie plusieurs synsets, il est ambigu.
Pour désambiguïser, Katz et al proposent aussi une approche basée sur le contexte local. Le contexte local d'un mot est défini comme étant la liste ordonnée des mots démarrant du mot utile le plus proche du voisinage gauche ou droit jusqu'au mot cible.
L'hypothèse de Katz et al., est que des mots utilisés dans le même contexte local (appelés sélecteurs), ont souvent des sens proches. Les sélecteurs des mots d'entrée sont extraits des contextes locaux gauche et droit, puis l'ensemble S de tous les sélecteurs obtenus est comparé avec les synsets de WordNet. Le synset qui a le plus de mots en commun avec S est sélectionné comme sens adéquat du mot cible.
La figure illustre le processus d'extraction de sélecteurs de "charge" dans une phrase donnée. La désambiguïsation de "charge" se fait dans deux contextes différents. En comparant les sélecteurs des mots d'entrée avec les synsets de WordNet, le sens de "charge" dans le premier contexte est associé (matché) avec le sens 3 de WordNet "charge, bill" et le deuxième sens est apparié avec le sens 4 de WordNet "appoint, charge".

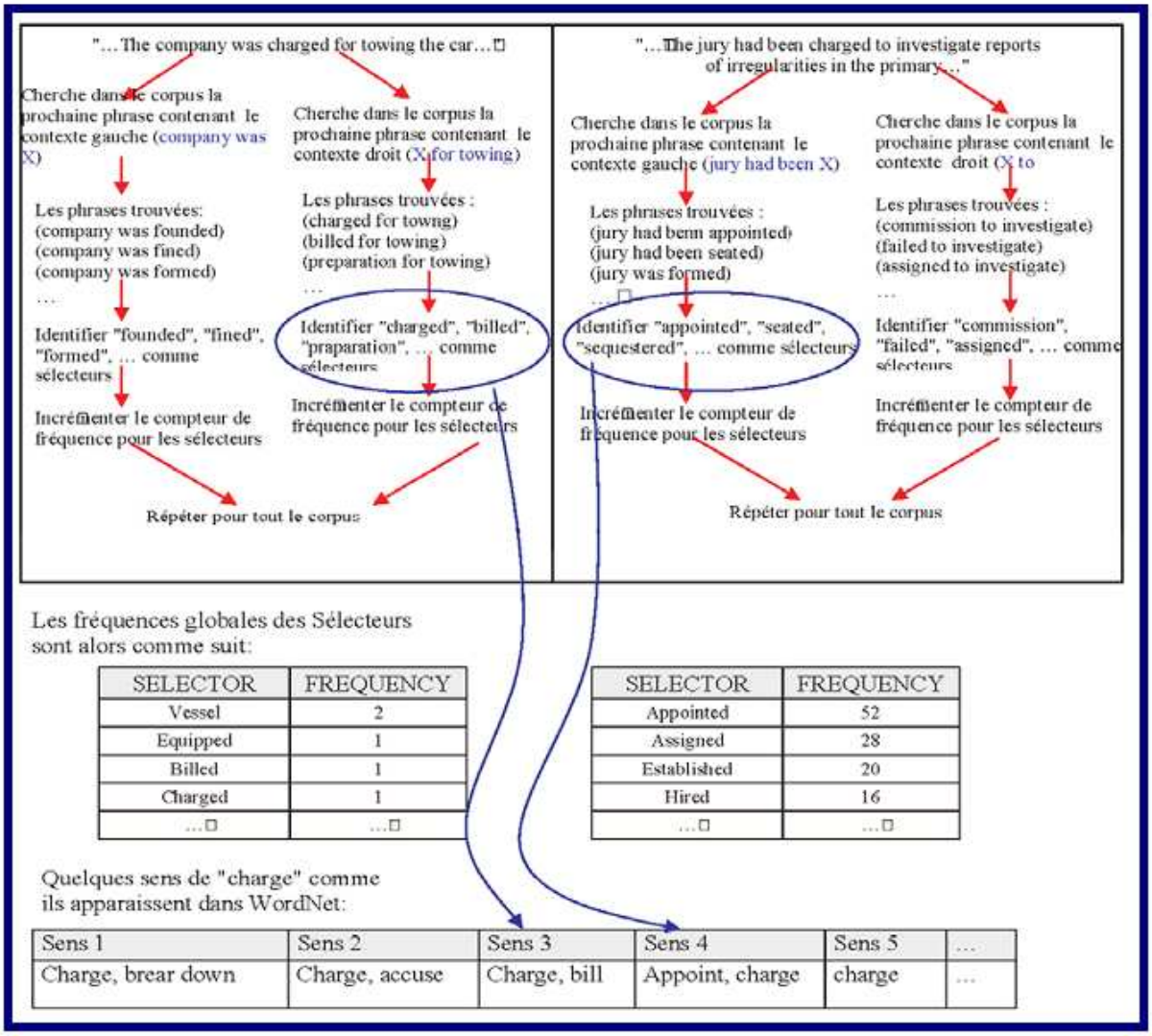


Figure 2.6 : Désambiguïsation de termes selon l'algorithme de Katz

Katz et al. ont testé leur désambiguïseur sur le corpus Semcor. La précision rapportée est de 60%. En incorporant ce désambiguïseur au système SMART, Katz et al. ont rapporté que leur algorithme n'améliorait pas les performances du système. Ceci pourrait être du aux erreurs de désambiguïsation.

Il existe bien d'autres exemples d'approches basées sur des ressources externes pour la désambiguïsation. Dans ce qui suit nous nous intéresserons spécialement à celles utilisant des ressources externes pour désambiguïser le domaine du discours.

II .7 INDEXATION SEMANTIQUE BASEE SUR LA DESAMBIGUÏSATION DU DOMAINE DU DISCOURS :

Le domaine est un ensemble de mots qui montrent un forte relation sémantique de par eux-mêmes. Les domaines sémantiques sont considérés comme liste de mots relatifs décrivant un sujet ou centre d'intérêt particulier. Partant de l'hypothèse qui stipule que bien souvent le sens

d'un mot est dépendant du domaine du discours (Part of Speech), plusieurs approches exogènes ont été proposées dont [Magnini & al., 02] et [Kolte & al., 09] .

Les auteurs de [Magnini & al., 02] ont introduit l'utilisation de WordNet Domains ,qui est une extension de WordNet, dans la tache de désambigüisation sémantique. Leur approche est basée sur l'idée que le sens d'un mot est souvent dépendant du domaine du discours (Domain of Speech). Par exemple, si on parle sur les ordinateurs, alors le sens le plus probable pour le mot ambigu « mouse » est le matériel informatique et non le rongeur. Le domaine du discours est établi en retrouvant le domaine le plus prélevant du discours. Ainsi un mot peut être désambiguïsé en se référant au domaine dominant des mots du contexte entourant le mot ambigu. La technique introduite a donné une amélioration de la précision passant de 0.48 en utilisant une indexation par lemmes à 0.67 et une amélioration du rappel passant de 0.25 en utilisant une indexation par lemmes à 0.48.

[Kolte & al., 09] ont présenté une approche se basant sur la même hypothèse précédente, à savoir que le sens d'un mot est souvent relatif au domaine du discours. Leur méthodes de désambiguïsation se base sur le contexte local pour retrouver le sens d'un mot polysémique. Elle consiste en deux étapes : d'abord retrouver le domaine auquel est associé un mot polysémique , ensuite retrouver le sens correct de ce mot dans le domaine préalablement fixé.

Ainsi chaque phrase est analysée et les mots vides sont éliminés. Pour chaque mot non vides restants (ce que Kolte désigne comme « content-word » ou mot de contenu) on cherche les domaines auxquels il est associé. Le domaine qui maximise la correspondance avec les domaines des autres mots restants de la phrase est le domaine choisi.

Une fois le domaine fixé, deux situations se présentent :

- ✓ la première est celle ou il n'y a qu'un seul sens du mot dans le domaine retrouvé, dans ce cas le sens est directement fixé.
- ✓ La deuxième est celle ou le mot ambiguë possède plusieurs sens dans le domaine fixé, auquel cas il faut passer à la désambiguïsation do sens du mot dans le domaine.

Pour désambiguïser le sens du mot dans le domaine, Kolte et al. Utilisent les relations de WordNet hyperonymie holonomie et méronymie. Ainsi pour chaque sens candidat du mot ambiguë, il faut retrouver ses hyperonymes (respectivement holonymes et méronymes) et calculer le nombre de leurs occurrences dans le contexte du mot ambiguë. Le sens qui maximise le score est choisit.

En plus des relations disponibles dans la base de données de wordnet, Kolte et al. Ont annoté leur base avec des relations supplémentaires a savoir, « the ability link », « the capabilty link » et enfin « the function link ».

- ✓ The ability link : la relation d'aptitude

Ainsi pour la phrase « A crane was flying across the river», le mot crane a les sens suivants :

1. Crane, Stephen Crane -- (United States writer (1871-1900))

2. Crane, Hart Crane, Harold Hart Crane -- (United States poet (1899-1932))
3. Grus, Crane -- (a small constellation in the southern hemisphere near Phoenix)
4. crane -- (lifts and moves heavy objects)
5. crane -- (large long-necked wading bird)

Grace à la relation de « ability link » et étant donné que le « crane is able to fly » le sens choisit est #5.

- ✓ The capability link : la relation de capacité

Dans la phrase « The chair asked members about their progress » le mot chair dispose des sens suivants :

1. chair -- (a seat for one person)
2. professorship, chair
3. president, chairman, chairwoman, chair, chairperson
4. electric chair, chair, death chair, hot seat

Étant donné que la personne « has capability to ask » le sens choisit est le sens 3.

- ✓ The function link :

Dans la phrase “Please keep the papers in the file”, le mot *file* peut avoir les sens suivants :

1. file, data file -- (a set of related records (either written or electronic) kept together)

2. file, single file, Indian file -- (a line of persons or things ranged one behind the other)

3. file, file cabinet, filing cabinet -- (office furniture consisting of a container for keeping papers in order)

4. file -- (a steel hand tool with small sharp teeth on some or all of its surfaces; used for smoothing wood or metal)

Étant donné que le mot *file* a la fonction de contenir les *papers* le sens choisit est le sens 3.

Cette approche a été testée sur le fichiers de Semcor2.1 a donné des résultats plutôt satisfaisant. Elle a en fait correctement désambiguïsé 5236 noms sur 5463. ce qui a donné une précision 63.92%.

II .8 CONCLUSION :

Au cours de ce chapitre, nous avons étudié l'indexation basée sur le sens des mots. Nous avons présenté un état de l'art des différentes approches proposées dans la littérature et quelques travaux dédiés à ces approches.

Bien que les avis divergent, il existe plusieurs méthodes [Mihalcea & al., 00] [Baziz & al., 04 ; 05] qui ont apporté leurs preuves et sont venues soutenir l'idée que l'indexation par les sens des mots était bénéfique à la RI [Schütze & al., 95].

L'objectif de notre travail est d'étudier et d'implémenter une approche d'indexation sémantique basée sur la désambiguïsation selon le domaine du discours. l'approche sera explicitée dans le chapitre suivant.

Chapitre III

Approche d'indexation sémantique

III .1 Introduction :

Après avoir présenté un état de l'art de l'indexation sémantique, nous entamons la description de l'approche que nous proposons. Cette approche s'inspire des travaux de [Kolte et al, 09]. En effet, elle se base sur le principe qui stipule que souvent le sens d'un mot dépend du domaine du discours. Ainsi, pour désambiguïser un mot polysémique, on tente d'abord de retrouver son domaine d'utilisation dans le contexte du document, puis on désambiguïse son sens dans le domaine ainsi sélectionné. Dans cette approche, la ressource linguistique *WordNet Domains* a été utilisée pour rechercher le domaine associé à un mot dans le document, puis WordNet a été utilisée pour rechercher son sens exact dans ce domaine. Notre approche, bien que fondamentalement basée sur l'approche de [Kolte & al, 09], elle y propose néanmoins une extension en vue d'améliorer cette dernière.

Notre proposition est ensuite implémentée dans la plate forme de RI Terrier, constituant ainsi une extension de la plate forme à la prise en charge de la RI sémantique. L'extension de Terrier constitue notre seconde contribution dans le cadre de ce travail de master. Terrier étendu est utilisé pour effectuer les tests et évaluations de notre proposition à la RI sémantique.

III.2 Description de notre approche :

III .2.1 Terminologie et Notations :

- ✓ *Un mot vide* désigne un mot appartenant à la liste des mots vides de la langue anglaise.

- ✓ *Un mot de contenu « content-word »* désigne un mot du texte n'appartenant pas à la liste des mots vides.

- ✓ *Un mot orphelin* désigne un mot qui n'a pas d'entrée dans WordNet 2.0.

- ✓ *L'ensemble* $L_{orphelins}$: Contient tous les mots orphelins d'un document donné.

- ✓ *L'ensemble* $L_{concepts}$: Contient les concepts (ou sens) associés aux mots de contenu désambiguïsés.

III .2.2 WordNet et WordNet Domains :

WordNet est une base de données lexicale développée depuis 1985 par des linguistes du laboratoire des sciences cognitives de l'université de Princeton, dirigé par le professeur George A. Miller. WordNet est un réseau sémantique de la langue anglaise. Nous utilisons cette ressource pour récupérer les différents sens d'un mot appelés *synsets.*

Wordnet Domains est une extension de WordNet qui permet de récupérer les domaines d'un synset. (Voir annexe 1 pour plus de détails).

Ces deux ressources sont la base de notre désambiguïsation.

III .2.3 Présentation de l'approche d'indexation sémantique de [Kolte & al, 09] :

Basé sur la désambiguïsation selon le domaine du discours, l'approche d'indexation de Kolte consiste en deux étapes. D'abord, l'identification des concepts basée sur la désambiguïsation du domaine et sur la désambiguïsation du sens dans le domaine sélectionné. Ensuite, pondération des concepts. La figure suivante montre une vue d'ensemble de cette approche d'indexation. Les étapes seront détaillées dans les points suivants.

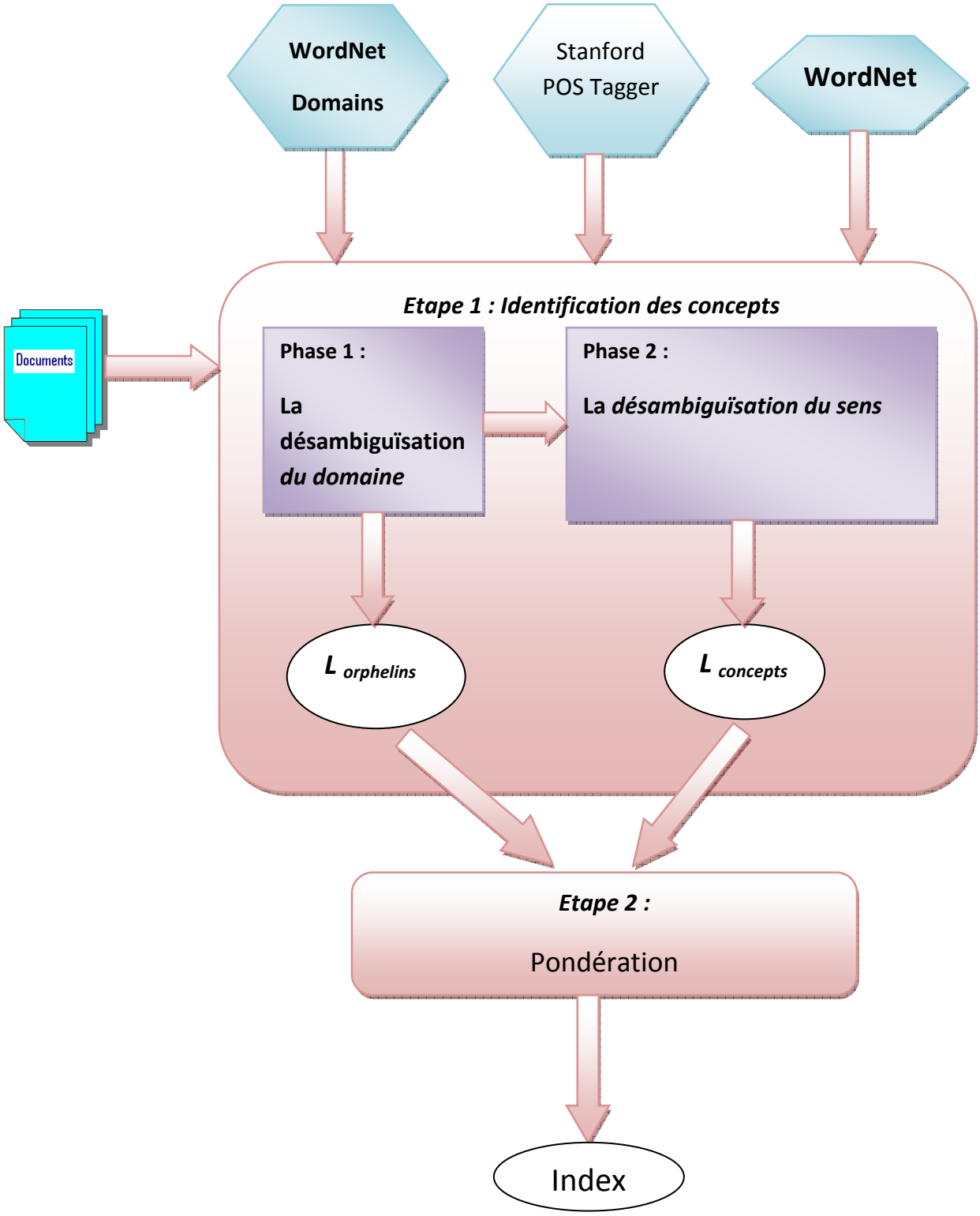


Figure 3.1 Vue d'ensemble de l'approche d'indexation

III .2.3.1 Identification des concepts :

Cette étape débute par l'identification des mots clés représentatifs du contenu du document. Cette étape consiste à analyser un document et en extraire les mots clés (ou plus généralement les termes) représentatifs de son contenu. A l'issue de cette étape deux types de mots sont identifiés : les mots orphelins, qui sont des mots n'ayant pas d'entrée dans WordNet et les mots simples (non orphelins). Un mot simple peut être polysémique. Il faut alors identifier son sens correct dans son contexte d'utilisation dans le document :C'est la désambiguïsation. Le contexte est considéré comme du texte se composant de tous les mots de contenu, "content-words", situés dans la phrase dans laquelle apparait l'occurrence du mot polysémique.

permet d'assigner un seul sens (ou *concept*) à chaque terme. La désambiguïsation s'effectue en deux étapes successives :

- ✓ **Phase 1** : La désambiguïsation *du domaine* qui permet de retrouver le domaine adéquat du mot cible
- ✓ **Phase 2** : La *désambiguïsation du sens* qui permet de retrouver le sens correct du mot cible dans le domaine identifié dans l'étape précédente.

III .2.3.1.1 La désambiguïsation du domaine :

Vu qu'un mot peut avoir plusieurs sens selon le domaine du discours, il peut donc appartenir à plusieurs domaines différents. L'objectif de cette étape est de retrouver le bon domaine auquel appartient le mot polysémique en tenant compte des domaines des autres mots de son contexte local (la phrase).

A cet effet, le texte analysé est d'abord syntaxiquement annoté. L'objectif de l'annotation syntaxique (ou *POS*[5] *tagging*), consiste à identifier la partie du discours (*Part Of Speech*) de chaque entité syntaxique du texte analysé. Dans notre cas, nous avons utilisé le *Stanford POS tagger*[6] pour ce faire. Une fois le texte taggué, on élimine les mots vides. Les mots de contenu sont insérés dans un premier sac B1.

[5] Part Of Speech (partie du discours ou catégorie syntaxique)

[6] http://nlp.stanford.edu/software/tagger.shtml

Pour chaque mot de contenu appartenant au sac B1, il s'agit ensuite de retrouver les domaines correspondants à sa catégorie syntaxique et les insérer dans un second sac B2. Les domaines correspondant à la catégorie syntaxique du mot (cible) à désambiguïser sont insérés dans le sac B3. Les domaines correspondant à la catégorie syntaxique du mot (cible) à désambiguïser sont insérés dans le sac B3.

> $(w_1, w_2, w_3 \ldots w_n)$ est le sac B1 contenant les mots-pleins tagués
>
> $(d_1, d_2, d_3 \ldots d_n)$ est le sac B2 contenant l'ensemble de tous les domaines correspondant aux mots de contenu avec leurs étiquettes syntaxiques.
>
> Chaque ensemble d_i contient tous les domaines possibles pouvant correspondre aux sens d'un mot de contenu.

Pour trouver le bon domaine du mot cible, chaque domaine du sac B3 est comparé à chacun des domaines appartenant à B2. Le domaine du mot cible qui maximise la correspondance avec les domaines des autres mots est sélectionné comme le domaine adéquat du mot cible. Les sens appartenant à ce domaine deviennent les sens probables du mot cible.
Notons, que le domaine générique *factotum*, n'a aucun sens réel, il n'est donc pas considéré dans le processus de désambiguïsation ci-haut.

Exemple :

Considérons la phrase "*The **virus** infected all files on the hard disk*." Cette phrase sera étiquetée par le POS tagger comme suit : "The/DT virus/NN infected/VBD all/DT files/NNS on/IN hard_disk/NN ./."

Les mots vides sont ensuite éliminés et les mots de contenu retenus sont les mots : *virus, infected, files* et *disk*.

Supposons que l'on veuille retrouver le sens du mot *virus,* le tableau suivant décrit les numéros des synsets et les domaines du mot à désambiguïser et des autres mots de contenu de la phrase.

Virus (noun sense)
Infected (verb sense)
Files (noun sense)
disk (noun sense)

b1

virus (noun sense) Target word	infected (verb sense)	files (noun sense)	hard_disk (noun sense)
01254816-factotum	00087224-medicine	06106818-telecommunications	03364489-computer science
13209397-factotum	00086241-medicine	07917489-factotum	
06179311-Computer_science	02503346-factotum	03215630-administration furniture	
	00585683-phychological features	03215329-building industry	

b2

01254816-factotum
13209397-factotum
06179311-Computer_science

b3

Tableau III .1 Contenu des sacs B1, B2 et B3.

En résumé les entrées de l'algorithme sont :

B1= {virus, infected, files, disk}

B2= {{factotum, factotum, computer science},{medicine, medicine, factotum, psychological features }, {telecommunication, factotum, administration, building industry}, {computer science}}

w_t= virus

B3= {factotum, factotum, computer science}

Vu que le domaine qui maximise la correspondance (hormis le domaine *factotum*) est le domaine *computer science,* le domaine du discours est fixé à *computer science* et le sens est celui appartenant à ce domaine.

Algorithme de désambiguïsation du domaine :

Entrée :

w_t : mot à désambiguïser

sacB1={w_1, w_2, … w_n}, les mots pleins de la phrase ou figure w_t

sacB2={{d_{11}, d_{12}, … , d_{1m}}, {d_{21},d_{22}, …, d_{2l}}, … ,{d_{n1},d_{n2} … d_{np}}}, contient chaque ensemble des domaines relatifs à chaque mot de la phrase.

sacB3={d_{t1}, d_{t2}, … , d_{tr}}, contient les domaines du mot w_t

S={s_1,s_2, … ,s_q}, contient les numéros des synsets de w_t

Sortie :

domain, le domaine retenu du mot w_t

nbSens,nombres de sens de w_t appartenant au domaine *domain*

```
domain    ←   w1 ;
nbSens ←  0 ;
max  ←  0 ;
Sd ={ } : ensemble des synsets appartenant au domaine domain
pour chaque mot à désambiguïser wt du sacB1
      score       ←  0 ;
      pour chaque domaine dti du sacB3 faire
            pour chaque domaine djk du sacB2    tel que ( j != t) faire
                  si (dti= djk) alors
                        score ← score + 1 ;
            fait ;
            si (score> max) et (dti != factotum) alors
            {     max    ←  score ;
                  domain   ←  dti;
            }
      fait ;
      pour chaque sens si appartenant à S faire
            si ( si appartient au domaine domain )
            {     ajouter si à Sd ;
                  nbSens ← nbSens +1 ;
            }
      fait ;
fait ;
```

III .2.3.1.2 La désambiguïsation du sens :

Après avoir retrouvé le domaine du discours de chaque mot de contenu dans son contexte d'utilisation dans le document, on s'intéresse aux sens de ce mot dans ce domaine. S'il existe un seul sens associé dans le domaine en question alors le ce sens est retenu comme sens adéquat du mot cible dans son contexte d'utilisation. Cependant, il peut arriver que le mot cible ait plusieurs sens dans le domaine sélectionné. On peut citer l'exemple du mot *bank* dont les sens bank#1, bank#6 et bank#8 appartiennent au domaine *economy.* Dans ce cas, la désambiguïsation de sens a pour tâche de retrouver parmi les sens probables du mot cible dans ce domaine, celui qui correspond au mieux à son utilisation dans le contexte.

L'approche de désambiguïsation des sens des mots de [Kolte & al, 09] est une approche par étapes qui se base successivement sur différentes relations hiérarchiques de WordNet, à savoir l'holonymie/méronymie, l'hyperonymie. Nous proposons de l'étendre avec une autre étape de désambiguïsation basée sur le glossaire issu de WordNet. En effet, à chaque synset dans WordNet est associé un glossaire qui regroupe une définition du concept avec éventuellement un ou plusieurs exemples du monde réel. On se propose d'utiliser le glossaire du mot pour le désambiguïser dans son contexte ; pour ce faire, on se propose de comparer le contexte du mot à désambiguïser avec le glossaire de chacun de ses sens issus du domaine préalablement désambiguïsé. Le sens qui maximise le score sera sélectionné comme étant le sens adéquat du mot cible.

Dans notre cas nous avons choisi d'insérer cette étape de désambiguïsation par le glossaire entre la désambiguïsation par la relation de méronymie et celle d'hyperonymie.

Le principe de chacune de ces désambiguïsations est d'abord explicité à travers les exemples suivants, puis les algorithmes correspondants sont formellement définis.

a. Désambiguïsation par la relation de holonymie (part_of):

Considérons la phrase "The trunk is the main structural member of a tree that supports the branches. ". Cette phrase sera tagguée comme suit :" `The/DT trunk/NN is/VBZ the/DT main/JJ structural/JJ member/NN of/IN a/DT tree/NN that/WDT supports/VBZ the/DT branches/NNS ./.`"

Le mot "trunk" a les holonymes suivants

1. PART OF: {12934526} <noun.plant> tree#1
3. PART OF: {05154650} <noun.body> body#1,
4. PART OF: {02929975} <noun.artifact> car#1,
5. PART OF: {02480939} <noun.animal> elephant#1
6. PART OF: {02482174} <noun.animal> mammoth#1

Vu que le contexte du mot "trunk" contient le nom « tree » l'algorithme va détecter que le bon sens est celui dont l'hyponyme maximise le score, donc le sens #1.

b. Désambiguïsation par la relation de méronymie (has_part):

Considérons la même phrase précédente à savoir, "He has a nice car, but it's door damaged.". Après tagging la phrase sera comme suit: "`He/PRP has/VBZ a/DT nice/JJ car/NN ,/, but/CC it/PRP 's/VBZ door/NN damaged/VBN ./.`"
Supposons que l'on veuille désambiguïser le mot "car" dont voici quelques méronymes :

1. HAS PART: {02593287} <noun.artifact> air bag#1
   HAS PART: {02663175} <noun.artifact> auto accessory#1
   HAS PART: {02858241} <noun.artifact> car door#1
   HAS PART: {02865022} <noun.artifact> car seat#1
   HAS PART: {02868511} <noun.artifact> car window#1
   …

2. HAS PART: {04197018} <noun.artifact> suspension#5

Vu que le contexte du mot "car" contient le nom « door » l'algorithme va détecter que le bon sens est celui dont les méronymes maximisent le score, donc le sens #1.

c. Désambiguïsation par la relation d'hypernymie (is_a_kind):

Considérons la phrase "He ate many dates. ". Cette phrase sera taguée comme suit "`He/PRP ate/VBD many/JJ dates/NNS ./.` "
Pour retrouver le bon sens du mot "date", dont les hypernymes sont :

1. {14297391} <noun.time> day#1, twenty-four hours#1, solar day#1, mean solar day#1
2. {14297391} <noun.time> day#1, twenty-four hours#1, solar day#1, mean solar day#1
3. {07809281} <noun.group> meeting1#3, get together#1
4. {14321000} <noun.time> point#6, point in time#1
5. {14263350} <noun.time> present#1, nowadays#1
6. {09310467} <noun.person> companion#1, comrade#1, fellow2#2, familiar2#2, associate1#2
7. {14299149} <noun.time> calendar day#1, civil day#1
8. {07234431} <noun.food> edible fruit#1

Vu que date est une sorte de "edible fruit", en recherchant son synonyme on tombe sur "eatable" qui apparait dans le contexte de date après lemmatisation. Donc le bon sens du mot "date" est le sens#8.

d. Désambiguïsation par le glossaire :

Considérons la phrase "economists work into financial institutions like banks and posts". Cette phrase sera taguée comme suit: `economists/NNS work/VBP into/IN financial/JJ institutions/NNS like/IN banks/NNS and/CC posts/NNS`Pour désambiguïser le nom "bank"

1. bank -- (a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home")
2. bank -- (a container (usually with a slot in the top) for keeping money at home; "the coin bank was empty")
3. bank -- (the funds held by a gambling house or the dealer in some gambling games; "he tried to break the bank at Monte Carlo")

En comparant le contexte du mot bank et le glossaire de chaque synset, le sens qui maximise le score est bank#1 donc le sens choisit est ce sens.

Dans notre cas, nous avons placé la désambiguïsation par le glossaire juste avant l'hyperonymie.

Algorithme de désambiguïsation du sens par la relation d'Holonymie :

Entrée :

w_t : mot à désambiguïser

sacB1={w_1, w_2, … , w_n}, contient les mots pleins de la phrase où figure w_t

S_d={s_1,s_2, … ,s_m}, contient les numéros des synsets de w_t appartenant au domaine *domain*

nbS : nombre de synsets appartenant au domaine *domain*

Sortie :

sens : le numéro du synset représentant le bon sens de w_t

nbSens : nombre de sens candidats de w_t après la désambiguïsation par l'holonymie

numSens : numéros des sens candidats de w_t après la désambiguïsation par l'holonymie

```
max  ←  0 ;
nbSens ←  0 ;
holonyms(si) : l'ensemble des holonyms du sens si de  wt
scoreHolonym : ensemble des scores de chaque holonyms(si)
si (nbS =1) alors       // un seul sens par domaine
      sens= s1 ;
sinon
{     pour chaque sens candidat si de wt
            déterminer holonyms(si), l'ensemble des holonyms du sens si
de  wt ;
            scoreHolonym(si) ← 0
            pour chaque mot wj appartenant au sacB1
                  si (wj appartient à holonyms (si)) alors
                        scoreHolonym(si) ←  scoreHolonym(si)  +1 ;
            fait ;
            si (scoreHolonym(si) > max) alors
                  max    ←  scoreHolonym(si) ;
      fait ;
      pour chaque sens candidat si de wt
            si (scoreHolonym(si)    = max)
             {    nbSens ← nbSens +1 ;
                  Ajouter i à numSens ;
            }
      fait;
      si (nbSens> 1) alors
            désambiguïser par la relation de méronymie ;
}
```

Algorithme de désambiguïsation du sens par la relation de Méronymie:

Entrée :

w_t : mot à désambiguïser

sacB1={w_1, w_2, … , w_n}, contient les mots pleins de la phrase où figure w_t

S_d={s_1,s_2, … ,s_m}, contient les numéros des sens candidats de w_t après la désambiguïsation par la relation de holonymie

Sortie :

sens : le numéro du synset représentant le bon sens de w_t

nbSens : nombre de sens candidats de w_t après la désambiguïsation par la méronymie

numSens : numéros des sens candidats de w_t après la désambiguïsation par la méronymie

max ← 0 ;

nbSens ← 0 ;

méronyms(s_i) : l'ensemble des méronyms du sens s_i *de* w_t

scoreMeronym : ensemble des scores de chaque méronyms(s_i)

pour chaque sens candidat s_i de w_t

déterminer méronyms(s_i), l'ensemble des méronyms du sens s_i *de* w_t ;

scoreMéronym(s_i) ← 0

pour chaque mot w_j appartenant au sacB1

si (w_j appartient à méronyms (s_i)) alors

scoreMéronym(s_i) ← scoreMéronym(s_i) +1 ;

fait ;

si (scoreMéronym(s_i) > max) alors

max ← scoreMéronym(s_i) ;

fait ;

pour chaque sens candidat s_i de w_t

si (scoreMéronym(s_i) = max)

{ nbSens ← nbSens +1 ;

Ajouter i à numSens ;

}

fait;

si (nb> 1) alors

désambiguïser par le glossaire;

Algorithme de désambiguïsation du sens par le glossaire:

Entrée :

w_t : mot à désambiguïser

sacB1={w_1, w_2, … , w_n}, contient les mots pleins de la phrase où figure w_t

S_d={s_1,s_2, … ,s_m}, contient les numéros des sens candidats de w_t après la désambiguïsation par la relation de méronymie

Sortie :

sens : le numéro du synset représentant le bon sens de w_t

nbSens : nombre de sens candidats de w_t après la désambiguïsation par le glossaire

numSens : numéros des sens candidats de w_t après la désambiguïsation par le glossaire

max ← 0 ;

nbSens ← 0 ;

gloss(s_i), le glossaire du sens s_i *de* w_t

scoreGloss : ensemble des scores de chaque gloss(s_i)

pour chaque sens candidat s_i de w_t

rechercher gloss(s_i), le glossaire du sens s_i *de* w_t ;

scoreGloss(s_i) ← 0 ;

pour chaque mot w_j appartenant au sacB1

si (w_j appartient à scoreGloss (s_i)) alors

scoreGloss (s_i) ← scoreGloss (s_i) +1 ;

fait ;

si (scoreGloss (s_i) > max) alors

max ← s scoreGloss (s_i) ;

fait ;

pour chaque sens candidat s_i de w_t

si (scoreGloss (s_i) = max)

{ nbSens ← nbSens +1 ;

Ajouter i à numSens ;

}

fait;

si (nb> 1) alors

désambiguïser par la relation d'hypernymie ;

Algorithme de désambiguïsation du sens par la relation de Hyperonymie:

Entrée :

w_t : mot à désambiguïser

sacB1={w_1, w_2, … , w_n}, contient les mots pleins de la phrase où figure w_t

S_d={s_1,s_2, … ,s_m}, contient les numéros des sens candidats de w_t après la désambiguïsation par le glossaire

Sortie :

sens : le numéro du synset représentant le bon sens de w_t

max ← 0 ;

nbSens ← 0 ;nombre de sens candidats de w_t après la désambiguïsation par l'hypernymie

numSens : numéros des sens candidats de w_t après la désambiguïsation par l'hypernymie

hypernyms(s_i), l'ensemble des hypernyms du sens s_i *de* w_t

scoreHypernym : ensemble des scores de chaque hypernyms(s_i)

pour chaque sens candidat s_i de w_t

déterminer hypernyms(s_i), l'ensemble des hypernyms du sens s_i *de* w_t ;

déterminer synonyms(s_i), l'ensemble des synonyms des hypernym(s_i) du sens s_i *de* w_t ;

scoreHypernym (s_i) ← 0

pour chaque mot w_j appartenant au sacB1

si (w_j appartient à synonyms (s_i)) alors

scoreHypernym(s_i) ← scoreHypernym(s_i) +1 ;

fait ;

si (scoreHypernym(s_i)> max) alors

max ← scoreHypernyms(s_i) ;

fait ;

pour chaque sens candidat s_i de w_t

si (scoreHypernym(s_i) = max)

{ nb ← nb +1 ;

Ajouter i à numSens ;

}

fait;

si (nb> 1) alors

sens = S_1 ; // sens par défaut ; les relations hiérarchiques ne permettent pas la désambiguïsation

III .2.3.2 Pondération des concepts :

Après extraction des concepts et des mots orphelins, la prochaine étape consiste à assigner des poids aux termes selon leur degré d'importance dans le document. Dans cette approche, on se base la fréquence d'occurrence des termes. A cet effet, on utilise la pondération *Tf*Idf* qui combine le facteur de pondération local $Tf_{i,j}$ et le facteur de pondération global Idf_i et la mesure résultante est w_{ij} qui représente le poids du terme i dans le document j

$$w_{ij} = tf_{ij} * idf_i = tf_{ij} * \frac{1}{df_i} = \frac{tf_{ij}}{df_i}$$

$tf_{i,j}$: mesure la fréquence (nombre d'occurrences) du terme i dans un document j

df_i: représente la fréquence documentaire (ou le nombre documents de la collection qui contiennent le terme t_i)

idf_i *:* mesure la fréquence documentaire inverse du terme i.

III.3 Extension de Terrier à l'indexation sémantique:

III.3.1 Présentation de Terrier :

Terrier[6]*, Terabyte RetrIEveR* est un moteur de recherche robuste et efficace, développé par le département informatique de l'université Glasgow en Ecosse. Il est open source et entièrement écrit en java. Terrier offre une plate forme idéale destinée à l'indexation de volumes importants de documents: jusqu'à 25 millions de documents. Il est en fait classé troisième parmi les SRI écrits en java.

Comme tout SRI Terrier offre la possibilité d'indexer des documents, d'effectuer une recherche et d'en évaluer les résultats (voir Annexe 2 pour plus de détails). Dans ce qui suit, nous allons nous intéresser au processus d'indexation et au processus de recherche que nous allons modifier afin d'intégrer notre module de désambiguïsation.

III.3.2 Le processus d'indexation de Terrier :

Pour indexer une collection de documents et générer l'index, Terrier est doté d'un processus d'indexation à quatre étapes :

- Splitter la collection de documents : consiste à parcourir la collection et à envoyer chaque document à l'étape suivante.

[6] http://www.terrier.org

- Extraire les termes : consiste à parser chaque document et en extraire les termes. Chaque terme trouvé sera envoyé au composant TermPipeline.

- Traitement des termes extraits : consiste à traiter tous les termes extraits par le TermPipeline qui va éliminer les mots vides et lemmatiser.

- Construction de l'index.

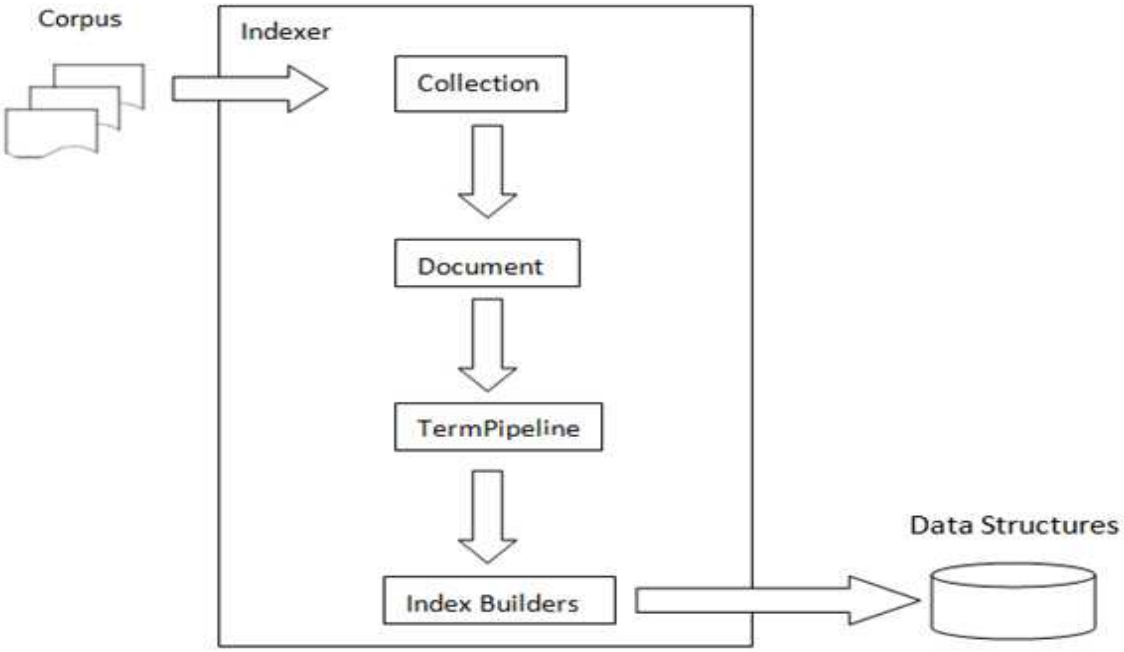


Figure 3.2 Processus d'indexation de Terrier

Toutes ces étapes, les modules en charge de leur exécution ainsi que les fichiers résultats de cette indexation sont présentés de façon plus détaillée en Annexe 2.

III.3.3 Le processus de recherche de Terrier :

Durant le processus de recherche, chaque requête doit passer par les étapes suivantes :

- Parsing : qui se charge de tokenizer la requête

- Pre-processing : qui applique le Termepipeline à la requête. Elimine les mots vides et les lemmatise.

- Matching : qui est responsable de l'initialisation du WeightingModel et du calcul des scores entre la requete et les documents.

- post-filtrage : va filtrer les résultats.

- post-traitement : peut modifier le ResultSet, par exemple, par un procédé QueryExpansion afin de générer un meilleur classement de documents.

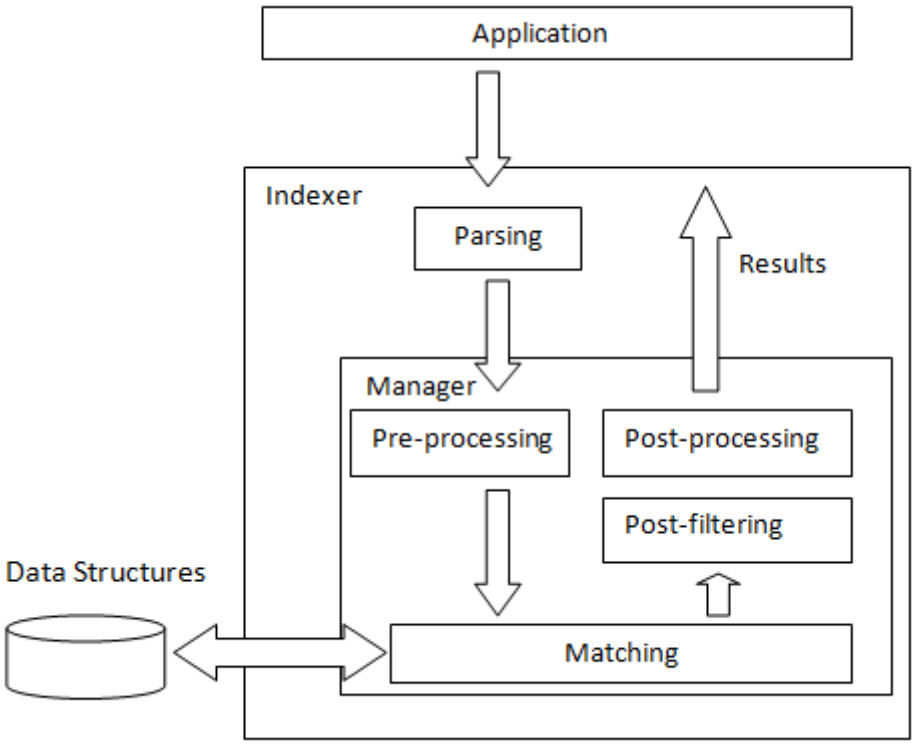


Figure 3.3 Processus de recherche de Terrier

Toutes ces étapes et les modules en charge de leur exécution seront détaillés en Annexe 2.

III.3.4 présentation de Sem-Sem-Terrier:

Notre objectif à travers cette proposition est d'étendre la plate forme Terrier à la prise en charge de l'indexation sémantique. Pour ce faire, nous préconisons d'intégrer notre approche d'indexation sémantique à *Terrier 3.5.* Nous appellerons la plateforme résultante ***Sem-Terrier***

Vu que notre approche est basée sur le contexte, chaque mot doit être désambiguïsé par rapport aux autres mots de la phrase (aussi bien dans les documents que dans les requêtes). Ceci implique que la désambiguïsation doit se faire préalablement à tout autre traitement sur la phrase comme la tokenization ou la lemmatisation. De ce fait, notre Désambiguïsation sera en fait vue comme une étape préliminaire à l'indexation de Terrier.

Sachant que le même procédé de désambiguïsation doit être appliqué aussi bien aux documents qu'aux requêtes, le module de désambigüisation sera intégré dans le processus d'indexation et dans le processus de recherche.

III.3. 4. 1 Processus d'indexation sémantique de Sem-Terrier:

Le processus d'indexation de terrier sera augmenté d'une phase supplémentaire réalisée par notre *Module de Désambiguïsation Sémantique.* Ainsi, avant de lancer la première phase du processus d'indexation de terrier, chaque document du corpus sera d'abord désambiguïsé par notre *Module de Désambiguïsation Sémantique.* L'indexation suivante se fera sur les documents préalablement désambiguïsés.

La figure suivante illustre le processus d'indexation de Sem-Terrier

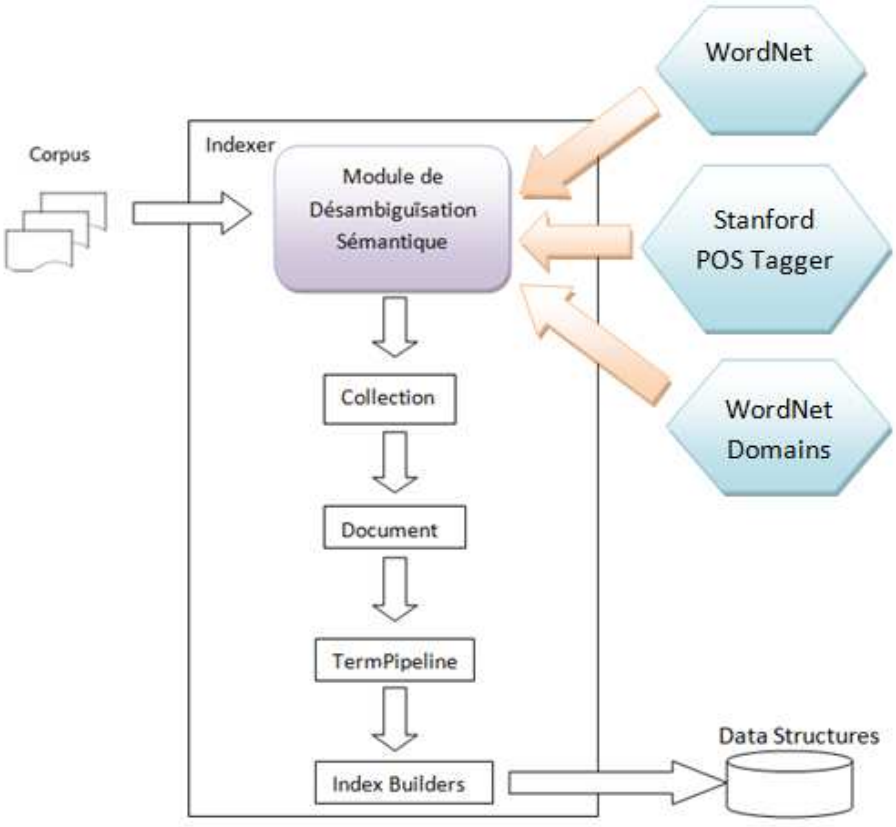


Figure 3.4 Processus d'indexation sémantique de Sem-Terrier

III.3. 4. 2 Processus de recherche sémantique de Sem-Terrier :

Le processus de recherche de terrier sera étendu d'une phase supplémentaire réalisée par notre *Module de Désambiguïsation Sémantique de la Requête.* Ainsi, avant de lancer la première phase du processus de recherche (ie. le parsing), chaque requête sera d'abord désambiguïsée par notre *Module de Désambiguïsation Sémantique de la Requête.*

La figure suivante illustre le processus de recherche sémantique de Terrier :

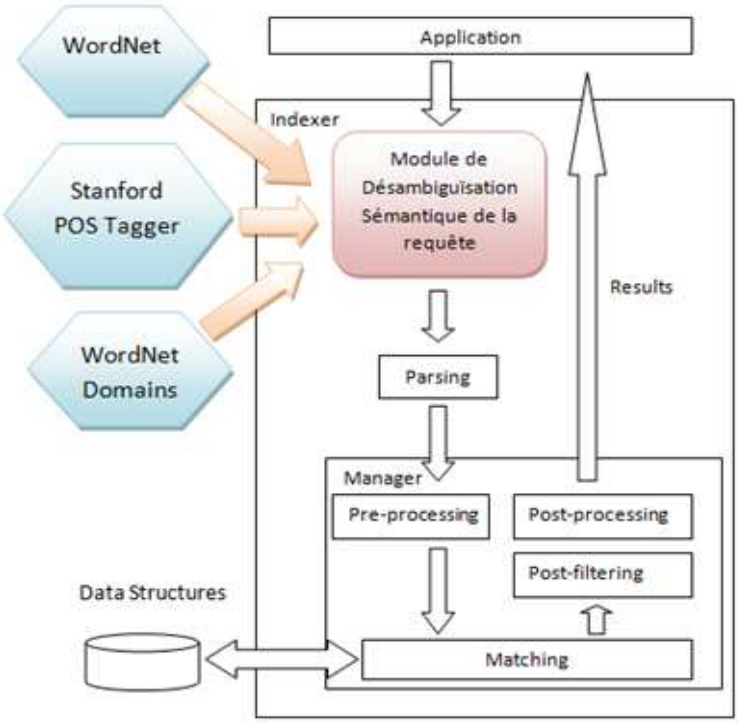


Figure 3.5 Processus de recherche sémantique de Sem-Terrier

A la fin de notre intégration , la plate forme de RI sémantique Sem-Terrier aura la structure suivante :

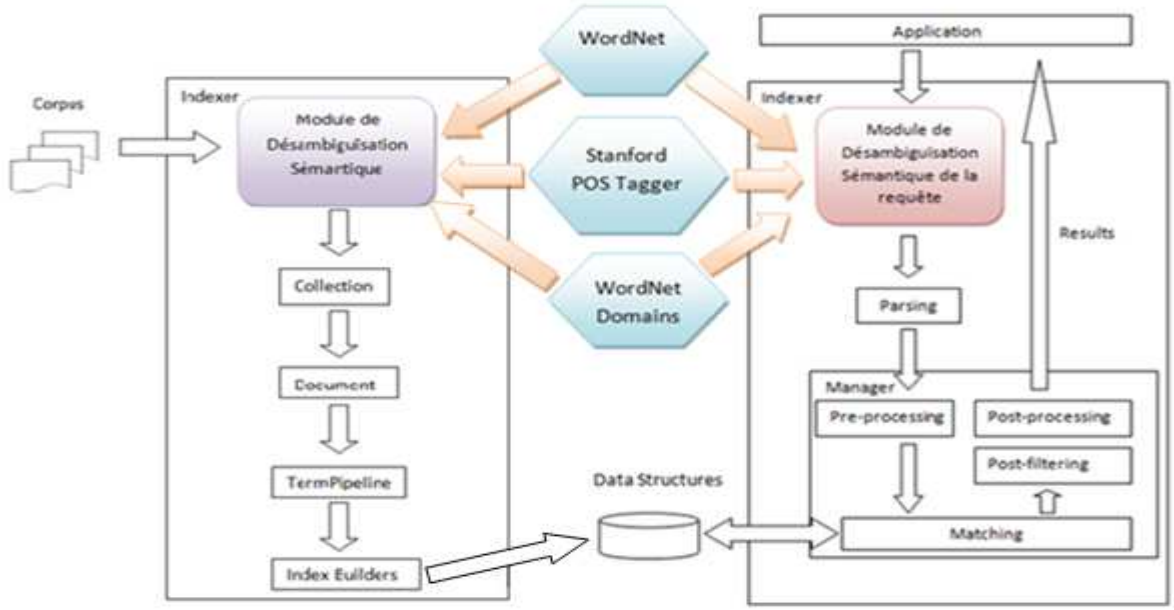


Figure 3.6 Présentation finale de Sem-Terrier

III .4 Conclusion :

Nous avons présenté dans de ce chapitre l'approche d'indexation sémantique s'inspirant de celle de Kolte basée sur la désambiguïsation selon le domaine du discours et sur la désambiguïsation des sens dans le domaine. Nous avons présenté notre proposition à l'amélioration de cette approche, qui consiste à améliorer la désambiguïsation du sens des mots avec le glossaire de WordNet. Nous avons ensuite proposé une architecture d'intégration de cette approche sémantique à la plate forme terrier. La structure de *Sem-Terrier ,* la plateforme résultante de l'intégration de notre module à Terrier, a été décrite.

Nous pouvons à présent implémenter et évaluer notre approche d'indexation sémantique en utilisant la plateforme de RI *Sem-Terrier*. Ceci fera l'objet du chapitre suivant.

Chapitre IV

Résultats et expérimentations

IV .1 Introduction :

Au cours du chapitre, précédent nous avons décrit la première partie de notre travail qui consiste en la présentation de notre approche d'indexation sémantique. Ensuite nous avons présenté la deuxième partie de notre travail qui consiste en l'intégration de ce module vers la plateforme de RI *Terrier 3.5* . A présent nous allons évaluer les résultats dans la plateforme résultante, *Sem-Terrier,* et les comparer à ceux de Terrier 3.5.

IV .2 Description de l'environnement technologique :

Au cours du développement de cette application, nous avons eu affaire à plusieurs composantes technologiques et API :

- ✓ **Eclipse** : nous avons utilisé l'IDE Eclipse comme Environnement de Développement.

- ✓ **WordNet 2.0** [8] : qui un dictionnaire développé par l'université de Princeton. Nous l'avons utilisé comme ressource pour la désambiguïsation du sens .

- ✓ **WordNet Domains** [9]: liste des domaines de chaque synset de WordNet 2.0 . Elle contient des le boublet <key Domain> qui sont respectivement l'identifiant du synset dans WordNet 2.0 et les domaines auxquels il est associé.

Nous l'avons utilisé pour retrouver le domaine de chaque Synset de WordNet.

Voici un échantillon de ce fichier :

```
…
00027929-n    factotum
00028549-n    sociology
00028764-n    linguistics telecommunication
00029305-n    metrology
…
```

JWNL API [10]: c'est une API qui permet d'interroger le dictionnaire, nous avons utilisé la version 1.3 qui est compatible avec WordNet 2.0.
On peut :
Initialiser le dictionnaire :

[8] http://www.wordnet.princeton.edu/
[9] http://wndomains.fbk.eu/download.html
[10] http://jwordnet.sourceforge.net/handbook.html

```
public static void initialize(String propsFile)
{                    // Creation de l'objet dictionary object
try {                JWNL.initialize(new ileInputStream(propsFile));
                     } catch (FileNotFoundException e) { e.printStackTrace(); }
                       catch (JWNLException e) {  e.printStackTrace(); }
```

Rechercher les Synsets d'un mot :

```
initialize("C:\\JAVA_JARS\\jwnl20\\config\\file_properties.xml");
Dictionary dict20 = Dictionary.getInstance();
String word="car" ;
IndexWord words =dict20.lookupIndexWord(POS.NOUN, word);
Synset[]syn=words.getSenses();
```

Rechercher les Holonyms, Meronyms et Hypernyms d'un synset:

```
ArrayList <String> holonyms, meronyms, hypernyms =new ArrayList <String>();
holonyms= getRelated (syn[i],PointerType.MEMBER_HOLONYM);
meronyms= getRelated (syn[i],PointerType.MEMBER_MERONYM);
hypernyms= getRelated (syn[i],PointerType.HYPERNYM);
```

IV .3 Evaluation Expérimentale :

Après avoir intégré notre module d'indexation sémantique à Terrier, on peut à présent tester l'apport de notre approche sur les performances de la recherche d'information.

Pour évaluer notre approche, nous avons utilisé la collection Muchmore[11] spécifique aux

[11] http://muchmore.dfki.de.about.html

documents médicaux. Pour des contraintes de temps, causés par la complexité de la méthode de désambiguïsation, spécialement l'utilisation du tagger, et l'accès au dictionnaire qui prennent un temps considérable, nous n'avons testé que sur 851 documents de la collection avec 10 requêtes.

- ✓ Extrait d'un document Muchmore (`Arthroskopie.00130217.txt`) :

The present study describes a new anteroinferior portal for the arthroscopic repair of Bankart lesions. The portal was performed in an outside-to-inside fashion with the patient in the beach-chair position. The blunt guide rod was inserted 8-10 cm distal from the palpable coracoid tip lateral of the deltopectoral groove, directed to the inferior glenoid rim at an angle of about 135 to the glenoid.

- ✓ Exemple de requête Muchmore

88: Approach of the correction of deformities in orthopedics.

- ✓ Exemple de jugement de pertinence :

6 0 DerChirurg.00710389.txt 1

6 0 DerChirurg.80690503.txt 1

6 0 Bundesgesundheitsblatt.00430449.txt 1

6 0 Bundesgesundheitsblatt.0043s003.txt 1

6 0 Bundesgesundheitsblatt.0043s009.txt 1

IV .4 Evaluation des résultats :

IV .4.1 Protocole d'évaluation :

Nous avons effectué notre évaluation selon *Protocol d'évaluation TREC :*

Pour chaque requête, les 1000 premiers documents restitués par le système sont examinés et des précisions sont calculées à différents points (à 1, 2, 3, 4, 5, 10, … 1000 premiers documents restitués). La précision à x (exemple précision à 5) définie le taux de documents pertinents parmi les x premiers documents retrouvés.

Une précision moyenne *MAP* est ensuite calculée pour chaque requête. Il s'agit de la moyenne des précisions de chaque document pertinent pour cette requête. La précision d'un document est la précision à x, tel que x est le rang de ce document dans l'ensemble des documents pertinents retrouvés. Finalement, la précision moyenne pour l'ensemble des requêtes est calculée permettant d'obtenir une mesure de la performance globale du système.

IV .4.2 résultats et tests :

Nous avons indexé 851 documents issus de Muchmore avec « Terrier classique » et indexé ces mêmes documents avec *Sem-Terrier*. Nous avons interrogé l'index avec dix requêtes. On peut à présent évaluer les résultats de cette recherche. Voici un tableau récapitulatif du contenu des fichiers d'évaluation qui montre une amélioration de le *Précision moyenne* et de la *R Précision* :

Information	Modèle classique Terrier 3.5 (pondération TF_IDF)	Modèle sémantique Sem-Terrier (pondération TF_IDF)
Number of queries Retrieved Relevant Relevant retrieved	10 7763 280 275	10 2877 280 266
Average Precision: R Precision :	0.5987 0.5612	0.6172 0.6018
Precision at 1 : Precision at 2 : Precision at 3 : Precision at 4 : Precision at 5 : Precision at 10 : Precision at 15 : Precision at 20 : Precision at 30 : Precision at 50 : Precision at 100 : Precision at 200 : Precision at 500 : Precision at 1000 :	0.7000 0.7500 0.7667 0.7750 0.7800 0.7000 0.6067 0.5450 0.4633 0.3820 0.2480 0.1310 0.0534 0.0275	0.7000 0.7500 0.7667 0.8000 0.8000 0.7500 0.6600 0.5950 0.4767 0.3800 0.2450 0.1300 0.0532 0.0266
Precision at 0%: Precision at 10%: Precision at 20%: Precision at 30%: Precision at 40%: Precision at 50%: Precision at 60%: Precision at 70%: Precision at 80%: Precision at 90%: Precision at 100%:	1.6256 1.5619 1.5206 1.2682 1.2470 1.2680 0.9404 0.6834 0.4444 0.2644 0.1659	1.5817 1.5644 1.5864 1.4453 1.1094 0.9619 0.9778 0.8068 0.5450 0.3972 0.1942
Average Precision:	0.5987	0.6172

Tableau 4.1 contenus des fichiers d'évaluation (fichiers .res)

Comme nous le voyons ici dans ce fichier résultat : la précision moyenne est passée de 0.5987 pour le modèle classique à 0.6172 pour le modèles sémantique. nous constatons aussi une amélioration de la R-Precision qui est passée de 0.5612 à 0.6018.

Voici deux graphes récapitulatifs des résultats présentés dans ce tableau pour les deux modèles étudiés :

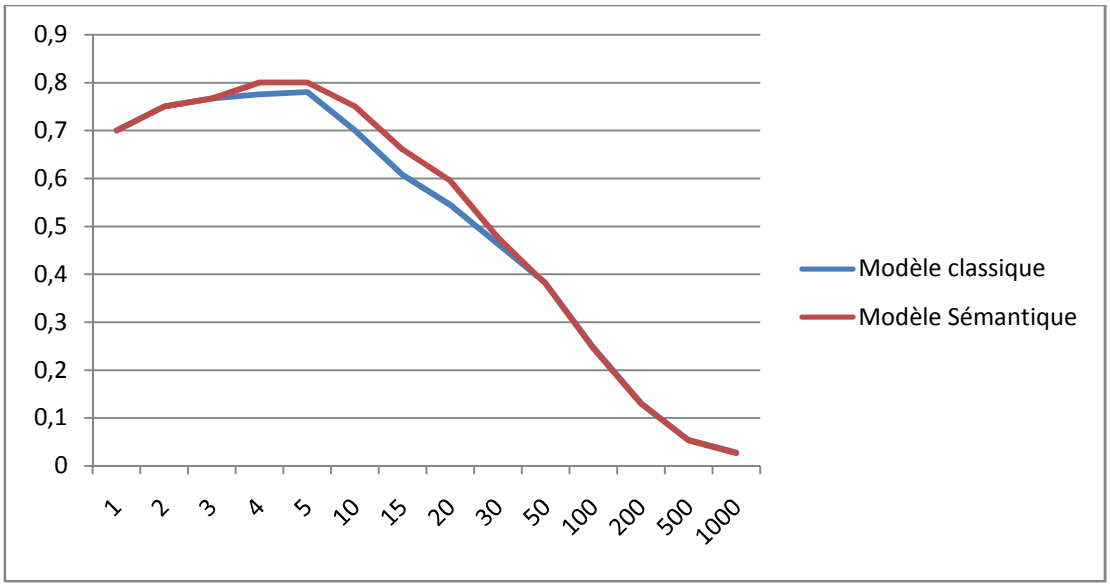


Figure 4.1 precision at x

La comparaison des courbes montre une amélioration certaine de la précision du modèle sémantique par rapport au modèle classique pour les précisions comprises entre : precision at 5 et precision at 30.

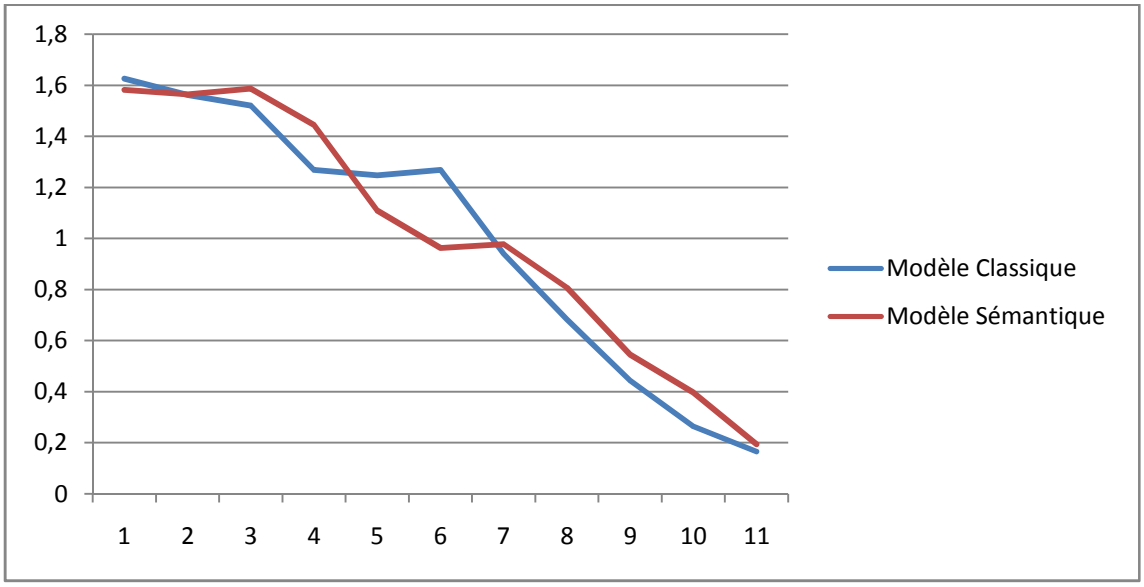


Figure 4.2 precision at x%

Voici une figure illustrant l'augmentation de la précision moyenne et de la R-precision

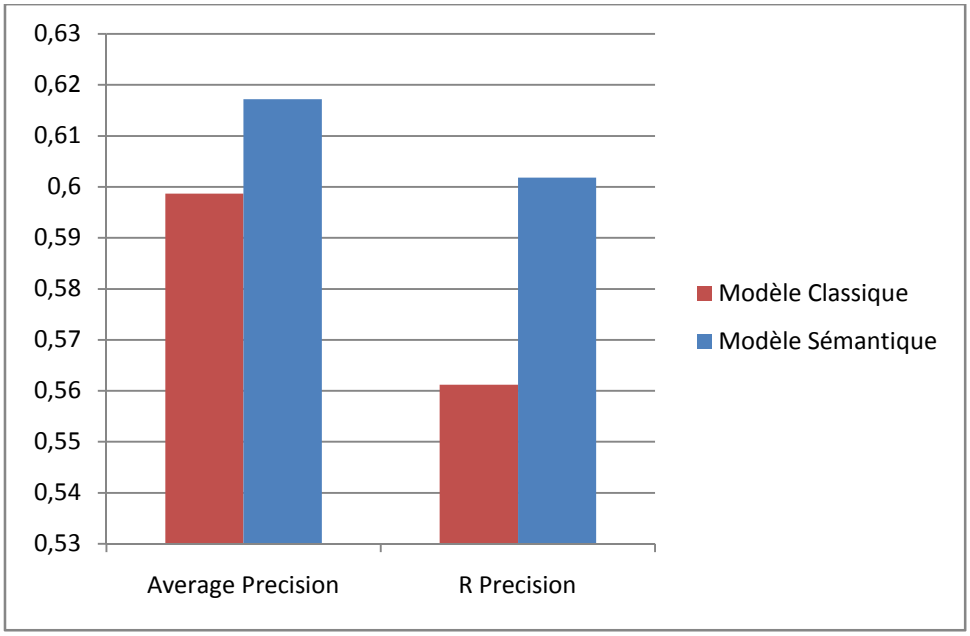


Figure 4.3 Augmentation de la précision moyenne et de la R-precision

IV . 5 Conclusion :

Nous avons décrit au cours de ce chapitre l'intégration de notre approche d'indexation sémantique à la plateforme de RI Terrier 3.5. cette intégration nous a permis d'évaluer notre approche. Nous avons pu constater une amélioration légère de la Précision moyenne et de la R-Précision. Cependant cette approche devrait être testée sur une collection plus volumineuse avec plus de requêtes pour pouvoir conclure de façon certaine sur l'efficacité de cette technique.

Conclusion générale et perspectives

Le travail présenté dans ce mémoire se situe dans le contexte général de l'utilisation de la sémantique pour la représentation de l'information dans les systèmes de recherche d'information et plus particulièrement dans le cadre de l'indexation sémantique basée sur la désambiguïsation du sens.

Nous avons consacré la première partie de ce travail à l'étude des principaux modèles de recherche et à la description du processus d'indexation. Nous avons aboutit au constat de certaines lacunes de tels systèmes basés sur une indexation dite classique. En effet, pour calculer la pertinence document-requête, ces systèmes basent leur comparaison sur le nombre de mots que le document partage avec la requête. Dans une telle approche un document contenant des termes de la requête et pourtant non pertinent est retrouvé, alors que des documents pourtant pertinents ne partageant pas de mots avec la requête sont ignorés.

Nous avons donc enchainé avec l'étude d'un autre type d'indexation « indexation sémantique », offrant la possibilité d'intégrer le sens des termes dans la représentation des documents et requêtes et de les indexer en conséquence. Nous avons étudié les principaux travaux en relation, dont l'approche de Kolte dont on s'est inspiré pour notre approche.

L'approche de Kolte se base sur la désambiguïsation selon le domaine du discours. Ainsi pour retrouver le sens d'un mot polysémique, on commence par rechercher son domaine, ensuite le sens de ce mot dans le domaine en utilisant les relations hiérarchiques de *WordNet, à savoir,* l'holonymie/méronymie, l'hyperonymie auxquelles nous avons proposé d'ajouter la désambiguïsation en utilisant le glossaire de *WordNet.*

Le résultat de notre travail se concrétise en Sem-*Terrier* qui est l'extension de la plateforme Terrier avec notre module d'indexation sémantique . Nous avons ainsi pu faire des tests sur une partie de la collection Muchmore. L'évaluation des résultats montre une amélioration de la précision moyenne de Sem-Terrier par rapport aux résultats obtenus en utilisant Terrier classique .

Sachant qu'on a placé cette méthode comme troisième dans la liste des méthodes à utiliser pour la désambiguïsation du sens ; il convient de faire des tests sur l'ordre de son

emplacement pour déterminer celui qui optimiserait éventuellement les résultats de la désambiguïsation.

L'amélioration de la précision moyenne relativement faible est peut être due au fait que la désambiguïsation n'utilise aucune similarité sémantique. En effet la désambiguïsation des domaines se limite à une correspondance lexicale entre les domaines du mot ambiguë et les domaines des mots non vides de son contexte. De même pour la désambiguïsation du sens dans le domaine, aucune similarité sémantique n'est utilisée entre le contexte du mot ambiguë et ses synsets.

Bien que le temps d'accès au dictionnaire et au tagger soit beaucoup trop long, ce qui constitue en soit un handicap pour l'indexation de grandes collections, nous pouvons, conclure que l'utilisation *WordNet* , des domaines de *WordNet Domains* et du *Stafford POS tagger* offre des perspectives assez intéressantes pour la recherche sémantique.

L'une des perspectives à envisager est d'utiliser la similarité sémantique dans la désambiguïsation du sens en plus de la désambiguïsation par les domaines de WordNet. L'autre est de proposer des méthodes de désambiguïsation pour les adjectifs et les verbes. En effet la méthode proposées en haut ne désambiguïse que les noms une fois le domaine fixé au préalable.

Et pour conclure, pour une utilisation fructueuse de *WordNet* et de *WordNet Domains* dans des temps raisonnable, il va de soit que l'amélioration du temps d'accès au dictionnaire WordNet est nécessaire et ce en revoyant sa structure.

Bibliographie

- **[Abi Chahine, 11] :** Indexation et recherche conceptuelles de documents pédagogiques guidées par la structure de Wikipédia. Thèse de Doctorat en Informatique. L'Institut National des Sciences Appliquées de Rouen, Octobre 2011.
- **[Amirouche, 08]:** Fatiha AMIROUCHE, Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets, Thèse de Doctorat en Informatique de l'université de Toulouse, 2008.
- **[Audibert, 03] :** Audibert L., Outils d'exploration de corpus et désambiguïsation lexicale automatique. Thèse de Doctorat en Informatique de l'Université de Provence. Décembre2003.
- **[Aussenac & al., 04]**: N. Aussenac-Gilles and J. Mothe. Ontologies as background knowledge to explore document collections. In Actes de la Conférence sur la Recherche d'Information Assistée par Ordinateur (RIAO)., pages 129–142, 2004.
- **[Ralalason, 10]** Bachelin RALALASON Représentation multi-facette des documents pour leur accès sémantique , Thèse de Doctorat en Informatique de l'université Paul Sabatier de Toulouse, décembre 2010.
- **[Baeza & al., 92]:** C. Fox. *Lexical analysis and stoplists*, pages 102{130. Frakes W B, Baeza-Yates R (eds) Prentice Hall, New jersey, 1992.
- **[Baeza-Yates & al., 99]:** R. Baeza-Yates and R. A. Ribeiro-Neto. Modern information Retrieval. New York : ACM Press ; Harlow England : Addison-Wesley, cop., 1999.
- **[Baziz & al., 04]** : Mustapha Baziz, Mohand Bo ughanem, Nathalie Aussenac-Gilles. The Use of Ontology for Semantic Representation of Documents. Dans : The 2nd Semantic Web and Information Retrieval Workshop(SWIR), SIGIR 2004, Sheffield UK, 29 juillet 2004. Ying Ding, Keith van Rijsbergen, Iad Ounis, Joemon Jose (Eds.), pp. 38-45.
- **[Baziz , 05] :** BAZIZ M., Indexation conceptuelle guidée par ontologie pour la recherche d'information, Thèse de Doctorat en Informatique de l'université Paul Sabatier de Toulouse, décembre 2005.

- **[Bell & al.]:** David Bell and Jon Patrick. Using WordNet Domains in a supervised learning word sense disambiguation system. Sydney technology research Group. School of information Technologies. University of Sydney, Australia.
- **[Biemann, 05]:** Chris Biemann - Semantic Indexing with Typed Terms Using Rapid Annotation. in Methods and Applications of Semantic Indexing. Workshop at the 7th International Conference on Terminology and Knowledge Engineering. Copenhagen Denmark, Tuesday 16th August 2005.
- **[Boucham, 09]** BOUCHAM Souhila Thème Une approche basée Ontologies pour l'indexation automatique et la Recherche d'Information Multilingue (RIM), 2009.
- **[Boughanem & al., 92] :** M. Boughanem. *les Systèmes de Recherche d'Information : d'un modèle classique à un modèle connexionniste*. PhD thesis, l'Université Paul Sabatier, 1992.
- **[Boughanem & al., 97]:** M. Boughanem and C. Soulé-Dupuy. Query Modification based on Relevance Back-Propagation. In *5th International Conference on Computer Assisted Information Retrieval, RIAO,* , pages 469–487. -, juin 1997. Dates de conférence : juin 1997 1997.
- **[Boughanem & al., 99]:** M. Boughanem, C. Chrisment, and C. Soule-Dupuy. Query modification based on relevance backpropagation in adhoc environment. *Information Processing and Management*, 35 :pages 121–139, 1999.
- **[Boughanem & al., 04] :** M. Boughanem, W. Kraaij, J.Y. Nie, Modeles de langue pour la recherche d'informations, dans Les systemes de recherche d'informations - Modeles conceptuels, ed. M. Ihadjadene, Hermes, pp. 163-184, 2004.
- **[Brajnik & al., 96]:** G. Brajnik, S. Mizzaro, , and C. Tasso. Evaluating user interfaces to information retrieval systems : a case study on user support. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 128.136, Zurich, 1996.
- **[Buitelaar & al., 04]:** Buitelaar, P., Steffen D., Volk, M., Widdows, D., Sacaleanu, B., Vintar, S., Peters, S., Uszkoreit, H., Ev aluation Resources for Concept-based Cross-Lingual IR in the Medical Domain In Proc. of LREC2004, Lissabon, Portugal, May 2004.
- **[Champclaux, 09]:** Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information. Thèse de Doctorat en Informatique de l'université de Toulouse, 2009.

- **[Cleveland & al., 00]** Cleveland, D. B. , & Cleveland, A. D. (2000). *Introduction to Indexing and Abstracting :* (3rded.) Libraries Unlimited.
- **[Croft, 95]:** Croft, W.B. *What Do People Want from Information Retrieval?* D-Lib Magazine, http://www.dlib.org/dlib/november95//11croft.html, November 1995.
- **[Daoud, 09]:** Mariam DAOUD Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche. Thèse de Doctorat en Informatique de l'université de Toulouse, 2009.
- **[Fagan, 87]:** Fagan, Joel L. 1987. Experiments in Automatic Phrase Indexing for Document Retrieval : A Comparison of Syntactic and Non-syntactic methods, PhD thesis, Dept. of Computer Science, Cornell University, Sept. 1987.
- **[Gibbs, 87]:** Gibbs, R. W. (1987). Mutual knowledge and the psychology of conversational inference. Journal of Pragmatics , 11(5), 561- 588.
- **[Gonzalo & al., 98]:** J. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In Proceedings of COLING-ACL '98 Workshop on Usage of Word.Net in Natural Language Processing Systems, Montreal, Canada, August.
- **[Gonzalo & al., 99]:** J. Gonzalo, A. Pefias, and F. Verdejo. Lexical ambiguity and information retrieval revisited. In Proceedings of EMNLP/VLC, 1999.
- **[Guarino & al., 99] :** Guarino, N., C. Masolo, and G. Vetere, OntoSeek: Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs, . 1999, National Research Council, LADSEBCNR: Padavo, Italy.
- **[Hains & al., 93]:** D. Haines and W.B. Croft. Relevance feedback and inference network. In 16*th Annual International ACM SIGIR Conference on Research and developement in Information Retrieval*, pages 2,11, 1993.
- **[Harman, 92a]:** D. Harman. Relevance feedback revisited. In 15*th Annual International ACM SIGIR Conference on Research and developement in Information Retrieval*, pages 1,10, 1992.
- **[Harter, 75]:** Harter, S.A probabilistic approach to automatic keyword indexing. part ii. an algorithm for probabilistic indexing.Journal of the American Society for Information Science (JASIS) 35, 3 (1975), 280–289.

- **[HLAOUA, 07] :** Lobna HLAOUA Reformulation de Requêtes par Réinjection de Pertinence dans les Documents Semi-Structures. Thèse de Doctorat en Informatique de l'université de Toulouse, 2007.
- **[Mallak, 11] :** Ihab Mallak De nouveaux facteurs pour l'exploitation de la sémantique
- d'un texte en Recherche d'Information
- **[Katz & al., 98]:** Özlem Uzuner, Boris Katz, Deniz Yuret : Word Sense Disambiguation for information Retrieval. AAAI/IAAI1999:985
- **[Kent & al., 55]** Allen Kent, Madeline M. Berry, Luehrs, and J. W. Perry. Machine literature sear-ching viii, operational criteria for designing information retrieval systems. American Documentation, 6(2) :93–101, 1955. page 81.
- **[Kolte & al., 09]**: S. G. Kolte and S. G. Bhirud K. J. Somaiya. WordNet: A Knowledge Source for Word Sense Disambiguation. International Journal of Recent Trends in Engineering, Vol 2, No. 4, November 2009 .
- **[Khan, 00] :** Latifur R. Khan, Ontology-based Information Selection, Phd Thesis, Faculty of the Graduate School, Univers ity of Southern California . August 2000.
- **[Kompaoré, 08]** N. D. Kompaoré. Fusion de systèmes et analyse des caractéristiques linguistiques des requêtes : vers un processus de RI adaptatif. Thèse de Doctorat en Informatique de l'université Paul Sabatier de Toulouse, juin 2008.
- **[Krovetz & al., 92]:** R. KROVETZ and W. B. CROFT. Lexical Ambiguity and InformationRetrieval.ACMTransactionsonInformationSystems,Vol.10,No2,pp.115_141. April 1992.
- **[Krovetz, 93]:** Krovetz R, "Viewing Morphology as an Inference Process", in Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 191-202, 1993.
- **[Krovetz,97]:**R.Krovetz. 1997. Homonymy and polysemy in information retrieval . In Proceedings of the 35 th Annual Meeting of the Association for Computational Linguistics (ACL-97}, pages 72-79.
- **[Luhn, 58]:** Luhn, H. The automatic creation of literature abstracts. IBM Journal of Research and Development 24, 2 (1958), 159–165.
- **[Magnini & al., 00]** Magnini B., Cavaglià G. 2000. Integrating Subject Field Codes into WordNet. Actes de LREC-2000, Second International Conference on Language Resources and Evaluation, Athènes, Grèce, pp. 1413-1418.

- **[Magnini & al., 02**]: BERNARDO MAGNINI, CARLO STRAPPARAVA, GIOVA NNIP EZZULO and ALFIO GLIOZZO. The role of domain information in Word Sense Disambiguation. Natural Language Engineering 8 (4): 359–373.2002.
- **[Maniez & al., 91]** : J. Maniez and E. de Grolier. A decade of research in classification. 1991.
- **[Maron & al., 60]:** M. Maron and J. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery*, 7 :pages 216{244, 1960.
- **[Mihalcea & al., 00]:** Mihalcea, R. and Moldovan, D.: Semantic indexing using WordNet senses. In Proceedings of ACL Workshop on IR & NLP, Hong Kong, October 2000. http://www.seas.smu.edu/~rada/papers/acl00.nlp_ir.ps.gz
- **[Mihalcea, 04]**: Mihalcea, R.: Co-training and Self-training for Word Sense Disambiguation. In: Proc. CoNLL, pp. 33–40 (2004).
- **[Miller, 95]:** G. Miller. Wordnet: A lexical database. Communication of the ACM, 38(11):39--41, (1995).
- **[Moldovan & al., 00]:** D. Moldovan and R. Mihalcea. Using wordnet and lexical operators to improve internet searches. In IEEE Internet Computing., pages 34–43, 2000.
- **[Ogawa & al., 91]:** Ogawa, Y., Morita, T., & Kobayashi, K. (1991). A fuzzy document retrieval system using the keyword connection matrix and a learning method. *Fuzzy sets and systems*, *39*(2), 163– 179.
- **[Ponte & al., 98]:** Ponte, J. M., and Croft, W. B. A language modeling approach to information retrieval. research and development in information retrieval. In Proc. of the International ACM-SIGIR Conference (1998), Proc. Of the International ACM-SIGIR Conference, pp. 275–281.
- **[Porter, 80]:** M.F. Porter. An algorithm for suffix stripping. pages 130–137, 1980.
- **[Pustejovsky, 95]:** Pustejovsky, J., Boguraev, B. & Johnston, M. A core lexical engine : The contextual determination of word sense (Tech. Rep.). Department of Computer Science, Brandeis University. (1995).
- **[Ren & al., 99]:** F. Ren, L. Fan, and J. Nie. Aak approach : How to acquire knowledge in an actual application system. IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, pages 136–140, 1999.

- **[Robertson & al., 76]:** S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27 :129–146, 1976.
- **[Robertson, 90]:** S.E. Robertson. On term selection for query expansion. *Journal of Documentation*, 46 :pages 359–364, 1990.
- **[Robertson & al., 94]:** ROBERTSON S. E., WALKER S., ≪ Some Simple Effective Approximations to the2-Poisson Model for ProbabilisticWeighted Retrieval ≫, Proceedings of SIGIR 1994, p. 232-241, 1994.
- **[Robertson & al., 97]:** S. E. Robertson, and S. Walker. On relevance weights with little relevance information. In Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, pages 16–24. ACM Press, 1997.
- **[Rocchio, 71]:** J.J. Rocchio. *"Relevance feddback in Information Retrieval", in The SMART Retrieval System : Experiments in Automatic Document Processing*. Prentice Hall Series in Automatic Computation, 1971.
- [**Roussey & al., 01]** : Roussey, C., Calabretto, S. et Pinon, J.-M. (2001a). A multilingual information system based on knowledge representation. pages 98111.
- **[Schütze & al., 95]:** H. Schütze and J. Pedersen. 1995. Information retrieval based on word senses. In Proceedings of the 4th Annual Symposiu m on Document Analysis and Information Retrieval, pages 161-175.
- **[Salton & al., 68]:** G. Salton, and M. Lesk. Computer Evaluation of Indexing and Text Processing. J. ACM 15(1): 8-36 (1968).
- **[Salton, 70]:** Gerard Salton - Automatic processing of foreign language document – Journal of the American Society for Information Science, May 1970.
- **[Salton, 71b]:** G. Salton. "A Comparison between manual and automatic indexing methods". Journal of the American Documentation, 20(1), pp. 6171, 1971.
- **[Salton & al., 83]:** Salton.G, E.A.Fox,H.Wu: Introduction to modern information retrieval, Mc Craw Hill international book company ISBNO-07-y665266-5, 1983.
- **[Salton & al., 83]:** Salton.G, E.A.Fox,H.Wu: Introduction to modern information retrieval, Mc Craw Hill international book company ISBNO-07-y665266-5, 1983.
- **[Salton & al., 83a]**: G. Salton and M.J. McGill. Introduction to modern information retrieval. McGraw Hill Publishing Company, New York, 1983.

- **[Salton & al., 83a]:** Salton, G., Fox, E., and Wu, H. Extended Boolean information retrieval. CommunicationsoftheACM,26(12),1983.
- **[Salton & al., 83c]:** G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.
- **[Salton, 88]:** Salton, G. Syntactic approaches to automatic book indexing. In Proc. of the annual meeting on Association for Computational Linguistics (ACL) (1988), Department of Computer Science, Cornell University, Ithaca, New York, pp. 204–210.
- **[Salton & al., 88]:** G. Salton and C. Buckley On the use of spreading activation methods in automatic information retrieval. *11th ACM-SIGIR Conference*. p. 147-160, 1988.
- **[Salton, 89]:** Salton, G., "Automatic Text Processing", Addison Wesley, 1989.
- **[Sanderson, 94]:** M. Sanderson. 1994. Word sense disambiguation and information retrieval. In Proceedings of the 17th Annual Internati onal ACM-SIGIR Conference on Research and Development in Information Retrieval, pages 142-151, Springer- Verlag.
- **[Singhal & al., 97]:** A. Singhal, M. Mitra, and C. Buckley. (1997). Learning routing queries in a query zone. In Proceedings of the 20th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Philadelphia, Pennsylvania, United States, July 27 - 31, 1997). N. J. Belkin, A. D. Narasimhalu, P. Willett, and W. Hersh, Eds. SIGIR '97. ACM Press, New York, NY, 25-32.
- **[Sowa, 84]:** J. Sowa. Conceptual Structures: information processing in mind and machine. In The System Programming Series, Reading: Addision Wesley publishing Company, 1984. 481 pages.
- **[Sparck, 79]:** K. Sparck Jones. Experiments in relevance weighting of search terms. Inf. Process. Manage. 15(3): 133-144, 1979.
- **[Sparck Jones, 86]:** Sparck Jones, K. (1986). Synonymy and semantic classification. Edinburgh, England : Edinburgh University Press.
- **[Voorhees, 93]:** Voorhees, E.M. Using WordNet to disambiguate word senses for text retrieval. Association for Computing Machinery Special Interest Group on Information Retrieval. (ACM-SIGIR-1993): 16thAnnual International Conference on Research and Development in Information Retrieval, 171–180. (1993).
- **[Voorhees, 94]:** E. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and*

development in information retrieval, pages 61.69, Dublin, Ireland, 1994. Springer-Verlag New York, Inc.

- **[Wong & al., 85]:** Wong, S., Ziarko, W. et Wong, P. (1985). Generalized vector spaces model in information retrieval. In Proc. of the 8th ACM-SIGIR conference, pages 18-25. Montreal, Quebec.
- **[Woods, 97]:** William A. Woods. 1997. Conceptual indexing : A better way to organize knowledge. Technical Report SMLI TR-97-61, Sun Microsystems Laboratories, Mountain View, CA, April. ww.sun.com/research/techrep/1997/abstract-61.html.
- **[Woods & al., 98]:** W. A. Woods and J. Ambroziak. Natural language technology in precision content retrieval. In Proceedings of the International Conference on Natural Language Processing and Industrial Applications, (NLP+IA), Moncton, Canada, 1998.
- **[Zadeh, 65]:** L. A. Zadeh. Fuzzy sets, Information and Control, 8, 338-353, 1965.
- **[Zemirli, 08] :** Wahiba Nesrine Zemirli, Modèle d'accès personnalisé à l'information basé sur les Diagrammes d'Influence intégrant un profil utilisateur évolutif . Thèse de Doctorat en Informatique de l'université de Toulouse, 2008.

Annexe 1 :

WordNet , WordNet Domains

Cette annexe sera consacrée à la présentation de WordNet et WordNet Domains qui sont les deux ressources sémantiques utilisées pour la désambiguïsation dans le cadre de ce travail de recherche.

1.1. Origine de WordNet :

WordNet est une base de données lexicale développée depuis 1985 par des linguistes du laboratoire des sciences cognitives de l'université de Princeton, dirigé par le professeur George A. Miller. WordNet est un réseau sémantique de la langue anglaise, qui se fonde sur une théorie psychologique du langage. La première version diffusée remonte à juin 1991.

Son but est de répertorier, classifier et mettre en relation de diverses manières le contenu sémantique et lexical de la langue anglaise, mais abusivement, certains la considèrent comme une ontologie, d'autres comme un thésaurus. Le système se présente sous la forme d'une base de données électronique qu' on peut télécharger sur un système local. Des interfaces de programmation sont disponibles permettant de l'interroger. Dans le cas de WordNet 2.0, l'API permettant de l'interroger est la JWNL 1.3 (Java WordNet Language version 1.3).

1.2. Structure de WordNet :

Dans WordNet, un nœud est appelé Synset (qui est un concept). Il contient trois parties:

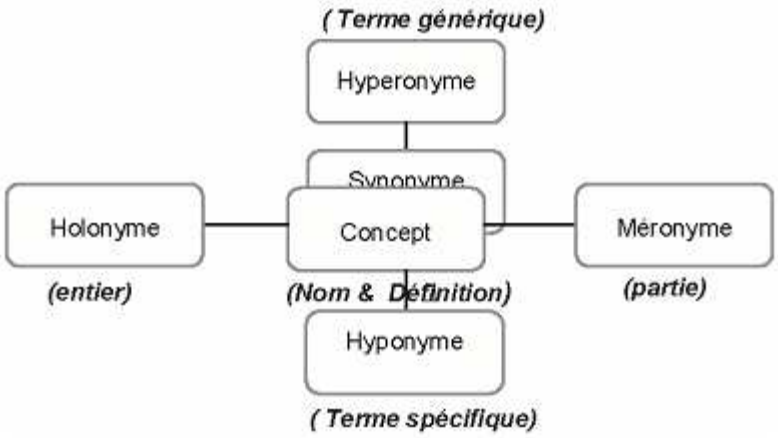
- Le terme représentant du Synset : c'est le terme pour lequel le concept (synset) est identifié. Dans WordNet, les termes les plus utilisés sont placés en premier
- Les termes synonymes : une liste de termes interchangeables séparés par des virgules

- Le glossaire : il est mis entre parenthèses et vient après le symbole "--". Il contient une définition du concept avec éventuellement un ou plusieurs exemples du monde réel (mis entre double côtes, "").

L'exemple suivant montre les nœuds correspondants aux différents sens de "mouse" dans WordNet:

```
1.   mouse -- (any of numerous small rodents typically resembling
diminutive rats having pointed snouts and small ears on elongated
bodies with slender usually hairless tails)

2.   shiner, black eye, mouse -- (a swollen bruise caused by a blow to
the eye)

3.   mouse -- (person who is quiet or timid)

4.   mouse, computer mouse -- (a hand-operated electronic de vice that
controls the coordinates of a cursor on your computer screen as you
move it around on a pad ;  on the bottom of the device is a ball that
rolls on the surface of the pad ;  "a mouse takes much more room than a
trackball")

5.   sneak,  mouse, creep, pussyfoot -- (to go stealthily or furtively
;  "..stead of sneaking around spying on the neighbor's house")

6.   mouse -- (manipulate the mouse of a computer)
```

Les nœuds sont ensuite reliés entre eux suivant différentes relations sémantiques.



Principales relations sémantiques dans WordNet

Voici les principales relations sémantiques reliant les nœuds :

- ✓ **Synonymie** , les synonymes étant associés à la classe Concept.

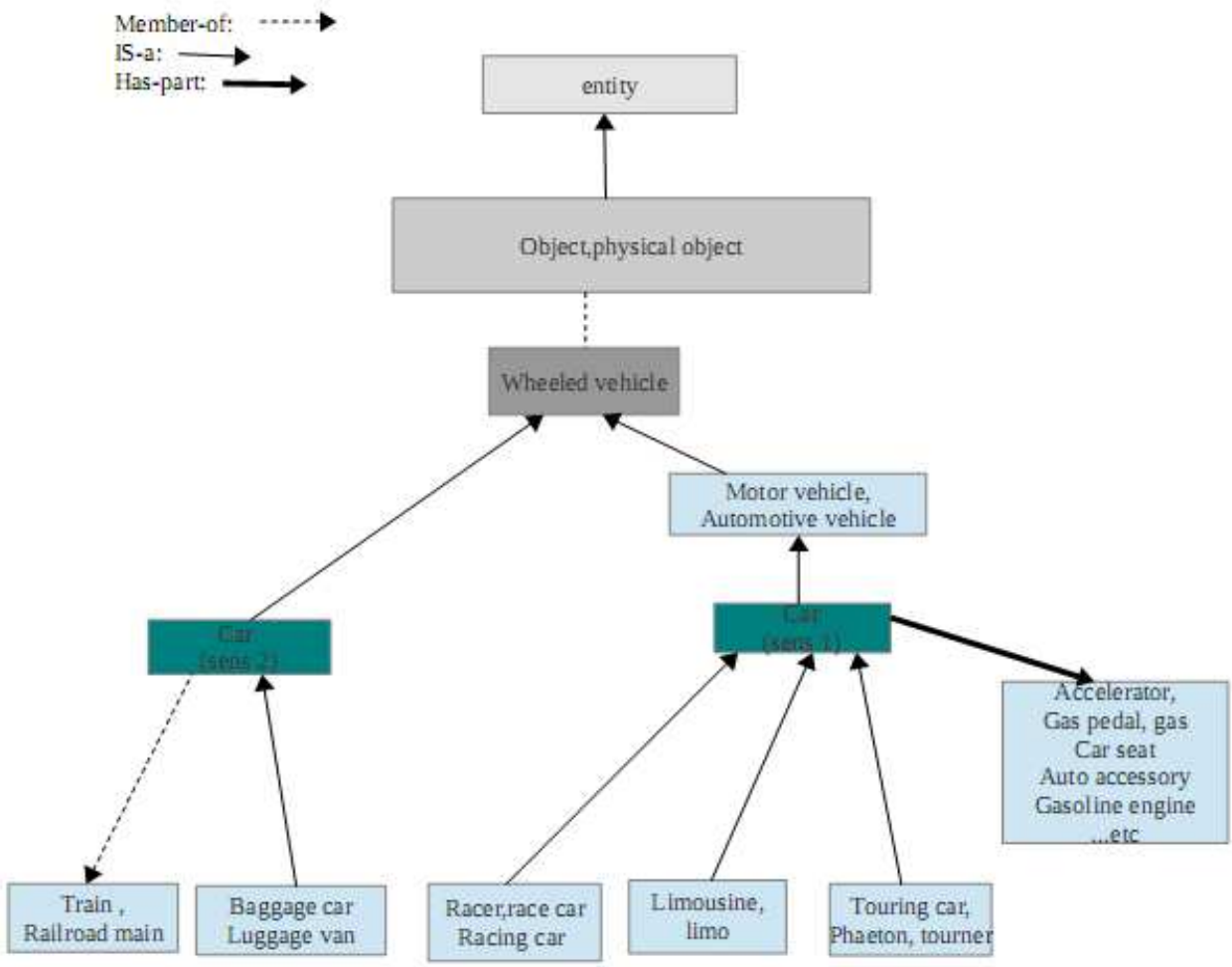
- ✓ **Relation Hyperonymie**: C'est le terme générique utilisé pour désigner une classe englobant des instances de classes plus spécifiques. Y est un hyperonyme de X si X est un type de (kind of) Y.
- ✓ **Relation Hyponymie**: C'est le terme spécifique utilisé pour désigner un membre d'une classe (relation inverse de Hyperonymie). X est un hyponyme de Y si X est un type de (kind of) Y.
- ✓ **Relation Holonymie**: Le nom de la classe globale dont les noms méronymes font partie. Y est un holonyme de X si X est une partie de (is a part of) Y.
- ✓ **Relation Méronymie**: Le nom d'une partie constituante (part of), substance de (substance of) ou membre (member of) d'une autre classe (relation inverse de l'holonymie). X est un méronyme de Y si X est une partie de Y. exemple : {voiture } a pour méronymes {{ porte}, {moteur }}.

1.3. Contenue de WordNet :

WordNet couvre la majorité des noms, verbes, adjectifs et adverbes de la langue Anglaise. Sa dimension ainsi que le domaine de la langue générale qu'il traite lui permette souvent de couvrir les sujets traités dans les collections de tests conventionnelles de la RI (TERC, CLEF). Ces dernières sont le plus souvent de type presse.

1.4. Les concepts dans WordNet :

Les concepts de ce réseau n'ont cependant pas vocation à sous-tendre un système de représentation des connaissances. Ils ont été appliqués en TALN dans de nombreux contextes, en particulier pour l'indexation sémantique de textes et à des fins de recherche documentaire



Exemple de contenu et de structure de WordNet

1.3. Limites de WordNet :

1.3.1 Informations manquantes :

WordNet ne précise pas l'étymologie, la prononciation, les formes de verbes irréguliers et ne contient que des informations limitées sur l'usage des mots.

1.3.2 Profusion de sens pour un mot donné :

La contrepartie de son importante couverture est que WordNet est très précis dans le sens des définitions. On a une granularité très (trop ?) fine des sens. Par exemple, le verbe to give (« donner ») n'a pas moins de 44 sens. Une telle profusion ne facilite pas une tâche de désambiguïsation lexicale.

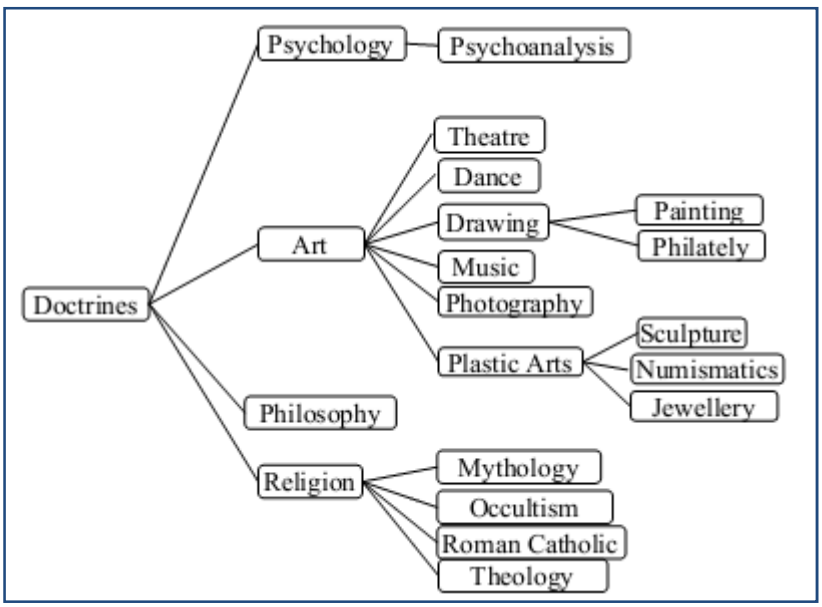
1.3.3 Absence de relations pragmatiques :

WordNet ne matérialise pas d'une façon formelle tout le sens contenu dans les définitions des termes. Par exemple, l'information qu'un chat ne rugit pas figure dans la définition, mais ne se retrouve formalisée dans aucune relation. De même, des relations pragmatiques telles que savon / bain (SOAP#1 / BATH#2) sont absentes de WordNet.

1.5 WordNet Domains :

WordNet Domains [Magnini & al., 00] est une extension multilingue de WordNet 2.0, développée à l'Instituto Trentino di Cultura (IT C-irst). La notion de domaine a été employée aussi bien en linguistique qu'en lexicographie pour marquer des usages des mots. Les domaines sémantiques offrent une manière naturelle d'établir des relations sémantiques entre les sens des mots, qui peuvent être utilisée avec profit en informatique linguistique. Dans WordNet Domains, chaque synset est annoté avec au moins une étiquette de domaine (par exemple Sport, Politique , Médecine , Economie…), choisie dans un ensemble d'environ deux cents étiquettes organisées hiérarchiquement.

Dans Wordnet Domains les domaines sont organisés en cinq arbres principaux. La figure suivante montre un fragment de l'un des cinq arbres principaux dans la Wordnet Domains original Hierarchy (WDH).



Fragment de La WDH

Un domaine peut inclure des synsets de différentes parties du discours et de différentes sous-hiérarchies de WordNet. Par exemple le domaine Médecine regroupe des sens de noms tels que DOCTOR#1 (le 1er sens du mot docteur) et HOSPITAL #1, et de verbes comme OPERATE#7.

L'information apportée par ces domaines est complémentaire à celles déjà présentes dans WordNet. Les domaines peuvent créer des regroupements homogènes des sens d'un même mot, avec comme effet secondaire de réduire la polysémie des mots dans WordNet

Le mot bank , par exemple, a dix sens dans WordNet 2.0. Trois d'entre eux (BANK #1, BANK #3 et BANK #6) sont regroupés au sein du domaine Economie, tandis que deux (BANK #2 et BANK #7) sont regroupés avec les étiquettes de domaine Géographie et Géologie.

Sens	Synset (Définition)	Domaines
#1	depository financial institution, bank, banking concern, banking company (a financial institution...)	*Economy*
#2	bank (sloping land ...)	*Geography, Geology*
#3	bank (a supply or stock held in reserve...)	*Economy*
#4	bank, bank building (a building...)	*Architecture, Economy*
#5	bank (an arrangement of similar objects...)	*Factotum*
#6	savings bank, coin bank, money box, bank (a container...)	*Economy*
#7	bank (a long ridge or pile...)	*Geography, Geology*
#8	bank (the funds held by a gambling house...)	*Economy, Play*
#9	bank, cant, camber (a slope in the turn of a road...)	*Architecture*
#10	bank (a flight maneuver...)	*Transport*

Tableau 1.1 Exemple des domaines associés aux différents sens du mot 'bank' dans WordNet Domains

Annexe 2

La plateforme de RI « Terrier »

2 .1 Présentation de Terrier

Terrier , Terabyte RetrIEveR est un moteur de recherche robuste et efficace, développé par le département informatique de l'université Glasgow en Ecosse. Il est open source et entièrement écrit en java. Il est utilisé aussi bien pour la recherche web, ad hoc et multilingue.

Terrier offre une plate forme idéale destinée à l'indexation de volumes importants de documents: jusqu'à 25 millions de documents. En plus de l'indexation terrier offre deux autres fonctionnalités qui sont la recherche et l'évaluation. En résumé :

- ✓ L'indexation classique : permet l'extraction des mots clés des documents appartenant à une collection et les stocke dans un index.
- ✓ Recherche : permet de retrouver des documents pertinents pour répondre aux requêtes formulés par l'utilisateur.
- ✓ Evaluation : permet de mesurer le degré de pertinence des résultats de la recherche aux requêtes formulées par l'utilisateur.

2.2 Installation de Terrier :

- **Exigences de Terrier** : La seule exigence de Terrier se compose d'un JRE 1.6.0 installé ou plus. On peut facilement télécharger le JRE, JDK (pour développer avec Terrier ou exécuter l'interface web), à partir du site web de Java[1] .
- **Téléchargement de Terrier** : Une copie de la version 3.5 Terrier peut être téléchargé à partir de la page d'accueil du projet *Terrier* [2]. Le site offre des copies des versions précompilés sous Unix et Windows. On peut y télécharger la version la plus récente de Terrier ou les versions antérieures.
- **Installation** : Afin de pouvoir utiliser Terrier on peut simplement extraire le contenu du fichier Zip téléchargé dans un répertoire. Terrier requiert la version Java 1.6 ou supérieur. Enfin, Terrier suppose que java.exe est dans le PATH, on doit donc s'assurer que Java \ bin est dans la variable d'environnement PATH.

2.3 Structure de Terrier

Une fois dézippé le fichier terrier-3.5 contient les éléments suivants :

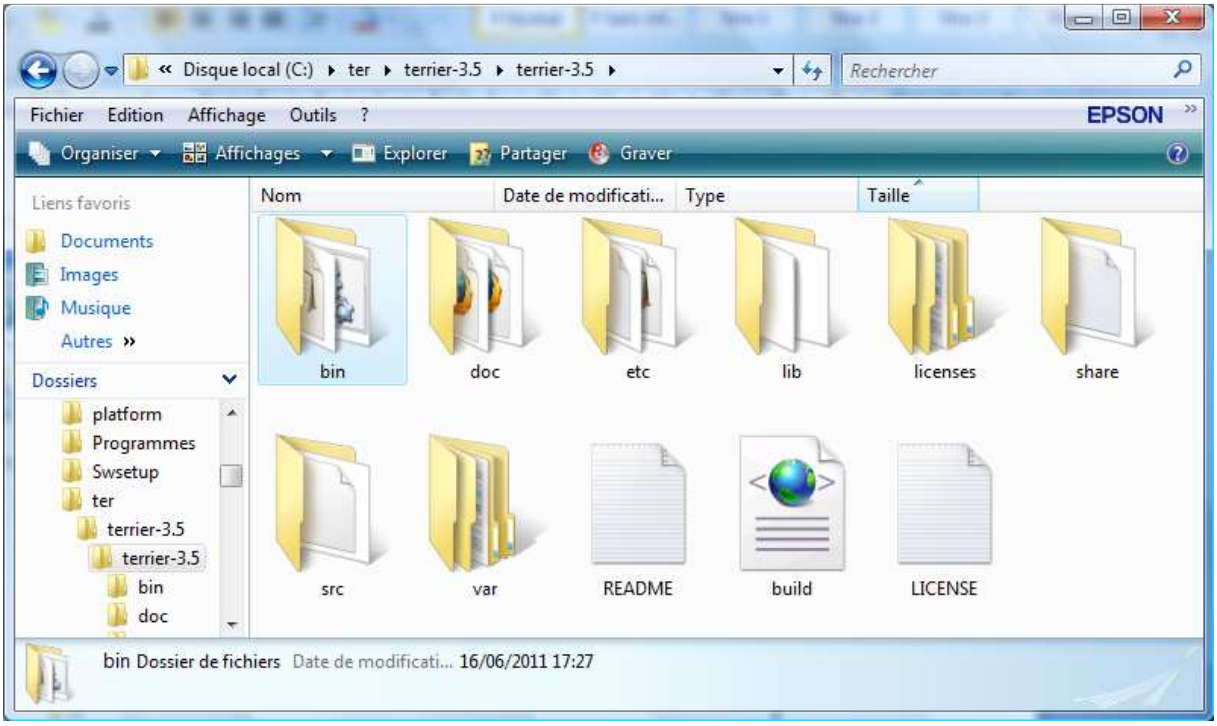


Figure 2.1 Structure de Terrier 3.5

- bin\ : Ensemble de scripts pour exécuter terrier.
- doc\ : Documentation relative à terrier.
- etc\ : Fichiers de configuration de terrier, le fichier terrier.properties.sample contient la plupart des propriétés de configuration de terrier.
- lib\ : Classes compilés de terrier et les différentes librairies externes utilisées par terrier

- share\ : Liste des mots vides (stopword-list.txt) et des exemples de documents à tester sur terrier
- scr\ : Code source Java de terrier
- var\ : Contient deux dossiers le premier est crée suite à une commande d'indexation, le deuxième est crée suite à une commande de recherche.
  - index\ Structures de données après indexation (fichier inverse, fichier lexicon, index direct, index documents)
  - result\ Résultats de la recherche et l'évaluation.

2.4 Applications de Terrier :

Terrier est livré avec trois applications:

- ✓ **Batch (TREC) Terrier** : Cela permet de facilement indexer, rechercher et évaluer les résultats sur les collections TREC. Un tutoriel sur la façon d'utiliser cette application est disponible sur le site officiel de Terrier[3].
- ✓ **Interactive Terrier** Cela permet de faire une recherche interactive. C'est un moyen rapide de tester Terrier. Étant donné que vous avez installé Terrier sur Windows, vous pouvez commencer Terrier Interactive en exécutant le fichier dans le répertoire `bin interactive_terrier.bat` Terrier. Sur un système Unix ou Mac, vous pouvez exécuter Terrier interactive en exécutant le fichier `interactive_terrier.sh`
- ✓ **Desktop Terrier** : Interface graphique de Terrier.

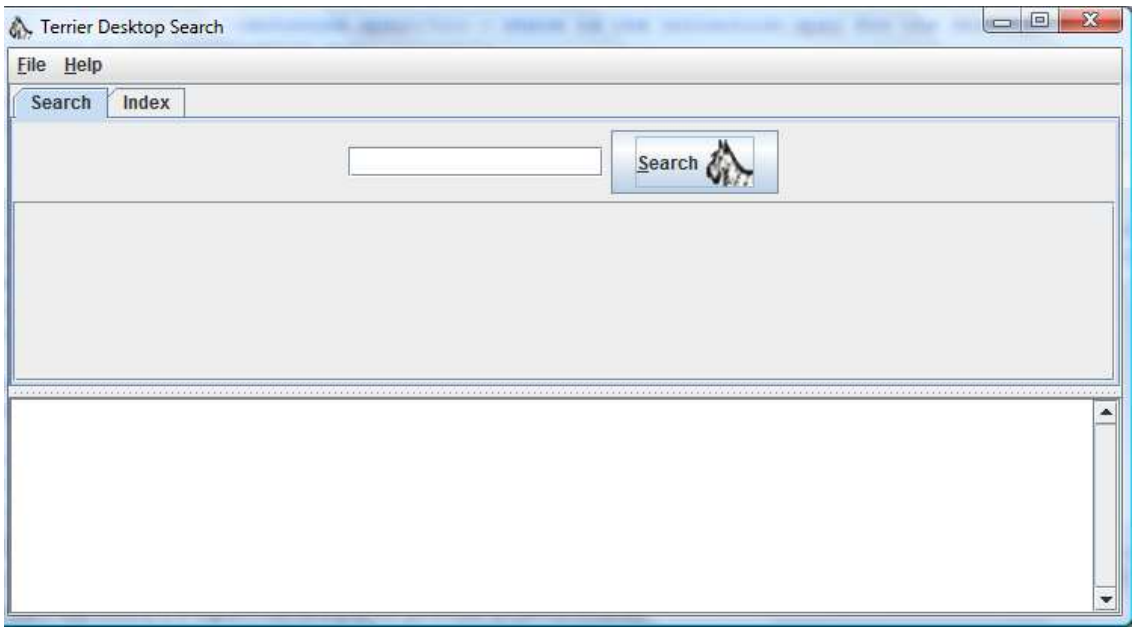


Figure 2.2 Interface du DesktopTerrier

2.5 Composants de Terrier :

2.5.1 L'API d'indexation :

L'indexation de terrier est divisé en quatre étages (procédures) :

1. Extraction de l'objet *Document* à partir de la *collection*. L'objet *collection* est une représentation des corpus reçus en entrée par Terrier.
2. Parcourir chaque document de la collection et en extraire les termes. Chaque terme trouvé sera envoyé au composant TermPipeline.
3. Traitement des termes extraits de chaque document par le TermPipeline.
4. Construction de l'index.

Le graphique ci-dessous donne un aperçu de l'interaction des principaux composants impliqués dans le processus d'indexation de Terrier.

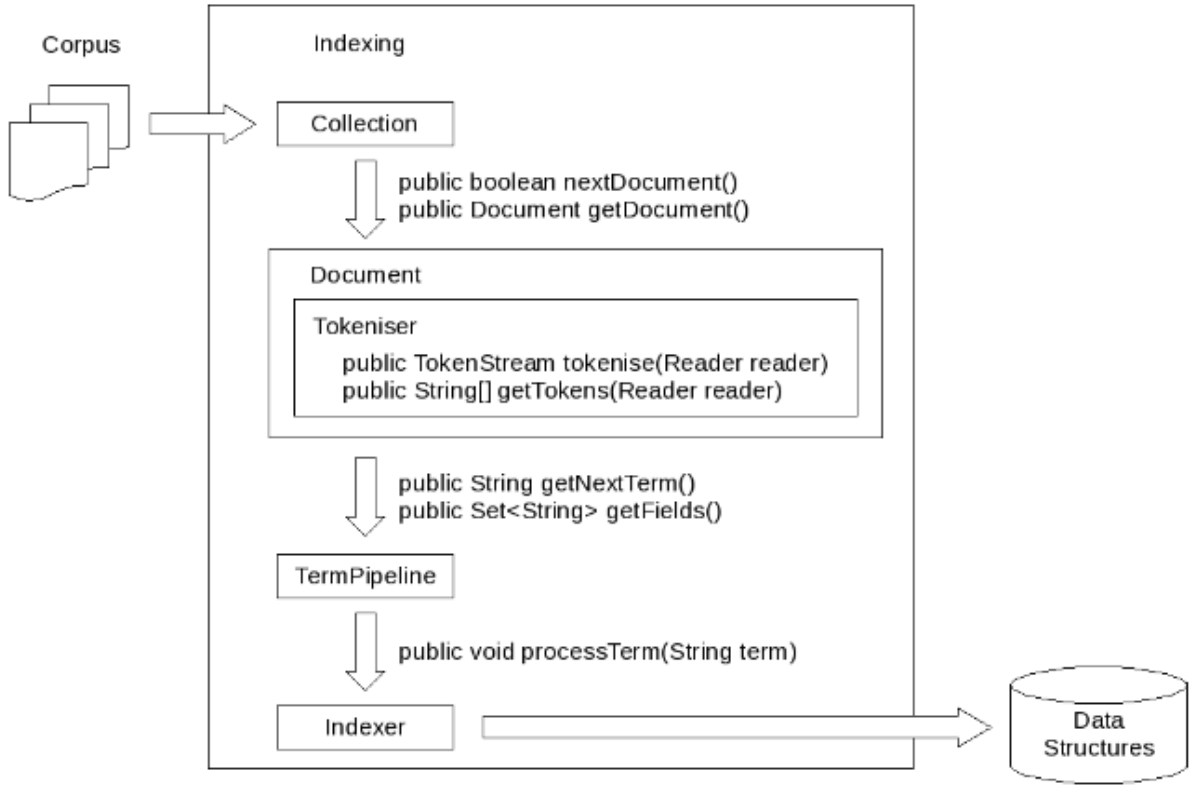


Figure 2.3 Processus d'indexation de Terrier

2.5.1.1 Collection:

Cette composante est une Interface dans le package org.terrier.indexing, elle représente un concept fondamental de l'indexation avec terrier. Son rôle est de Splitter une collection (préalablement construite à partir de corpus) en documents. Pour cela elle fait appel à plusieurs de ses méthodes telles que :

- ✓ public Document getDocument();
- ✓ public boolean nextDocument();
- ✓ public String getDocid();
- ✓ public boolean endOfCollection();

Plusieurs Classes implémentent l'interface collection selon le format du document :

- ✓ SimpleFileCollection: PDF, TXT, HTML,…..
- ✓ TrecCollection
- ✓ SimpleXMLCollection : XML
- ✓ SimpleMedlineXMLCollection
- ✓ TRECUTFCollection
- ✓ WARC018Collection
- ✓ WARC09Collection.

2.5.1.2 Document :

Ce composant est une interface qui se trouve dans le package org.terrier.indexing. cette interface englobe une concept important, celui de Document. Son rôle est de parcourir les documents et d'en extraire les *Termes* en utilisant *Tokeniser* .

Plusieurs méthodes sont utilisées :

- ✓ public String getNextTerm();
- ✓ public boolean endOfDocument();

Plusieurs Parseurs sont disponibles selon le format du document

- ✓ HTMLDocument
- ✓ FileDocument
- ✓ MSExcelDocument

2.5.1.3 Term Pipeline :

C'est une interface qui se trouve dans le package org.terrier.terms. son rôle consiste en le traitement des termes extraits. Il peut transformer les termes ou supprimer des termes qui ne devraient pas être indexés. En définitive il :

- ✓ Elimine les mots vides (Stopwords)
- ✓ Lemmatise les termes selon la langue : pour l'anglais l'algorithme de lemmatisation utilisé est celui de Porter (PorterStemmer).

2.5.1.4 Indexer :

Ce composant est responsable de la gestion du processus d'indexation et entre autre de la construction de l'index. Il instancie les termes pipelines et initialise les *Builders* qui sont en charge de l'écriture de l'index sur le disque dans la structure de données appropriée.

Terrier offre deux types d'indexeur : *BasicIndexer, BlockIndexer*.

2.5.1.5 Data Structures (Structures d'Index):

- **BitFile**

- **Direct Index**
  - Id document
  - Longueur document
  - Byte offset dans Direct Index

- **Document Index** Index
  - Id Terme
  - Fréquence Terme
  - #Filds (#of fields bits)

- **Inverted Index :** Fichier inverse
  - Id Terme
  - Id document
  - Fréquence terme dans le document
  - #Filds (# of fields bits)

- **Lexicon :** Informations sur chaque terme de la collection
  - Terme
  - Id terme
  - Nombre documents qui contiennent le terme
  - Fréquence terme dans la collection
  - Offset terme dans le fichier inverse

- **Meta Index :** informations additionnelles (méta informations) sur chaque document comme son *docno* ou son URL.

Exemple d'indexation

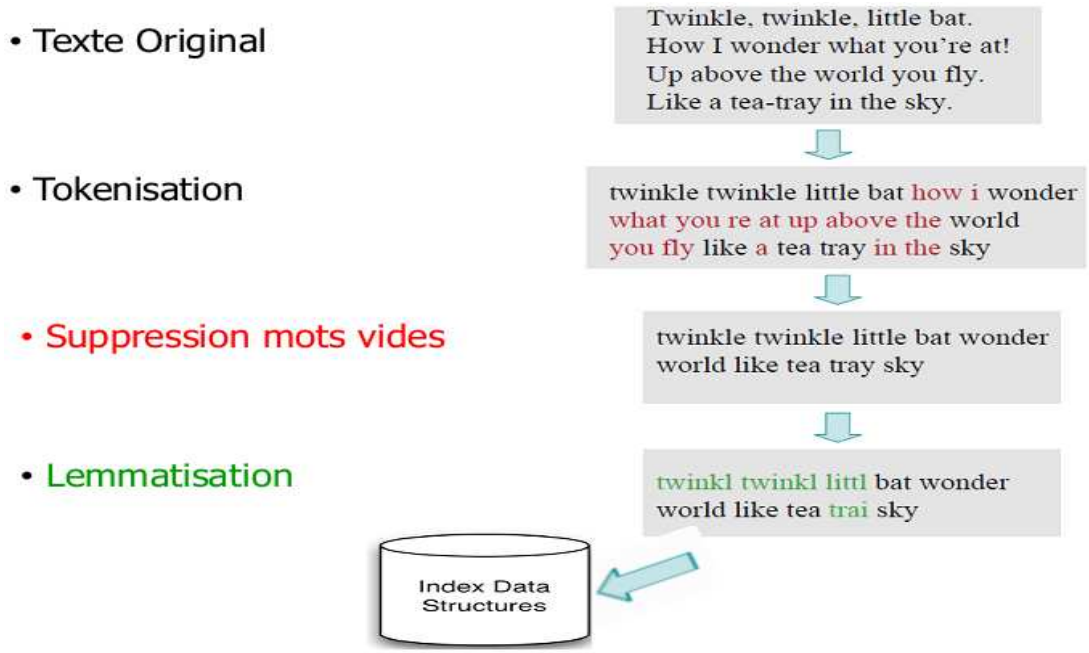


Figure 2.4 Exemple d'indexation

2.5.2 L'API de recherche :

Une fois l'index construit, la phase de recherche peut débuter :

- Chaque requête sera analysée et une instanciation d'un objet de requête aura lieu.
- La requête sera transmise à la composante *Manager* qui aborde le pré-traitement de la requête, en l'appliquant au TermPipeline configuré.
- Après le pré-traitement, la requête sera transmise à la composante Matching qui est responsable de l'initialisation du WeightingModel approprié et DocumentScoreModifiers. Une fois tous ces éléments ont été instancié le calcul des scores des documents à l'égard de la requête aura lieu.
- Ensuite, le post-filtrage et post-traitement ont lieu. Le post-filtrage va filtrer les résultats. Ensuite, en post-traitement, le ResultSet peut être modifié , par exemple, QueryExpansion élargit la requête, puis appelle à nouveau *Matching* afin de générer un meilleur classement de documents.

Le graphique ci-dessous donne un aperçu de l'interaction des principaux composants impliqués dans la phase de recherche de Terrier que nous allons détailler dans ce qui suit :

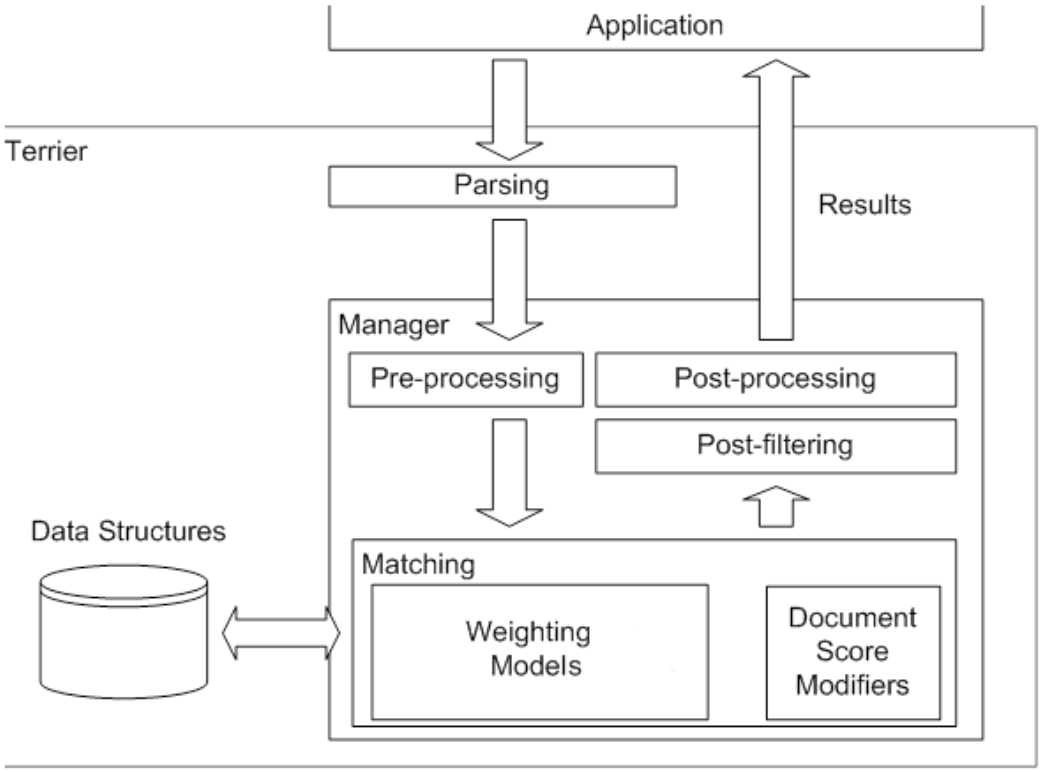


Figure 2.3 Processus d'indexation de Terrier

2.5.2.1 Query

C'est une Classe abstraite dans le package org.terrier.querying.parser qui modélise la requête. Un objet Query est crée pour chaque requête.

Terrier supporte Trois modèles de requête:

- ✓ *SingleTermQuery*: modèle de requête avec un seul terme.
- ✓ *MultiTermQuery* : modèle de requête avec plusieurs termes.
- ✓ *FieldQuery* : Terme qualifié par un champ (Exemple: dans le titre du document)

2.5.2.2 Manager :

Ce module est chargé de la gestion de la recherche il fait appel à quatre sous modules

1. ***Pre-processing***: Appliquer Tokeniser et TermPipeline
2. ***Matching***: Déterminer les documents qui répondent à la requête en initialisant:
   a. *WeightingModels* : Assigner un score pour chaque terme de la requête dans le document (Pondération). Plusieurs Modèles de pondération sont disponibles dans org.terrier.matching.models dont : TF_IDF, BM25.
   b. *DocumentScoreModifiers* : Modifier le score d'un document en fonction du langage de la requête (ou expansion de la requête)
3. ***Post-filtring***: Filtrer les documents pertinents pour la requête

4. ***Post-processing***: Reclasser les documents pertinents s'il y a une expansion de la recherche.

2.5.2.3 Set-Results

Se charge de récupérer les résultats de Post-processing et de retourner les documents selon leur degré de pertinence.

Langage de requetes :

term1 term2	Documents qui contiennent term1 ou term2
{term1 term2}	Documents qui contiennent term1 ou term2, les deux termes sont traités comme synonymes
term1^2.3	Le score du term1 est multiplié par 2.3
+term1 +term2	Documents qui contiennent term1 et term2
+term1 -term2	Documents qui contiennent term1 et ne contiennent pas term2
title:term1	Documents dont leur titre contient term1
term1 -title:term2	Documents qui contiennent term1 mais le titre ne contient pas term2
"term1 term2"	Documents dont term1 et term2 apparaissent dans la phrase
"term1 term2"~n	Documents dont term1 et term2 apparaissent dans un bloc d'une distance n. L'ordre n'est pas important

2.6 Utilisation du Batch (TREC) Terrier :

Le Batch(TREC) Terrier peut être utilisé pour l'indexation, la recherche et l'évaluation des résultats. Dans ce qui suit, nous décrirons les principales commandes relatives à ces trois taches sous Windows.

Avant tout, dans l'invite de commande, spécifier le chemin vers le répertoire bin

Dans l'invité de commandes : Spécifier le chemin vers le dossier bin de terrier pour démarrer :

*cmd>**cd** <Path vers .. \terrier-3.5\ bin >*

2.6.1 Indexation :

Avant d'indexer il faut d'abord supprimer le contenu du dossier *var\index* et initialiser Terrier pour une nouvelle indexation en spécifiant le chemin vers la collection TREC à indexer avec la commande *trec_setup :*

*cmd>.**bin\Trec_setup** <Chemin \vers \ … \collection>*

cela se traduit par la création dans etc\ le fichier *collection.spec* qui contient le chemin vers les documents à indexer.
Une fois la collection spécifiée on peut lancer l'indexation avec la ommande *trec_terrier.*

- ✓ Pour une indexation classique two_pass :

*cmd>.**bin\Trec_terrier -i***

- ✓ pour une indexation classique Single_pass (Direct Index non construit : Gain d'espaces mémoire). Utilisé pour les collections complexes et volumineuses :

*cmd>.**bin\Trec_terrier -i -j***

Terrier offre la possibilité de changer les propriétés du processus d'indexation et même des autres tâches. On peut consulter les propriétés possibles dans le fichier *terrier.properties.sample*. Pour modifier une propriété il faut l'ajouter ou la modifier au fichier *terrier.properties*
Par exemple on peut :

- Changer le type de la collection pour indexer une collection autre que TREC

```
trec.collection.class=TRECWebCollection
```

- Définir la taille max des id des documents*:*

```
indexer.meta.forward.keylens=500
```

- Modifier la taille max d'un terme dans le doc :

```
max.term.length =500
```

- Ignorer la suppression des mots vides ou ignorer la lemmatisation :

```
termpipelines= WeakPorterStemmer
```

```
termpipelines= Stopwords
```

- Sauter les termepipelines :

```
termpipelines.skip
```

2.6.2 Recherche

Après avoir ajouté la propriété trec.topics qui spécifie le chemin vers le fichier texte contanant les requêtes dans terrier.properties:

```
trec.topics=<Path vers le fichier txt contenant les requêtes>
```

on peut exécuter la commande de recherche :

*cmd>.**bin\Trec_terrier -r***

le résultat de la recherche est stocké dans le fichier évaluation (.res) est dans ..\terrier-3.5\var\result\ x.res

On peut aussi modifier les propriétés du processus de recherche, par exemple on peut :

- Indiquer au Parser : Analyser une requête simple par ligne

```
trec.topics.parser=SingleLineTRECQuery
```

- Configuration du modèle de pondération

```
Trec.model= BM25

Trec.model=TF_IDF
```

- Prendre en charge les termes dont Idf faible lors de la recherche

```
ignore.low.idf.terms=false
```

2.6.3 Evaluation

Après avoir Spécifié dans terrier.properties le chemin vers le Rel_Ass, qui contient les documents pertinents pour chaque requête :

```
trec.qrels=<path vers fichiers Rel_Ass.qrels>
```

on peut lancer le processus d'évaluation avec la commande

*cmd>.**bin\Trec_terrier –e***

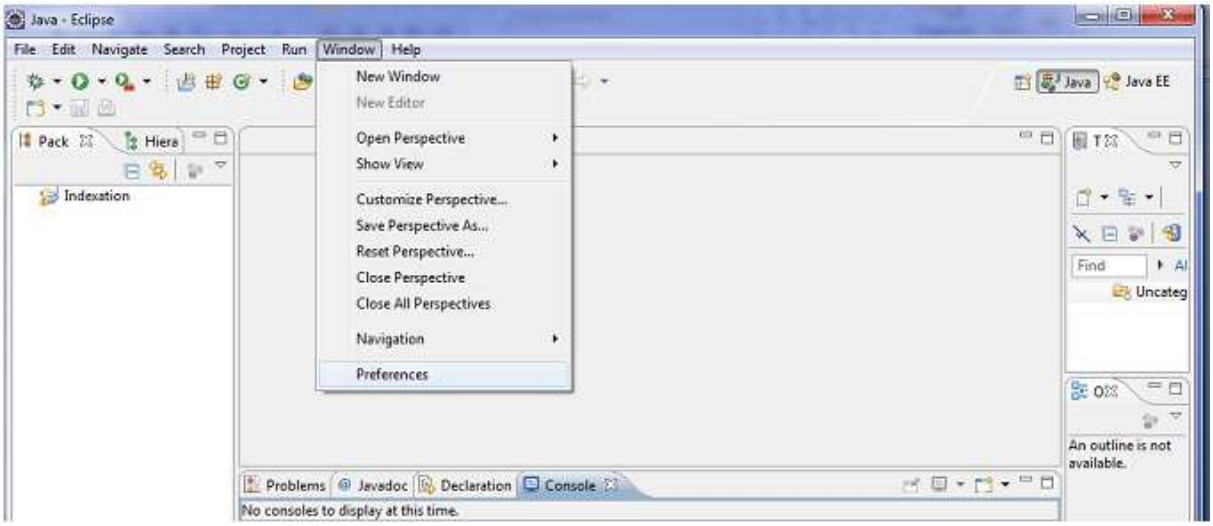
le résultat de l'évaluation est stocké dans le fichier évaluation (.eval) est dans ..\terrier-3.5\var\result\ x.eval

2.7 Compilation de Terrier

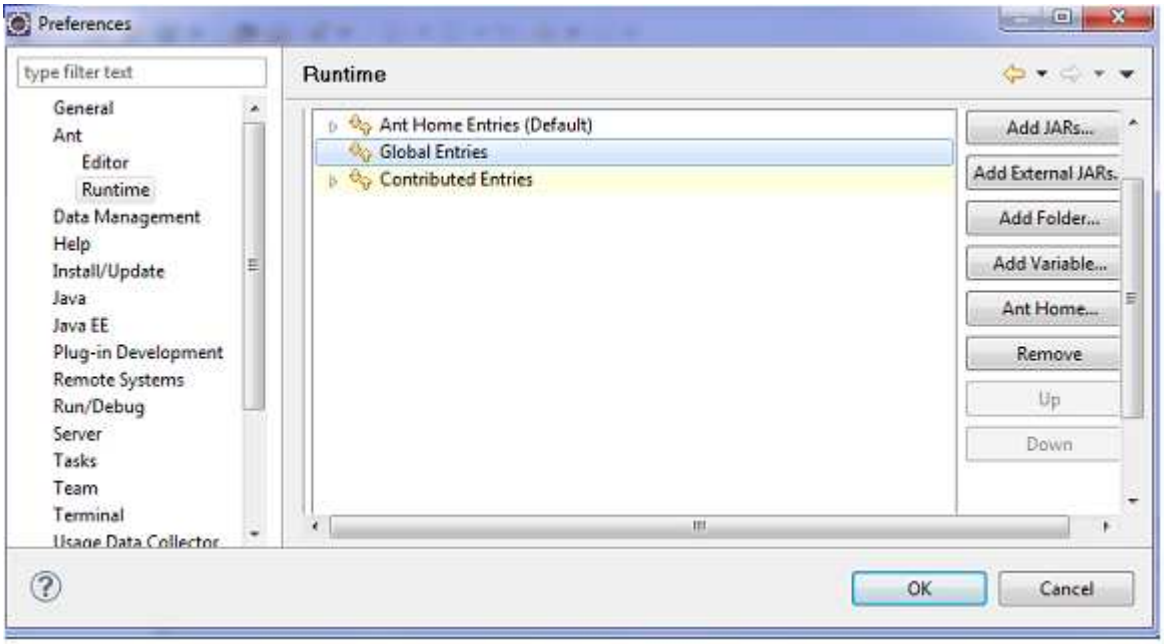
Bien que ce ne soit pas tâche facile, l'un des avantages majeurs de Terrier est son extensibilité. En effet, il est possible de modifier son code source ou d'intégrer de nouvelles composantes. Pour cela une recompilation de terrier est nécessaire.

Pour compiler Terrier, utiliser le Ant de l'environnement IDE Eclipse et le fichier build.xml de terrier 3.5.

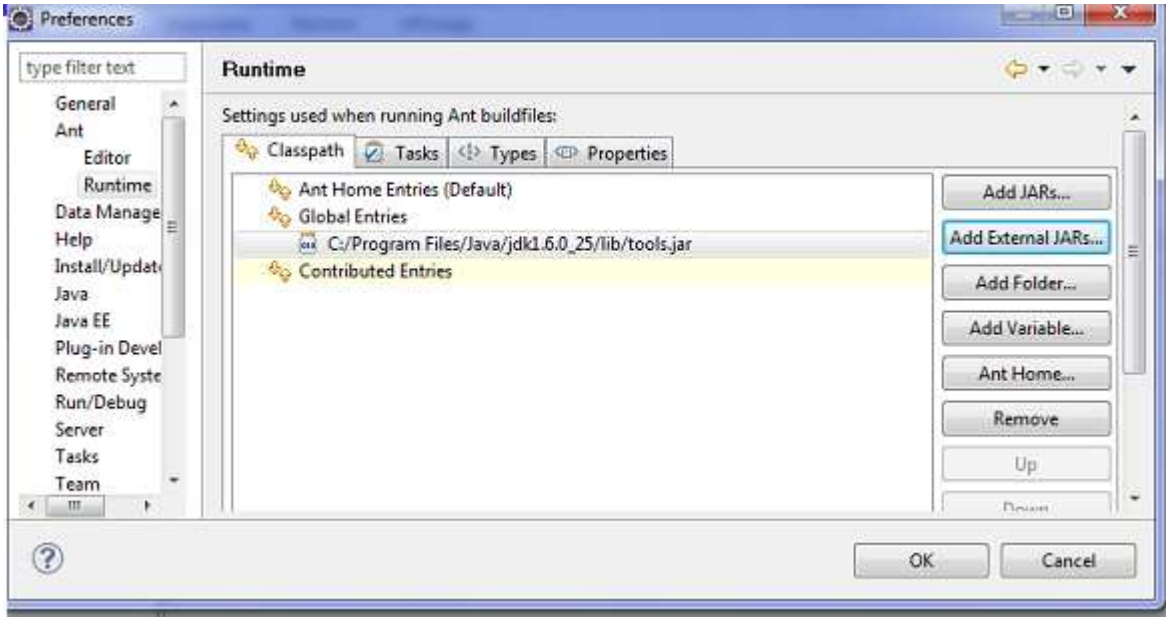
Configuration Ant de l'IDE Eclipse



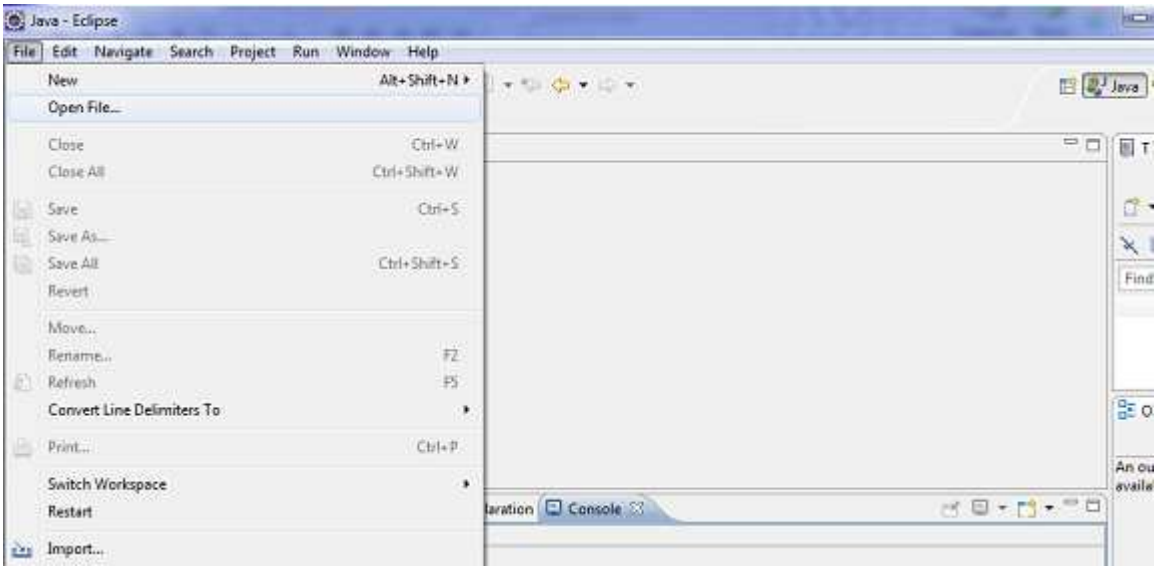
Configuration Ant : 1- Window->Preferences



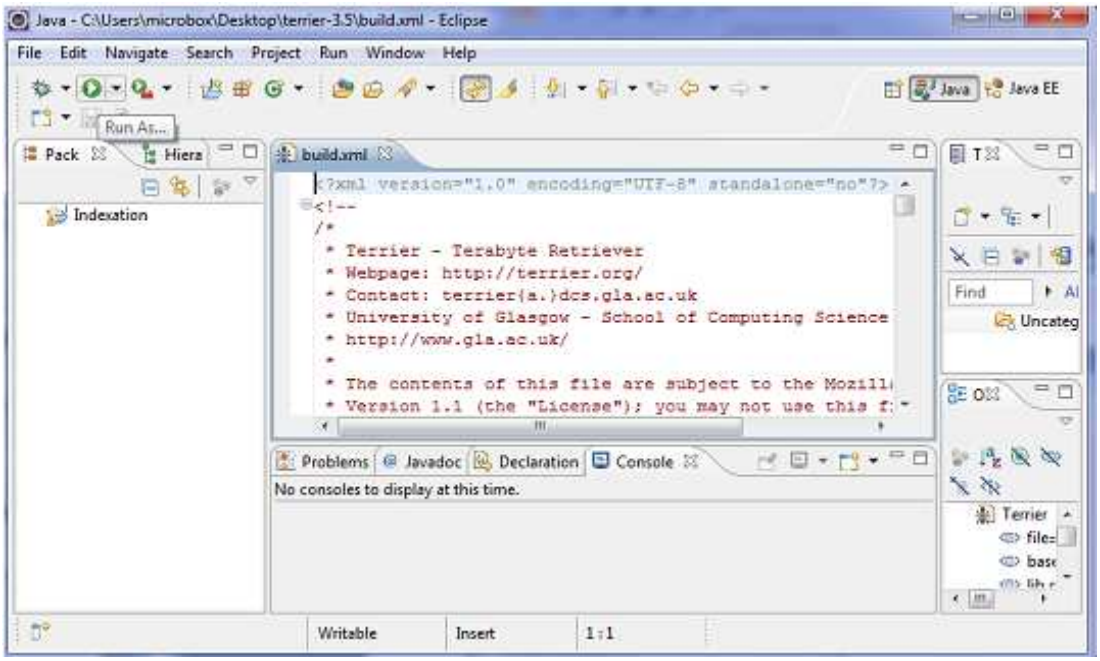
Configuration Ant : 2- Ant ->Runtime->Global Entries



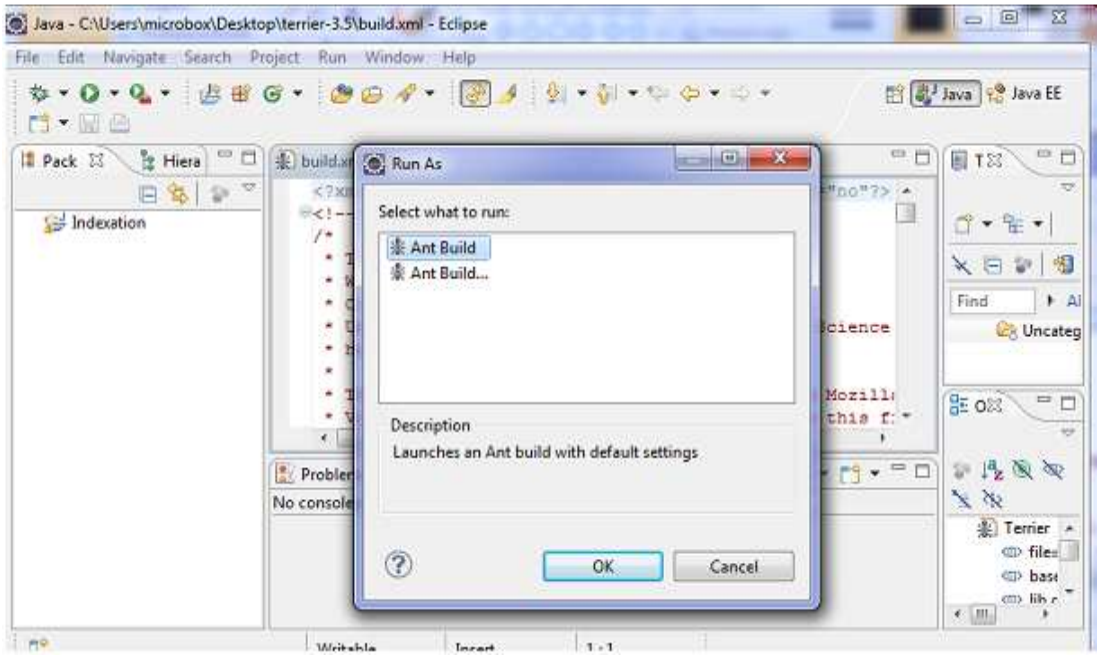
Configuration Ant : 3- Ajouter avec add Externals JARS le fichier ../Java/ jdk1.6.0_25/ lib/tools.jar



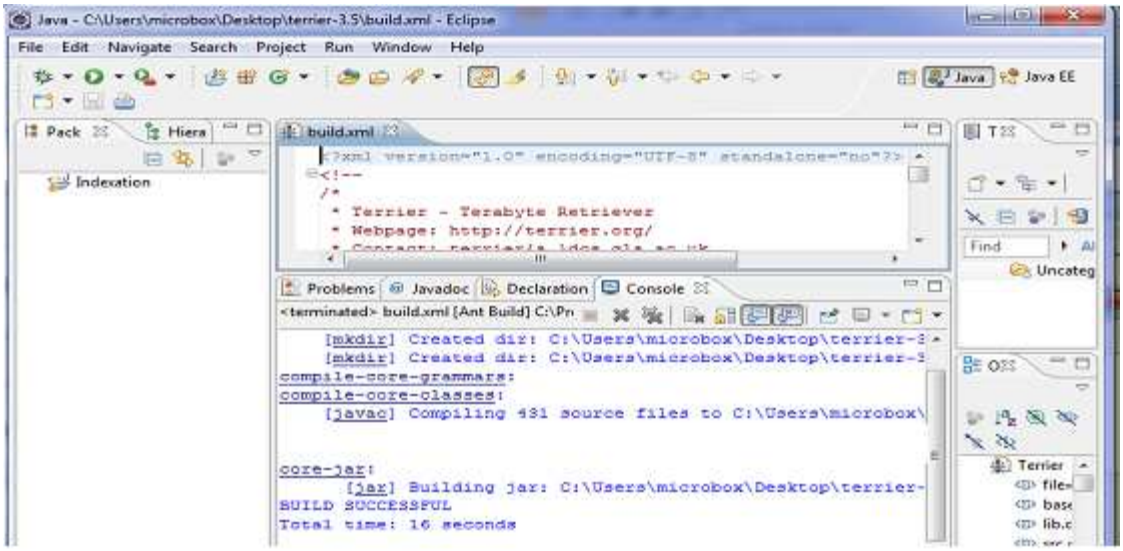
Ouvrir le fichier build.xml de terrier



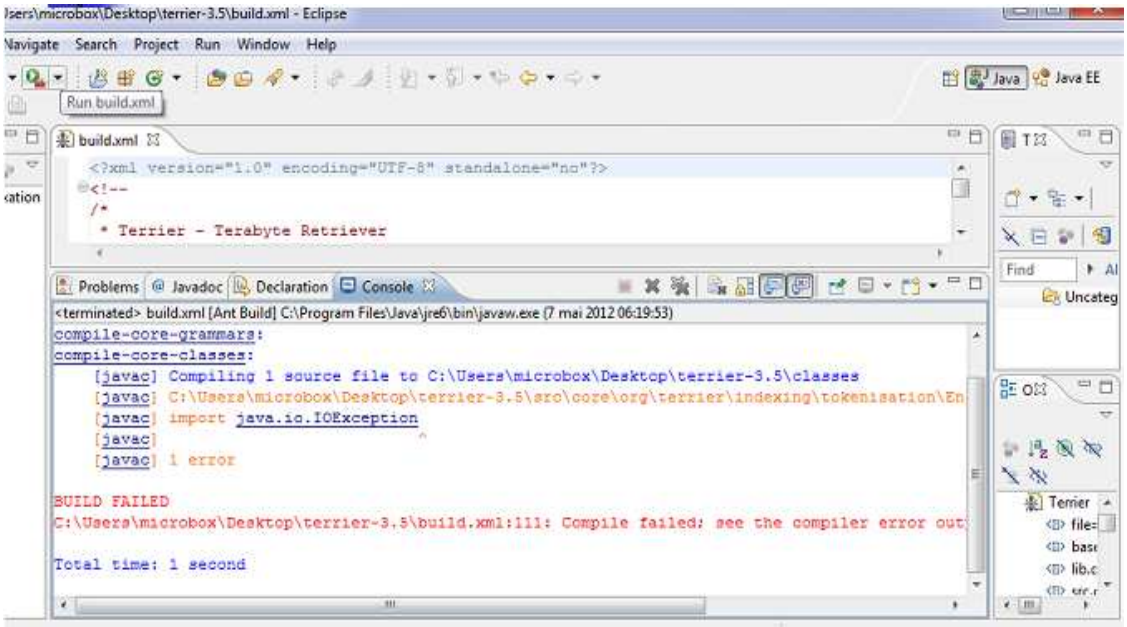
Exécuter le script : Bouton Run



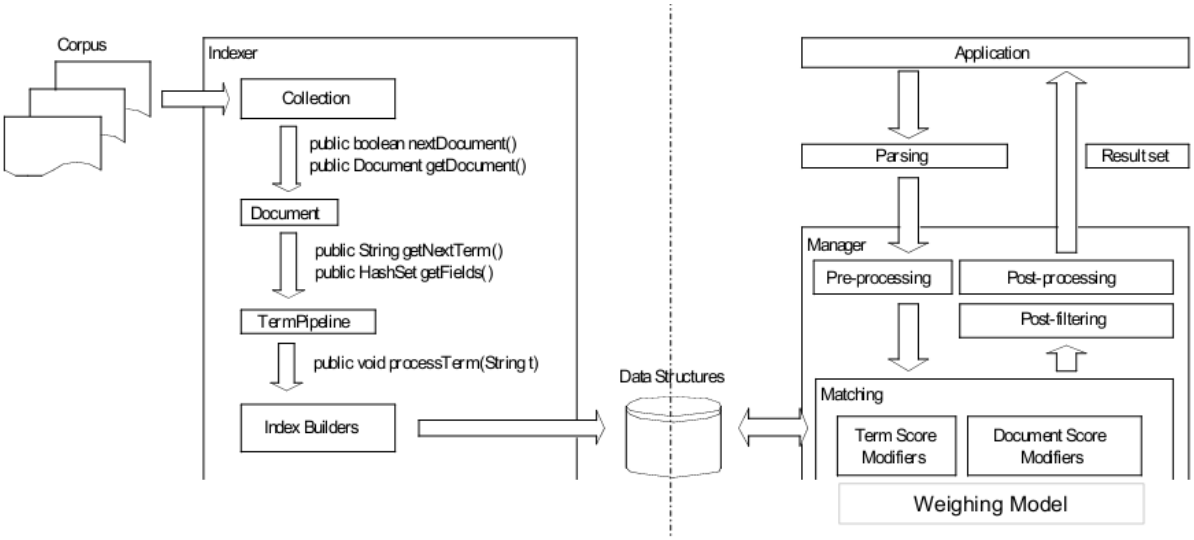
Exécuter Ant Build

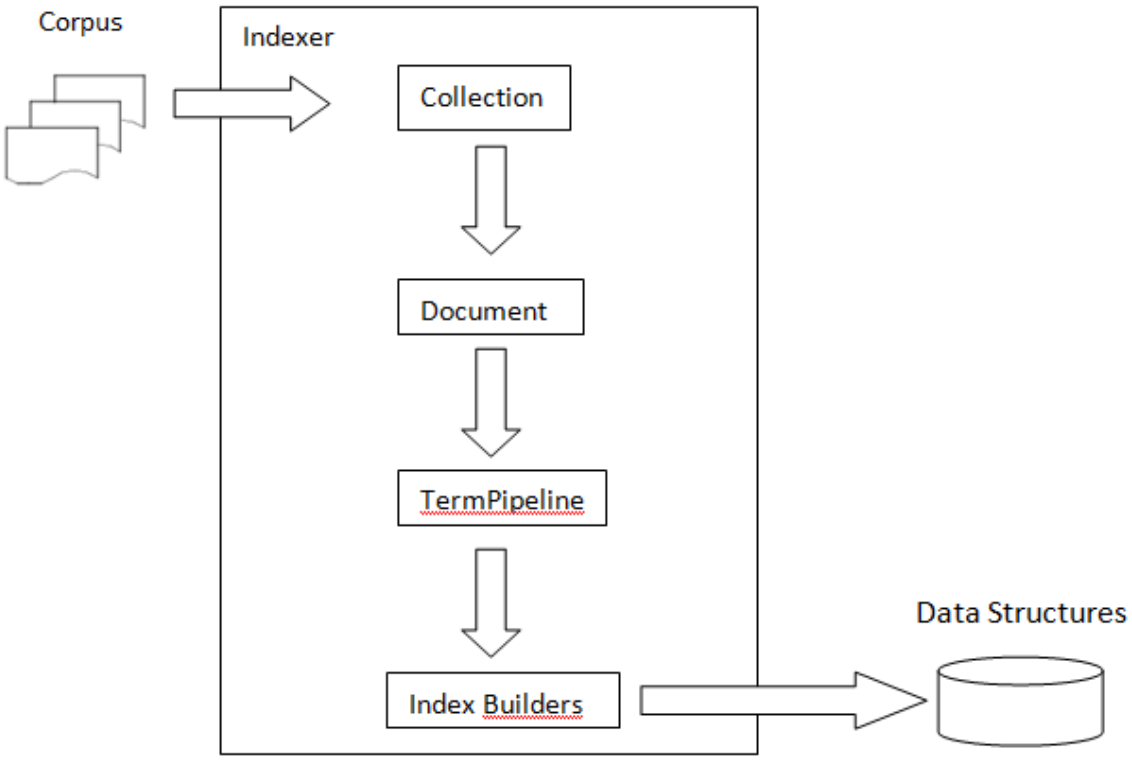


BUILD SUCCESSFUL : Compilation réussie



BUILD FAILED : Compilation échouée


Corpus
Indexer
Collection
public boolean nextDocument()
public Document getDocument()
Document
public String getNextTerm()
public HashSet getFields()
TermPipeline
public void processTerm(String t)
Index Builders
Data Structures
Application
Parsing
Result set
Manager
Pre-processing
Post-processing
Post-filtering
Matching
Term Score Modifiers
Document Score Modifiers
Weighing Model


Corpus
Indexer
Collection
Document
TermPipeline
Index Builders
Data Structures

Annexe 3

Stanford POS Tagger

Stanford Part-of-Speech Tagger est un étiqueteur morphosyntaxique et un lemmatiseur développé par *The Stanford Natural Language Processing Group,* il est distribué librement à des fins d'évaluation, de recherche ou d'enseignement. Pour l'étiquetage, il implémente une méthode probabiliste (arbres de décision) nécessitant une phase d'entraînement ; il est donc possible de développer une version spécifique selon la langue pour laquelle on souhaite l'utiliser. Une version dédiée à l'Anglais est ainsi disponible sur la page d'accueil du *Stanford POS Tagger* ; seul le fichier de paramètres varie : le moteur probabiliste reste inchangé. La liste des catégories grammaticales utilisées par *Stanford POS Tagger* pour l'Anglais est présentée dans le **tableau 3.1** ci-dessous. Cette liste correspond aux jeux d'étiquetage *Treebank Tag* set.

Catégories	*Signification*
CC	Conjonction
CD	Nombre
DT	Determinant (*the, a, all, and, both, etc*…)
EX	*There*
FW	Mot ou expression étrangère
IN	Preposition (*across, after, as, for, in, etc…*)
JJ	Adjectif
LS	Référence
MD	Auxiliaires (*can, should, may, would, will, might*)
NN	Nom
NNP	Nom propre
NNPG*	Nom de groupe (société, association, etc…)
NNPL*	Nom de lieu
NNP'	Nom de personne
NNS	Pluriels
PDT	All, both, that, this
POS	*'s* possessif
PRP	Pronom personnel
PRP$	Pronom possessif
RB	Adverbe
RBR	Adverbe de comparaison (*better*, etc…)

SPUNC′	Ponctuation forte
TO	Infinitif to
UH	Interjection
UNK	Mot inconnu
VB	Verbe
VBD	Verbe au passé
VBG	Participe présent
VBN	Participe passé
VBP	Auxiliaires (*be, do, have, is, does, has*)
WDT	*That, whatever, which*
WP	*Whadya, what, who, whom ,adjectifs interrogatifs relatifs*
WP$	Whose
WPUNC′	Ponctuation faible
WRB	*How, when, whence, whenever, where, whereby, wherever, why*
ZTRM′	Fin de phrase

Liste des catégories grammaticales.