REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire

de fin d'études

En vue de l'obtention du diplôme de Master 2 en informatique

Options : Conduite de Projet Informatique

Thème :

MODÈLE DE LANGUE MIXTE COMBINANT ENTITÉS NOMMÉES ET MOTS SIMPLES

Proposé et dirigé par :

Mr HAMMACHE Arzeki

Réalisé par :

Melle BEDRANE Ferroudja

Melle HOUALI Nawel

Melle REZZOUG Faiza

2012/2013

Remerciements

« On parvient rarement à ses fin par ses propres moyens, il faut toujours compter sur quelqu'un d'autre »

D'après Marie-Claude Bussières- Tremblay

Nous remercions DIEU pour le courage et la patience qu'il nous a donné afin de mener ce projet a terme.

Nous tenons à exprimer nos remerciements avec un grand plaisir et un grand respect à notre promoteur M HAMMACHE Arezki.

Sa disponibilité et Ses précieux conseils nous ont permis de réaliser ce travail dans les meilleures conditions.

Nous sommes Très reconnaissants envers les personnes qui nous ont aidés sans pour autant nous connaitre.

Nous remercions les membres du jury qui nous ont fait l'honneur de bien vouloir juger notre travail.

Nos sincères sentiments vont à tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce projet. En particulier nos chères familles et nos amis(es).

FAIZA, NAWEL & FERROUDJA

DEDICACES

Je dédie ce modeste travail à :

Ma chère tante OURDIA et cher oncle SAID

Mes chers parents

Mon mari et ma belle-famille

Mes grands-mères

Mes sœurs et leurs maris, mes nièces et neveux

Mes oncles maternels, paternels et leurs familles

Mes cousins, cousines et tous mes proches

Toutes mes amies : HAMIDA, FARIDA, FATIMA, SOUAD…

Ainsi qu'à mes binômes FAIZA et NAWEL

A la mémoire de mes chers grand- pères ALI et RAMDANE,

neveu ADEL, oncle BELKACEM et mon cousin BELAID.

Ferroudja

Dédicaces

Je dédie ce travail :

A la personne la plus chère à mes yeux, à ma mère qui a tout sacrifié pour ses enfants, qui a veillé à notre éducation, qui, sans elle je ne serai pas ce que je suis ;

A mon père que j'adore, en signe de reconnaissance pour tous les sacrifices, lui qui m'a supporté et m'a donné la force d'être aujourd'hui ce que je suis ;

A toute ma famille

A tous les étudiants de ma promotion spécialement à : Zina, Dahmane, Aliane et Anissa.

A mes binômes Nawel et Ferroudja

A tous mes ami (e) s

A toutes les personnes que je connaisse et que je n'ai pas citées.

A tous ceux que j'aime

F A IZ A

Dédicaces

Je dédie ce travail à mon cher et regretté père

A ma douce mère en demandant à Dieu de la préserver pour nous tous.

A mes chers sœurs et frères, ayant toujours été à mes côtés à m'encourager et à me soutenir ;

A mon cher mari Amar

A mes beaux parents ;

A mes beaux -frères et belles-sœurs ;

A mes neveux et à ma nièce ;

A toute la famille Zemirli et les collègues du Moulin de Belloua que je considère comme ma deuxième famille

A tous mes amis et à tous ceux qui m'ont inscrite dans leur entourage ;

A mes chères binômes Faiza et Farroudja.

Nawel HOUALI

Sommaire

Sommaire

Sommaire

Chapitre II : Les Entités Nommées

Chapitre III : Approche et implémentation

Sommaire

Sommaire

Liste des figures

Liste des tableaux

Introduction générale

L'intérêt stratégique porté à l'information sous ses différentes facettes, combinée à l'avènement explosif d'Internet et autres services de l'information sont des facteurs déterminants qui justifient la multiplication de directions de recherche ayant pour objectif de mettre en œuvre des processus automatiques d'accès à l'information, sans cesse plus performants.

De nos jours, la quantité d'information disponible dans le monde explose. Des études statistiques ont estimé qu'elle double tous les 20 mois. Le volume de l'information, la taille des collections, ainsi que le nombre d'utilisateurs sont toujours en croissance vertigineuse.

Ainsi, Dans ce contexte, les systèmes de recherche d'information sont devenus les principaux moyens d'accès à l'information. Pourtant, nombre d'entre eux présentent deux faiblesses : d'une part, le nombre de documents retournés en réponse à un utilisateur est trop souvent important; d'autre part, c'est toujours à l'utilisateur de localiser l'information dont il a besoin dans les documents renvoyés. Notons que plus la collection devient gigantesque et plus le problème devient aigu et la performance des systèmes tend à diminuer.

L'objectif fondamental de la recherche d'information consiste à mettre en œuvre un mécanisme d'appariement entre requête utilisateur et documents d'une base afin de restituer l'information pertinente.

L'élaboration d'un processus de recherche d'information pose alors des problèmes liés tant à la modélisation qu'à la localisation de l'information pertinente. En effet, la recherche d'information induit un processus d'inférence de la sémantique véhiculée par l'objet de la requête, en se basant sur une description structurelle des unités d'informations.

Dans ce cadre, les travaux ciblent la compréhension fidèle et exhaustive de la requête utilisateur. Des modèles de représentation sémantique et mécanismes d'indexation ont été à cet effet, proposés. Cependant, la modélisation des unités textuelles représentatives du contenu sémantique des requêtes, ne saurait être concluante sans la mise en œuvre de modèles représentatifs des documents d'une part, et mécanismes d'appariement requête-document d'autre part.

Les modèles de recherche et représentation d'information sont basés sur un processus de mise en correspondance entre requêtes utilisateurs et documents de la collection. Le mécanisme de recherche détermine alors, sur la base d'un degré de pertinence supposé des documents, ceux qui répondent au besoin de l'utilisateur. De nombreux modèles et stratégies sont développés dans la littérature.

Le modèle de langue est un nouveau cadre probabiliste pour la description du processus de la RI. Parmi les propriétés de ce modèle est son fondement mathématique solide. Cependant, comme la majorité des modèles de RI classique, le modèle de langue se base sur l'hypothèse d'indépendance entre terme. Une telle hypothèse pose le problème d'ambigüité des termes.

Dans le but d'accroître les performances des modèles de recherche de base, de nombreuses stratégies sont mises en œuvre afin d'y être greffées. Ces stratégies exploitent diverses sources d'évidence : relations sémantiques définies dans le thesaurus, classes et contextes d'utilisation des concepts, résultats de recherche, jugement de pertinence des utilisateurs, éléments de la théorie de l'information, heuristiques etc...

Pour notre part, nous nous intéressons à la mise en œuvre d'une stratégie d'optimisation de la recherche basée sur la reconnaissance d'entités nommées qui est une sous-tâche de l'activité d'extraction d'information dans des corpus documentaires. Elle consiste à rechercher des objets textuels (c'est-à-dire un mot, ou un groupe de mots) catégorisables dans des classes telles que noms de personnes, noms d'organisations ou d'entreprises, noms de lieux, quantités, distances, valeurs, dates, etc., utilisées comme un outil puissant d'optimisation.

L'organisation retenue pour la présentation de notre travail et le domaine dans lequel il s'inscrit, s'articule autour de trois chapitres.

Le premier chapitre traite de la recherche d'information et de son processus ainsi que les différents modèles et stratégies de recherche et de représentation d'information proposés dans la littérature.

Le second chapitre présente les entités nommées, le processus de reconnaissance et d'extraction ainsi qu'aux outils dédiés à leurs manipulation notamment l'outil Gate.

Le troisième chapitre présente globalement notre approche. Nous décrivons le fonctionnement général du SRI proposé. La deuxième partie présente la mise en œuvre de stratégies de recherche d'information à travers la description de notre approche d'optimisation de requête. On y présente également les résultats d'évaluations réalisées dans le but de valider notre approche d'optimisation de requête. Cette évaluation a pour but de mesurer l'efficacité d'introduire les entités nommées dans le processus de recherche.

En conclusion, nous dressons un bilan de notre étude. Nous présentons ensuite les perspectives d'évolution de ces travaux.

Une annexe est enfin présentée pour décrire la plateforme Terrier, utilisé pour la réalisation de notre approche.

Recherche d'information

I.1 Introduction

La recherche d'informations (RI) est une branche de l'informatique qui s'intéresse à la collecte, à l'organisation et à la sélection d'informations répondant aux besoins des utilisateurs. La RI est une discipline relativement ancienne qui s'intéresse à répondre pratiquement à la problématique suivante : comment retrouver un document parmi une collection de documents qui satisfait un besoin utilisateur ?

La RI est mise en œuvre à travers un système de recherche d'informations (SRI) qui offre des techniques et des outils permettant de localiser et de visualiser l'information pertinente, répondant à un besoin en information exprimé par un utilisateur sous forme de requête.

L'indexation est une phase très importante pour un SRI car de sa qualité dépend la qualité des réponses du système et donc les performances de ce dernier. Afin de palier l'ambiguïté de la langue dans l'indexation classique due à une indexation par entités lexicales du vocabulaire, de nouvelles approches ont été introduites celles-ci s'intéressent principalement à la représentation des documents et aux requêtes par des concepts plutôt que par les mots eux-mêmes.

Dans ce chapitre nous présentons les concepts de base de la RI classique.

I.2 Système de Recherche d'Information

Un SRI permet de retrouver les documents pertinents à une requête utilisateur, à partir d'une base de documents volumineuse [Nie, 04].

Dans cette définition, on distingue trois notions clés : **documents, requête, pertinence**.

- **Document :** Le document représente le conteneur élémentaire d'informations, exploitable et accessible par le SRI. Un document peut être un texte, une page WEB, une image, une bande vidéo, etc. Nous appelons document toute unité qui peut constituer une réponse à un besoin en information exprimé par un utilisateur.

- **Requête :** Une requête constitue l'expression du besoin en information de l'utilisateur.

- **Pertinence :** Le but de la RI est de trouver seulement les documents pertinents. La notion de pertinence est très complexe. De façon générale, dans un document pertinent, l'utilisateur doit pouvoir trouver les informations dont il a besoin. C'est sur cette notion de pertinence que le système doit juger si un document doit être donné à l'utilisateur comme réponse. Cette notion de pertinence peut être appréhendée à deux niveaux :

- ***Le niveau utilisateur* :** à ce niveau, l'utilisateur a un besoin d'informations dans sa tête, et il espère obtenir les documents pertinents pour répondre à ce besoin. La relation entre le besoin d'informations et les documents attendus est la relation de pertinence (idéale ou absolue).

- ***Le niveau système*** : à ce niveau, le système répond à la requête formulée par l'utilisateur, par un ensemble de documents trouvés dans la base de documents qu'il possède. Remarquez que la requête formulée par l'utilisateur n'est qu'une description partielle de son besoin d'informations. Beaucoup d'études ont montré qu'il est très difficile, voire impossible, de formuler une requête qui décrit complètement et précisément un besoin d'informations. Du côté de document, il y a aussi un changement entre les deux niveaux : les documents que l'on peut retrouver sont seulement les documents inclus dans la collection de documents. On ne peut souvent pas trouver des documents parfaitement pertinents à un besoin. Il arrive souvent qu'aucun document pertinent n'existe dans la collection.

I.3 Le processus général de RI

Les différentes étapes du processus de RI, sont représentées schématiquement par le processus général (voir Fig. I.1) [Baaziz ,05]. La figure illustre particulièrement :

- les notions de documents et de requêtes qui sont des conteneurs d'informations,
- les opérations d'indexation et d'appariement qui permettent globalement de traiter la requête dans le but de sélectionner des documents à présenter à l'utilisateur.

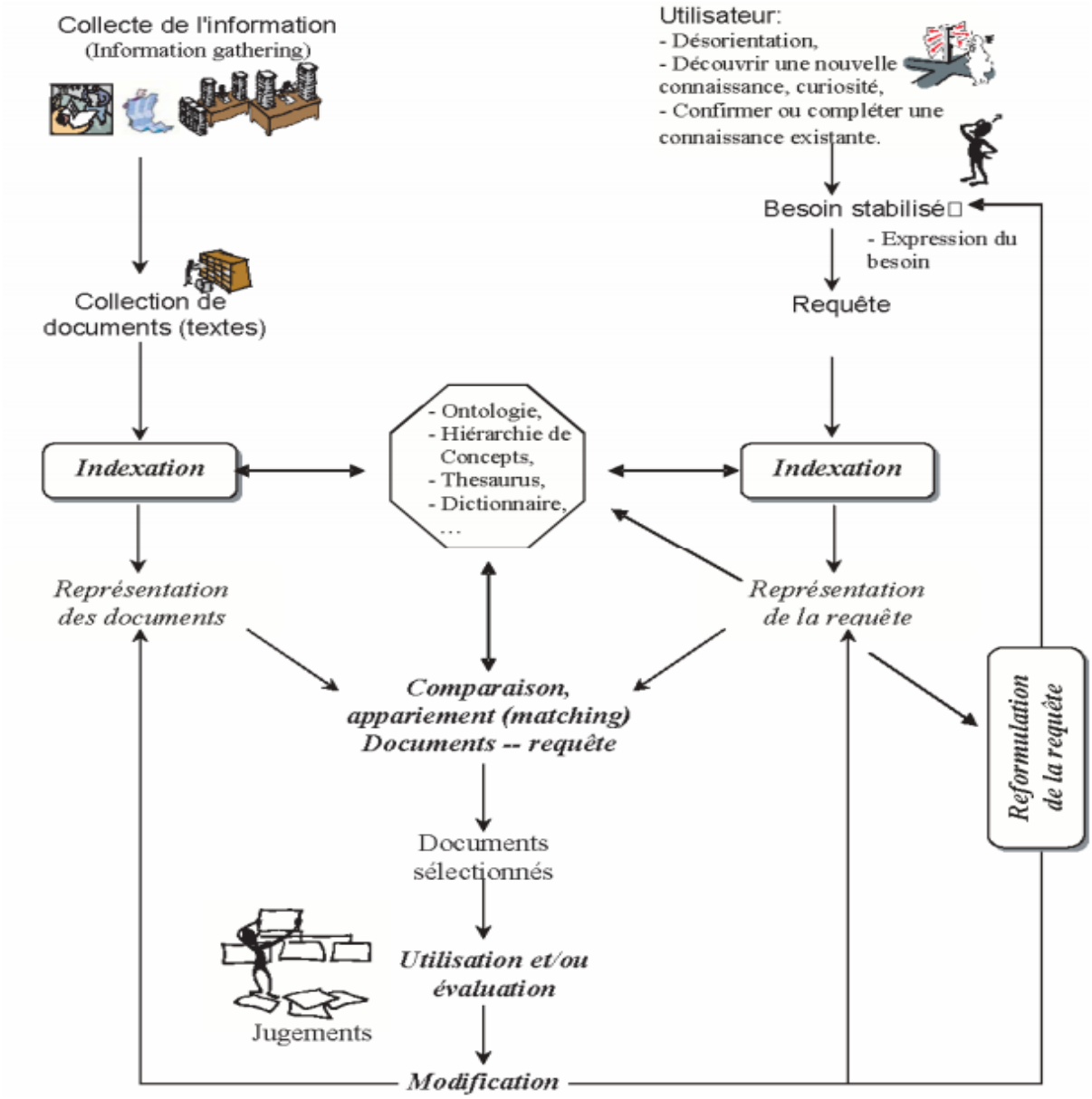


Fig. I .1 . *Processus général de recherche d'informations*

L'objectif fondamental d'un processus de RI est de sélectionner les documents ‘‘les plus proches’’ du besoin en informations de l'utilisateur décrit par une requête. Ceci induit deux principales phases dans le déroulement du processus : indexation et appariement requête/documents. Une troisième phase est également introduite, elle se nomme expansion de requête.

I.3.1. Indexation

Afin d'assurer la recherche dans des conditions acceptables de coût et d'efficacité, une étape primordiale doit s'effectuer avant l'étape de recherche effective de l'information. Cette étape consiste à analyser le document lors de l'organisation du fond documentaire afin de

produire un ensemble de mots clés, appelés aussi descripteurs, que le système pourra gérer aisément puis utiliser dans le processus de recherche ultérieur. Cette opération est appelée indexation (Salton,71 ; SparkJones,79 ; Rijsbergen,79 ; Deerwester et al.,90 ; Soule Dupuy, 90). Cet ensemble de mots clés peut être regroupé dans un thésaurus (Crouch et al., 89 ; Crouch et al.,92 ; Frakes et al., 92), mais en pratique, un thesaurus représente une notion plus large qu'une liste de mots clés. Il regroupe plusieurs relations de types linguistique (équivalence, association, hiérarchie) et statistique (pondération).

Le résultat de l'indexation est un ensemble de termes définissant ce qui est appelé le langage d'indexation. On peut distinguer deux types de langage d'indexation [Baaziz,05] :

- **Langage contrôlé** : il s'agit d'un lexique figé de descripteurs. L'indexation est alors le plus souvent manuelle, parfois semi-automatique ; un professionnel choisit un ou plusieurs descripteurs pour représenter le document.

- **Langage libre** : Les descripteurs sont extraits automatiquement des documents, ou de la requête de l'utilisateur. Ici, un document est le plus souvent indexé par la liste des mots qui le composent.

Aussi l'indexation peut être manuelle, automatique ou semi-automatique :

- **Manuel :** L'indexation manuelle, est réalisée par un expert qui après son analyse des documents, et en utilisant son savoir faire, propose des termes représentants le contenu des documents considérés. L'indexation manuelle est fondée sur le jugement d'un être humain, et est cependant trop dépendante de l'état de connaissances des indexeurs, ce qui induit une subjectivité des résultats, de ce fait, elle n'est pas toujours efficace. Elle est non seulement très couteuse en temps, mais inapplicable sur des collections de taille importante. Elle est cependant utilisée notamment dans le monde de la documentation, comme dans les bibliothèques et sert en général à la classification des ouvrages par thèmes. Dans ce cadre, un ensemble prédéfini de mots clés et de catégories est défini précisément pour faciliter l'indexation et la formulation des requêtes.

- **Semi-automatique :** Le choix final reste au spécialiste du domaine correspondant ou documentaliste, qui intervient souvent pour établir des relations sémantiques entre mots-clés et choisir les termes significatifs.

- **Automatique :** A l'aide d'un processus entièrement informatisé, elle a la même finalité que l'indexation manuelle et elle est réalisée avec une machine (ordinateur).

Nous décrivons en détail dans ce qui suit l'indexation automatique qui regroupe un ensemble de traitements automatisés sur un document :

I.3.1.1. L'analyse lexicale

L'analyse lexicale est le processus qui permet de convertir le texte d'un document en un ensemble de termes. Un terme est une unité lexicale ou un radical (Fox,92). L'analyse

lexicale permet de reconnaitre les espaces de séparation des mots, des chiffres, des ponctuations, etc [Sauvagnat,05].

I.3.1.2 Élimination des mots vides

Extraire les termes significatifs et éviter les mots vides est l'un des problèmes majeurs de l'indexation. Pour l'élimination de ces mots vides on utilise une liste appelée stopliste (ou anti-dictionnaire) qui contient tous les mots non significatifs comme les propositions (ex: ''de'', ''à'', …) et les pronoms (''aucun'', ''tout'', ''ou'', …) ….etc.

Le traitement lié à une stop-liste est très simple : quand on rencontre un mot dans un texte, on doit d'abord examiner s'il apparaît dans cette liste. Si oui, on ne le considère pas comme un index.

I.3.1.3 Lemmatisation

La lemmatisation désigne l'analyse lexicale du contenu d'une page (ou d'un document) regroupant les mots d'une même famille, chacun des mots d'un contenu se trouve ainsi réduit en une entité appelée lemme.

La lemmatisation regroupe les différentes formes que peut revêtir un mot, soit le nom, le pluriel, le verbe à l'infinitif, etc. Par exemple on peut citer médecine, médical, médicalement, médecin, etc. et il n'est pas forcément nécessaire d'indexer tous ces mots alors qu'un seul suffirait à représenter le concept. Frakes et Baeza-yates (Frakes,92) distinguent cinq types stratégiques de lemmatisation, la table de consultation (dictionnaire), l'élimination des affixes (on peut par exemple citer l'algorithme de Porter (Porter, 80)), la troncature, les variétés de successeurs ou encore la méthode des n-gramme (Adamson and J. Boreham, 74). [Sauvagnat,05]

I.3.1.4 Extraction de descripteurs

L'extraction et le choix des descripteurs s'effectuent d'une façon totalement automatisée : L'intérêt de cette approche réside dans sa capacité à traiter les textes nettement, plus rapidement, et de ce fait elle est particulièrement adaptée aux corpus volumineux, parmi les descripteurs on distingue les catégories suivantes :

- **Mots clés :** Ce sont des mots simples, des expressions composées d'un ou plusieurs mots (uni-termes ou multi-termes) qui décrivent au mieux le contenu d'un texte. L'ensemble des descripteurs de documents forme le langage d'indexation de la base documentaire.
- **Termes composés :** Il est important de reconnaître les mots composés car ce sont des unités de sens. C'est d'autant plus important lorsque ce sens ne se déduit pas des mots composants. Par exemple : « arbre à cames » ou « pomme de terre ».
- **Entités nommées :** une entité nommée est une appellation générique utilisée pour la catégorisation d'un certain nombre d'objets textuels rencontrés dans un document. Cette appellation recouvre par exemple les noms de personnes, des lieux, ou des organismes (raisons sociales). Par extension, on y ajoute l'adresse mails, les numéros de téléphone, les codes postaux et toute information de ce genre que l'on peut y relier. (c'est la catégorie qui nous intéresse dans notre cas)

- **Expressions :** regroupe l'une des catégories citées ci-dessus ou bien une combinaison de ces catégories.

I.3.1.5 Pondération

La pondération des termes permet de mesurer l'importance d'un terme dans un document. Cette importance est souvent calculée à partir de considérations et d'interprétations statistiques (ou parfois linguistiques). L'objectif est de trouver les termes qui représentent le mieux le contenu d'un document. Si on dresse une liste de l'ensemble des mots différents d'un texte quelconque classés par ordre de fréquences décroissantes, on constate que la fréquence d'un mot est inversement proportionnelle à son rang de classement dans la liste.

Cette constatation est énoncée formellement par la loi de Zipf (Zipf, 49) [Sauvagnat,05] :

Rang * fréquence = constante

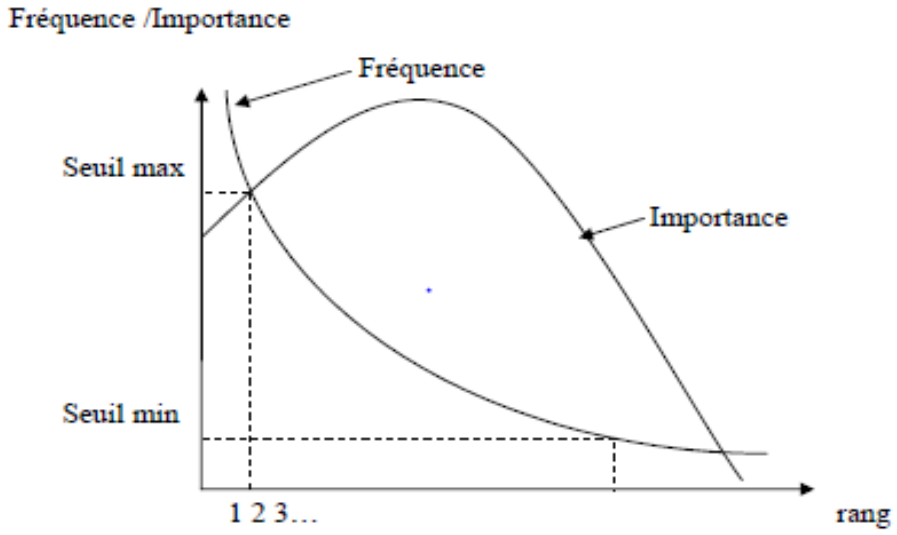


Fig. I.2 : *Importance d'un terme en fonction de sa fréquence d'apparition dans un document*

A. La Loi de Zipf : Zipf explique la courbe hyperbolique de la distribution des termes par ce qu'il appelle : Le principe du moindre effort ; il considère qu'il est plus facile pour un auteur d'un document, de répéter certains termes que d'en utiliser des nouveaux. La relation entre la fréquence et le rang des termes permet de sélectionner les termes représentatifs d'un document.

La sélection de termes discriminants par la loi de Zipf consiste alors à éliminer respectivement les termes de fréquences très élevées puisqu'ils ne permettent pas la distinction entre les documents de la collection et puis les termes de fréquences très faibles (exemple, qui sont rarement utilisés dans une requête). En utilisant cette approche, la taille du langage d'indexation d'une collection peut être réduite considérablement.

B. Conjecture de Luhn : La loi de Zipf est à la base de la conjecture de Luhn (Luhn, 58). La conjecture de Luhn émet une hypothèse sur l'information contenue dans les termes

(informativité) d'un document. L'informativité mesure la quantité de sens et le pouvoir d'expression des mots dans un texte en fonction de leur fréquence. Elle considère que les descripteurs non pertinents sont les descripteurs de rangs faibles (très fréquents), car ce sont des mots qui reviennent souvent, ils n'ont pas de pouvoir discriminant. Par exemple, dans un corpus de documents informatiques, le mot ordinateur est très fréquent et ne va pas permettre de différencier les documents entre eux. La conjecture de Luhn considère aussi ceux de rangs élevés (très rares), comme peu pertinents, en effet ces mots sont rares et donc peu utilisés. Les descripteurs qui sont pertinents sont les descripteurs de rangs intermédiaires.

Cette conjecture est utilisée pour diminuer la taille des index des documents. Deux seuils de fréquence (seuil max et seuil min), sont fixés pour éliminer les termes dont le contenu informatif est jugé faible. Seuls les termes entre ces deux seuils sont alors pertinents pour représenter les documents.

La correspondance entre l'informativité c.à.d. l'importance d'un terme ou le pouvoir expressif et la fréquence d'apparition dans un document est illustrée dans la figure I.2.

Ainsi, en choisissant les mots qui ont des fréquences entre les deux seuils, on espère obtenir les mots dont l'informativité est la plus élevée.

En conclusion, la loi de Zipf et la conjecture de Luhn soulignent qu' :

- un terme d'indexation qui apparait trop fréquemment dans un texte est un mot vide qui ne joue aucun rôle sémantique dans ce texte, son rôle est seulement syntaxique, donc, il ne doit pas être utilisé dans le langage d'indexation.
- un terme d'indexation présent dans l'ensemble des documents de la collection ne peut pas discriminer entre ces documents et donc, n'apporte aucun pouvoir discriminant à la recherche.
- un terme d'indexation de fréquence intermédiaire est considéré comme significatif, il représente le contenu sémantique de l'unité documentaire et appartient au langage d'indexation.

C. Pondération en TF*IDF : L'idée derrière la pondération des termes d'indexation est d'affecter aux termes d'un document, un poids pour traduire son importance dans le document, donc son degré d'informativité. Dans un SRI, la majorité des méthodes de pondération sont construites par la combinaison de deux facteurs : un facteur de pondération local et un second facteur de pondération globale.

- **Pondération Locale :** La pondération locale permet de mesurer la représentativité locale d'un terme. Elle prend en compte les informations locales du terme par rapport à un document donné. Elle indique l'importance du terme dans ce document. Les fonctions de pondération locales les plus utilisées sont les suivantes :

❖ ***Fonction brute de tf_{ij} (term frequency) :*** correspond au nombre d'occurrence du terme dans le document est supérieure ou égale à 1, 0 sinon ;

❖ ***Fonction logarithmique*** : combine tf_{ij} avec un logarithme, elle est donnée par :

$$\alpha + \log (tf_{ij}) \qquad \textbf{(I.1)}$$

où **: α** est une constante

- ***Fonction normalisée*** : permet de réduire les différences entre les valeurs associées aux termes du document. Elle est donnée par la formule suivante :

$$\mathbf{0.5 + 0.5} \ * \frac{tf}{\mathbf{Max}\ t_i \in D_i\ (tf_{ii})} \qquad \textbf{(I.2)}$$

Où : max $t_i \in$ Dj (tf_{ij}) est la plus grande valeur de tf_{ij} des termes du document D_j.

- **Pondération Globale :** La pondération globale prend en compte des informations concernant un terme par rapport à la collection de documents. Elle indique la représentativité globale du terme dans l'ensemble des documents de la collection. Un poids plus important doit être donné aux termes qui apparaissent moins fréquemment dans la collection : les termes qui sont utilisés dans de nombreux documents sont mois utiles pour la discrimination que ceux qui apparaissent dans peu de documents. Par conséquent, un facteur de pondération globale qui dépend de la fréquence inverse dans le document a été introduit. Il est souvent désigné par *IDF* (pour inverse of document frequency) et peut être selon l'une des déclinaisons suivantes :

$$Idf = \log\frac{N}{ni} \qquad \text{Ou} \qquad Idf = \log\frac{N-ni}{N} \qquad \textbf{(I.3)}$$

Où : n_i est le nombre de documents contenants le terme t_i et N le nombre total de documents dans la collection sont souvent référencées sou le nom de *TF*IDF*.
Le poids d'un descripteur t_i pour un document D_j dans un corpus est donné par la formule :

$$W_{ij} = tf_{ij} * idf_{ij} \qquad \textbf{(I.4)}$$

Cela signifie qu'un terme avec une forte pondération est présent fortement dans un document mais pas dans les autres.

La mesure *TF*IDF* est une bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles homogènes. Cette mesure a eu en revanche un succès très limité dans les corpus de tailles variables.

1.3.1.6 Création de L'index

Afin de répondre plus rapidement à une requête, des structures de stockage particulières sont nécessaires pour mémoriser les informations sélectionnées lors du processus d'indexation. Les moyens de stockage les plus répandus sont les suivants : les fichiers inverses (”inverted files”), les tableaux de suffixes (”suffix arrays ”) et les fichiers de signatures (”signature files”). Les fichiers inverses sont actuellement le meilleur choix possible pour la plupart des applications et sont utilisés dans la plupart des systèmes commerciaux.

Dans cette structure, chaque document peut être représenté par une liste de mots clés qui décrivent le contenu du document pour la recherche. Une recherche rapide, peut être réalisée si nous inversons ces mots clés. Il faut noter que les documents ne sont pas les seuls à être indexés : les requêtes sont également perçues comme des listes de mots-clés.

I.3.2 Appariement Documents / Requêtes

Les SRI intègrent un processus de recherche/décision qui permet de sélectionner l'information jugée pertinente pour l'utilisateur. A cet effet, une mesure de similitude (correspondance) entre la requête indexée et les descripteurs des documents de la collection est calculée. Seuls les documents dont la similitude dépasse un seuil prédéfini sont sélectionnés par le SRI.

La fonction de correspondance est un élément clé d'un SRI car la qualité des résultats dépend de l'aptitude du système à calculer une pertinence des documents la plus proche possible du jugement de pertinence de l'utilisateur (Tmar, 02) [Zemirli,04].

Il existe deux types d'appariement :

- **Appariement exact**

  Le résultat est une liste de documents respectant exactement la requête spécifiée avec des critères précis. Les documents retournés ne sont pas triés

- **Appariement approché**

  Le résultat est une liste de documents sensés être pertinents pour la requête. Les documents retournés sont triés selon un ordre de mesure. Cet ordre reflète le degré de pertinence document/requête.

I.3.3 La Reformulation de Requêtes

Il est souvent difficile pour l'utilisateur de formuler son besoin exact en information. Par conséquent, les résultats que lui fournit le SRI ne lui conviennent pas parfois. Retrouver des informations pertinentes en utilisant la seule requête initiale de l'utilisateur est aujourd'hui très difficile, et ce à cause du volume croissant des bases documentaires. Afin de faire correspondre au mieux la pertinence utilisateur et la pertinence du système, une étape de reformulation de la requête est souvent utilisée [Boubekeur,08].

La reformulation de requêtes peut être réalisée avec ou sans intervention de l'utilisateur.

I.3.3.1 Reformulation Manuelle

Il s'agit de la stratégie de reformulation de la requête la plus populaire [Bouidghaghen, 11]. On la nomme communément réinjection de la pertinence ou relevance feedback. Dans un cycle de réinjection de pertinence, on présente à l'utilisateur une liste de documents sélectionnés par le système comme réponse à la requête initiale. Après les avoir examinés, l'utilisateur indique ceux qu'il considère pertinents. L'idée principale de la réinjection de pertinence est de sélectionner les termes importants appartenant aux documents jugés pertinents par l'utilisateur, et de renforcer l'importance de ces termes dans la nouvelle formulation de la requête.

Cette approche est associée aux systèmes de recherche booléens. On peut procéder à la Reformulation de requête en utilisant un vocabulaire contrôlé (thésaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes pour compléter sa requête [Aliane et al,04].

I.3.3.2 Reformulation Interactive

Dans une reformulation interactive l'utilisateur joue un rôle actif. A l'inverse de la reformulation automatique, ici, ce sont le système et l'utilisateur qui sont, ensemble, responsables de la détermination et du choix des termes candidats à la reformulation. Le système joue un grand rôle dans la suggestion des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes. L'utilisateur examine cette liste et décide du choix des termes à ajouter dans la requête. C'est donc l'utilisateur qui prend la décision ultime dans la sélection des termes [Aliane et al,04].

I.3.3.3 Reformulation Automatique

Également dite pseudo ré-injection de pertinence. Dans ce cas, l'utilisateur n'intervient pas, l'extension de la requête est effectuée en exploitant les premiers documents retournés par le SRI en réponse à la requête initiale (blind feedback) ou à partir d'une ressource externe qui peut être un thesaurus, une ontologie, etc [Bouidghaghen, 11].

La reformulation automatique de requêtes permet de générer une requête plus adéquate à la recherche d'information dans l'environnement du SRI, que celle initialement formulée par l'utilisateur. Son principe est de modifier la requête de l'utilisateur par ajout de termes significatifs et/ou par réestimation de leur poids.

Cette reformulation intervient dans un processus plus général d'optimisation de la fonction de pertinence. Celle-ci a pour but de rapprocher la pertinence système de La pertinence utilisateur. Elle se présente comme une opération primordiale dans un SRI.

I.4 Les Modèles de RI

Un modèle de recherche d'informations est formellement décrit par un quadruplet [**D, Q, F, R (q_i, d_j)**] (Yates, 99) où :

D : ensemble des représentants des documents de la collection,
Q : ensemble des représentants des besoins en informations,
F : schéma du support de représentation des documents, requêtes et relations associées.

R (q_i, d_j) : fonction d'ordre associée à la pertinence.

Nous présentons ici les modèles les plus couramment utilisés pour la RI, notamment le modèle booléen, le modèle vectoriel, le modèle probabiliste et le modèle de langue. (Grossman, 98) détaille les différents modèles de RI [Zemirli,04].

I.4.1 Le Modèle Booléen

Cette approche est très efficace pour des requêtes utilisant des termes très spécifiques ou portant sur des domaines techniques particuliers avec leur vocabulaire propre mais son intérêt reste néanmoins limité. [Ihadjadene,04]

La "Stratégie de Recherche Booléenne" est le plus ancien de tous les modèles de RI conventionnels. Les premiers systèmes de gestion de bibliothèques sont des exemples classiques ayant une longue histoire avec la recherche Booléenne. Le modèle booléen, propose la représentation d'une requête sous forme d'une expression logique. Les termes d'indexation sont reliés par les connecteurs logiques **ET(∧)**, **OU(∨)** et **NON(¬).** La correspondance RSV ($\mathbf{D_j}$, $\mathbf{Q_k}$), entre une requête $\mathbf{Q_k}$ et un document $\mathbf{D_j}$ est déterminée comme suit :

$$\mathbf{RSV(D_j,Q_i)=1 \ si \ Q_i \in D_j} \ ; \ \mathbf{0} \text{ sinon,} \qquad \textbf{(I.5)}$$

$$\mathbf{RSV(D_j,Q_i \wedge Q_j)=1} \ \text{ si } \mathbf{RSV(Dj,Q_i)=1 \ et \ RSV(D_j,Q_j)=1} \ ; \ \mathbf{0} \text{ sinon,} \qquad \textbf{(I.6)}$$

$$\mathbf{RSV(D_j,Q_i \vee Q_j)=1} \ \text{ si } \mathbf{RSV(D_j,Q_i)}=1 \text{ ou } \mathbf{RSV(D_j,Q_j)}=1 \ ; \ \mathbf{0} \text{ sinon,} \qquad \textbf{(I.7)}$$

$$\mathbf{RSV(D_j, \neg Q_i)=1} \ \text{ si } \mathbf{RSV(D_j,Q_i)=0} \ ; \ \mathbf{0} \text{ sinon,} \qquad \textbf{(I.8)}$$

Sachant que $\mathbf{Q_i}$ est un terme de la requête $\mathbf{Q_k}$,

Le modèle de recherche Booléen est reconnu pour sa force à faire une recherche très restrictive et obtenir, pour un utilisateur expérimenté, une information exacte et spécifique. [Baaziz,05]

- **Avantages :** Simple à mettre en œuvre.

- **Inconvénients :**

- L'inconvénient majeur du modèle booléen réside dans sa caractéristique de fournir une réponse binaire (les documents contiennent les termes demandés ou ne les contiennent pas). Ceci induit un volume de réponses important sans ordre spécifique des documents résultants. Cette approche est très stricte, ne classe les documents que dans catégories : pertinents ou non-pertinents c-à-d.
- Tous les termes des documents ont la même importance, alors qu'un terme apparaissant dans le titre qui représente la sémantique de texte doit être plus important que les autres termes.
- Les utilisateurs manipulent très mal les opérateurs logiques.

I.4.2 Le Modèle Vectoriel

Le modèle vectoriel introduit par Salton (Salton,71), repose sur les bases mathématiques des espaces vectoriels. Dans ce modèle, les documents et les requêtes sont représentés dans un espace vectoriel engendré par l'ensemble des termes d'indexation. .

Chaque document est représenté par un vecteur : $\mathbf{D_{j=}}_{(}\mathbf{d_{1j}, d_{2j},... d_{ij},..., d_{Tj})}$

Chaque requête est représentée par un vecteur : $\mathbf{Q}_{=(}\mathbf{q_1, q_2,... q_i,..., q_T)}$

Avec : $\mathbf{d_{ij}}$: Poids du terme $\mathbf{t_i}$ dans le document $\mathbf{D_j}$,

$\mathbf{q_i}$: Poids du terme $\mathbf{t_i}$ dans la requête **Q**.

T est le nombre total de termes issus de l'indexation de la collection des documents

Les termes de poids nul représentent les termes absents dans un document alors que les poids positifs représentent les termes assignés.

La fonction de calcul du coefficient de similarité entre chaque document D, représenté par le vecteur $(\mathbf{d_{1j}, d_{2j},... d_{ij},..., d_{Tj}})$, et la requête Q, représentée par le vecteur $_{(}\mathbf{q_1, q_2,... q_i,..., q_T)}$ est appelée Retrieval Status Value ou RSV.

Ce coefficient de similarité est calculé sur la base d'une fonction qui mesure la colinéarité des vecteurs documents et requête. On peut citer notamment les fonctions suivantes : [Zemirli,04].

Fonction	**Formule**
Produit scalaire	$\mathbf{RSV}\ (\mathbf{Q}.\boldsymbol{D_j}) = \sum_{i=1}^{T} \boldsymbol{q_i} * \boldsymbol{d_{ij}}$ **(I.09)**
Coefficient de Dice	$\mathbf{RSV}\ (\mathbf{Q}.\boldsymbol{D_j}) = \frac{2\sum_{i=1}^{T} q_{ik} * d_{ij}}{\sum_{i=1}^{T} (q_{ik}^2 + d_{ij}^2)}$ **(I.10)**
Mesure de Jaccard	$\mathbf{RSV}\ (\mathbf{Q}.\boldsymbol{D_j}) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\sum_{i=1}^{T} q_i^2 + \sum_{i=1}^{T} d_{ij}^2 - \sum_{i=1}^{T} q_i * d_{ij}}$ **(I.11)**
Mesure de cosinus	$\mathbf{RSV}\ (\mathbf{Q}.\boldsymbol{D_j}) = \frac{\sum_{i=1}^{T} q_i * d_{ij}}{\left(\sum_{i=1}^{T} q_i^2\right)^{1/2} * \left(\sum_{i=1}^{T} d_{ij}^2\right)^{1/2}}$ **(I.12)**

Tab. I.1 *Fonctions du modèle vectoriel*

- **Avantage :**

- Facile à mettre en œuvre.
- Les documents sont en effet restitués dans un ordre décroissant de leur degré de similarité avec la requête.

- **Inconvénient :** le modèle vectoriel présente le principal inconvénient lié à l'indépendance mutuelle des termes d'indexation.

I.4.3 Le Modèle Probabiliste

Le modèle de recherche probabiliste utilise un modèle mathématique fondé sur la théorie de la probabilité. Le processus de recherche se traduit par calcul de proche en proche, du degré ou probabilité de pertinence d'un document relativement à une requête. Pour ce faire, le processus de décision complète le procédé d'indexation probabiliste en utilisant deux probabilités conditionnelles :

- $\boldsymbol{P(w_{ji}/Pert)}$ **:** Probabilité que le terme $\boldsymbol{t_i}$ occure dans le document $\boldsymbol{D_j}$ sachant que ce dernier est pertinent pour la requête
- $\boldsymbol{P(w_{ji}/NonPert)}$**:** Probabilité que le terme $\boldsymbol{t_i}$ occure dans le document $\boldsymbol{D_j}$ sachant que ce dernier n'est pas pertinent pour la requête.

Le calcul d'occurrence des termes d'indexation dans les documents est basé sur l'application d'une loi de distribution (type loi de poisson) sur un échantillon représentatif de documents d'apprentissage.

En posant les hypothèses que :

- La distribution des termes dans les documents pertinents est la même que leur distribution par rapport à la totalité des documents
- Les variables « documents pertinents », « document non pertinent » sont indépendantes, la fonction de recherche est obtenue en calculant la probabilité de pertinence d'un document *D*, notée *P (Pert|D)* [Risjbergen, 79] :

$$\boldsymbol{P(Pert/D) = \frac{P(D/pert) * p(pert)}{p(D)}} \quad \textbf{(I.13)}$$

$$\boldsymbol{P(NonPert/D) = \frac{P(D/Nonpert) * p(Nonpert)}{p(D)}} \quad \textbf{(I.14)}$$

L'ordre des documents est basé sur l'une des deux méthodes :

- Considérer seulement les termes présents dans les documents et requêtes.
- Considérer les termes présents et termes absents dans les documents et requêtes.

La similitude entre requête et document est calculée comme suit :

$$RSV(Q_K, D_J) = C\sum_{i=1}^{T} q_{Ki}d_{ji} + \sum_{i=1}^{T} f_{ij}q_{Ki}d_{ji}\, log\frac{N.n_i}{n_i} \quad (\mathbf{I.15})$$

Où : $\mathbf{f_{ij}} = \frac{\mathbf{tf_{ij}}}{\mathbf{max\, tf_j}}$

C : est une Constante.

De manière générale, le modèle probabiliste présente l'intérêt d'unifier les représentations des documents et des concepts.

- **Avantages du modèle probabiliste :** le modèle de recherche probabiliste est plus efficace que le modèle de recherche booléen, mais moins performant que le modèle de recherche vectoriel.

- **Inconvénients du modèle probabiliste :** Il n'existe pas de méthode d'estimation de la pertinence des termes avant toute extraction de document pertinent. Cette estimation se fait à posteriori.

I.4.4 Le Modèle de Langue

Depuis leur introduction en recherche d'informations (RI), les modèles de langue sont distingués par leur efficacité et leur fondement mathématique solide, d'une part et d'autre part, il permet de combiner de différentes informations sur un document et sur des requêtes [Boughanem et al,04].

I.4.4.1 Modèles de Langue en Linguistique Informatique

Le terme modèle de langue est emprunté de la linguistique informatique, où l'objectif d'un modèle de langue est de capter les régularités linguistiques par une et plusieurs fonctions probabilistes. Ainsi, nous allons décrire des méthodes développées pour la modélisation statique de langue en linguistique informatique.

- **Idée de base**

Un modèle de langue se construit en utilisant « une fonction de probabilité P qui assigne une probabilité P(s) à un mot ou une séquence de mots (s) dans une langue ». On appelle langue un corpus de documents. Cette fonction permet d'estimer la probabilité d'une séquence quelconque de mots dans la langue modélisée ou de façon plus générale, d'estimer la probabilité de générer cette séquence de mots à partir du modèle de langue.

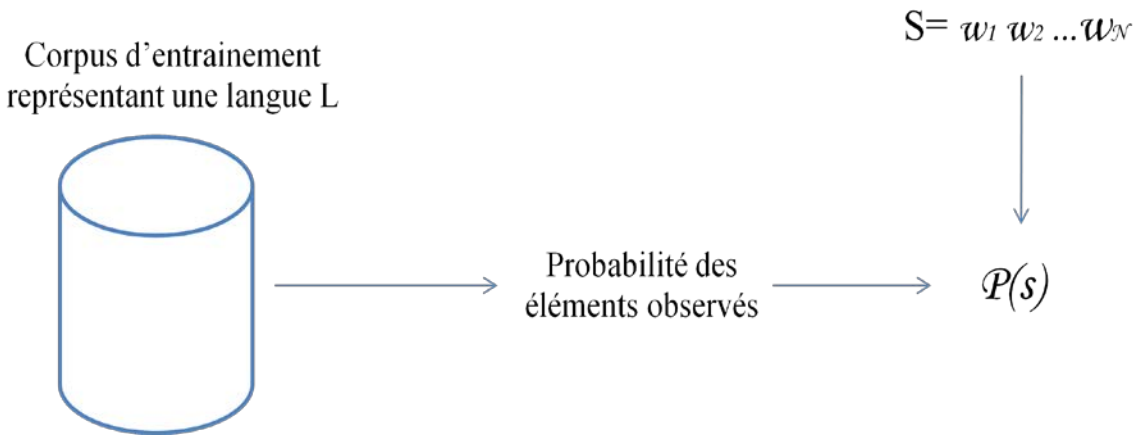


***Fig. I.3** : principe des modèles de langue*

Pour estimer la probabilité de la phrase **S = w_1 w_2 …w_n** appartenir à la langue L, nous calculons la probabilité de la phrase d'être dans le modèle ML de la langue L :

$$P(s) = \prod_{i=1}^{n} P_{ML}(w_i \mid w_1 .. w_{i-1}) \quad \textbf{(I.16)}$$

Si on utilise la règle de chaîne en théorie de probabilité pour calculer P(s), il y a souvent trop de paramètres (c'est-à-dire $P(w_i|w_1 \ldots w_{i-1})$) à estimer, et ceci est souvent impossible à réaliser. Ainsi dans les modèles de langue utilisés en pratique, des simplifications sont souvent faites. En général, nous supposons qu'un mot w_i ne dépend que de ses n-1 prédécesseurs immédiats.

Nous utilisons dans ce cas un modèle de langue n-gramme. En particulier, les modèles souvent utilisés sont les modèles Uni-gramme, Bi-gramme et Tri-gramme exprimés comme suit :

Uni-gramme : $$\mathbf{P(s)} = \prod_{i=1}^{n} P(w_i) \quad \textbf{(I.17)}$$

Bi-gramme : $$\mathbf{P(s)} = \prod_{i=1}^{n} P(w_i \mid w_{i-1})\, w_i = \frac{P(w_{i-1}\, w_i)}{P(w_{i-1})} \quad \textbf{(I.18)}$$

Tri-gramme : $$\mathbf{P(s)} = \prod_{i=1}^{n} P(w_i \mid w_{i-2}\, w_{i-1}) = \prod_{i=1}^{n} \frac{P(w_{i-2}\, w_{i-1}\, w_i)}{P(w_{i-2}\, w_{i-1})} \quad \textbf{(I.19)}$$

Pour estimer les probabilités, on utilise un corpus de documents représentatif de la langue à modéliser. Si le corpus est suffisamment grand, il permet de faire l'hypothèse que reflète la langue en général et ainsi le modèle de langue correspond approximativement au modèle de langue pour le corpus.

Le calcul de probabilité d'un mot w dans un corpus C se base sur l'estimation de vraisemblance maximale du terme w :

$$P_{ML}(\alpha) = \frac{|\alpha|}{\sum_{\alpha_j \in C} |\alpha_j|} = \frac{|\alpha|}{|C|} \quad \textbf{(I.20)}$$

Où:

|α| est la fréquence d'occurrence du n-gramme **α** dans ce corpus, $\boldsymbol{\alpha_j}$ est un n-gramme de la même langueur que **α**, et **|C|** est la taille du corpus (c'est-à-dire le nombre total d'occurrences de mots)

Nous donnons ici un exemple simple pour illustrer l'estimation de la probabilité uni-gramme ainsi que son utilisation pour calculer la probabilité d'une phrase.

Supposons un petit corpus contenant 10 mots, avec les fréquences comme montrées dans la Table suivante :

Mots	Le	Un	Prof	ML	Dit	Aime	De	Langue	Modèle	RI
Fréquence	3	2	2	1	2	1	4	2	1	2
$P_{ML}(\alpha\|C)$	0.15	0.1	0.1	0.05	0.1	0.05	0.2	0.1	0.05	0.1

***Tab.I.2** : Exemple de l'estimation de la probabilité uni-gramme*

En utilisant l'estimation de vraisemblance maximale, nous obtenons les probabilités comme illustrées dans la table (note : la fréquence totale de mots dans ce corpus est $|C| = 20$).

En utilisant ces probabilités estimées, nous pouvons calculer la probabilité de construire la séquence *s* = « **le prof aime le ML** » dans cette langue comme suit :

$P(s) \approx P(s|C) = P(\textbf{le}|C) * P(\textbf{prof}|C) * P(\textbf{aime}|C) * P(\textbf{le}|C) * P(\textbf{ML}|C) = 0{,}15 * 0{,}1 * 0{,}05 * 0{,}15 * 0{,}05.$

Par simplification d'écriture, on note P(s) au lieu de $P_{ML}(s)$ en donnant au préalable la langue modélisée.

Le principe général des modèles de langue se résume donc en deux grandes étapes : l'apprentissage du modèle de langue et l'évaluation de la probabilité d'appartenance d'un document à ce modèle. Ainsi, les paramètres du modèle de langue M_{Lx} (2) s'estime en se basant sur les caractéristiques statistiques de langue extraites du corpus d'entrainement (1). A partir de ce modèle de langue M_{Lx}, la probabilité du document D d'appartenir à la langue X s'évolue par $P(D|M_{Lx})$ (3)

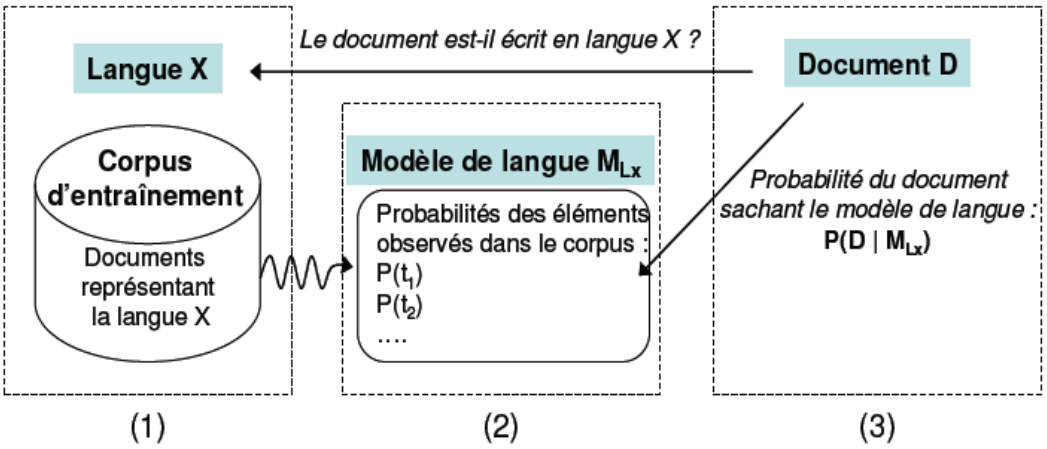


Fig. I.4 : *Principe générale des modèles de langue.*

- **Techniques de lissage**

Plus le corpus utilisé pour ces estimations est grand, plus on peut espérer obtenir des estimations de probabilité justes. Cependant, quelle que soit la taille du corpus d'entraînement, il y a toujours des mots ou des séquences de mots absents du corpus. Pour ces mots ou séquences de mots, leur estimation de probabilité est **0**. La conséquence de cette probabilité *nulle* est qu'on attribuera une probabilité *nulle* à toute séquence de mots ou des phrases contenant un mot ou un *n-gramme* non rencontré dans le corpus. Par exemple, la phrase *s* = « ***le prof dit non*** » aura une probabilité ***P(s\|C)* = 0**.

En d'autres termes, les modèles ainsi construits ne sauraient reconnaître que les phrases dont les *n-grammes* sont tous apparus dans le corpus. Ce sont donc des modèles très limités. Afin de généraliser les modèles, on voudrait assouplir cette attribution systématique de probabilité nulle aux mots ou séquences de mots non rencontrés.

Cette procédure qui consiste à attribuer une probabilité non-nulle á ces éléments est appelée le *lissage*. Le lissage peut aussi être vu comme une façon d'éviter le sur entraînement d'un modèle sur un corpus, et de doter le modèle d'une plus grande capacité de généralisation.

Le principe de lissage peut être résumé ainsi : Au lieu de distribuer la totalité de masse de probabilité sur les *n-grammes* vus dans le corpus d'entraînement, on enlève une partie de cette masse et la redistribue aux *n-grammes* non vus dans le corpus. De cette façon, les *n-grammes* absents du corpus vont recevoir une probabilité *non-nulle*.

Sur la façon d'enlever et de redistribuer une partie de masse de probabilité, il y a une série de méthodes proposées dans la littérature. Ici, nous présentons quelques-unes classiques.

A. Lissage de Laplace

Le lissage de Laplace consiste à ajouter la fréquence 1 à tous les *n-grammes*. Cette méthode est aussi appelée la méthode « *ajouter-un* ». Pour un *n-gramme* α, sa probabilité est estimée comme suit (où *V* est l'ensemble du vocabulaire d'indexes) :

$$P_{ajouter\text{-}un}(\alpha|C)=\frac{|\alpha|+1}{\sum_{\alpha_i \in V}(|\alpha_i|+1)} \quad \textbf{(I.21)}$$

On peut remarquer que cette méthode simple a un problème fatal : Si le corpus ne contient qu'une petite portion des *n-grammes* parmi tous les *n-grammes* possibles (et c'est souvent le cas dans la pratique, même pour un grand corpus), la majeure partie de la masse de probabilité sera distribuée uniformément sur les *n-grammes* non vus dans le corpus. Les *n-grammes* vus dans le corpus ne joueront qu'un rôle mineur dans la définition du modèle. On ne peut donc pas s'attendre à une très bonne performance (c'est-à-dire de reconnaître les phrases autorisées d'une langue correctement).

B. Lissage Good-Turing

L'idée de ce lissage est de modifier la fréquence d'occurrence observée de la façon suivante : Soit un *n-gramme α* qui apparaît *r* fois dans le corpus. On modifie cette fréquence à ***r**** :

$$\mathbf{r^* = (r+1)\frac{n_{r+1}}{n_r}} \quad \textbf{(I.22)}$$

Où n_r est le nombre de *n-grammes* apparus *r* fois dans le corpus. Ainsi, l'estimation de la probabilité devient la suivante :

$$P_{GT}(\alpha) = \frac{r^*}{\sum_{\alpha_i \epsilon C}|\alpha_i|} \quad \textbf{(I.23)}$$

Dans cette méthode, on applique une diminution (*discount*) de fréquence de l'ordre de $\frac{r^*}{r}$ au *n-gramme α* et cette fréquence est redistribuée sur les *n-grammes* non vus.

Quand la fréquence *r* est grande, le nombre n_r de *n-grammes* de cette fréquence est petit. Le *lissage Good-Turing* dans ce cas effectue une grande modification de la valeur de *r*. L'estimation *Good-Turing* pour les *n-grammes* de grande fréquence a donc tendance d'être instable. Dans l'autre bout du spectre, l'estimation de *Good-Turing* pour les *n-grammes* de faibles fréquences sont plus stables. C'est souvent pour ces derniers *n-grammes* que le *lissage Good-Turing* est recommandé.

C. Lissage Backoff

Le lissage « *Backoff* » consiste à utiliser un modèle du même ordre (par exemple, *bi-gramme*) si le *n-gramme* est observé dans le corpus, mais utiliser un modèle d'ordre inférieur (par exemple, *uni-gramme*) si ce n'est pas le cas. Par exemple, dans le lissage *Katz*, on peut combiner le modèle *bi-gramme* avec un modèle *uni-gramme* comme suit :

$$P_{Katz}(w_i|w_{i-1}) = \begin{cases} P_{GT}(w_i|w_{i-1}) \text{ si } |w_i w_{i-1}|>0 \\ \alpha(w_{i-1})\, P_{Katz}(w_i) \text{ sinon} \end{cases} \quad \textbf{(I.24)}$$

Où $\alpha(w_{i-1})$ *est un paramètre de normalisation.*

Dans cette méthode, la diminution de fréquence utilisée *dans* P_{GT} est redistribuée au modèle d'ordre *inférieur (uni-gramme).* $\alpha(w_{i-1})$ est un paramètre qui détermine la part de cette redistribution à m_i, déterminée comme suit :

$$\alpha\,(w_{i-1}) = \frac{1-\sum_{w_i:|w_{i-1}w_i|>0} P_{GT}(w_i|w_{i-1})}{1-\sum_{w_i:|w_{i-1}w_i|>0} P_{ML}(w_i)} \qquad \textbf{(I.25)}$$

D. Lissage par Interpolation

Le lissage par interpolation, par exemple de Jelinek-Mercer, consiste à combiner un modèle avec un ou des modèles d'ordre inférieur systématiquement, plutôt que d'utiliser ce dernier seulement dans le cas de fréquence 0 comme dans « Backoff ». Pour une combinaison de modèle *bi-gramme* avec le modèle *uni-gramme*, on a :

$$P_{JM}\,(w_i|w_{i-1}) = \lambda_{w_{i-1}} P_{ML}(w_i|w_{i-1}) + \left(1-\lambda_{w_{i-1}}\right) P_{JM}(w_i \qquad \textbf{(I.26)}$$

Où $\lambda_{w_{i-1}}$ est un paramètre déterminé de telle manière à maximiser l'espérance des données.

E. Lissage Dirichlet

Dans cette méthode, la fréquence d'un mot w_i dans le document D est incrémentée de $\mu P_{ML}(w_i|D)$, où μ. est un paramètre appelé pseudo-fréquence. La probabilité $P_{Dir}\,(w_i|D)$ d'un mot selon le modèle de langue du document devient la suivante :

$$P_{Dir}\,(w_i|D) = \frac{tf(w_i,D)+\mu P_{ML}(w_i|C)}{|D|+\mu} \qquad \textbf{(I.27)}$$

Où $|D|$ est la taille du document (le nombre total d'occurrences de mots), et $tf\,(w_i, D)$ est la fréquence du mot w_i dans D.

I.4.4.2 Modèle de langue en recherche d'information

Dans cette section, nous allons décrire quelques approches proposées utilisant des modèles de langues pour la RI. Avant de détailler, il est important de réaliser que la RI n'a pas le même but que la modélisation statistique de langue en linguistique informatique. Le but visé des modèles de langues en linguistique informatique est de déterminer la légitimité d'une séquence de mots en une langue, ou de la générer à partir du modèle. Dans cette tâche, la séquence de mots acceptée doit être conforme aux critères linguistiques d'une langue, telles les contraintes syntaxiques et morphologiques. Pour la RI, le but est différent : on vise à retrouver les documents dont la sémantique est pertinente à celle de la requête. Dans la tâche de RI, les contraintes syntaxiques et morphologiques deviennent moins importantes, mais la notion de pertinence est centrale. Comme on peut observer, cette notion est absente dans les

méthodes de modélisation de langue qu'on vient de décrire. Toutefois, les deux domaines ont certaines caractéristiques en commun :

- Ils traitent d'une langue naturelle ;
- Ils disposent d'un grand volume de textes.

Ainsi, les idées et les approches développées dans la linguistique informatique peuvent être adaptées pour la RI. C'est cette intuition que les approches de la RI basées sur les modèles de langue tentent d'exploiter.

- **Modèle de Ponte et Croft**

En recherche d'information, les modèles de langue ont été introduits en 1998 par *Ponte* et *Croft* (Ponte,98) qui voient le score d'un document face à une requête comme la probabilité que la requête soit générée par le modèle du document. Ainsi,

$$\boldsymbol{Score\ (D, Q) = P\ (Q|M_d)} \qquad \textbf{(I.28)}$$

Plusieurs façons d'exprimer ce score demeurent envisageables. Ponte et Croft combine un modèle de langue du document et un modèle de langue du corpus. Dans cette approche, non seulement la probabilité des mots d'appartenir à la requête est prise en compte mais également la probabilité de ne pas rencontrer les mots n'appartenant pas à la requête entre en jeu. Ceci permet de différencier un document contenant beaucoup de sujets de la requête d'un document en contenant peu par la prise en compte des mots absents de la requête. Toutefois, cette approche s'avère coûteuse en calcul si le nombre de termes absents de la requête est important, ce qui est généralement le cas.

- **Modèle de Hiemstra et al. et de Miller et al.**

Hiemstra (Hiemstra,98) propose d'évaluer le score du document face à une requête en utilisant une approche par interpolation :

$$\boldsymbol{Score\ (D, Q) = \prod_{t \in Q} (\alpha\ P_{ML}(t_i|D) + (1 - \alpha) P_{ML}(t_i|C))} \qquad \textbf{(I.29)}$$

Miller et al. (Miller,98 ; Miller,99) reformule le modèle de Hiemstra à l'aide d'un modèle de Markov caché à deux états

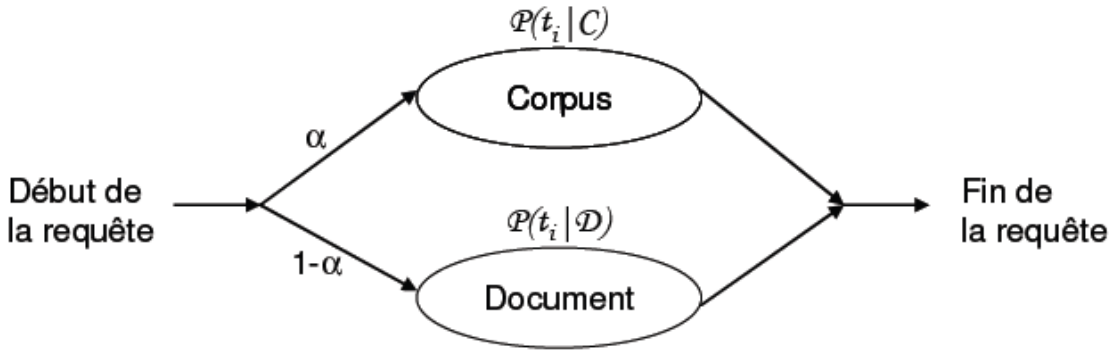


Fig I.5. : *Modèle de Morkov caché à deux états*

Enfin, Ng (Ng,99 ; Ng,00) offre une variante du modèle de la langue basée sur le ratio de vraisemblance :

$$\textbf{Score (D, Q)} = \sum_{t \in Q} log \frac{(\alpha P_{ML}(t|D) + (1-\alpha) P_{GT}(t|C))}{P_{GT}(t|C)} \quad \textbf{(I.30)}$$

$P_{GT}(t|C)$ correspond à un lissage de type Good-Turing.

I.5 Évaluation des SRI

L'évaluation d'un système de recherche d'information peut être appréhendée selon deux aspects : un aspect efficience et un aspect efficacité. L'aspect efficience dépend de l'évaluation cognitive de l'utilisateur, tels que la facilité d'utilisation du système, rapidité d'accès, temps de réponse à une requête, présentation des résultats, etc. L'aspect efficacité concerne la capacité du système à sélectionner le maximum de documents pertinents et un minimum de documents non pertinents. Nous nous intéressons dans cette section à présenter l'aspect efficacité, qui est souvent mesuré par deux paramètres Rappel/Précision.

I.5.1 Les Mesures d'évaluation

- **Mesures de précision et de rappel**

Ce sont les deux métriques les plus utilisées pour évaluer la performance d'un système de recherche d'information. Pour présenter ces deux mesures, la figure I.6 introduit le partitionnement de l'ensemble B des documents restitués par le SRI en deux sous-ensembles : un sous-ensemble de documents non pertinents et un sous-ensemble de documents pertinents.

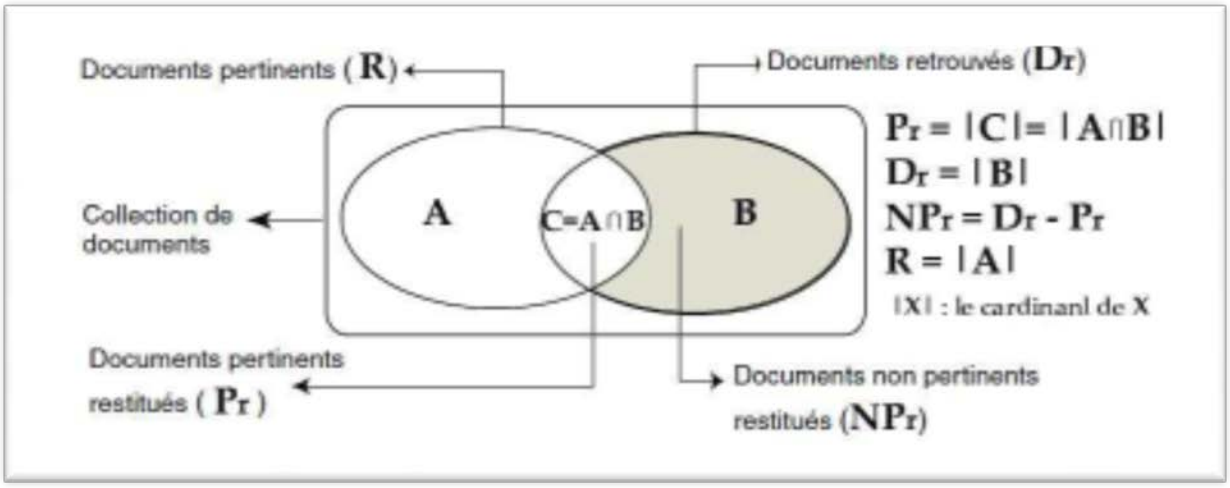


Fig I.6 *Partition de la collection pour une requête*

Les taux de rappel et de précision sont définis comme suit :

- **Taux de rappel**

Le rappel (Ra) mesure la capacité du système de retrouver tous les documents pertinents répondant à une requête. Autrement dit, il mesure le pourcentage de documents pertinents, selon une liste de référence (ensemble des documents pertinents de la collection), que le système a trouvé :

$$\boldsymbol{Rappel\ (Ra)} = \frac{\textbf{Nombre de documents pertinents retrouvés}}{\textbf{Nombre de documents pertinents dans la collection}} = \frac{\textbf{Pr}}{\textbf{R}} \qquad \textbf{(I.31)}$$

- **Taux de précision :**

La précision (P) mesure la capacité du système de renvoyer tous les documents pertinents à une requête. Autrement dit, elle mesure le pourcentage de documents retrouvés qui sont pertinents :

$$\boldsymbol{Précision\ (P)} = \frac{\textbf{Nombre de documents pertinents retrouvés}}{\textbf{Nombre total de documents retrouvés}} = \frac{\textbf{Pr}}{\textbf{Dr}} \qquad \textbf{(I.32)}$$

- **Le bruit (B)**

Mesure le taux de documents non pertinents restitués par rapport au total des documents restitués .Il s'exprime comme suit :

$$\boldsymbol{Bruit\ (B)} = \frac{\textbf{Nombre de documents non pertinents retrouvés}}{\textbf{Nombre de documents retrouvés}} = \frac{\textbf{NPr}}{\textbf{Dr}} \qquad \textbf{(I.33)}$$

Aussi le bruit est exprimé en fonction de la précision : $\mathbf{B = 1 - P}$ **(I.34)**

- **Le silence (S)**

Mesure le taux de documents pertinents non restitués par rapport au total des documents pertinents de la collection. Il s'exprime par :

$$\boldsymbol{Silence(S) = \frac{\text{Nombre de documents pertinents non retrouvés}}{\text{Nombre de documents pertinents}} = \frac{R-Pr}{R}} \qquad \textbf{(I.35)}$$

Aussi le silence est exprimé en fonction du rappel : $\mathbf{S = 1 - Ra}$ **(I.36)**

Il est facile de remarquer que la précision et le rappel sont liés : quand l'un augmente l'autre diminue. Ainsi, pour un système, on a une courbe de précision-rappel qui a en général la forme suivante :

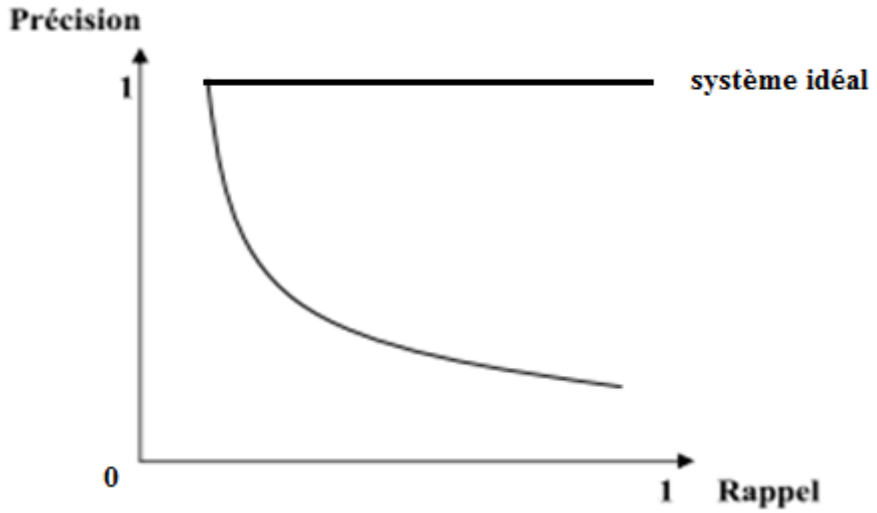


***Fig. I.7** : Allure d'une courbe de rappel-précision*

Cette figure montre les variations du rappel et de la précision qui sont incluses dans l'intervalle [0.1] et que ces variations sont inversement proportionnelles. Dans la pratique, ces mesures n'atteignent pas la valeur de 1, qui serait le synonyme du système idéal.

- **Autres mesures d'évaluation les plus courantes**

- MAP (Mean Average Precision) La MAP mesure la capacité du modèle d'appariement ou d'un SRI à pouvoir sélectionner les documents pertinents, en réponse à un ensemble de requêtes. La MAP (Voorhees,01) permet de comparer des systèmes sur les précisions à différents niveau de coupe de la réponse du système (et non pas seulement à 5 documents). Plus précisément, la précision moyenne pour une requête est la moyenne des précisions obtenues chaque fois qu'un document pertinent est retrouvé. Lorsqu'un document non pertinent est retrouvé, sa précision est égale à 0, la formule suivante donne la méthode de calcul de la MAP [Bouidghaghen, 11]

$$MAP = \frac{\sum_{q \in Q} AP_q}{|Q|} \quad \textbf{(I.37)}$$

Où **APq** est la précision moyenne d'une requête **q**, **Q** est l'ensemble des requêtes et **|Q|** est le nombre de requêtes. Cette mesure peut être qualifiée de globale puisqu'elle combine différents points de mesure. Cette précision peut être aussi calculée à différents niveaux de rappel **(0%, 10%, 20%, . . . ,100%),** elle est alors appelée : précision moyenne interpolée (MAiP). Pour chaque niveau de rappel, les valeurs calculées sont moyennées sur tout l'ensemble des requêtes.

- **MRR** (Mean Reciprocal Rank) : une autre mesure basée rang est la métrique Mean Reciprocal Rank. Elle permet d'évaluer le nombre de documents qu'il faut considérer avant de retrouver le premier document pertinent. Elle est égale à la moyenne calculée sur l'ensemble des requêtes, du rang du premier document pertinent. [Bouidghaghen, 11]

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{rank\ i} \quad \textbf{(I.38)}$$

MRR est nulle pour une requête si aucun document pertinent n'est retourné par le système. Cependant, MRR donne un score élevé pour un système qui retourne des documents pertinents en haut de la liste présentée à l'utilisateur. Cette mesure est couramment utilisée dans les systèmes Questions-Réponses où l'utilisateur s'intéresse à recevoir la bonne réponse en premier rang.

I.5.2 Collection de test

Pour arriver à une telle évaluation, on doit connaître les réponses idéales de l'utilisateur. Ainsi, l'évaluation d'un système se fait à l'aide d'un corpus de test.
On notera que le calcul de ses mesures entre dans le cadre de collection de tests qui englobe :

- Un ensemble de documents.
- Un ensemble de requêtes.
- La liste de documents pertinents pour chaque requête.
- Des mesures et des critères quantifiables.

Depuis l'apparition de l'expression « recherche d'information » dans le mémoire de fin d'étude de Calvin Mooers en 1950, le monde de la RI n'a cessé de développer des outils et méthodes de recherche et, en parallèle des campagnes d'évaluation pour tester les méthodes et outils développés.

Parmi les compagnes de test qui ont été dirigées dans le cadre du développement de la communication entre l'industrie, l'université et l'état en mettant en place des moyens favorisant les échanges d'idées sur la recherche et l'évaluation en RI retrouve la compagne TREC(Text Retrieval Conférence).[Ihadjaden,04]

- **Les compagnes TREC :** Les conférences TREC (Text REtrieval Conferences) constituent à ce jour l'initiative la plus importante dans le domaine de l'évaluation des

systèmes de recherche d'information. Lancées en 1991 (bien que la première conférence se soit tenue en novembre 1992), onze compagnes ont été organisées jusqu'à présent par le NIST (National Institue of Standards and Technologiy) avec le soutien financier de la DARPA (Defence Advanced Research Projects Agency). Les objectifs de TREC sont d'encourager la recherche, de faciliter les échanges entre différentes communautés (recherche, enseignement, industrie et administrations), d'accélérer le transfert technologique de la recherche vers l'industrie et d'améliorer et rendre accessible les méthodes d'évaluation. Dès l'origine, un aspect important de TREC a été la volonté de tester les systèmes sur des corpus de documents de grande taille**.**

I.6 Conclusion

Dans ce chapitre, nous avons présenté les concepts de bases de la recherche d'information classique. Nous y avons abordé la définition et l'architecture générale d'un SRI, les techniques d'indexation utilisées en RI, les principaux modèles de recherche d'informations, ainsi que les méthodes d'évaluation adoptées pour attester de la qualité d'un SRI. Principalement, le modèle de recherche d'informations se base sur l'utilisation des termes simples comme descripteurs. Cependant, l'utilisation de ce type de descripteurs pose le problème d'ambigüité. Pour y remédier, des types plus élaborés de descripteurs sont proposés et utilisés, parmi eux, les entités nommées qui sont des unités textuelles particulières. Il s'agit de reconnaître les noms propres (personnes, lieux, organisations) mais aussi les expressions temporelles (dates, durées, horaires) et les noms de quantités (monétaires, unités de mesure, pourcentages).

Notre travail s'insère dans cette direction, à savoir l'utilisation des entités nommées que nous traiterons dans le chapitre suivant.

Entités Nommées

II.1 Introduction

La notion d'Entité Nommée (EN) est utilisée pour regrouper tous les éléments du langage qui font référence à une entité unique et concrète, appartenant à un domaine spécifique (*humain, économique, géographique,* etc.). On désigne traditionnellement par le terme « EN » les noms propres au sens classique, les noms propres dans un sens élargi mais aussi les expressions de temps et de quantité. Dans ce chapitre nous allons présenter les entités nommées en détails, particulièrement leur utilisation dans différents domaines et de leur extraction. [Chinchor,98]

II.2 Définitions des entités nommées

Même s'il n'existe pas de définition standard, on peut dire que les EN sont des types d'unités lexicales particuliers qui font référence à une entité du monde concret dans certains domaines spécifiques notamment humains, sociaux, politiques, économiques ou géographiques et qui ont un nom (typiquement un nom propre ou un acronyme). [LeMeur et al.,04]

Autre définition via le nom propre : qui déclare nom propre : « Toute expression associée dans la mémoire à long terme à un particulier en vertu d'un lien dénominatif conventionnel stable ». Cette définition, plus souple, permet d'inclure les noms propres descriptifs, qu'il est convenu dans le monde du traitement automatique des langues, d'appeler entités nommées [Tolone,06]

Une entité a généralement une existence relativement stable dans le temps, même si cette existence a un début (naissance, fondation, dépôt, formation...) et une fin (mort, dissolution, faillite, disparition...) et si l'entité évolue entre temps. Pour appréhender plus simplement cette notion, on s'appuiera de manière générale sur le « principe du catalogue » pour savoir si on a affaire à une EN. Ainsi si on peut aisément imaginer l'EN supposée comme étant une entrée d'un catalogue, annuaire, dictionnaire ou index alors celle-ci sera bien une EN. [LeMeur et al,04]

II.3 Les différentes catégories des entités nommées

Parmi les catégories des entités nommées nous citons :

II.3.1 Variations des EN

Il existe plusieurs types de variations d'EN :

Une **variation graphique** peut être aussi simple que l'utilisation ou non de majuscules (ex. : *Parti Socialiste* et *Parti socialiste*), la présence ou non de points dans les sigles ou acronymes (ex. : *P.M.E.*, *P.M.E* et *PME*). Un autre cas courant concerne les sigles ou acronymes (ex. : *AFNOR* et *Association Française de Normalisation*, *DSK* et *Dominique Strauss Khan*).

Une **variation syntaxique** peut correspondre à un changement dans l'ordre des mots (ex. : *Jacques Chirac* et *Chirac Jacques*) ou à l'alternance entre des paires syntaxico-sémantiquement équivalentes (ex. : *le Président de la Côte d'Ivoire* et *le Président ivoirien*).

Les **variations lexicales** imposent une normalisation plus complexe (ex. : *Jacques Chirac* et *le Président de la République française)*. Elles sont de plus souvent amalgamées avec les autres variations (ex. : *Jacques Chirac*, *J.Chirac*, *Chirac*, *le Président Chirac*, *le Président de la République française*).

Les **ellipses** constituent un dernier exemple de variations complexes où le contexte a une très grande importance (ex. : *école normale sup, normale sup, normale*). [Tolone,06]

II.3.2 Classes sémantiques

Ces classes explicitent la nature des entités nommées.

Certaines entités sont cependant **ambiguës** pour cause d'homographie (ex. : *Washington* est à la fois une personne, une ville et un état américain). Dans ce cas, le choix de l'entité à utiliser se fera grâce au contexte syntaxique et aux indices externes.

Dans le cas d'entités **poly-référentielles**, une même entité nommée peut appartenir à plusieurs classes (ex. : l'*ONU* est à la fois un lieu et une organisation : « *L'ONU a décidé que…* » et « *Le président s'est rendu à l'ONU* »). Son sens pourra également être désambiguïsé grâce au contexte syntaxique et aux indices externes.

Les **traits sémantiques** permettent de spécifier les classes sémantiques (ex. : l'entité *Président Jacques Chirac* des classe « noms de personnes » a pour traits fonction = président, prénom = Jacques et patronyme = Chirac). [Tolone,06]

II.3.3 Application d'Extraction d'Entités Nommées (EEN)

L'Extraction des entités nommées (EEN) est la combinaison des méthodes pour indiquer des entités nommées dans des documents et les utilisé dans des objets différents.

L'EEN a plusieurs applications en réel. Dans cette partie, nous présentons quelques applications dans des systèmes de traitement des informations automatiques par l'ordinateur. [Nguyen,02]

II.3.3.1 Extraction des informations du texte

Normalement, quand on veut extraire des informations dans les textes libres comme des informations de la personne (nom, adresse, numéro de téléphone, lieu domicilié), l'utilisateur doit lire des documents et note toutes ces informations dans un tableau. Mais le travail est morne notamment avec les grandes données. Le système d'extraction des entités nommées peut tirer automatiquement ces informations. [Nguyen,02]

La phase d'extraction d'entités nommées consiste à mettre en place un système de détection et de typage des entités d'intérêt dans un texte. Notre objectif, ici, se limite à l'extraction des entités de type « Person », « Organization » et « Location », dans des textes écrits en anglais. Nous détaillons par la suite notre façon de procéder pour réaliser cet objectif.

II.3.3.2 Répondre automatiquement à des questions

L'EEN joue un rôle important dans le système de réponse automatique. Le système peut savoir le nom de la personne et fournit des services correspondants.

Nous exprimons ci-dessous le système de réponse des notes dans le concours à l'université. Aujourd'hui, pour savoir les notes du concours. Des personnes doivent préparer un message avec le format fixé. Le message contient le numéro d'identité. Le système obtenu ce numéro d'identité et trouve des notes dans la base de données. Mais Si l'utilisateur ne connaît pas le format par exemple, il a seulement le nom et la date de naissance ou autres informations, alors, il est impossible de réaliser la recherche. [Nguyen,02]

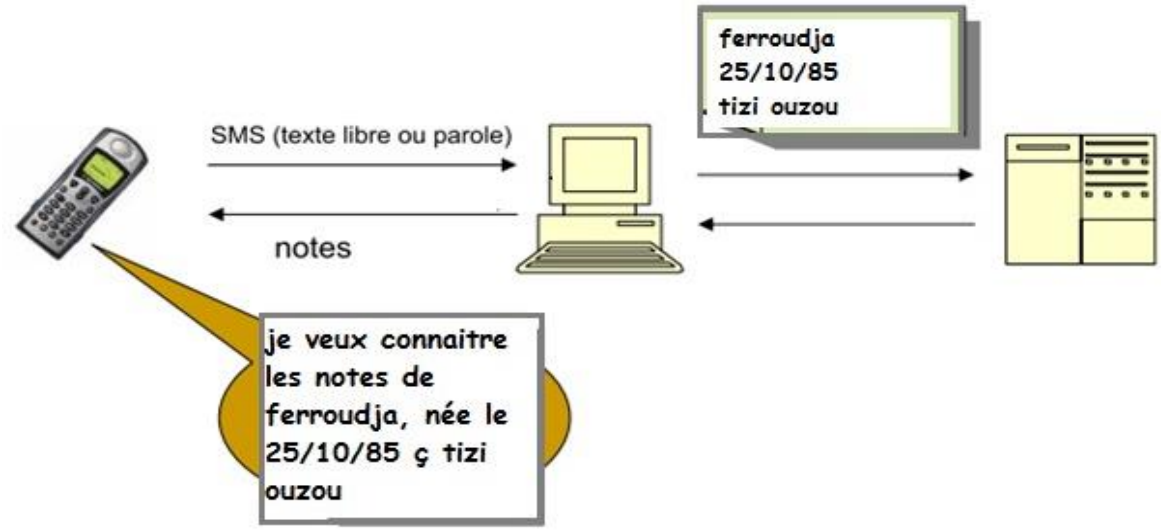


***Fig. II.1** : Modèle du système de réponse automatique fourni des notes de concours*

Dans ce système, l'utilisateur peut envoyer un message avec n'importe quel format. Il contient des informations nécessaires (par ex. le nom, la date de naissance, etc.). Le système reconnaît des entités nommées et les utilise pour trouver des notes correspondantes

Cependant, l'EEN est une partie du système. Le système peut intégrer le module de reconnaissance des paroles ou autre pour améliorer des fonctions. [Nguyen,02]

II.3.3.3 Améliorer des résultats du système de recherche.

Des moteurs de recherche aujourd'hui utilisent souvent des mots clés pour chercher des informations via Internet. Ils donnent toutes informations concernées comme étant des mots clés. Par exemple, on veut trouver toutes les informations sur les lieux, ils s'appellent SAID AMIROUCHE. S'il utilise ce mot clé, le moteur donne plusieurs informations. Il comprend des noms de personne et le nom de location.

Dans l'exemple ci-dessous, nous présentons un module pour améliorer les résultats de recherche.

Premièrement, l'utilisateur utilise un groupe de mots comme des mots clés. Le système d'EEN extrait des entités nommées dans le groupe des mots. Il va utiliser cette EN pour trouver des documents. Enfin, le système d'EEN filtre des résultats pour diminuer des informations superflues.

Nous pouvons voir plus clair dans le graphe-ci

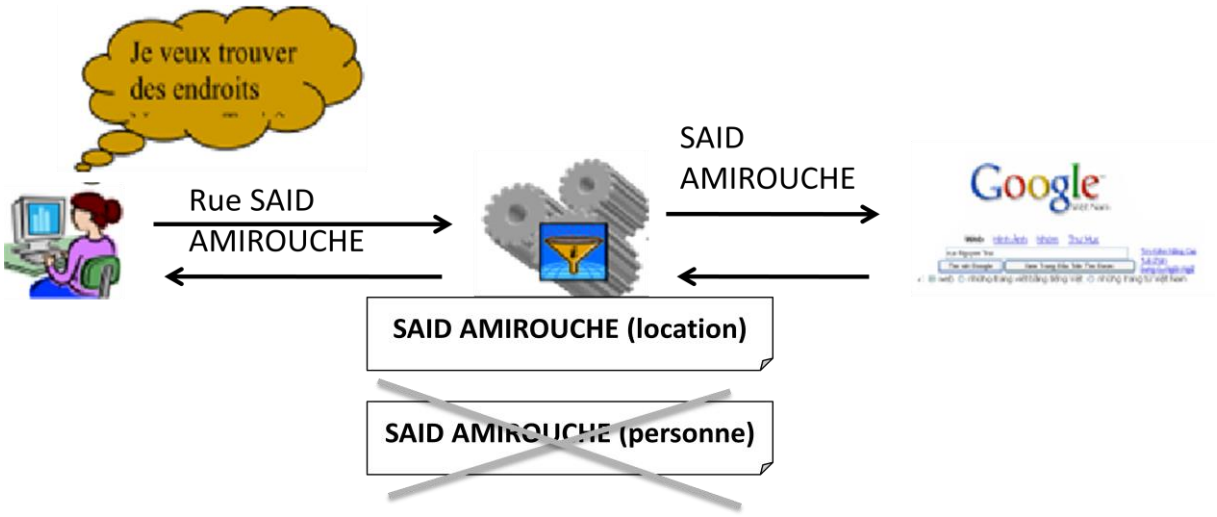


***Fig. II.2** : Modèle du système d'amélioration des résultats de recherche*

II.4 Les typologies d'Entités Nommées

Réaliser la reconnaissance d'entités nommées suppose simultanément, de les détecter et leur associer un type. Pour ce faire, il faut au préalable disposer d'une typologie au sein de laquelle les types appropriés pourront être sélectionnés. Les typologies construites à cet effet peuvent avoir une plus ou moins grande complexité. Dans la plupart des travaux, les typologies forment une partition des entités nommées selon des critères d'ordre sémantique. Les types peuvent ensuite être organisés, sous forme de hiérarchies ou d'ontologies. Ceci peut être formalisé comme une relation d'ordre entre ces types. Dans la plupart des cas, les entités nommées sont généralement associées aux feuilles de cette typologie, plus rarement à n'importe quel nœud.

Nous ne discutons pas en détail ces considérations, mais notons néanmoins qu'il pourrait s'avérer nécessaire à l'avenir, dans des situations d'ambiguïté ou de rôles multiples du référent, de tenir compte d'une plus grande finesse pour la reconnaissance des entités nommées, par exemple à l'aide de :

– **Types généraux :** une entité nommée peut être associé à un nœud non feuille de la typologie.

– **Types multiples :** une entité nommée peut être associée à plusieurs types (facettes) au sein de la typologie.

– **Treillis :** la typologie est organisée comme un treillis plutôt qu'une hiérarchie : un type peut avoir plusieurs sur-types.

– **Structures de traits :** à une entité nommée peuvent-être associés un ensemble de traits, un raisonnement ultérieur permettant de déduire, pour une entité, son ou ses type(s) selon ses traits définis.

Les entités nommées y sont réparties en 7 types primaires et 32 sous-types, dont voici la liste des 7 types :

– **Personnes :** personnes individuelles, personnes collectives.

– **Lieux :** lieux administratifs (ville, région, nations, surannations), lieux physiques (géographiques, hydrologiques, astrologiques).

– **Organisations :** entreprises, administrations.

– Temps : dates (absolues ou relatives) et horaires (absolus ou relatifs).
– Montants : quantités, durées.
– Produits : objets manufacturés, œuvres artistiques, œuvres médiatiques, produits financiers, logiciels, récompenses, voies, doctrines, lois.
– Fonctions : fonctions individuelles, fonctions collectives.

Cette typologie ajoute donc aux entités nommées traditionnelles les produits et les fonctions. Elle ajoute une granularité supplémentaire sous-types) aux types principaux (ou primaires). Nous notons que les produits recouvrent une assez grande diversité de sous-types. [Damien,12]

II.5 Les différentes méthodes d'extraction d'entité nommée :

Il existe trois types principaux de systèmes pour extraire les noms propres.

II.5.1 Les systèmes à base de règles

La majorité des systèmes utilisent cette approche. Les systèmes typiques à base de règles utilisent les preuves externes et internes, ainsi que des dictionnaires de noms propres pour les repérer. Les règles sont écrites à la main.

On note que les stratégies à base de règles ne sont pas idéales pour des corpus composés de textes ne répondant pas à des critères rédactionnels stricts, par exemple, les e-mails. C'est le système qui est utilisé ici.

II.5.2 Les systèmes à apprentissage

Ils construisent leurs connaissances automatiquement grâce à un apprentissage sur un corpus d'entraînement.

En 1999, Collins et Singer ont créé un classifieur basé sur un apprentissage non supervisé travaillant sur des textes anglais. Les seules règles sont au nombre de 7 et sont extrêmement simples :

New York est un lieu, California est un lieu, U.S. est un lieu, Tout nom qui contient Mr. Est un nom de personne, Tout nom qui contient Incorporated est un nom d'organisation, I.B.M. est une organisation, Microsoft est une organisation.

Ces 7 règles de base (seed rules) permettent à l'algorithme d'apprentissage de s'amorcer et de déduire de nouvelles règles. Par exemple, avec la phrase Mr. Cooper, a président of …, le système infère que président prédit un nom de personne car Mr. Prédit un nom de personne.

II.5.3 Les systèmes hybrides

Ils utilisent des règles écrites à la main mais construisent aussi une partie de leurs règles à l'aide d'informations syntaxiques et d'informations sur le discours tirées de données d'entraînement grâce à des algorithmes d'apprentissage, des arbres de décisions. [Tolone,06]

II.6 Les outils de traitement d'EN

Ce tableau résume les principaux outils de traitement d'EN :

POS Tagger I (Tree Tagger)	
Auteur(s)	Helmut Schmid
Description	Étiquetage morphosyntaxique et lemmatisation (Outil probabiliste (HMM) basé sur des arbres de décision
Langue(s)	Multilingue
Site web	http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger
Licence	gratuit pour l'évaluation, la recherche et l'enseignement
Plateforme	Multiplateforme
Langage	C++
Référence(s)	Schmid (1994, 1995)
Institution	Institute for Natural Language Processing (IMS), Universite de Stuttgart
POS Tagger II (LIA Tools : LIA_TAGG)	
Auteur(s)	Frédéric Béchet
Description	Étiquetage morphosyntaxique et lemmatisation
Langue(s)	Français
Site web	**http://pageperso.lif.univ-mrs.fr/~frederic.bechet/download_fred.htm**
Licence	GPL
Plateforme	Multiplatform
Institution	Laboratoire d'Informatique Fondamentale (LIF), Université Aix Marseille
POS Tagger III (Stanford POS Tagger)	
Description	Entropie maximale (CMM) partie du discours (POS) tagger
Langue(s)	Anglais, Arabe, Chinois, Allemand
Site web	**http://www-nlp.stanford.edu/software/tagger.shtml**
Licence	GPLv2
Plateforme	Multiplateforme
Langage	Java
Référence(s)	Toutanova et Manning (2000) ; Toutanova, Klein, Manning, et Singer (2003)
Institution	Stanford NLP Group, Université de Stanford
POS Tagger IV (Illinois Part of Speech Tagger	
Auteur(s)	Nick Rizzolo
Description	C'est un état de la tagger NER d'art que les étiquettes de texte avec entités nommées (personnes / organisations / lieux / divers).
Langue(s)	Anglais
Site web	**http://cogcomp.cs.illinois.edu/page/software_view/3**
Licence	Gratuit
Plateforme	Multiplateforme
Langage	Java
Institution	Cognitive Computation Group (CCG), University of Illinois
Reconnaissance des entités nommées I (LIA Tools : LIA_NE)	
Auteur(s)	Frédéric Béchet
Description	Reconnaissance des entités nommées
Langue(s)	Français
Site web	**http://pageperso.lif.univ-mrs.fr/~frederic.bechet/download_fred.htm**

Licence	GPL
Plateforme	Multiplatforme
Institution	Laboratoire d'Informatique Fondamentale (LIF), Université Aix Marseille
Reconnaissance des entités nommées II (Stanford Named Entity Recognizer)	
Description	Modèle conditionnel aléatoire de champ de séquence, avec des caractéristiques bien conçues pour la reconnaissance d'entités nommées.
Langue(s)	Anglais
Site web	**http://www-nlp.stanford.edu/software/CRF-NER.shtml**
Licence	GPLv2
Plateforme	Multiplateforme
Langage	Java
Référence(s)	Finkel, Grenager, et Manning (2005)
Institution	Stanford NLP Group, Université de Stanford
Reconnaissance des entités nommées III (Illinois Named Entity Tagger)	
Auteur(s)	Lev Ratinov
Description	C'est un état de la tagger NER d'art que les étiquettes de texte avec entités nommées (personnes / organisations / lieux / divers)
Langue(s)	Anglais
Site web	**http://cogcomp.cs.illinois.edu/page/software_view/4**
Licence	Gratuit
Plateforme	Multiplateforme
Langage	Java
Référence(s)	Ratinov et Roth (2009)
Institution	Cognitive Computation Group (CCG), University of Illinois

***Tab. II.1**. Les outils d'extraction d'EN (named entity tool recognition –stanford)*

Pour l'extraction des entités nommées dans notre cas nous avons utilisé l'outil GATE, la section suivante traite son fonctionnement.

II.7 GATE (General Architecture for Text Engineering)

Créée par les chercheurs de l'université de Sheffield (Royaume-Uni), GATE est largement utilisé par les experts en TAL et dispose d'une grande communauté d'utilisateurs. Cela lui permet de disposer d'un ensemble de solutions d'aide et de support (forum, liste de diffusion, foire aux questions, wiki, tutoriels, etc.), point particulièrement important lorsque l'on débute avec un tel outil. Plateforme open source en Java dédiée à l'ingénierie textuelle. Par ailleurs, les créateurs de GATE proposent des formations pour améliorer son niveau ainsi que des certifications permettant de faire valoir ses compétences à un niveau professionnel. [Serrano,10]

II.7.1 Fonctionnement général

L'environnement GATE peut être utilisé en tant que librairie ou grâce à une interface graphique, solution que nous avons privilégiée ici. L'interface graphique permet de charger de nouveaux plugins et ressources, de les paramétrer et de les combiner au sein d'une même chaine de traitement. L'outil repose sur le principe d'une chaine de traitement « pipeline » composée de plusieurs modules dits Processing Resources (PR) appliqués successivement sur

un ou plusieurs textes dits Language Resources (LR). Les documents donnés en entrée peuvent être un simple texte ou un corpus, fournis soit par un « copier-coller » soit par une URL. Les différents composants annotent chacun à leur tour le texte en prenant en compte les annotations précédentes puis le document est retourné à l'utilisateur au format XML. Les annotations correspondent généralement à l'association d'attributs « features » à une zone de texte. Ceux-ci se présentent sous la forme propriété = « valeur », où la valeur peut être une chaîne de caractères ou un nombre.

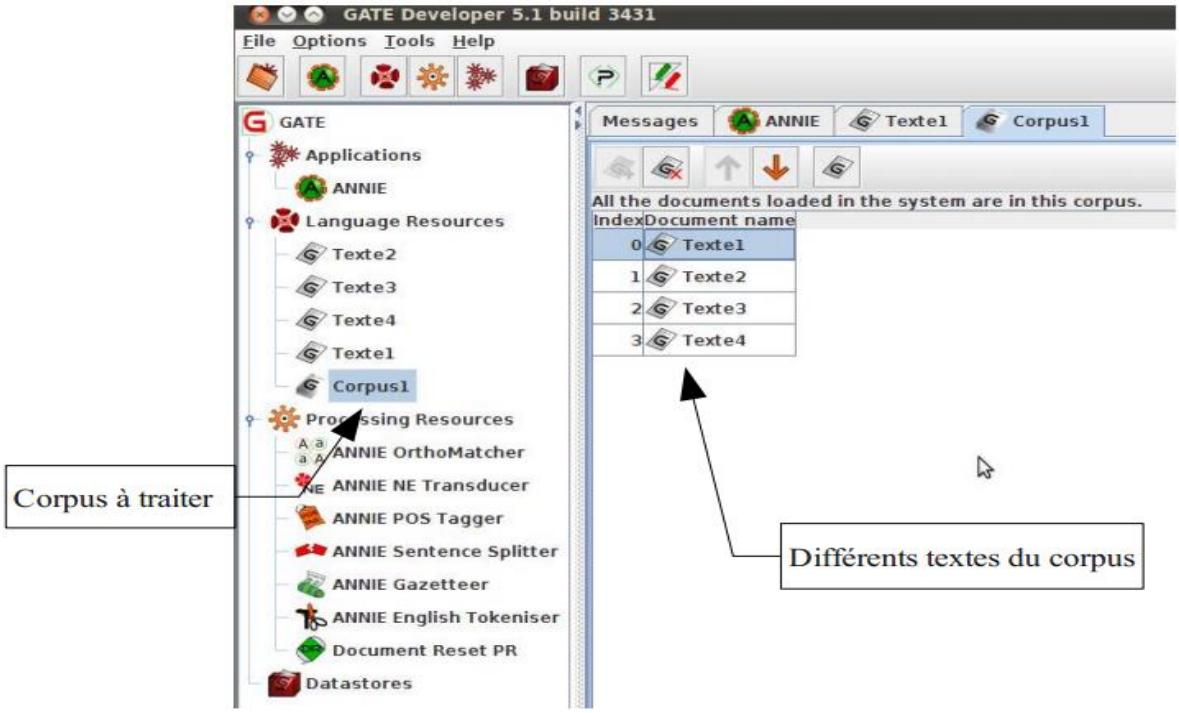


***Fig II.3** : Exemple de corpus dans GATE*

II.7.2 Le formalisme JAPE

Une partie des différents modules proposés dans GATE est basée sur JAPE (Java Annotation Patterns Engine), un transducteur à états finis permettant de reconnaître des expressions régulières sur les annotations. Ce système s'avère très utile en extraction d'informations car il permet de définir les contextes d'apparition des éléments à extraire pour ensuite les repérer et les annoter. Le principe est de combiner différentes annotations « basiques » (tokens, syntagmes, relations syntaxiques, etc.) pour en créer de nouvelles plus complexes (entités nommées, relations, évènements, etc.) : cela revient à l'écriture de règles de production et donc à l'élaboration d'une grammaire régulière.

Une grammaire JAPE se décompose en plusieurs phases exécutées consécutivement et formant une cascade d'automates à états finis. Chaque phase correspond à un fichier « .jape » et peut être constituée d'une ou plusieurs règle(s) écrite(s) selon le formalisme associé à JAPE. Classiquement, ces règles sont divisées en deux blocs : une partie gauche (« Left Hand Side » ou LHS) définissant un motif d'annotations à repérer et une partie droite (« Right Hand Side » ou RHS) contenant les opérations à effectuer sur ce motif. Le lien entre ces deux parties se fait par l'attribution d'une étiquette au motif (ou à ses constituants) en LHS et par sa

réutilisation en RHS pour y appliquer les opérations nécessaires. Pour plus de clarté, prenons l'exemple d'une règle simple :

```
1.  Rule: OrgAcronym
2.  ((
3.   {Organization}
4.   {Token.string == "("}
5.   ({Token.orth == "allCaps"}):org
6.   {Token.string == ")"}
7.  )
8.  -->
9.  :org.Organization = {rule = "OrgAcronym"}
```

Fig. II.4 *Exemple de règle avec le formalisme JAPE*

L'objectif de celle-ci est d'annoter en tant qu'organisation les acronymes entre parenthèses positionnés après une annotation de type « Organization ». Tout d'abord, l'on commence par donner un nom à la règle (ligne 1). Les lignes 2 à 7 définissent le motif à repérer dans le texte. Le signe « --> » (ligne 8) sert de séparateur entre LHS et RHS. Enfin, la dernière ligne (ligne 9) exprime l'opération souhaitée. Précisons quelques règles syntaxiques de base du formalisme JAPE :

• La partie gauche de la règle est toujours entre parenthèses,
• La partie droite débute par le signe « --> »,
• Les types d'annotation sont encadrés par des accolades (ex : {Organization}),
• « Token.string » permet d'obtenir la valeur de la propriété « string » associée à l'annotation « Token »,
• « :org » permet d'étiqueter une partie du motif en LHS pour l'utiliser en RHS,

La ligne 9 attribue une annotation de type « Organization » au segment étiqueté « org » en LHS ; l'ajout de la propriété « rule » à cette annotation permet d'indiquer quelle règle en est à l'origine.

La liste des annotations utilisées en LHS de la règle est déclarée en début de phase grâce à l'attribut « Input » : par exemple, « Input : Lookup, Token, Person ». Par ailleurs, l'attribut « Control » permet de définir l'ordre d'exécution des différentes règles d'une phase et « Debug » d'obtenir un affichage des éventuels conflits rencontrés entre règles.

Pour finir, précisons qu'un système de macros est également disponible : une macro permet de définir et de nommer une séquence d'annotations afin de la réutiliser plus rapidement par la suite.

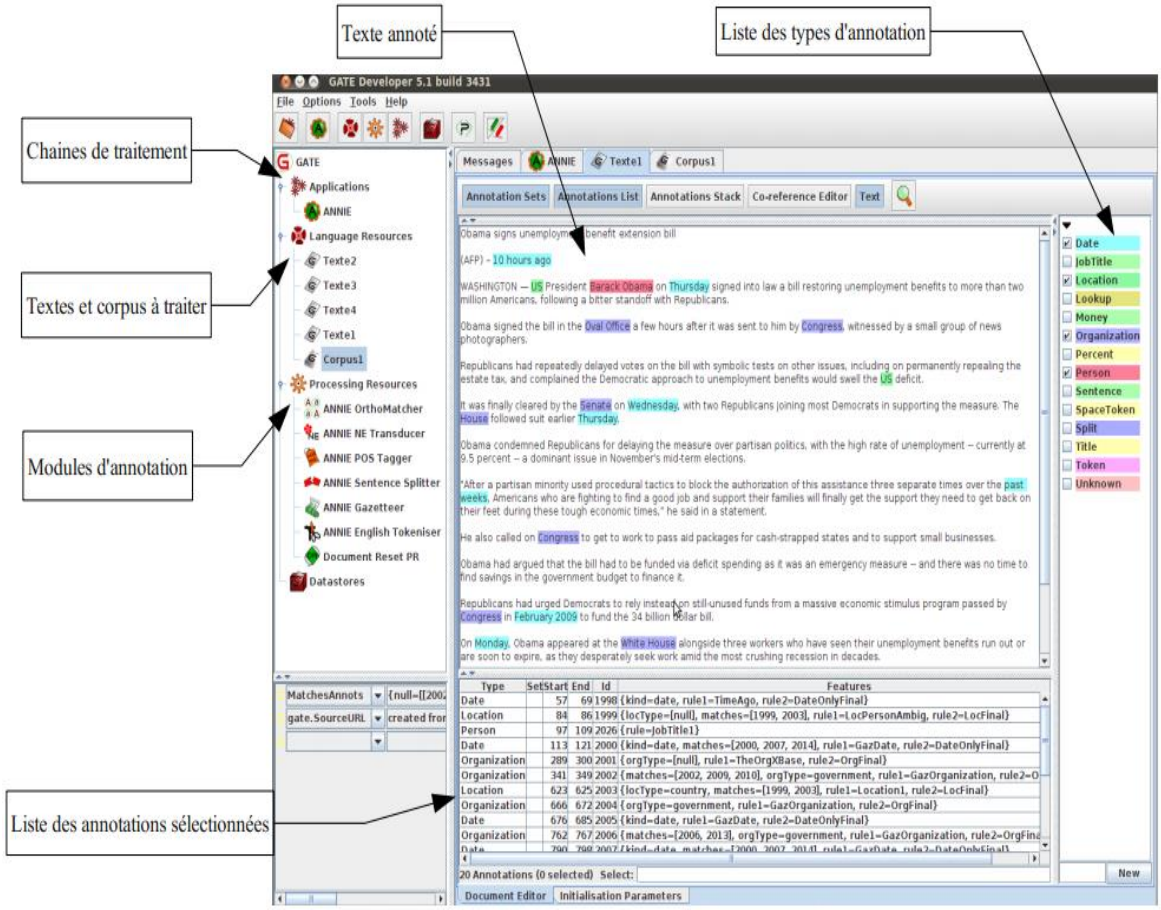


***Fig II.5** : Exemple de texte annoté dans GATE*

II.7.3 Quelques plugins intéressants

Par un système de plugins, GATE met à disposition de ses utilisateurs un grand nombre de modules dédiés à l'analyse textuelle. Les plus courants sont les segmenteurs (Tokenizers), les étiqueteurs morpho-syntaxiques (Part Of Speech Taggers), les lexiques (Gazetteers) ou encore les transducteurs (JAPE18transducers.

Dans cette partie, nous présentons différents plugins disponibles dans GATE qui nous ont été utiles pour réaliser les différentes tâches d'extraction.

Tout d'abord, nous devons parler d'ANNIE (A Nearly-New Information Extraction system), chaine complète dédiée à l'extraction d'information et plus particulièrement à l'extraction d'entités nommées pour l'anglais. Ce plugin fourni par les créateurs de GATE est basé sur le langage JAPE et comprend différents modules d'annotation :

• Document Reset : supprime toutes les annotations précédentes mises sur le document.

• English Tokenizer : découpe en mots (« tokens ») un texte anglais.

• Gazetteer : cherche les éléments d'une liste (appelée aussi « gazetteer ») dans le texte et les annote en tant que « Lookup ».

• Sentence Splitter : découpe le texte en phrases.

• POS Tagger : ajoute à l'annotation Token mise par le « tokenizer » une propriété « category » indiquant la catégorie morpho-syntaxique du mot en question.

• NE Transducer : transducteur JAPE permettant de repérer un certain nombre d'entités nommées (Person, Organization, Location, Date, URL, Phone, Mail, Adress, etc.).
• OrthoMatcher : ajoute des relations d'identité référentielle entre les entités nommées annotées précédemment.
Pour finir, précisons que la manipulation de larges corpus dans l'environnement GATE nous a été facilitée par le système de « datastore ». Celui-ci permet, en optimisant la gestion de la mémoire, de manipuler facilement et rapidement une grande quantité de textes mais aussi de sauvegarder les modifications éventuelles faites sur ces données. Un « datastore » peut être créé à partir d'un dossier existant mais l'on peut également y stocké soit un corpus soit un texte seul depuis l'interface GATE. L'ensemble des textes contenus dans le « datastore » pourront ensuite être ouverts simultanément en un simple clic.

II.8 Conclusion

Dans ce chapitre nous avons décrit la notion d'entité nommée, la typologie et les différentes méthodes et outils d'extraction de ces entités nommées. Dans notre cas nous nous sommes limité à l'extraction des entités de type « Person », « Organization » et « Location », dans des textes écrits en anglais dans la collection « TIME » pour les utiliser comme des descripteurs, pour améliorer la pertinence de la RI. Dans le chapitre suivant nous décrivons l'approche d'intégration des entités nommées dans le processus de la RI et de pallier au problème d'ambigüité lié aux entités nommées (EN>=1).

Approche et implémentation

III.1 Introduction

Le modèle de langue est un nouveau cadre probabiliste pour la description du processus de la RI. Parmi les propriétés de ce modèle est son fondement mathématique solide. Cependant, comme la majorité des modèles de RI classique, le modèle de langue se base sur l'hypothèse d'indépendance entre termes. Une telle hypothèse pose le problème d'ambigüité des termes. [Hammache et al, 13]

Pour y'aller au-delà de cette hypothèse, deux catégories d'approches ont été investies. La première se base sur l'utilisation de la notion de proximité entre termes (Tao,07 ; Lv,09 ; Zhao,09). La seconde considère l'utilisation de la dépendance entre termes (Miller,99 ; Song,99 ; Srikanth,02 ; Metzler,05).

III.2 Approche proposée

Notre approche s'inscrit dans cette dernière catégorie. Nous proposons dans notre approche d'estimer le modèle de document des entités nommées (voir termes simples) comme une mixture des deux modèles : le modèle de document des termes simples et le modèle de document des entités nommées.

L'approche proposée est applicable sur les termes composés, nous l'avons adapté sur les entités nommées ayant une taille égale à 2. Pour les entités nommées ayant une taille égale à 1, on a utilisé la fréquence initiale.

III.3 Un modèle de langue mixte pour la RI

L'objectif de notre approche est de mieux représenter le contenu sémantique des documents et des requêtes en introduisant une certaine sémantique dans leurs représentations. À cette fin, nous proposons dans ce chapitre un modèle de langue mixte (LM_EN) pour la recherche d'information qui combine les termes simples et les entités nommées.

Une des contributions de ce modèle est la manière dont le modèle de langue des entités nommées est estimé. Plus précisément, dans la plupart des travaux existants sur les modèles de langue, un modèle de document est estimé par le comptage, soit des termes simples ou des entités nommées. Nous pensons, que le simple comptage des n-grammes est susceptible de biaiser l'importance réelle (poids) des entités nommées (n-grammes) dans les documents. En effet, deux intuitions ont guidé notre réflexion, nous pensons d'abord que les mots composants d'une entité nommée n'apportent pas la même contribution dans le poids final d'entités nommées. Nous introduisons à cet effet la notion de dominance de mot composant dans une entité nommée. Nous considérons que les mots composants dominants contribuent plus que les autres mots. Deuxièmement, nous supposant que l'auteur d'un document utilise les mots composants isolément pour exprimer l'entité nommée comme abréviation après un nombre d'occurrences d'entités nommées. Nous proposons alors une nouvelle formule de calcul de fréquence des entités nommées qui prend en compte aussi la fréquence de ses termes composants.

Enfin, nous pensons que, la prise en compte de tous les n-grammes (n> 1), (n-grammes est composé de *n* mots non vide adjacents) peut introduire du bruit. Nous proposons de considérer que les n-grammes, à savoir "les entités nommées ", qui sont fréquents dans la collection.

III.4 Description du modèle

Nous considérons une requête $\mathbf{Q}$ et un document $\boldsymbol{D}$ représentés dans le vocabulaire $\boldsymbol{V} = \{\mathbf{T_1}, \dots, \mathbf{T_m}, \mathbf{t_1}, \dots, \mathbf{t_n}\}$ composé de termes simples $\mathbf{t_i}$ et d'entités nommées$\mathbf{T_j}$**.** Une entité nommée peut être formé de deux ou plusieurs termes simples adjacents non vides. Elles sont extraites des documents et utilisées comme unités d'indexation supplémentaires.

En suivant la logique du modèle de langue et en considérant que le contenu d'un document comporte à la fois des mots simples et des entités nommées, on peut considérer ainsi, que le document a été généré par un modèle mixte. Chacun produisant un type de terme.

Nous supposons donc que le modèle de document peut être estimé à l'aide de deux modèles : un modèle des termes simples ($\mathbf{M_{D_t}}$) et un modèle des entités nommées ($\mathbf{M_{D_T}}$). Ainsi, étant donné une requête $\boldsymbol{Q}$, exprimée par des termes simples et des entités nommées, le modèle d'appariement document/requête que nous proposons combine les deux modèles de la manière suivante :

$$\mathbf{P(Q|D)} = \prod_{\mathbf{t_i} \in \mathbf{Q}} \mathbf{P(t_i|D)} \times \prod_{\mathbf{T_j} \in \mathbf{Q}} \mathbf{P\left(T_j|D\right)} \qquad \textbf{(III.1)}$$

Chaque terme simple et entité nommée dans la requête est estimé en combinant les deux modèles de documents. Formellement, on l'exprime comme suit :

$$\mathbf{P(t_i|D)} = \boldsymbol{\lambda}\mathbf{P\left(t_i\middle|M_{D_T}\right)} + \mathbf{(1-\lambda)P\left(t_i|M_{D_t}\right)} \qquad \textbf{(III.2)}$$

$$\mathbf{P\left(T_j|D\right)} = \boldsymbol{\alpha}\mathbf{P\left(T_j|M_{D_T}\right)} + \mathbf{(1-\alpha)} \prod_{\mathbf{t_k} \in \mathbf{T_j}} \mathbf{P\left(t_k|M_{D_t}\right)} \qquad \textbf{(III.3)}$$

Où $\boldsymbol{\lambda}$ et $\boldsymbol{\alpha}$ $\boldsymbol{\epsilon}$ **[0, 1]** sont des paramètres de lissage, $\mathbf{P\left(T_j|M_{D_T}\right)}$ et $\mathbf{P\left(t_i|M_{D_t}\right)}$ peuvent être évaluées en utilisant n'importe quel modèle de langue uni-gramme. Nous avons pour notre part opté pour le lissage de Dirichlet, car il a donné de meilleurs résultats que les autres méthodes de lissage. Les deux modèles sont estimés comme suit [Zhai,02] :

$$\mathbf{P_{Dir}\left(t_i\middle|M_{D_t}\right)} = \frac{\mathbf{F(t_i,D_t)+\mu P(t_i|C_t)}}{\mathbf{|D_t|+\mu}} \qquad \textbf{(III.4)}$$

Où $\mathbf{F(t_i, D_t)}$ est la fréquence du terme simple $\mathbf{t_i}$ dans le document $\boldsymbol{D}$, $\mathbf{P(t_i|C_t)}$ est le modèle de langue de la collection (la fréquence globale du terme est utilisée), $\mathbf{|D_t|}$ est la longueur du document exprimée avec des termes simples, $\boldsymbol{\mu}$ est le paramètre de lissage.

De la même manière on a :

$$\mathbf{P_{Dir}\left(T_j\middle|M_{D_T}\right)} = \frac{\mathbf{F\left(T_j,D_T\right)+\mu P(T_j|C_T)}}{\mathbf{|D_T|+\mu}} \qquad \textbf{(III.5)}$$

Où $\mathbf{P(T_j|C_T)}$ est le modèle de langue de la collection, $\mathbf{F\left(T_j, D_T\right)}$ est la fréquence de l'entité nommée $\mathbf{T_j}$ dans le document $\boldsymbol{D}$, elle peut être calculée par un simple comptage des

occurrences d'entité nommée ou en utilisant notre méthode de comptage décrite dans la section III.6, et $|\mathbf{D_T}|$ est la longueur du document représentée par des entités nommées.
Nous détaillons dans les sections suivantes comment la fréquence de l'entité nommée et la probabilité $\mathbf{P}(\mathbf{t_i}|\mathbf{M_{D_T}})$ sont calculées. Nous introduisons tout d'abord dans la section suivante la notion de dominance d'un terme.

III.5 Domination d'un terme

La pondération des entités nommées demeure un problème ouvert en RI. En effet il n'existe pas de schéma bien accepté pour la pondération des entités nommées. La manière la plus simple de pondération des entités nommées est l'utilisation du facteur de pondération $\boldsymbol{tf}$ (en comptant le nombre d'occurrences d'une entité nommée dans un document). Des alternatives ont été proposées ; parmi elles l'adaptation du schéma de pondération $\boldsymbol{tf.idf}$ (Croft,91 ; Huang,01), mais aucune amélioration notable n'est constatée.
Cependant, ces schémas de pondération ne tiennent pas compte d'un facteur important qui est l'importance des termes composants dans l'entité nommée ; dans les schémas précédents cette importance est considérée identique. Or, dans la réalité un des termes composants peut être plus important que les autres termes, nous les nommons les termes dominants. Par exemple, le terme *«**Germany**»* est plus important que le terme *«**West**»* dans l'entité nommée *«**West Germany**»*.

Nous considérons intuitivement que la dominance d'un terme est déterminée par sa spécificité, nous proposons de l'estimer de la manière suivante :

$$\boldsymbol{imp(t) = {}^{N}/_{df}} \qquad \textbf{(III.6)}$$

Où $\boldsymbol{df}$ est le nombre de documents où le terme $\boldsymbol{t}$ apparaît, et $\boldsymbol{N}$ le nombre de documents dans la collection TIME.

Nous attribuons à chaque terme simple $\boldsymbol{t}$ sa probabilité de dominance dans son entité nommée $\boldsymbol{T}$ comme suit :

$$\boldsymbol{P(t|T) = \frac{imp(t)}{\Sigma_{t_i \in T}\left(imp(t_i)\right)}} \qquad \textbf{(III.7)}$$

III.6 La fréquence des entités nommées revisitée

Notre deuxième intuition dans la pondération des entités nommées est la suivante : « Nous supposons que l'auteur d'un document utilise les composants d'une entité nommée isolément pour faire référence à cette entité nommée comme abréviation après un certain nombre d'occurrences d'entité nommée». Par exemple un document contenant l'entité nommée *«John Kennedy»*, l'auteur utilise le terme *«Kennedy»* simplement pour désigner l'entité nommée *«John Kennedy»*. Afin de prendre en compte cette hypothèse, nous proposons de lisser la fréquence d'entité nommée en tenant compte de la fréquence de ses

termes composants relativement à leur dominance dans l'entité nommée. La nouvelle fréquence d'une entité nommée est exprimée alors comme suit :

$$\mathbf{F^n(T) = F(T) + \sum_{i=1}^{\#T} P(t_i|T) \times F(t_i)} \quad \textbf{(III.8)}$$

Où $\mathbf{F^n(T)}$ représente la nouvelle fréquence (revisitée) de l'entité nommée **T**, $\mathbf{F(T)}$ est la fréquence initiale de l'entité nommée **T**, $\mathbf{P(t_i|T)}$ est la probabilité de dominance du terme $\mathbf{t_i}$ dans l'entité nommée **T**, $\mathbf{F(t_i)}$ est la fréquence du terme $\mathbf{t_i}$ dans le document, et $\mathbf{\#T}$ est la taille de l'entité nommée **T** (nombre de termes).

Pour illustrer cette seconde intuition, nous avons pris dans la collection TIME un ensemble d'entités nommées et nous avons examiné la validité de cette hypothèse dans tous les documents de cette collection où ces entités apparaissent. La table suivante représente les résultats obtenus. Nous avons adopté la notation suivante : un «+ » pour designer que l'hypothèse est vérifiée un «-» pour signifier que l'hypothèse n'est pas vérifiée.

Entités nommées	**Documents**	**Termes simples seuls**	**Vérifiée**
CAPE TOWN	*TIME-110*	*Cape, town*	+ +
	TIME-254	*Cape, town*	+ +
	TIME-442	*Cape, town*	+ -
BRITAIN HAD	*TIME-442*	*Britain, had*	+ +
	TIME-442	*Britain, had*	+ +
CONGRESS PARTY	*TIME-304*	*Congress, party*	+ +
	TIME-413	*Congress, party*	+ +
	TIME-424	*Congress, party*	- +
	TIME-501	*Congress, party*	+ +
	TIME-503	*Congress, party*	+ +

Tableau III.1 Exemple de termes simples référençant les entités nommées

Nous pouvons noter dans les exemples ci-dessus que notre hypothèse est satisfaite dans certains cas. Pour vérifier cette hypothèse automatiquement une analyse syntaxique et sémantique doit être réalisée.

Pour illustrer la méthode de pondération des entités nommées, nous prenons l'exemple suivant :

Soit un document D=" *TIME-501*", une entité nommée T=*" congress party "*, qui contient les deux termes simples suivant t_1 = "*Congress*" et t_2 = "*Party*". Les statistiques sur ces termes sont les suivantes :

F(*T*) = 3 : Fréquence initiale de l'entité nommée dans le document *D*.

F(t_1) = 1: Fréquence de terme t_1 apparaissant seul dans le document *D*.
F(t_2) = 3 : Fréquence de terme t_2 apparaissant seul dans le document *D*.
df(t_1)=21 : Nombre de documents dans la collection contenant le terme t_1 .
df(t_2)=153 : Nombre de documents dans la collection contenant le terme t_2.
P(*t1*|*T*)=(153)/(153+21)= 0.87 : Probabilité de dominance de terme t_1 dans *T*.
P(*t2*|*T*)=(21)/(153+21)= 0.12 : Probabilité de dominance de terme t_2 dans *T*

La fréquence revisitée de l'entité nommée " congress party " dans le document "TIME-501" est calculée de la manière suivante :

$$F^n(T) = 3 + 1 * 0.87 + 3 * 0.12 = 4{,}23$$

Ainsi, la fréquence revisitée de l'entité nommée « *congress party »* dans le document "*TIME-501* " est égale à 4.23, alors que sa fréquence calculée par simple comptage du nombre d'occurrence est de 3.

III.7 Estimation de la probabilité $\mathbf{P(t_i|M_{D_T})}$

Afin d'estimer cette probabilité, nous proposons un modèle similaire au modèle de traduction (Berger,99). Par conséquent, nous exprimons cette probabilité comme suit :

$$\mathbf{P(t_i|M_{D_T}) = \sum_T (P(t_i|T) \times P_{Dir}(T|M_{D_T}))} \quad \textbf{(III.9)}$$

Dans cette formule, le passage d'un terme simple vers un document ***D*** est réalisé à travers toutes les entités nommées contenant le terme simple.

Cependant, comme nous l'avons mentionné précédemment, nous supposons que, l'auteur lorsqu'il utilise un terme simple dans un document, il ne peut renvoyer qu'à une entité nommée donnée. Nous considérons que cette entité nommée est la plus fréquente qui contient ce terme simple dans le document. Cette entité nommée noté $\widehat{\boldsymbol{T}}$ est sélectionnée par la formule suivante :

$$\widehat{\boldsymbol{T}} = \boldsymbol{argmax}_{\forall\, T \in D_T \wedge t_i \in T} \left(\mathbf{P(t_i|T) \times P_{Dir}(T|M_{D_T})}\right) \quad \textbf{(III.10)}$$

En introduisant le facteur $\widehat{\boldsymbol{T}}$ dans la formule (9), elle peut être réécrite de la manière suivante :

$$\mathbf{P(t_i|M_{D_T})} = \boldsymbol{P}(\mathbf{t_i}|\widehat{\boldsymbol{T}}) \times \boldsymbol{P_{Dir}}(\widehat{\boldsymbol{T}}|\mathbf{M_{D_T}}) \quad \textbf{(III.11)}$$

III.8 Présentation de l'environnement

III.8.1 Langage de programmation utilisé

Langage java [Delannoy,09]

Java est un langage de programmation orienté objet mis en point par la firme SUN Microsystems. Il est caractérisé par les points suivants :

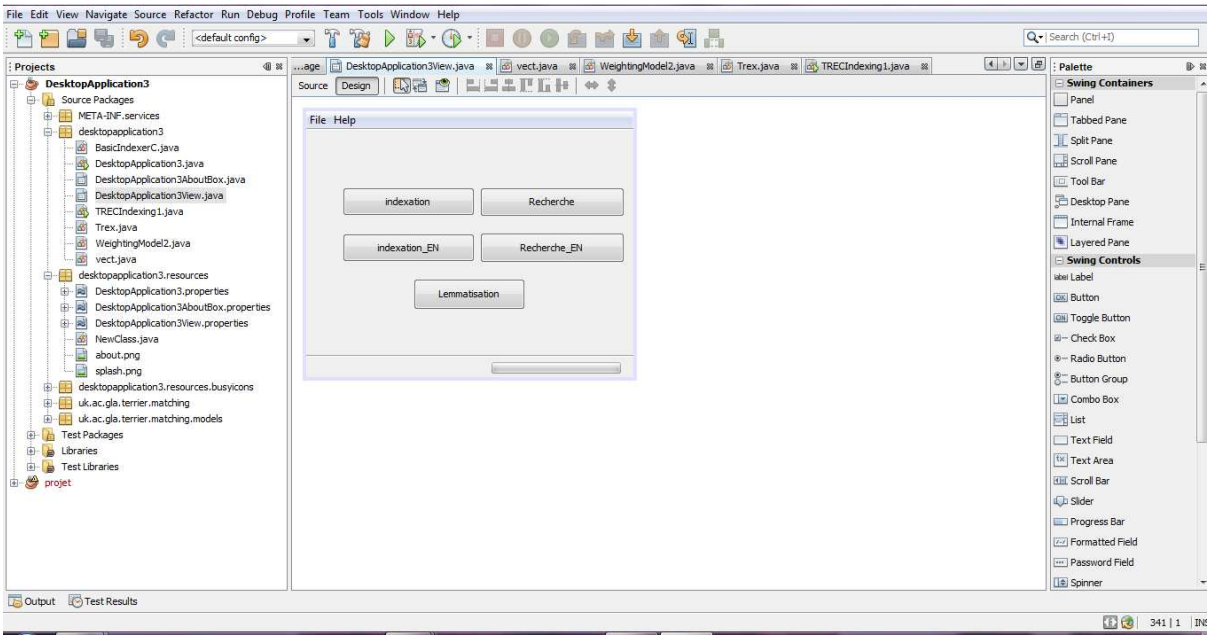
- Java est indépendante de toute plate forme : Une application en java fonctionne sur n'importe quel environnement (Unix, Windows,…) disposant d'une JVM (Java Virtual Machine). (La machine virtuelle java)
- Java est un langage orienté objet : Dans ce type de programmation, on manipule des objets qui s'échangent des messages, non pas des procédures et fonctions.
- Java est extensible à l'infini : java est écrit en java. Idéalement, toutes les catégories d'objets (appelées classes) existant en java sont définies par extension d'autres classes, en partant de la classe de base la plus générale : la classe objet. Pour étendre le langage il suffit donc de développer de nouvelles classes.
- Java est un langage de haute sécurité : java a été développé dans un souci de sécurité maximale. L'idée maîtresse est qu'un programme comportant des erreurs ne doit pas pouvoir être compilé. Ainsi les erreurs ne risquent pas d'échapper au programmeur et de passer les procédures de tests. En détectant les erreurs à la source, on évite qu'elles se propagent en s'amplifiant.
- Java est un langage compilé : java est un langage compilé, c'est-à-dire qu'avant d'être exécuté, il doit être traduit dans le langage de la machine sur laquelle il doit fonctionner. Cependant, contrairement à de nombreux compilateurs, java traduit le code source dans le langage d'une machine virtuelle appelée JVM (Java Virtual Machine). Le code traduit appelé bytecode, ne peut pas être exécuté directement par le processeur d'une machine.
- Java doté en standard de bibliothèques de classes : Il est doté en standard de bibliothèque de classes très riches comprenant la gestion des interfaces graphiques (fenêtres, boites de dialogue, contrôles, menus, graphisme), la programmation multi_threads (multitâches), la gestion des exceptions, les accès aux fichiers et au réseau. L'utilisation de ces bibliothèques facilite grandement la tâche du programmeur lors de la construction d'applications complexes. [Lemy et al, 97]

III.8.2 Présentation de NetBeans

NetBeans est un projet open source fondé par Sun Microsystems. L'IDE NetBeans est un environnement de développement permettant d'écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'IDE NetBeans.
L'IDE NetBeans est un produit gratuit, sans aucune restriction quant à son usage.

L'installation de l'IDE NetBeans nécessite l'installation de la JDK (Java Developpement Kit) le kit de développement java compatible avec la version d'IDE. [Mounier]

Pour concevoir notre système nous avons utilisé la version NetBeans **6.7.1.**
L'interface de notre application est donnée dans la figure suivante FigIII.1.



FigIII.1 : l'interface de notre application

III.8.3 La plateforme terrier

Terrier est une plate-forme dédiée à la recherche d'information. Elle implémente les différents modules intervenant dans le processus de RI classique et offre en plus un cadre pour l'évaluation des résultats de recherche pour différentes applications. Terrier a été largement éprouvée (Middleton et al,07). Le choix de cette plate-forme est dû aussi à sa capacité à traiter de grandes collections de documents telles que les collections TREC.
Terrier est un logiciel open source écrit en java et permet :

- **L'indexation classique :** extraction des mots clés des différents documents appartenant à une collection et les stocker dans un index.
- **Recherche des documents :** permet de générer des résultats aux requêtes formulées par l'utilisateur.
- **Évaluation des résultats de la recherche** : permet de mesurer le degré de pertinence des résultats de la recherche aux requêtes formulées par l'utilisateur.[Ounis et al,06]

III.8.3.1 Processus d'indexation de terrier

L'indexation dans terrier est divisée en quatre procédures et à chaque procédure, des classes java peuvent être ajoutées pour la personnalisation du système. Les quatre procédure sont :

1. Splitter la collection de document : consiste à parcourir l'ensemble du corpus reçu en entrée par terrier et envoyer chaque document à l'étape suivante.

2. Extraction des termes (Tokenize document) : qui consiste à parser chaque document reçu et en extraire les différents termes.
3. Traitement des termes extrait avec TermPiplines : consiste en l'élimination des mots vides et la lemmatisation des termes.
4. La construction de l'index.

Les différentes classes associées au processus d'indexation sont organisées dans un ensemble de package dont on trouve :

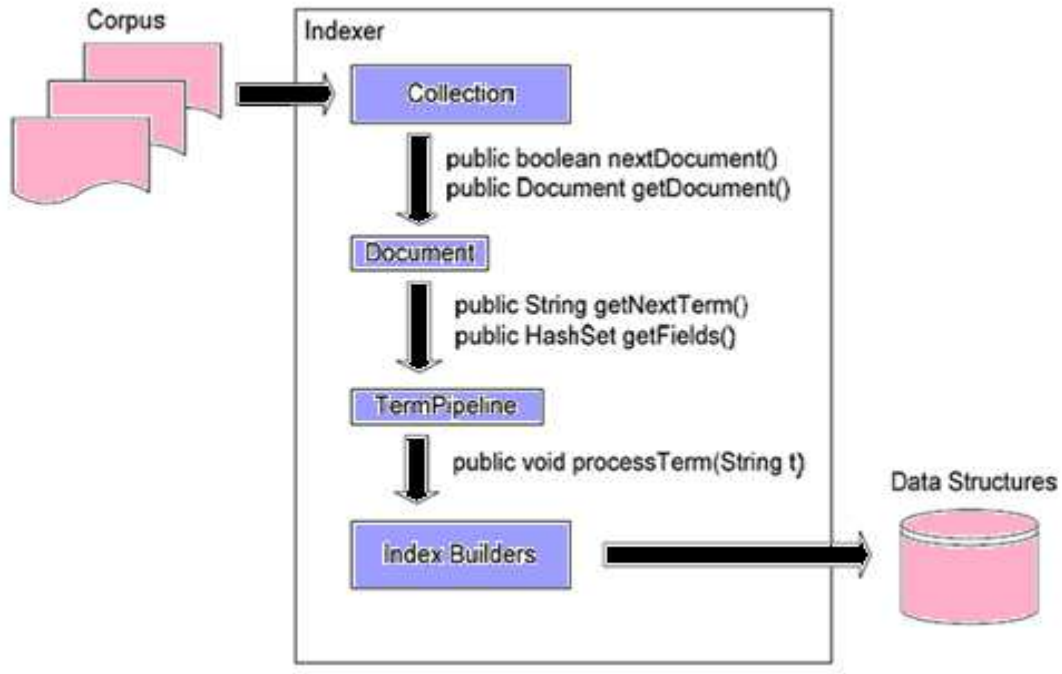
Org.terrier.indexing : ce package contient les différentes classes permettant de réaliser un ensemble d'opérations sur la collection des documents, dans le but d'extraire les termes de tous les documents de la collection.

Org.terrier.terms : Les classes qui se trouve dans ce package permettent d'effectuer un ensemble de traitement sur les termes extraits. Parmi ces traitements l'élimination des mots vides, lemmatisation des termes, …etc.

Org.terrier.structures : Les classes de ce package permettent la construction d'un ensemble de structures ou un ensemble de données est stocké, parmi ces structure on a : [Ounis et al,06]

- ✓ Lexicon : qui stocke les informations de chaque terme de la collection ;
- ✓ Inverted index : ou le fichier inverse est stocké ;
- ✓ Document Index : qui contint des informations sur les différents documents de la collection ;
- ✓ …etc.

La figure ci –dessous donne un aperçu de l'indexation des principaux composants impliqués dans le processus d'indexation.



FigIII.2 : processus d'indexation de terrier

III.8.3.2 Le processus de recherche de Terrier

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

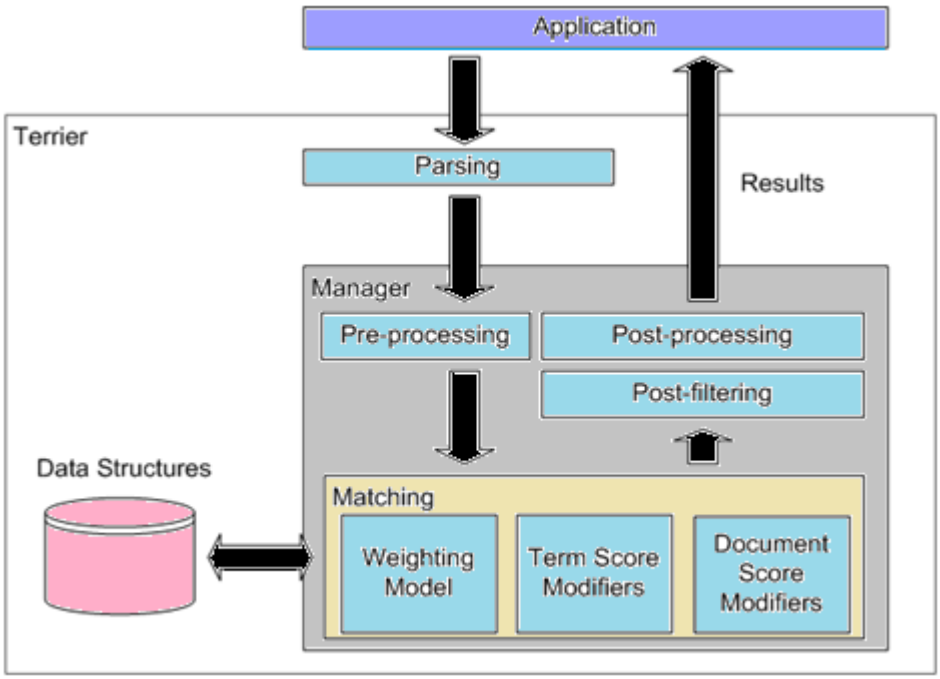
1. Parsing : qui se charge de tockenizer la requête.
2. Pré-processing : qui applique le TermePipeline à la requête. Elimine les mots vides et les lemmatise.
3. Matching : qui est responsable de l'initialisation du WeightingModel et du calcul des scores entres la requête et les documents.
4. Post-filtrage : va filtrer les résultats.
5. Post-traitement : peut modifier le résultatSet, par exemple, par un procédé QueryExpansion afin de générer un meilleur classement de documents. [Ounis et al,06]

Parmi les packages ou se trouve les classes associées au processus de recherche on a :

Org.terrier.Matching : contient les classes qui permettent de déterminer les documents répondant à la requête.

Org.terrier.matching.models : on trouve dans ce package les différents modèles de pondération dont TF-IDF, BM25, …etc. [Ounis et al,06]

La figure suivante donne un aperçu de l'interaction des composants terrier dans la phase de recherche.



FigIII.3 : processus de recherche de terrier

III.8.3.3 Utilisation de terrier : « TREC Terrier »

L'utilisation de TREC Terrier pour l'indexation, la recherche et l'évaluation sur le système d'exploitation Windows 7. Sont faites comme suit [Ounis et al,06] :

– **Indexation**

Pour indexer des documents avec terrier on peut utiliser cette méthode. Dans l'invite de commande on :

- Spécifier le chemin vers terrier : cd<le chemin vers terrier>\bin
- Spécifier la collection de document à indexer : trec_setup <le chemin de la collection à indexer>.Si on ouvre le fichier collection.spec on s'aperçoit que terrier à mis dans ce dernier tous les chemins vers les fichiers contenus dans le répertoire donné automatiquement.
- L'indexation est lancée par la commande suivante : « trec_terrier –i ».

S'il n'y a pas d'erreurs, on trouvera dans var\ index les fichiers résultants de l'indexation.

– **Recherche**

Pour faire une recherche sur une collection de document déjà indexés on utilise la commande « trec_terrier –r ». Mais avons, on doit :

- Donner à terrier les requête et ce en modifient etc\terrier.properties comme suit : Trec_topics = « le chemin vers les requêtes ».
- Spécifier le modèle de pondération à utiliser dans le fichier etc\trec.models.

Dans notre cas nous avons utilisé le modèle Dirich dans la recherche classique et le modèle model_TC dans la recherche des entités nommées.

- Les résultats de la recherche sont stockés dans : « var\result\nom_fonction_pondération.res ».

– **Évaluation**

Pour faire une évaluation sur une collection de document déjà indexée on utilise la commande « trec_terrier –e ».

III.8.4 Collection TIME

Nom :	TIME
Collection portant sur :	Divers domaines
Nombre de documents :	425
Nombre de requêtes :	81
Volume de la collection :	1,46 Mo

Tableau III.2 : description de la collection TIME.

TIME est composée de document au nombre de 425 tirés du magazine TIME de l'année 1963 et d'un nombre de requêtes raisonnable qui est de 81 accompagné d'un fichier de jugement de pertinence.

La collection TIME est une collection portant sur des domaines différents. Elle se caractérise par sa haute efficacité de recherche par rapport aux autres collections typiques.

Un document est identifié comme suit : /Numéro du document/, par exemple : « 18.txt ».
Les documents et les requêtes sont composés de textes simples.

Exemple de document (18.txt) :

<DOC>
<DOCNO>18.txt</DOCNO>
<TEXT>
RUSSIA WHO'S IN CHARGE HERE? IT WAS IN 1954 THAT NIKITA
KHRUSHCHEV LAUNCHED HIS GRANDIOSE «VIRGIN LANDS «GAMBLE. PART OF THE PLAN WAS TO PLOW UP 32 MILLION ACRES OF MARGINAL LAND IN KAZAKHSTAN, AND SETTLE IT WITH COMMUNIST «PIONEERS, «WHO WERE TO PLANT AND PRODUCE HUGE QUANTITIES OF DESPERATELY NEEDED GRAIN WITHIN TWO YEARS .NIKITA'S SCHEME FLOPPED. THERE WAS NOT ENOUGH RAINFALL, AND THE PIONEERS DID NOT TAKE TO TRACTOR LIFE ON THE BLEAK FRONTIER. EXCEPT
FOR 1958, EACH HARVEST HAS BEEN LOWER THAN THE PREVIOUS YEAR'S. WORST YEAR OF ALL WAS 1962, WHEN THE VIRGIN LANDS DELIVERED ONLY HALF THEIR QUOTAS. NATURALLY, KHRUSHCHEV TAKES NONE OF THE BLAME FOR THE FIASCO. THREE YEARS AGO HE FOUND A SCAPEGOAT IN KAZAKHSTAN PARTY BOSS NIKOLAI BELYAEV, FIRED HIM FOR HIS «ERRORS. » LAST WEEK BELYAEV'S SUCCESSOR, DINMUKHAMED KUNAEV, WAS SIMILARLY BOUNCED FOR «LAPSES «IN HIS WORK . FOR GOOD MEASURE, MOSCOW ALSO PURGED THE FORMER PREMIER OF THE TERRITORY FROM THE LOCAL PARTY'S CENTRAL COMMITTEE. IT WAS PERHAPS NO COINCIDENCE THAT NIKOLAI IGNATOV, 61, A ONETIME KHRUSHCHEV CRONY, LAST WEEK ABRUPTLY LEFT HIS POST AS A SOVIET DEPUTY PREMIER AFTER ONLY NINE MONTHS ON THE JOB. FARM EXPERT IGNATOV HAD THE MISFORTUNE TO BE BOSS OF A SPECIAL COMMITTEE TO BOOST FOOD PRODUCTION.
</TEXT>

Exemple de requête :

<top>
<num> Number: 2
<title>
EFFORTS OF AMBASSADOR HENRY CABOT LODGE TO GET VIET NAM'S PRESIDENT DIEM TO CHANGE HIS POLICIES OF POLITICAL REPRESSION.
</top>

Des jugements de pertinence sont associés aux requêtes selon le format suivant :
« Numéro de la requête/0 /Numéro du document/1 »

1 0 268.txt 1
1 0 288.txt 1
1 0 304.txt 1
2 0 326.txt 1
2 0 334.txt 1
3 0 326.txt 1
3 0 385.txt 1
4 0 370.txt 1

Pour l'évaluation des performances de notre modèle, nous avons utilisé la mesure de la précision moyenne, MAP (Mean Average Precision), qui est une mesure largement utilisée pour l'évaluation de l'efficacité des Systèmes de Recherche d'Information.
La précision moyenne donne une vue globale des performances d'un modèle de recherche d'information au travers d'un ensemble de requêtes.

III.9 Implantation de l'approche

Nous décrivons dans ce qui suit les différentes étapes que nous avons suivis pour l'indexation et l'évaluation des requêtes.

1- **Le prétraitement de la collection** : en premier lieu, nous procédons au prétraitement de la collection. Nous parsons les documents ; nous éliminons les mots vides et la lemmatisation des termes.
2- **L'extraction des entités nommées** : pour l'extraction des entités nommées nous avons utilisé l'outil GATE. . L'outil repose sur le principe d'une chaine de traitement « pipeline » composée de plusieurs modules dits « Processing Resources » appliqués successivement sur un ou plusieurs textes dits « Language Resources ».
3- **Indexation, recherche et évaluation** : nous avons utilisé la plate-forme de RI Terrier 2.1 pour l'indexation, la recherche et l'évaluation de notre approche. Les documents sont indexés en utilisant les entités nommées reconnus dans l'étape 2 et les mots simples utilisés traditionnellement en RI.

III.9.1 Architecture de notre approche

Les étapes de notre approche sont structurées et illustrées dans la figure ci-dessous :

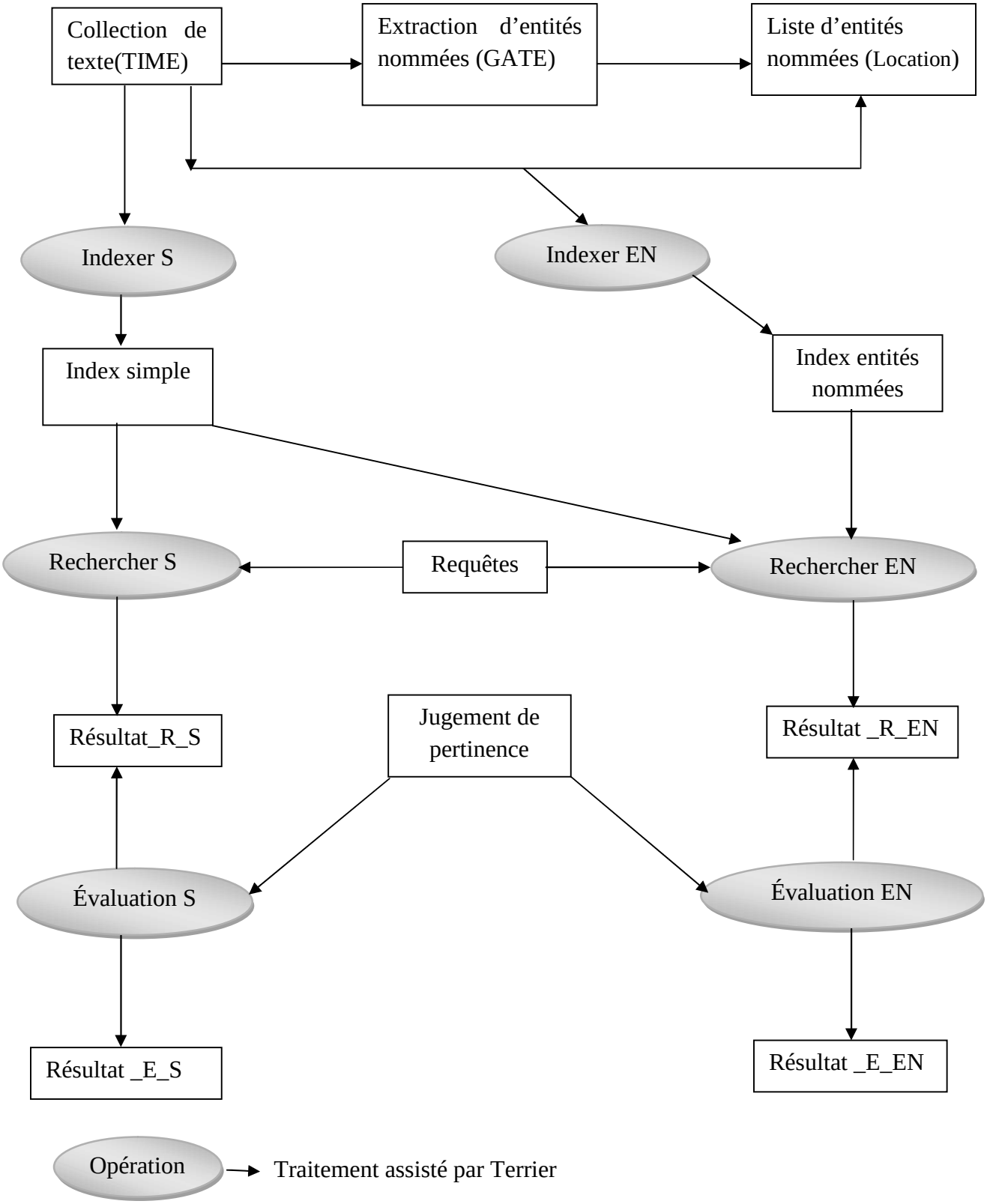


Fig. III.4 : Architecture de notre approche

III.10 Évaluation

Dans cette partie nous présentons l'évaluation des performances de notre modèle. Nous avons procédé de la manière suivante :

III.10.1 Résultats d'évaluation

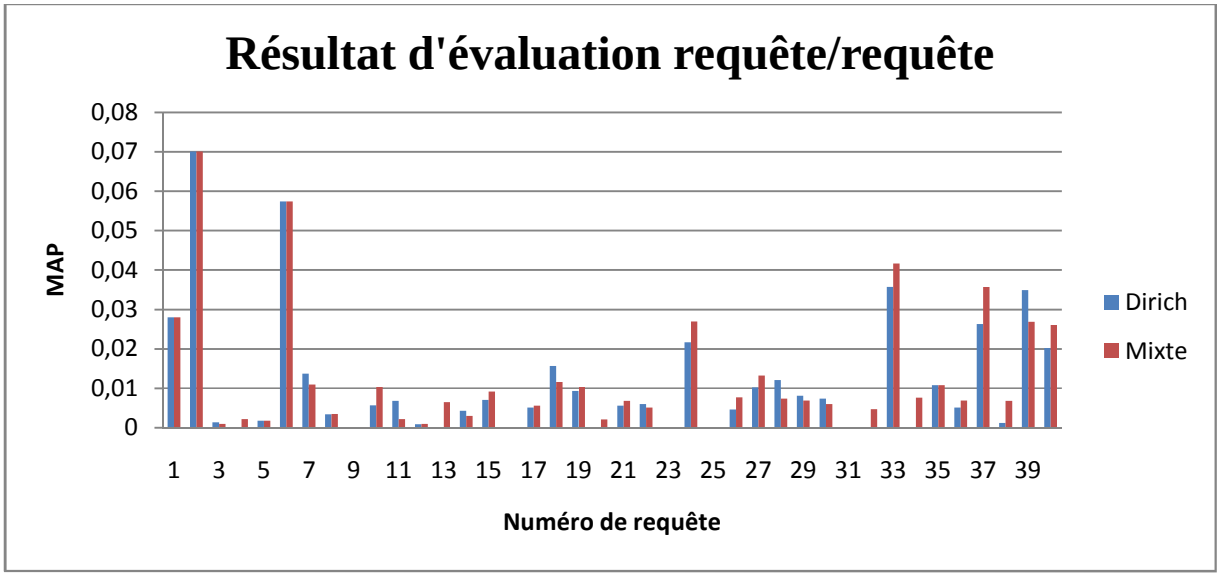
Information	Recherche avec termes simple	Recherche avec entités nommées (location)	Taux d'amélioration
Nombre de documents pertinents	321	321	
Nombre de documents pertinents sélectionnés	117	153	
Précision Moyenne	0.0087	0.0093	+ 6.89 %
Rappel	0.3644	0.4766	+ 30.79 %

Tableau III.3 : Résultats d'évaluation

Après avoir exécuter la recherche avec les termes simples, et la recherche avec les entités nommées, nous avons remarqué que la précision moyenne et le rappel ont augmenté (amélioré) dans la recherche avec entités nommées, et ça est dû à l'augmentation du nombre de documents pertinents sélectionnés dans la recherche.

III.10.2 Résultat d'évaluation requête par requête :

Pour avoir une vision et une analyse plus claire et plus profonde on a effectué l'évaluation requête par requête dans la collection TIME, le graphe suivant illustre cette analyse :



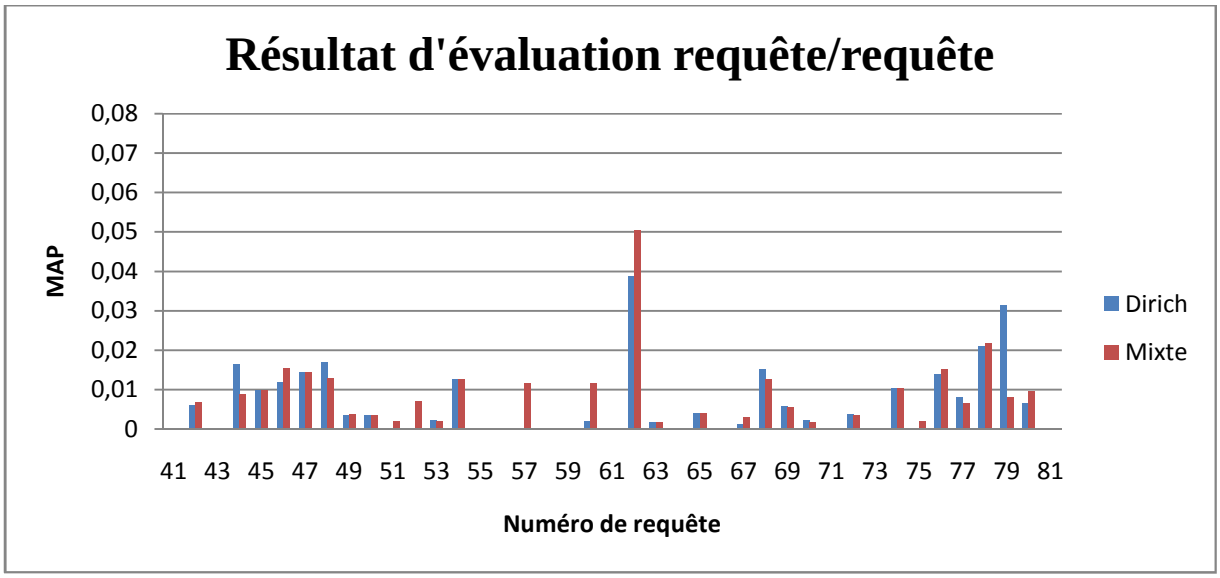


Fig. III.5 : Graphe représentant les résultats d'évaluation requête par requête pour les deux recherches avec la collection TIME

Dirich : recherche termes simples
Mixte : recherche modèle mixte (EN+TS).

La comparaison entre les résultats d'évaluation requête par requête pour les deux recherches sont représentés dans le tableau suivant :

Requête	Recherche termes Simples	Recherche modèle mixte	Améliorations
1	0,028	0,028	0
2	0,0701	0,0701	0
3	0,0014	0,001	-
4	0	0,0022	+
5	0,0018	0,0018	0
6	0,0574	0,0574	0
7	0,0137	0,011	-
8	0,0034	0,0035	+
9	0	0	0
10	0,0057	0,0104	+
11	0,0068	0,0022	-
12	0,0009	0,001	+
13	0	0,0065	+
14	0,0043	0,003	-
15	0,0071	0,0092	+
16	0	0	0
18	0,0051	0,0056	+
19	0,0157	0,0116	-
20	0,0093	0,0104	+
21	0	0,0021	+

22	0,0056	0,0068	+
23	0,006	0,0051	-
24	0	0	0
25	0,0217	0,027	+
26	0	0	0
27	0,0046	0,0077	+
28	0,0102	0,0132	+
29	0,0121	0,0074	-
30	0,0081	0,0069	-
31	0,0074	0,006	-
32	0	0	0
33	0	0,0047	+
34	0,0357	0,0417	+
35	0	0,0076	+
36	0,0108	0,0108	0
37	0,0051	0,0069	+
38	0,0263	0,0357	+
39	0,0012	0,0068	+
40	0,0349	0,0269	-
41	0,0202	0,0261	+
42	0	0	0
43	0,0060	0,0066	+
44	0	0	0
45	0,0165	0,0087	-
46	0,0097	0,0097	0
47	0,0117	0,0154	+
48	0,0143	0,0143	0
49	0,0168	0,0129	-
50	0,0034	0,0036	+
51	0,0035	0,0035	0
52	0	0,0019	+
53	0	0,0069	+
54	0,0021	0,0018	-
55	0,0126	0,0126	0
56	0	0	0
57	0	0	0
58	0	0,0115	+
59	0	0	0
60	0	0	0
61	0,0018	0,0114	+
62	0	0	0
63	0,0386	0,0503	+
64	0,0015	0,0017	+
65	0	0	0
66	0,004	0,004	0
67	0	0	0
68	0,0012	0,0029	+
69	0,0151	0,0125	-

70	0,0056	0,0054	-
71	0,0022	0,0017	-
72	0	0	0
73	0,0037	0,0035	-
74	0	0	0
75	0,0104	0,0104	0
76	0	0,0019	+
78	0,0139	0,0152	+
79	0,0081	0,0064	-
80	0,021	0,0216	+
81	0,0313	0,008	-
82	0,0064	0,0096	+
83	0	0	0

Tableau III.4 : Comparaison entre les résultats d'évaluation requête par requête obtenus de la recherche simple et de la recherche avec le modèle mixte.

L'approche mixte a permet d'améliorer 34 requêtes par rapport au modèle avec termes simples (dirich). 28 requêtes n'ont pas été amélioré (nulle), et 19 requêtes ont été diminué.

Le graphe suivant illustre cette analyse de l'évaluation des résultats requête/requête

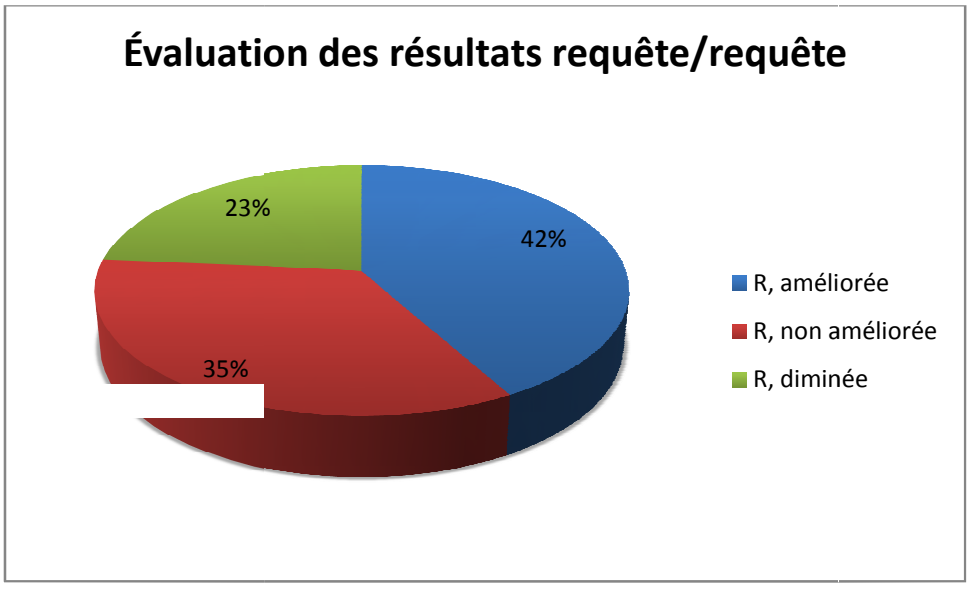


Fig. III.6 : Graphe représentant l'analyse de l'évaluation des résultats requête/requête.

La recherche de l'information avec modèle mixte est améliorée par apport à la recherche avec termes simples d'un pourcentage de 6,89 %.

III.11 Conclusion

Dans ce chapitre, nous avons décrit l'implémentation de notre approche. Nous avons abordé les différentes expérimentations que nous avons réalisées, d'abord nous avons détaillé l'architecture générale, ensuite passant à la présentation de notre environnement de développement et la définition des différents outils utilisés, après à notre application et finalement les expérimentations et évaluations que nous avons réalisées, et cela confirme l'intérêt de l'approche du modèle mixte combinant les termes simples et les entités nommées.

L'évaluation est une étape importante dans le domaine de la recherche d'information du fait qu'elle permet de juger la pertinence des documents restitués par le système de recherche d'information et donc d'évaluer les performances d'un SRI.

Conclusion générale

Notre travail se situe dans le cadre de l'utilisation des entités nommées pour l'amélioration de la qualité de la recherche. L'approche que nous avons adoptée consiste à combiner le modèle des termes simples avec celui des entités nommées pour en créer un modèle mixte.

Nous avons vu que le problème majeur de la traduction de requêtes est l'ambigüité des termes sources. La pluparts des systèmes de recherches d'informations(SRI) utilisent des mots simples pour représenter des documents et des requêtes. Il est reconnu depuis longtemps que cette représentation est insuffisante à cause de l'ambigüité des mots isolés et du non prise en compte des relations entre les mots.

Naturellement il faut choisir un modèle de RI qui supporte une telle représentation. Notre choix s'est fixé sur les modèles de langue étant données les performances qu'ils ont montrées.

Afin de sélectionner les documents susceptibles de répondre à une requête, un SRI évalue la pertinence d'un document vis-à-vis d'une requête, mais les documents retournés par le SRI, ne répondent pas toujours au besoin de l'utilisateur. Pour prendre en compte cette difficulté, un modèle mixte est utilisé, afin d'obtenir des requêtes optimales, le fait que pour sélectionner les bons termes à ajouter à la requête il faut au préalable choisir les bons documents dans lesquels on cherche les termes pour booster la recherche.

Les expérimentations et évaluations que nous avons réalisées en utilisant la collection TIME sur la plateforme de recherche d'information Terrier, ont permet de valider notre approche. Plus précisément, les résultats obtenus mettent en exergue l'intérêt de l'approche d'un modèle mixte.

Pour cela nous pensons que l'utilisation de modèle mixte pour combiner les termes simples et les entités nommées peut encore évoluer vers plus de performance, à savoir :

- Utiliser d'autres types d'EN: personne, organisation
- Tester l'approche sur de grandes collections: TREC

Annexe

La plateforme de RI Terrier 2.1

Cet annexe est principalement consacré à la présentation (Version 2.1), une plateforme de RI de haute performance et évolutive qui permet le développement rapide et à grande échelle de nouvelles applications de recherche d'information.

Introduction

Terrier (TeRabyte RetrIeveR) a été développe par le département informatique de l'université de Galasgow. C'est un logiciel open source entièrement écrit en Java. Il est utilisé avec succès pour la recherche Ad hoc, la recherche Web et la recherche inter-langage dans des environnements centralises et distribues.

Terrier offre plusieurs modèles de pondération de documents et d'expansion de requetés basées sur le Framework DFR (Divergence From Randomness). Comme tous les moteurs de recherche Terrier possède les principales facettes suivantes :

- ***Indexation :*** Permet l'extraction des termes des différents documents du corpus (basic indexed unit).
- ***Recherche :*** Permet de générer des résultats aux requêtes formulées par les utilisateurs.

1. Installation de Terrier

1.1. Préalablement

Pour pouvoir utiliser Terrier il est nécessaire d'installer une JRE (1.5.0 ou plus). La JDK peuvent être téléchargé sur le site de java

http://www.oracle.com/technetwork/java/javase/dowloads/index.html

1.2. Installation

Après avoir téléchargé Terrier version 2.1 sur la page d'accueil du projet Terrier www.terrier.org , créer un nouveau répertoire et dézippé Terrier dans ce dernier.

2. La structure des répertoires de Terrier

Terrier contient un ensemble de répertoires et ils sont structures comme suit :

- bin\ : Contient les scripts nécessaires pour démarrer Terrier.
- doc\ : Contient la documentation relative a Terrier.
- etc\ : Contient les fichiers de configurations de Terrier.
- lib\ : Contient les classes compilées de Terrier et les différentes librairies externes utilises par Terrier.
- licences\ : Contient les informations sur la licence des différents composants inclus dans Terrier.
- share\ : Contient la liste des mots vides (stop word list).
- scr\ : Contient le code source d Terrier.
- var/ : Il contient les deux fichiers :
  - index\ : Contient les structures de donnees : fichier inverse, fichier lexicon, index direct, document index.
  - results\ : Contient les resultats de la recherche.

3. Les applications de Terrier

Terrier vient avec trois applications :

- **Groupe (TREC) Terrier :** Ceci te permet facilement de classer, rechercher, et évaluer des résultats sur des collections de TREC.

- **Terrier Interactif :** Ceci vous permet à de faire la récupération interactive. C'est une manière rapide d'examiner Terrier.

- **Terrier Desktop :** Une application de bureau de recherche d'échantillon.

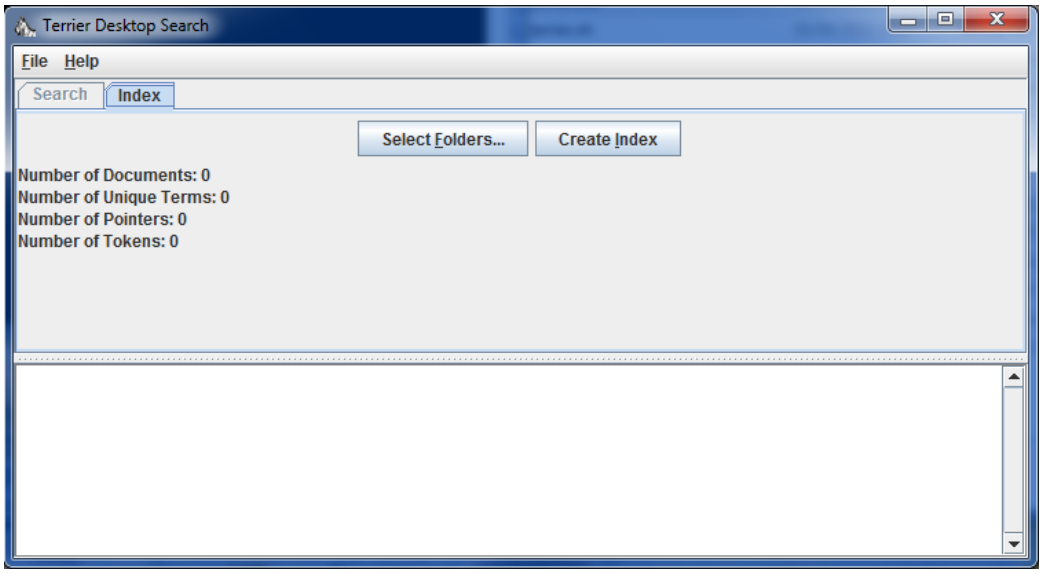


Fig3.1 : Présentation de l'interface Desktop Terrier.

4. L'API d'Indexation de Terrier

L'indexation dans Terrier est devisée en quatre procédures :

1. Extraction de l'objet Document de la collection qui est générée par l'ensemble des corpus reçu en entrée par Terrier.
2. Parcourir chaque document de la collection pour extraire les termes ainsi que les informations relatives.
3. Traitement des termes extrait on utilisant TermPipelines (mots-vide, lemmatisation).
4. La construction de l'inde via l'utilisation de la classe Index Builder.

Le graphique ci-dessous donne une vue d'ensemble de l'interaction des composants principaux impliqués dans le processus d'indexation.

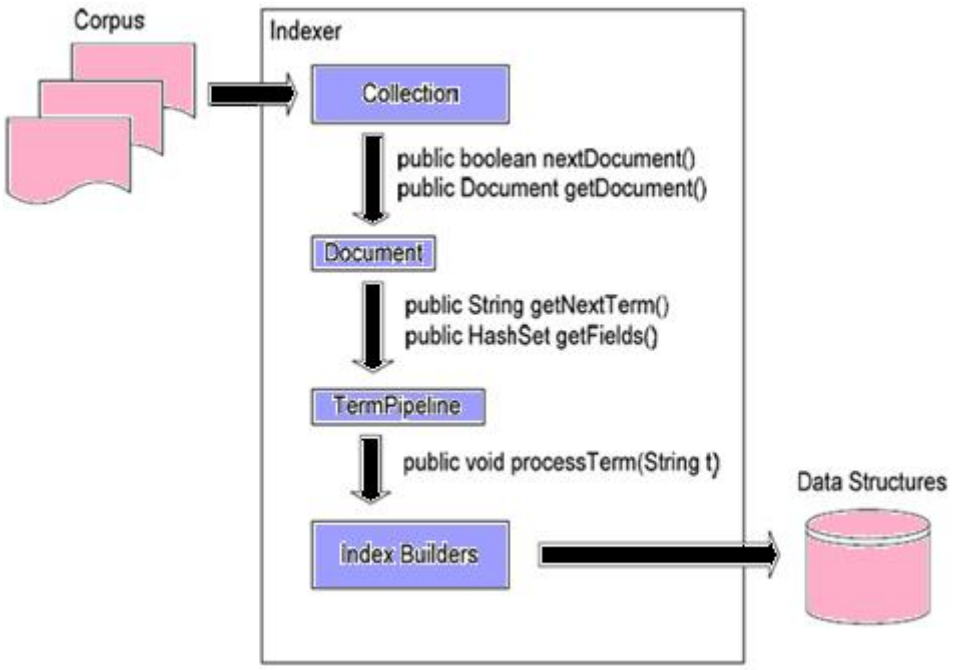


Fig3.2 : processus d'indexation de terrier

Dans ce qui suit nous présentons les quatre étapes de ce processus :

1. Collection

Ce composant met en résume le concept le plus fondamental à l'indexation avec Terrier. Une Collection c'est-à-dire un ensemble de documents. La « collection », c'est un objet qui représente le corpus, c'est-à-dire un ensemble de documents. Collection est une interface qui se trouve dans le package org.terier.indexing.

Utilisée généralement par terrier pour intégrer de nouvelles sources de données (nouveau corpus) et cela en ajoutant une nouvelle classe Java qui implémente cette interface. Elle permet de parcourir un ensemble de documents et de renvoyer un objet document en faisant appel à sa méthode nextDocument().

Terrier offre plusieurs classes qui implémentent cette interface telle que SimpleFileCollection, SimpleMedlineXMLCollection, SimpleXMLCollection, TRECCollection, TRECUTFCollection, WARC018Collection et WARC09Collection.

2. Document

C'est une interface qui se trouve dans le package org.terrier.indexing. Cette composante englobe le concept de Document. Les classes qui impl mentent cette interface s'occupe de parcourir les documents et dont extraire les différents termes. Terrier possède plusieurs parseurs de documents, par exemple : HTMLDocument, FileDocument, MSExcelDocument, etc.

3. TermPipeline

C'est une interface qui se trouve dans le packageuk.ac.gla.terrier.terms. Cette composante reçoit l'ensemble des termes extrait des documents. Cette étape subdivise en deux sous processus

3.1 Eliminations des mots vides (stopword-removal)

Ce processus permet de supprimer les mots vides dans les documents tel que les pronoms, les connecteurs, les espaces, etc. Le but est d'améliorer le résultat d'indexation.

3.2 **La** *normalisation (stemming)*

Ce processus permet de retrouver les différentes formes d'un mot et les représenter sous formes unique, dont le but est d'avoir une forme suffisante à la représentation des différentes apparitions des autres formes.

Terrier utilise plusieurs méthodes de normalisation comme différentes variantes de l'algorithme de porter.

Apres les trois étapes successives précédentes, une étape de construction des structures d'index vient pour compléter le processus d'indexation

4. Indexer

Cette composant est responsable de gérer le processus d'indexation, construire l'index et de son écriture et dans la structure de données appropriée.

Terrier offre deux types d'indexeur : BasicIndex, BlokIndex.

Terrier consiste en quatre structures de données :

- ✓ Vocabulary/Lexicon (data.lex)
- ✓ Inverted Index (data.if)

- ✓ Document Index (data.docid)
- ✓ Direct Index (data.df

Le tableau 1. Ci-dessus permet de représenter le contenu de ces différentes structures index :

Index Structure	***Contents***
Lexicon	Term Term id Document Frequency Term Frequency Byte offset in inverted file Bite offset in inverted file
Inverted Index	Document id Term Frequency Fields (#of fields bits) Blok Frequency [Block id]
Document Index	Document id Document Length Document Number Byte offset in direct file Bite offset in direct file
Direct Index	Term id Term Frequency Fields(#of fields bits) [Block id]

Tableau1: Présentation des différentes structures Index.

L'indexation en Terrier peut être configurée on changeant les propriétés approprié dans le fichier *ets\terrier.properties* . Chaque étage cité auparavant possède ces propres propriétés.

5. L'API de recherche de Terrier

La recherche s'effectue sur les structures d'index précédemment crées et cela âpres analyse de la requête par le système. Cette opération vise a retourner des document triés selon l'ordre décroissant de leurs scores.

Terrie utilise trois composantes principales pour la recherche : Query, Manager, Matching.

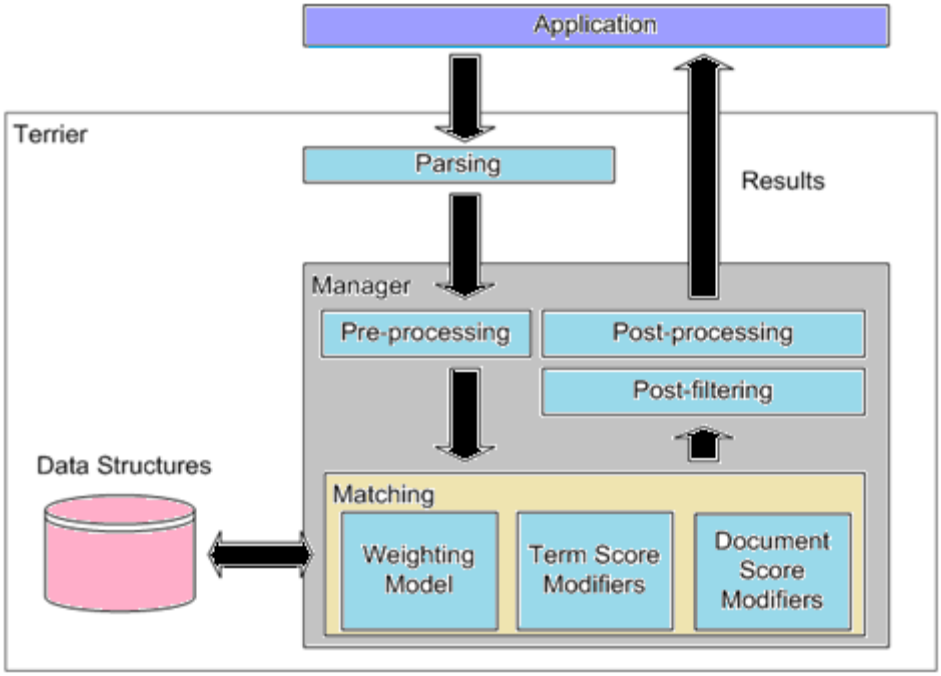


Figure 2. Présentation du processus de recherche de terrier

5.1 Query

C'est l'entrée que l'application offre a Terrier (voir la Figure2) .Le module Matching est responsable de déterminer quel document correspond à une requête spécifique et attribut un score au document qui respect la requête.

Query est une classe abstraite, qui se trouve dans le package ac.gla.terrier.querying.parser et qui modélise les requêtes. Elle consiste soit en un sub-queries ou bien un query termes. Un objet Query est créé pour chaque requête.

Terrier offre un support pour chaque type de requêtes :

- SingleTermeQuery : modèle de requête avec un seul terme.
- MultitermeQuery : modèle de requête qui contient plus d'un terme.
- File Query : modèle de requête qualifié par un champ (field).

5.2 Manager

C'est le module qui est chargé de la gestion de la recherche .Dans un premier niveau il lit chaque requête en utilisant le parseur approprié (l'analyse : qu'il s'agit de la reconnaissance des mots de la requête, elle se déroule de la même façon que pour les documents dans l'étape d'indexation. Puis il créait un second niveau de gestion associé à chaque requête en récupérant l'ensemble des paramètres nécessaires et en spécifiant un modèle de pondération puis il crée un fichier résultat (.Res) ou il associe à chaque requête les documents qui lui sont pertinents. Le résultat de chaque requête est donné par le second niveau de gestion.

Le second niveau de gestion est responsable de l'ordonnancement et de la coordination des opérations principales de haut niveau sur une seule requête.

Ces opérations sont : Pre-processing, Matching, Post-processing, Post-filtring

- **Pres-processing (pretraitement)**

  Cette étape est constituée des processus d'élimination des mots vides, de la lemmatisation et normalisation de termes. Ce prétraitement s'opère sur les termes précédemment extrais par l'analyse.

- **Matching**

  Le composant Matching est responsable de déterminer les documents correspondant à la requête spécifié et donne un score au document qui vérifie la

requéte.il utilise l'interface *Weighting Modele* pour assigner un score à chaque mot de la requête dans le document.

Terrier contient un nombre important de modèle de pondération .parmi ces modèles nous pouvons citer les suivants :

- Tf-IDF : ou la formule de TF est donnée par Robertson et IDF par Spark Jones.
- Lemur TF-IDF : le modèle proposé par Lemur.
- BM25 : le modèle probabiliste BM25.
- Hiemsta-LM : le modèle de langage proposé par Hiemstra.

Terrier dispose de plusieurs fonctions de modification de scores que ce soit pour les termes ou les documents, cela dans le but par exemple : d'accroitre la priorité d'un terme donné ou bien d'une phrase.

❖ **Post-processing /filtering (post-traitement/Filtrage)**

Après l'étape précédente, nous obtenons un ensemble de documents avec leurs scores respectifs. A ce niveau deux autres processus sont disponibles pour améliorer les résultats, il s'agit du Post-traitement et Post-filtring.

❖ **Post-traitement**

Ce traitement permet la modification du résultat de plusieurs manières, nous pouvons citer l'une d'elle.

✓ **La reformulation de la requête**

Nous parlons dans ce cas de pseudo-relance feedback qui permet soit l'expansion de la requête avec l'ajout de nouveaux termes, soit la repondération des termes de la requête. Cette reformulation permet d'apporter de nouvelles informations, mais peut aussi engendrer du bruit.

Terrier déploie différents modèles de reformulation de la requête, il permet aussi l'extension et l'ajout d'autres modèles.

Après à la fin de la reformulation, terrier doit à nouveau rappeler les fonctions d'appariement.

- **Post-Filtrage**

Cette opération est plus simple que la précédente, elle permet soit l'inclusion ou l'exclusion d'un document. Elle est surtout utilisée pour les besoin de restriction de l'utilisateur pour un domaine particulier.

- **Resultat**

A la suite des étapes précédentes, Terrier s'occupe de retourner un ensemble de documents triés selon l'ordre décroissant de leurs scores.

6. Modification Terrier

L'une des caractéristiques principales de terrier est, son extensibilité que ce soit sur ses processus d'indexation, dans ses modèles de pondération ou dans ses étapes de reformulation de la requête. Ces extensions peuvent aussi survenir dans le cas d'intégration de nouvelles collections de documents spéciales ou personnelles qui peuvent servir pour l'évaluation.

Terrier permet aussi la modification du code source pour intégrer tout changement nécessaire. Pour que toute modification soit prise en compte, il est nécessaire de recompiler tout le code source de Terrier.

7. Utilisation de Batch (TREC) Terrier

Dans cette section nous décrivant l'utilisation de Batch(TREC) Terrier pour l'indexation, la recherche et l'évaluation sur le système d'exploitation Windows(XP) 7.

- ***Indexation***

1. Allez au répertoire ou Terrier est installé ou utilisant la commande *cd* :

   >>cd terrier 2.1

2. Initialisation de terrier pour l'indexation d'une nouvelle collection TREC:

   >>.\bin\trec_setup <le chemin absolu du répertoire contenant les documents à indexer>

3. Maintenant vous pouvez indexer la collection TREC.

>>.\bin\trec_terrier-i

Remarque

Pour indexer une collection autre qu'une Collection TREC il est nécessaire d'apporter quelques modifications au fichier *terrier.properties.*

- ***Recherche et Evaluation***

Avant toute recherche ou évaluation il est nécessaire de réaliser les trois étapes suivantes :

1. Spécification du fichier de jugement de pertinence dans le fichier *etc\trec.qrels.*

```
#add the qrels files to use for evaluation
D:\Master2\terrier\qrel-wsj.txt
```

Figure3: Exemple d'un fichier trec.qrels

2. Spécification des fichiers contenant les requétes (topics file) dans le fichier *etc\trec.topics.list.*

```
#add the topics files to use for querying
D:\Master2\terrier\topics.51-100
```

Figure 4: Exemple d'un fichier trec.topics.list

3. Spécification du modèle de pondération (ex. TF_IDF) à utiliser dans le fichier *etc\trec.models.*

Apres que ses étapes soient faites la recherche ou l'évaluation peuvent être lancé sur Terrier.

4. Pour lancer la recherche dans Terrier il suffit d'exécuter la commande : .*bin**trec_terrier-r*

5. Les résultants de la recherche peuvent être évalués en exécutant la commande : .*bin**trec_terrier-e*

Bibliographie

- **[Aliane et al,04]** H. Aliane, Z. Alimazighi, R. O. Boughacha, T. Djelliout, « Un Système de reformulation de requêtes pour la recherche d'information », Centre de Recherche sur l'Information Scientifique et Technique, Université des Sciences et de la Technologie Houari Boumedienne, Alger, Algérie.
- **[Baaziz,05]** M. Baaziz, « Indexation conceptuelle guidée par ontologie pour la recherche d'information » Université Paul Sabatier de Toulouse, 2005
- **[Boughanem,05]** ''Recherche d'information'' Université Paul Sabatier de Toulouse.2005
- **[Boughanem et al,04]** Mohand Boughanem, Wessel Kraaij, Jian-Yun Nie, DIRO « Modèle de langue pour la recherche d'information » Institut de Recherche en Informatique de Toulouse, 118 route de Narbonne 31000 Toulouse Cedex 9, France ; TNO TPD, 2600 AD Delft, Pays bas ; DIRO, Université de Montréal, CP. 6128, succursale Centre-ville, Montréal, Québec, H3C 3J7 Canada.
- **[Boubekeur,08]** : Fatiha Boubekeur-Amirouche, thèse de doctorat « contribution à la définitions de modèles de recherche d'information flexibles basés sur le CP-Nets », université de Toulouse III-Paul Sabatier ; 01/07/2008.
- **[Bouidghaghen,11]** Ressad-Bouidghaghen Ourdia, 2011 : Accès contextuel à l'information dans un environnement mobile, approche basée sur l'utilisation d'un profil situationnel de l'utilisateur et d'un profil de localisation des requêtes. Université Toulouse III, Paul Sabatier.
- **[Chinchor,98]** : Chinchor N. « MUC-7 Named Entity Task Definition (version 3.5) », in *Proceedings of the 7th Message Understanding Conference (MUC-7)*, Fairfax, VA, 1998.
- **[Damien, 12].** Damien Nouvel, « Reconnaissance Des Entités Nommées Par Exploration De Règles D'annotation (Interpréter les marqueurs d'annotation comme instructions de structuration locale) » , 2012
- **[Delannoy,09]** C. Delannoy : Programmer en java 5 et 6, 5 e édition, EYROLLES, 2009.
- **[Hammache et al, 13]** Hammache A, Boughanem M & Ahmed-Ouamer R « Combining compound and single terms under language model framework », KAIS, 2013.
- **[Ihadjadene,04]** Madjid Ihadjaden, livre « les systèmes de recherche d'information, modèle conceptuel », LAVOISIER, 2004.
- **[LeMeur et al.,04]** Céline LeMeur, Sylvain Galliano, Edouard Geoffrois Conventions d'annotations en Entités Nommées 2004 http://www.afcp-parole.org/ester/docs/convention_en_old.pdf
- **[Lemy et al, 97]** L.Lemy et CHARLES L. PERKINS : Apprenez java 1.1, Simon & Schuster Macmillan.1997.
- **[Nie ,04]** Jian-Yun Nie, cours hiver 2004 ; Module Recherche d'Information ; Université Montréal. Département d'informatique et de recherche opérationnelle (I.R.O.) Hiver 2004 http://www.iro.umontreal.ca/~nie/IFT6255.
- **[Nguyen, 02]** Nguyễn Thiện Giáp. Từ vựng học tiếng Việt. Nxb Giáo dục, H., 2002,

  trang 326–330.

- **[Mounier]** H. Mounier, Support de cours Java Structures de données Notions en Génie Logiciel et Programmation Orientée Objet, Université Paris Sud.
- **[Ounis et al,06]** Ladh Ounis et al, Terrier information Retrieval platform, university of galasgow, galasgow G12 8QQ, UK
- **[Risjbergen, 79]** Information retrieval. Butterworths, 1979.
- **[Sauvagnat,05]** Karen Sauvagnat, thèse doctorat « modèle flexible pour la recherche d'information dans des corpus de documents semi-structurés », université Paul Sabatier, Toulouse, 2005.
- **[Serrano, 10]** Serrano Laurie : Modélisation d'une ontologie de domaine et des outils d'extraction de l'information associées pour l'anglais et le français, Université Stendhal Grenoble, France.
- **[Tolone, 06]** Tolone Elsa, Extraction d'entités nommées par les graphes d'unitex, projet 2006
- **[Zemirli,04]** Nesrine W.Zemirli, mémoire « vers le développement d'un système de recherche d'information personnalisé intégrant les profils utilisateurs », université Paul Sabtier, Toulouse, 2004.
- **[Zhai,02]]** ChengXiang Zhai « Risk Minimization and Language Modeling in Text Retrieval », School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 2002.

Webiographie

http://www.terrier.org/docs/v2.2.1/terrier_develop.html

http://www.netbeans.org

http://www.wikipedia.org

http://www.mscerts.programming4.us/fr/239656.aspx