الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Master Académique

Thème :

Réalisation d'un système parallèle pour l'indexation automatique des documents dans la recherche

Encadré Par :

Ds.SOUSSI Hakim

Présenté par :

BOUNAR Wennassa

Juin 2017

REMERCIEMENTS

Nous tenons tout d'abord à remercier notre encadreur Dr.SOUSSI Hakim pour tour le suivi, l'aide, l'orientation, les conseils précieux qu'il nous a apporté tout au long de la réalisation de ce travail.

On remercie également toutes les personnes qui nous ont soutenus pour bien accomplir notre travail.

Je dédie ce mémoire à

A mes chers parents ma mère et mon père pour leurs patience, leurs amours, leurs soutiens et leurs encouragement.

Ainsi ma sœur et mes frères

A mes amis et camarades,

A ma sœur soraya pour tous leurs encouragements et leurs soutiens

A mes professeurs a qui je dois gratitude et respect, pour leur temps dévouement et implications.

A tous ceux qui m'ont aidé tout au long de mes études.

Aux étudiants qui prendront le temps de lire ce manuscrit

RESUME:

Pour avoir un système de recherche de qualité, il est important que son système d'indexation reflète au mieux le contenu de la collection originale.

L'indexation des documents est une étape primordiale car elle détermine de quelle manière les connaissances contenues dans les documents fournis sont représentées, elle permet aussi de réduire la complexité des documents et les rendre plus faciles à manipuler. Elle a lieu à chaque ajout d'un document dans l'ensemble des documents étudiés.

L'indexation automatique implique une analyse automatique du contenu de chaque document de la collection. Cette analyse comprend plusieurs étapes, le but étant d'extraire les termes représentatifs du contenu et d'évaluer leur pouvoir de représentation et de caractérisation du document dans lequel ils apparaissent.

Le but de ce travail est de mettre au point un système parallèle pour l'indexation automatique des documents dont le but est d'accroitre les performances des systèmes de recherche d'informations (SRI) en termes de temps par apport aux systèmes séquentiels classiques.

Mots-clés : Recherche d'information, indexation, parallélisme.

SOMMAIRE

REMERCIEMENTS I

Je dédie ce mémoire à II

SOMMAIRE IV

LISTE DES FIGURES VII

INTODUCTION GENERALE : 1

CHAPITRE I : LA RECHERCHE D'INFORMATION 2

1. INTRODUCTION: 3

2. PROCESSUS DE LA RECHERCHE D'INFORMATION : 5

2.1. Concepts de base de la recherche d'information 5

2.2. Le processus de recherche d'information 5

2.2.1. Les données d'entrée 6

2.2.2. La représentation de l'information (L'indexation) 7

2.2.3. L'appariement document / requête 8

2.2.4. Mécanismes de reformulation de requête 8

2.2.5. Le reclassement en recherche d'information (l'apprentissage) 9

2.2.6. La recherche 10

3. LES GRANDES ETAPES DU PROCESSUS DE LA RECHERCHE D'INFORMATION : 11

3.1. L'indexation 11

3.1.1. Le prétraitement linguistique (analyse lexicale) 12

3.1.2. La sélection ou filtrage par un anti-dictionnaire 13

3.1.3. La radicalisation ou La normalisation linguistique 14

3.1.4. La pondération 15

3.1.5. L'indexation inversée 16

3.2. Les modèles de recherche d'information 19

3.2.1. Les modèles booléens 20

3.2.2. Le modèle vectoriel 23

3.2.3. Modèle probabiliste 25

3.3. Evaluation d'un système de recherche d'information 26

3.3.1. Collection de test 27

3.3.2. Mesures d'évaluation 27

4. CONCLUSION 28

CHAPITRE II : LES ALGORITHMES DE RACINISATION 29
1. INTRODUCTION 30
2. LES ALGORITHMES DE RACINISATION 30
2.1. Algorithme de Lovins 31
2.1.1. Définitions : 31
2.1.2. Limites de l'algorithme de lovins 31
2.2. L'algorithme de Paice et Husk 32
2.2.1. Définitions : 32
2.2.2. Limites de l'algorithme de Paice et Husk 32
2.3. Algorithme de Krovetz 33
2.3.1. Définitions : 33
2.3.2. Limites de l'algorithme de Paice et Husk 33
2.4. Algorithme de EDA 33
2.5. L'algorithme de Porter 34
2.5.1. Définitions : 34
2.5.2. Principe de l'algorithme de Porter 34
2.5.3. L'algorithme : 36
2.5.4. Limites de l'algorithme 39
2.6. L'algorithme de carry 40
2.6.1. Définitions : 40
2.6.2. Limites de l'algorithme 40
3. CONCLUSION 41
CHAPITRE III : CONCEPTION ET IMPLEMENTATION DU SYSTEME 42
1. INTRODUCTION : 43
2. ARCHITECTURE DU SYSTEME : 43
2.1. Le système d'indexation séquentiel : 43
2.2. Le système d'indexation parallele : 43
3. LE PROCESSUS DU SYSTEME : 45
3.1. L'analyse lexicale : 45

LISTE DES FIGURES

Figure 01 : Processus de la recherche d'information [Brini, 05] 6
Figure 02 : Les différents traitements effectués lors de l'indexation [Champc, 09] 12
Figure 03: différents modèles de la recherche d'information [Boub, 08] 20
Figure 04: requêtes booléennes sous forme de diagramme de Venn [Champc, 09] 21
Figure 05 : Architecture globale du système 44
Figure 06 : Exemple sur l'étape élimination de la ponctuation. 45
Figure 07 : Exemple sur l'étape de la casse 46
Figure 08 : Exemple sur l'étape le filtrage. 46
Figure 09 : Exemple d'exécution du système d'indexation 47
Figure 10 : Exécution du système sur l'exemple précédent avec le système séquentiel. 48
Figure 11 : Exécution du système sur l'exemple précédent avec le système parrallel. 49

INTODUCTION GENERALE :

La Recherche d'Information (RI) peut être définie comme une activité dont la finalité est de localiser et de délivrer un ensemble de documents à un utilisateur en fonction de son besoin en informations. Cependant, vu le volume important des documents disponibles, on doit trouver ceux qui correspondent au mieux à l'attente de l'utilisateur.

En informatique, le terme de *recherche* signifie la discipline informatique qui traite la problématique d'accès à l'information pertinente dans une masse de données souvent importante. Or, Elle peut se définir comme l'ensemble d'opérations, méthodes et procédures qui permettent de retrouver à partir d'une collection de documents, l'information pouvant répondre à une question sur un sujet précis.

Dans un contexte documentaire, un SRI est crée pour gérer une collection de documents stockés sous forme d'une représentation intermédiaire permettant de refléter aussi fidèlement que possible leur contenu sémantique.

Le rôle du SRI peut être défini comme l'ensemble des procédures et des opérations permettant la gestion, la représentation, l'interrogation, la recherche, le stockage et la sélection des informations répondant aux besoins d'un utilisateur.

L'interrogation du fonds documentaire à l'aide d'une requête nécessite la représentation de cette dernière sous une forme compatible avec celle des documents. Les fonctionnalités d'un SRI peuvent être déduites du processus global de la RI.

Ce travail est consacré à la réalisation d'un système d'indexation parallèle dont le but est d'amélioré l'efficacité en terme de temps. Le système parallèle sera comparé à un autre qui est séquentiel afin d'évaluer les résultats.

Ce mémoire est organisé en trois chapitres :

Le premier chapitre sera consacré aux concepts de base de la RI, ainsi qu'au processus global de la RI et à l'importance des opérations qui composent ce processus. On verra aussi les grandes étapes du processus de la recherche d'information, à savoir l'indexation, les modèles de recherche d'information et l'évaluation des systèmes de recherche d'information.

Le deuxième chapitre présentera quelques algorithmes existants qui traitent la radicalisation tels que les algorithmes de Porter et de Paice.

Dans le troisième chapitre, une description du modèle issu de cette recherche ainsi que l'implémentation et les résultats obtenus seront présentés et discutés.

Enfin, une conclusion finale et les perspectives viendront résumer le travail effectué ainsi que les éventuelles améliorations qui peuvent être ajoutés.

CHAPITRE I : LA RECHERCHE D'INFORMATION

CHAPITRE I : LA RECHERCHE D'INFORMATION

1. INTRODUCTION:

Dans le passé, la recherche d'information était un domaine réservé uniquement à l'intersection des techniques documentaires et de la science informatique. Aujourd'hui, grâce au développement des moteurs de recherche sur le web, on peut l'utilisée dans tous les domaines tel que ; la recherche des horaires des transports en communs, trouver un médicament, ou encore consulter la liste des filmes disponible au cinéma.

Avec l'explosion de la quantité d'informations mises à disposition du grand public et des entreprises, notamment via Internet, l'automatisation du stockage et de la consultation de l'information est devenue un besoin essentiel. Le développement d'outils et de méthodes pour gérer ces quantités d'informations est bien plus qu'un besoin, une nécessité.

Le terme « recherche d'information » a été introduit par Calvin Mooers en 1950 [Moer,50], il signifie la discipline qui traite la problématique d'accès à l'information pertinente dans une masse de données souvent importante.

Au début des années soixante, quelques années après l'invention de l'ordinateur, la RI est apparue comme une réponse au besoin de gérer l'explosion de la quantité d'informations.

La recherche d'information a été utilisée pour la première fois officiellement lors du lancement de la conférence RIAO (Recherche d'Information Assistée par Ordinateur) en 1985, à Grenoble. On parlait auparavant de « recherche documentaire » ou « d'informatique documentaire ».

Très tôt, le monde de la recherche s'est intéressé aux SRI qui sont des outils dans lesquels sont mis en œuvre des techniques et des mécanismes assurant la gestion automatique des informations documentaires, afin de répondre à un besoin d'information.

On peut distinguer cinq grandes périodes dans l'histoire des systèmes de recherche d'information:

1- Période de développement des SRI, avant 1980 : trois études (Ives, Hamilton, Davis1980, 1981, 1982) concluent que la grande majorité des recherches publiées sont alors de nature non empirique (soit conceptuelle, soit technique) sans formulation d'hypothèses claires.

2- Période de la théorisation des SRI, 1980-1985 : trois études (Hamilton et Ives 1982, Culnan et Swanson 1984, Culnana 1980-1985) montrent des progrès importants vers une tradition de recherche cumulative et un abandon des recherches techniques : développement et accumulation de connaissances, travail en équipe, construction de théories, production de références propres à la discipline.

3- Période positive, 1985-1990 : deux études (Culnan 1987, Barki et al.1988) montrent que l'orientation positive de la discipline se confirme alors, avec une rigueur méthodologique associé. le niveau d'analyse individuel, qui était dominant, a baissé au profit du niveau organisationnel devenu majoritaire, et du niveau inter-organisationnel aujourd'hui émergent. Les études empiriques sur le terrain prennent une place de plus en plus importante.

4- Période de la diversification, 1990-2000 : (Cheon et al., 1993; Alavi et Carlson, 1992; Reix et Fallery, 1996; Claver et al., 2000;) le monolithisme thématique autour du développement des SRI se termine, bien qu'encore majoritaire, il a perdu de l'importance face au domaine informationnel. Des études démontrent une diversification des objets de recherche (vers la gestion des organisations, le travail collaboratif …) et un renouvellement plus rapide des thèmes d'application (nouvelles technologies, internet, ERP…).

5- Période du contexte, depuis 2000 : une étude (Sdorva et al. 2008) montre l'importance attachée au contexte social et la manière dont les individus, les groupes et les organisations interagissent avec les SRI. Ceci traduit d'une certaine manière d'alignement de la recherche sur les questions d'ordre du jour dans les organisations [Rod, 10].

2. PROCESSUS DE LA RECHERCHE D'INFORMATION :

2.1. Concepts de base de la recherche d'information

La recherche des informations est la mise en œuvre des techniques et des moyens permettant de retourner les documents pertinents d'une collection en réponse à un besoin en information d'un utilisateur.

Elle consiste à mettre en correspondance les représentations des informations contenues dans un fonds documentaire avec celles des besoins de l'utilisateur.

2.2. Le processus de recherche d'information

Un Système de Recherche d'Information SRI a pour fonction de permettre à l'utilisateur d'accéder à des documents qui contribuent à combler son besoin d'information, exprimé sous forme de requête, qui motive sa recherche. Ainsi le système peut être vu par l'utilisateur comme un instrument de prédiction de la pertinence des documents d'un corpus par rapport à sa requête.

Nous détaillerons les modèles de recherche les plus importants et nous présenterons en quoi les choix que nous adoptons diffèrent.

Du coté du système, le processus de RI est composé de plusieurs fonctions [Champc, 09]:

- L'indexation des documents et des requêtes.
- La mise en correspondance requête-documents avec un ordonnancement des documents quand le modèle le permet.
- La restitution des documents reconnus pertinents par rapport à la requête.

Du point de vue utilisateur, le processus de recherche est composé de deux fonctions principales :

- L'interrogation du système par une requête
- L'analyse des documents restitués par le système et une éventuelle reformulation de requête.

La figure 01 représente les fonctionnalités d'un SRI. Ce dernier répond aux besoins des utilisateurs en retournant des documents via une requête

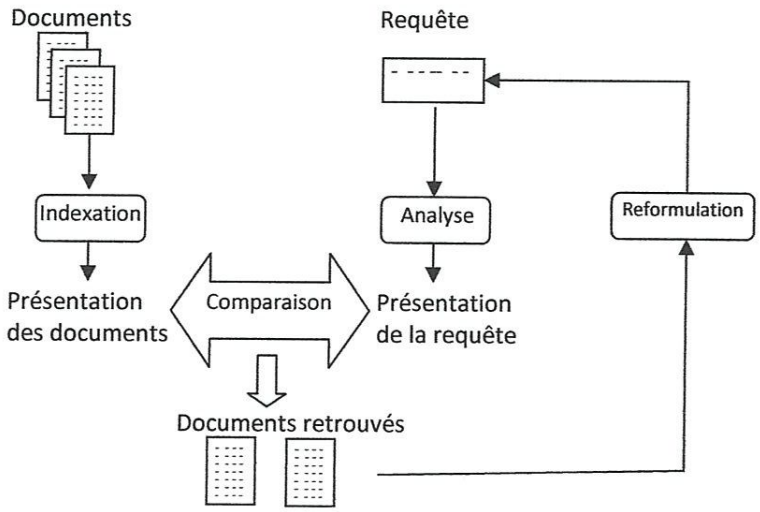


Figure 01 : Processus de la recherche d'information [Brini, 05]

2.2.1. Les données d'entrée

a. Les requêtes

La requête est créée par l'utilisateur, c'est elle qui initie le processus de recherche. Elle traduit un besoin d'information, c'est-à-dire une nécessité ressentie de combler une déficience constatée en information, une lacune ou un défaut [Schutz et al., 1973].

La requête n'est donc qu'une représentation possible d'un besoin en information. La requête peut être exprimée de différentes manières. Les concepts de « requête » et « besoin » sont souvent confondus.

La requête doit contenir les concepts clés du besoin et les relations entre ces concepts.

La requête initiale est vue en RI comme un moyen permettant d'initialiser le processus de sélection d'informations pertinentes.

Elle est issue d'une analyse conceptuelle du besoin d'information qui est effectuée dans l'esprit de l'utilisateur de façon plus ou moins précise.

En effet, l'utilisateur tente de traduire son besoin d'information en une requête.

La requête formulée peut avoir la forme d'une expression en langue naturelle, ou encore d'une liste de concepts avec éventuellement un degré d'importance associé, ou encore une formule logique de concepts coordonnés par des opérateurs logiques [Champc, 09].

Une fois la requête exprimée, il est nécessaire de lui donner une forme utilisable par un SRI pour entamer le processus de recherche.

b. Les documents

Un document veut dire toute base de connaissance, fixée matériellement, susceptible d'être utilisée pour une consultation, étude ou recherche.

Un document est une trace d'activité humaine, laissée dans l'objectif d'être interprétée par des personnes souvent différentes du ou des personnes à l'origine de cette même trace. Avec l'avènement des ordinateurs, le document quitte son support matériel natif (le papier pour l'écrit et les bandes analogiques pour les documents audiovisuels) et devient numérique. Il est alors stockable sous la forme d'une représentation binaire.

Pour réaliser d'une façon efficace un SRI, le système à concevoir doit réaliser plusieurs fonctionnalités:

–L'indexation des documents de la collection et du besoin utilisateur (La requête).

–L'appariement des représentations requête-documents pour le calcul de la pertinence des documents en réponse à un besoin utilisateur.

–Eventuellement pour certains SRI, la reformulation de la requête.

2.2.2. La représentation de l'information (L'indexation)

Afin de réduire la complexité des documents et les rendre plus faciles à manipuler, le document doit être transformé.

Indexer un document c'est élire ses termes représentatifs afin de générer la liste des termes d'indexation et ajouter à l'index de la collection, pour chacun de ces termes, la liste des références de chaque document le contenant.

Par référence on entend identifiant, c'est-à-dire un moyen de retrouver de façon non ambigüe des documents ou un document ou une partie de document où le terme apparaît.

Dans un SRI, dont l'objectif final est de retourner une liste de documents pertinents par rapport à une requête utilisateur, il est nécessaire de pouvoir rechercher les documents de la collection dont le contenu ressemble ou correspond au contenu de la requête.

La recherche implique une méthode de tri et la comparaison de contenu implique une analyse à défaut de pouvoir directement comparer les concepts véhiculés dans le document à ceux présents dans la requête.

L'indexation des documents est une étape primordiale car elle détermine de quelle manière les connaissances contenues dans les documents fournis sont représentées. Elle a lieu à chaque ajout d'un document dans l'ensemble des documents étudiés.

2.2.3. L'appariement document / requête

La comparaison entre le document et la requête revient à calculer un score représentatif de la ressemblance entre le document et la requête.

Ce score de similarité entre le document et la requête est donné par une fonction nommée Retrieval Status Value RSV(d,q), où d est un document et q est une requête.

Traditionnellement le système de recherche retourne à l'utilisateur une liste de documents classés par RSV.

Cette fonction est fondamentale pour la RI car c'est elle qui détermine comment comparer la requête aux documents indexés.

Le processus d'indexation et la fonction d'appariement constituent les deux éléments essentiels du modèle de recherche [Champc, 09].

2.2.4. Mécanismes de reformulation de requête

La reformulation de requête consiste, à partir d'une requête initiale formulée par l'utilisateur, à construire une requête qui répond mieux à son besoin informationnel.

Les techniques de reformulation de requête se classifient en méthodes locales et méthodes globales.

Les méthodes locales ajustent une requête relativement aux documents qui sont retournés comme documents pertinents pour la requête initiale. Elles se basent sur la technique dite de réinjection de pertinence (relevance feedback).

Les méthodes globales se basent sur l'expansion de requête en s'appuyant sur des ressources linguistiques (thésaurus ou ontologies), ou sur des techniques d'associations de termes telles que les règles d'association [Boub, 08].

La reformulation consiste a reajuster les poids des termes de la requête ou à rajouter des termes reliés à ceux de la requête initiale. Elle peut être manuelle (avec intervention de l'utilisateur) ou automatique.

L'ajout d'un terme peut se faire de plusieurs manières : soit par une interaction entre l'utilisateur et le système soit de façon automatique, on parle alors de réinjection de pertinence aveugle.

2.2.5. Le reclassement en recherche d'information (l'apprentissage)

C'est un domaine en plein essor qui concerne aussi bien la RI que l'apprentissage automatique.

L'apprentissage automatique est un outil pratique qui permet de régler les paramètres de manière automatique, de combiner plusieurs sources d'informations.

En RI, l'apprentissage automatique a donné lieu à la tâche d'apprentissage à ordonner (learning to rank).

L'objectif de l'apprentissage automatique est de concevoir et d'appliquer des méthodes pour apprendre automatiquement à partir de peu de données afin d'obtenir un meilleur classement des documents, de sorte à ce que l'on puisse trier des documents sur la base de leur degré de pertinence ou de ressemblance basée sur les caractéristiques.

Il existe trois familles de modèle à apprentissage : les modèles génératifs, les modèles discriminatifs et les modèles hybrides.

- **Les modèles génératifs** :

Ils font l'hypothèse que les documents peuvent être générés par un modèle de langage [Croft et al., 2003].

Un modèle de langage est un ensemble de propriétés et de contraintes sur des séquences de mots obtenues à partir d'exemples qui permet de déterminer la probabilité qu'une phrase quelconque puisse être générée par le modèle.

Les données d'entraînement sont directement utilisées pour estimer les paramètres du modèle et enrichir le modèle du document.

La probabilité qu'un document appartienne à la classe des documents pertinents pour la requête est estimée par la probabilité conditionnelle qu'un document pris au hasard appartienne à l'ensemble des pertinents [Champc, 09].

- **Les modèles discriminatifs** :

Ils cherchent d'abord à maximiser la qualité de la classification puis, dans un second temps, une fonction de coût va réaliser l'adaptation du modèle de classification final.

Ces méthodes ont toutes deux été appliquées avec succès à la classification de documents [Ogilvie et al., 2003].

- Les approches hybrides :

Ils utilisent une combinaison des approches génératives et discriminatives, par exemple en faisant un entraînement discriminatif avec une approche générative pour la modélisation et une approche discriminante pour l'entraînement ou encore en combinant les prédictions.

Les modèles discriminatifs se montrent plus performants si l'on dispose de nombreux exemples d'apprentissage, dans le cas contraire ce sont les modèles génératifs qui dominent.

2.2.6. La recherche

Cette tache constitue le cœur de la RI.

Un système de recherche transcrit un besoin d'information donné sous la forme d'une requête constitué de mots clés. Lorsque l'utilisateur examine le résultat de la recherche, il voit les documents triés par ordre décroissant de pertinence.

Dans la phase de recherche il existe plusieurs modèles parmi lesquels on trouve le modèle booléen qui est le premier modèle à être utilisé dans la RI.

Si la requête est une expression booléenne, l'utilisation de l'index inversé permet de trouver facilement et en un temps minimal tous les documents qui satisfont cette requête.

En revanche, les systèmes booléens purs ne permettent pas de retrouver les documents similaires au besoin d'information de l'utilisateur et ne contenant pas exactement les termes de la requête.

Plusieurs modèles ont été développés pour palier ce problème, depuis les modèles vectoriels jusqu'aux modèles probabilistes et plusieurs stratégies ont apparu afin d'enrichir ces différents modèles. Ces stratégies consistent à étendre la requête afin d'y inclure des termes similaires mais non mentionnés originellement par l'utilisateur.

3. LES GRANDES ETAPES DU PROCESSUS DE LA RECHERCHE D'INFORMATION :

3.1. L'indexation

L'indexation consiste à extraire des documents les mots les plus discriminants encore appelés index. Cette première tâche est généralement effectuée en marge du processus de recherche car la construction des index peut être assez longue en fonction du nombre de documents de la collection ainsi que de la taille des documents. Les index ont un caractère réducteur car tous les termes d'un document ne sont pas importants à prendre en compte pour la recherche.

L'indexation des documents est une étape primordiale car elle détermine de quelle manière les connaissances contenues dans les documents fournis sont représentées. Elle a lieu à chaque ajout d'un document dans l'ensemble des documents étudiés.

La première étape du processus d'indexation est l'analyse de la ressource afin d'en extraire les caractéristiques les plus importantes et les structurer par la suite dans l'un des modèles de recherche d'information.

L'indexation dcs documents du corpus a pour objectif d'éviter au système de les analyser à chaque interrogation, en effet l'index créé permet d'établir le lien vers les documents indexés à travers les mots-clés représentatifs de leurs contenus.

Salton [Sal, 88a] [Sal, 88b] distingue trois types d'indexation: l'indexation manuelle, semi-automatique et automatique.

- **indexation manuelle :** chaque document est analyse par un spécialiste du domaine correspondant, ou par un documentaliste, qui détermine, selon ses connaissances, les unités syntaxiques ou mots-clés qui lui semblent les plus significatifs pour représenter le contenu du document. Les mots clés sont ensuite regroupés dans une liste plus ou moins structurée, utilisée pour les opérations d'indexation et d'interrogation.
- **indexation automatique:** cette opération est réalisée à l'aide d'un processus entièrement informatise, qui consiste à déterminer les unités syntaxiques ou mots-clés représentatifs des concepts et des thèmes exprimes dans le document. Les méthodes existantes vont d'une simple extraction de mots simples (unitermes) `a des analyses linguistiques et/ou statistiques permettant d''établir des relations sémantiques entre ces mots et de pondérer ces mots en fonction de leurs apparitions.
- **indexation semi-automatique :** cette méthode est une combinaison des deux précédentes. Le choix final reste au spécialiste du domaine ou documentaliste, qui

intervient souvent dans l''etablissement de relations sémantiques entre les mots-clés et le choix des multi-termes significatifs.

La figure 02 représente les opérations traditionnellement effectuées sur les données textuelles lors de l'indexation :

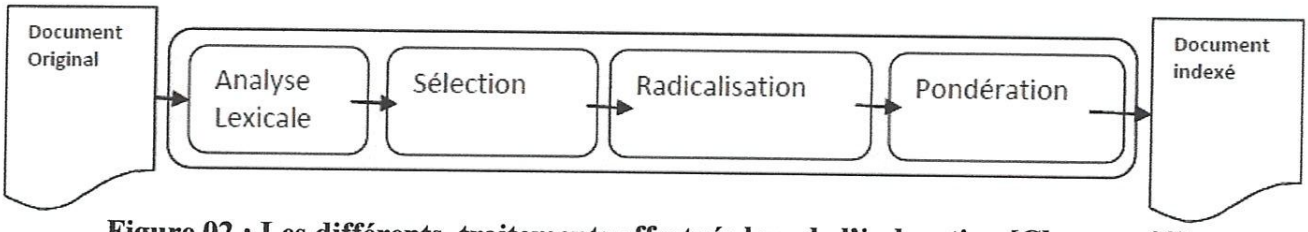


Figure 02 : Les différents traitements effectués lors de l'indexation [Champc, 09]

3.1.1. Le prétraitement linguistique (analyse lexicale)

C'est l'étape conduisant à la construction de l'index inversé et de la présentation documentaire à partir de collection de documents donnée [Amin.Gauss, 13].

Cette étape est généralement indépendante de la requête de l'utilisateur, mais elle dépend du type de source de données. Elle consiste à transformer la source de données brutes utilisées pour l'expansion des requêtes dans un format qui peut être traité plus efficacement par la suite. Dans le reste de cette section nous présenterons pour chaque source de données employée dans l'expansion la méthode de prétraitement adoptée.

Segmentation

La segmentation en anglais « tokenisation » [Amin.Gauss, 13] . Elle consiste à éliminer la répétition est on obtient un ensemble de mots à traiter, cette étape est difficile car une mauvaise segmentation a un impacte négatif direct sur les résultats de la recherche.

Une bonne segmentation dépend de la prise en comptes des spécifications de la langue des textes traités.

En effet, chaque langue a sa spécificité propre et une simple segmentation par des espaces et des signes de ponctuation conduite généralement à des résultats d'indexation médiocres, comme pour la langue chinoise, il y a pas des espace alors la segmentation devient difficile, alors que pour les langues indo-européennes la segmentation est plus aisée en présence des espace et des ponctuations.

Normalisation textuelle

La normalisation de mots est le processus qui transforme tous les mots d'une même famille sous une forme normale ou canonique de façon à ce que l'appariement entre les termes d'une requête et ceux de l'index puissent avoir lieu, et ce malgré les différences qui peuvent exister entre les séquences de caractères de leurs mots associés.

Le non appariement des termes de l'index et ceux des requêtes a été, depuis longtemps, un des obstacles majeurs à l'amélioration des performances des moteurs de recherches.

La normalisation textuelle rend les mots d'une même famille sous leur forme canonique en effectuant quelques transformations superficielles sur les séquences de caractère de ces mots.

Elle est dépendante de la langue. En français, les règles communément utilisées sont: les ponctuations, la casse, les accents, les dates et les valeurs monétaires.

3.1.2. La sélection ou filtrage par un anti-dictionnaire

Un anti-dictionnaire (stoplist) ou liste de mots vides, est une liste de mots qui tendent à être avec une fréquence élevée dans tous les documents d'une collection et qui n'apportent que peu d'information sur le contenu d'un document.

C'est une liste de mots vides qui contient généralement les articles, pronoms, prépositions, les mots outils, ainsi que les mots athématiques c'est-à-dire présents dans le document pour l'introduire ou le présenter mais n'ayant pas de réels rapports avec le sujet traité.

Le traitement lié à un anti-dictionnaire est très simple. Quand un mot est rencontré dans un texte à indexer, s'il apparaît dans l'anti-dictionnaire, il n'est pas considéré comme un index.

La suppression des mots vides doit être contrôlée car elle influence la qualité de la recherche. Il est évident que le rôle d'un mot dans un document dépend du contexte dans lequel il est employé, et qu'il peut avoir un pouvoir d'information différent dans un autre contexte.

Ainsi, un anti-dictionnaire devrait être dépendant de la collection c'est-à-dire constitué en fonction de la collection et mis à jour avec de nouveaux documents.

Néanmoins, en pratique on utilise souvent un anti-dictionnaire clef en main, par exemple l'anti-dictionnaire du système SMART [Salton, 88].

3.1.3. La radicalisation ou La normalisation linguistique

La radicalisation consiste à ramener un mot fléchi sous sa forme canonique.

Dans la littérature, une différence morphologique entre racine et radical est faite : la racine est la forme abstraite servant de base de représentation à tous les radicaux qui en sont les manifestations. En effet le radical d'un mot est une simple réduction du nombre de lettres de ce mot, et celui-ci peut différer (avoir plus ou moins de lettres) que la racine morphologique correcte.

Il existe deux types de radicalisation qui sont **[Amin.Gauss, 13]**:

a- La racinisation : procédé qui cherche à regrouper les différentes variantes morphologiques d'un mot autour d'une même racine, par un ensemble de règles pour supprimer les flexions et les suffixes des mots.

Par exemple, les mots suivant ont la même racine morphologique

danser, danse, danses, dansent, dansé, dansée, dansant => ***danse***

Néanmoins, la racinisation peut aussi ramener des mots à des formes qui ne sont pas de vrais mots :

cheval, chevalier, chevaux, chevalerie, chevaucher => *cheva*

En RI, le but de la racinisation est de trouver les formes fléchis des mots d'une requête. Mais, le risque est qu'on peut aussi ramener des documents qui contiennent les mots ayant les mêmes racines que les mots de la requête mais qui sont sémantiquement totalement différents de ces derniers.

b- La lemmatisation : procède qui fait une analyse linguistique poussée destinée à enlever les variantes flexionnelles des mots afin de les ramener sous leur forme lemmatisée ou encyclopédique.

Pour les verbes, on doit obtenir la forme à l'infinitif, pour les adjectifs la forme masculin singulier et pour les noms la forme singulier.

A l'opposition de la racinisation, le résultat de lemmatisation n'est qu'autre que les mots de l'encyclopédie.

Pour l'exemple précédent, *chevaux* sera agrégé à *cheval* mais pas à *chevaucher* ou *chevalerie*

Les algorithmes qui permettent de ramener un mot à un radical sont appelés des algorithmes de radicalisation et les algorithmes qui permettent de ramener un mot à un radical particulier qu'est sa racine des Lemmatisers.

Les algorithmes de radicalisation peuvent être :

Linguistiques, parmi eux citons [Porter, 80].

Automatiques quand ils se basent sur des méthodes statistiques comme par exemple les n-grammes [Adamson, 74].

un mélange des deux comme [Krovetz, 93][Paice, 96].

Ils peuvent également se baser sur des lexiques afin de valider ou d'invalider une tentative de transformation d'un mot en radical [Savoy, 93].

Les algorithmes de suppression des affixes (préfixes et suffixes) sont les plus fréquents [Frakes, 1996].

Dans notre travail, On va utiliser l'algorithme de Porter.

L'idée de l'algorithme de Porter est de ramener un mot de langue anglaise à un radical en supprimant sa terminaison. Pour cela il applique successivement plusieurs règles de transformation visant à supprimer le pluriel, les participes passés puis les différentes dérivations telles que «able», «ness», « tly ».

3.1.4. La pondération

La pondération d'un terme d'indexation est l'association de valeurs numériques à ce terme de manière à représenter son pouvoir de discrimination pour chaque document de la collection. Cette caractérisation est liée au pouvoir informatif du terme pour le document donné.

La fréquence relative d'un terme dans un document est représentative du pouvoir de représentation du terme pour le document, dans le même temps, la fréquence absolue d'un terme dans la collection est caractéristique du pouvoir de discrimination du terme pour les documents.

Il est donc important de prendre en compte la fréquence relative et la fréquence absolue d'un terme lors de sa pondération.

La pondération c'est l'association d'une valeur appelée poids à un terme.

Les mesures de pondération les plus répondues dans la littérature sont :

- La loi de Zipf [Zip, 49], elle considère que les termes des documents s'organisent suivant une loi inversement proportionnelle à leur fréquence d'apparition dans le corpus appelé rang.
- La conjecture de Luhn [Lhun, 57], elle repose sur le principe d'éliminer les termes ayant un rang inférieur à un seuil minimal ou supérieur à un seuil maximal et de ne maintenir que les termes ayant un rang intermédiaire. L'idée sous-jacente est que les rangs faibles sont affectés souvent aux mots vides de sens marquant une fréquence d'apparition élevée dans le corpus alors que les rangs élevés sont attribués aux

termes rarement apparus dans le corpus et dans les deux cas de figure, ces termes sont vus comme peu pertinents.

- le TF.IDF [Sal, 68; Spa, 79] est le schéma de pondération le plus utilisé, il combine deux facteurs de pondération sont : le facteur TF (Term Frequency), qui mesure la fréquence d'un terme dans le document où il apparaît [Sal, 68] et le facteur IDF (Inverse Document Frequency), qui mesure sa fréquence dans tout le corpus [Spa, 79].

Malgré ces limitations, cette représentation est largement utilisée en catégorisation et partitionnement de documents ainsi que dans les modèles vectoriels de recherche.

Elle est de plus très similaire à celle utilisée dans les modèles probabilistes de recherche d'information et présente l'avantage d'être facilement manipulable et exploitable par des techniques statistiques génériques.

Quand un SRI retourne un document à l'utilisateur, celui-ci en tire de l'information. Cette information a un sens pour un utilisateur donné, il se peut que la même information sémantique puisse prendre une importance plus ou moins grande, susciter un intérêt plus ou moins vif selon les individus et les circonstances. L'information a donc un sens pour un utilisateur donné qui lui attribue une certaine valeur.

3.1.5. L'indexation inversée

L'utilisation d'une structure qui fait correspondre chaque terme du vocabulaire à la liste des documents qui le contiennent est la façon la plus rapide pour trouver un terme d'une requête donnée dans une collection de documents. Cette structure s'appelle un index inversé.

Les deux types de structure de données les plus utilisées pour la construction de l'index inversé sont des tables de hachage ou des arbres.

Avec les arbres, il est possible de mener des recherches sur des termes tronqués (par exemple par leur préfixe) alors que les tables de hachage ne donnent pas cette possibilité.

Par contre, la consultation sur les tables de hachage est beaucoup plus rapide.

a- Indexation dans les collections statiques :

Dans les collections statiques, les documents ne sont pas modifiés, supprimés, ajoutés au cours du temps.

Il existe différentes étapes de création d'index inversé [Amin.Gauss,13] pour une collection statique qui sont:

- L'extraction de paires, en opérant une passe complète sur la collection. Ces identifiants sont généralement des clés numériques uniques associées à chaque terme et chaque document de la collection.
- Le tri des paires suivant les clés d'identifiants de documents
- Le regroupement des paires : construire pour chaque identifiant de terme, la liste des identifiants de documents dans lesquels le terme apparait. On peut ajouter à cette liste d'autres informations comme le ***df*** et le ***trf.***

Indexation par bloc à base de tri : La phase la plus complexe est la phase de tri qui pour de grandes collections de documents ne peut être réalisé en mémoire.

Dans ce cas, pour éviter de stocker la liste complète des paires à trier sur le disque et d'avoir recours à des accès disque avec de temps important, on stocke les paires d'identifiants (terme, document) par bloc en mémoire et on effectue le tri avant de passer au bloc suivant.

L'index inversé est obtenu en fusionnant toutes les listes d'identifiants résultantes après balayage du document complet stocké sur le disque.

Indexation distribuée : elle est utilisée pour de documents volumineux, ou l'indexation ne peut être effectuée sur une seule machine.

On utilise le cluster de calcul et de traitement de données en mode distribué pour construire l'index inversé des termes. Il peut être divisé en deux étapes :

- La collection de données est divisée en différentes parties dont chacune peut être traitée rapidement par un ordinateur appelé analyseur
- Chaque intervalle de paires d'identifiant de termes et de documents est ensuite assigné par un serveur à des ordinateurs qui vont trier les paires suivant la clé d'identifiant de terme de documents. En regroupant l'information dans les paires triées, des listes correspondant aux identifiants de termes sont ensuite écrites sur le disque pour chaque intervalle [Amin.Gauss, 13].

b- Indexation dans les collections dynamiques :

Elle consiste à actualiser l'index après ajouts, modifications ou suppressions des documents d'une collection dynamique.

Une solution triviale à ce problème est de construire périodiquement un nouvel index et de transférer la recherche sur ce dernier une fois sa construction terminée.

L'inconvénient majeur de cette stratégie est qu'elle est couteuse en termes de temps, alors la performance d'un système de recherche à base de cette stratégie peut se dégrader rapidement pendant le laps de temps qui sépare la mise à jour de la collection et celle de l'index.

Deux méthodes utilisant l'indexation dynamique ont été proposées qui sont :

- Technique de mise à jour directe sur place : elle ajoute directement les nouvelles listes d'identifiants de documents existants dans l'index en cours. Mais, il faut s'assurer que l'espace disque disponible est suffisant pour contenir la nouvelle liste sinon l'ancienne liste concaténée à la nouvelle doit être déplacée vers un nouvel emplacement où le tampon de la mémoire contenant la nouvelle liste puisse être enregistré.

La complexité de cette stratégie réside donc dans sa gestion de l'espace libre sur le disque.

- Technique de fusion : elle consiste à garder un index axillaire en mémoire à coté de l'index principal et fusionner les deux, une fois que la taille de l'index auxiliaire devient trop grande.

Après la fusion, un nouvel index sera entièrement crée sur le disque et l'ancien index ainsi que l'index auxiliaire en mémoire seront supprimés.

Cette stratégie conserve un bit d'invalidation sur les documents supprimés et utilise cette information pour présenter ou non un document dans la liste retournée correspondant à une recherche.

L'avantage de cette stratégie, est que lors de la fusion, l'ancien index inversé peut être sollicité pour répondre à des requêtes ; mais son inconvénient est la grande complexité dans la maintenance et la mise en œuvre, car à chaque mise à jour la totalité de l'index inversé sera lu et écrit sur le disque [Amin.Gauss, 13].

3.2. Les modèles de recherche d'information

Dans la RI, un modèle permet de créer une représentation interne d'un document ou d'une requête, afin de pouvoir créer une correspondance entre les deux.

Un Ri est représenté par un ensemble de documents, d'une liste de requêtes, d'un schéma de représentation des documents et des requêtes ainsi que les relations qui leur ont associé, et d'une fonction d'évaluation de la pertinence des résultats.

L'indexation choisit les termes pour représenter le contenu d'un document ou d'une requête, le modèle permet de donner une interprétation des termes choisis pour représenter le contenu d'un document.

Les modèles sont des programmes qui aident les utilisateurs à trouver les informations qu'ils cherchent dans une collection de documents textuels ou éventuellement multimédias

Étant donné un ensemble de termes pondérés issus de l'indexation, le modèle remplit deux fonctionnalités:

- La première est de créer une représentation interne pour un document ou pour une requête basée sur ces termes,
- La seconde est de définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (ou similarité).

Le modèle joue un rôle central dans la RI. C'est lui qui détermine le comportement clé d'un SRI.

De nombreux modèles existent (Figure 03). On présentera d'abord le modèle booléen qui est historiquement un des premiers modèles étudiés et qui a servi de point de départ aux recherches du domaine puis le modèle vectoriel (approche algébrique) qui sert de base à notre modèle (approche basée sur les graphes) et enfin, le modèle probabiliste qui, classifie les documents en fonction de leur probabilité de pertinence pour la requête.

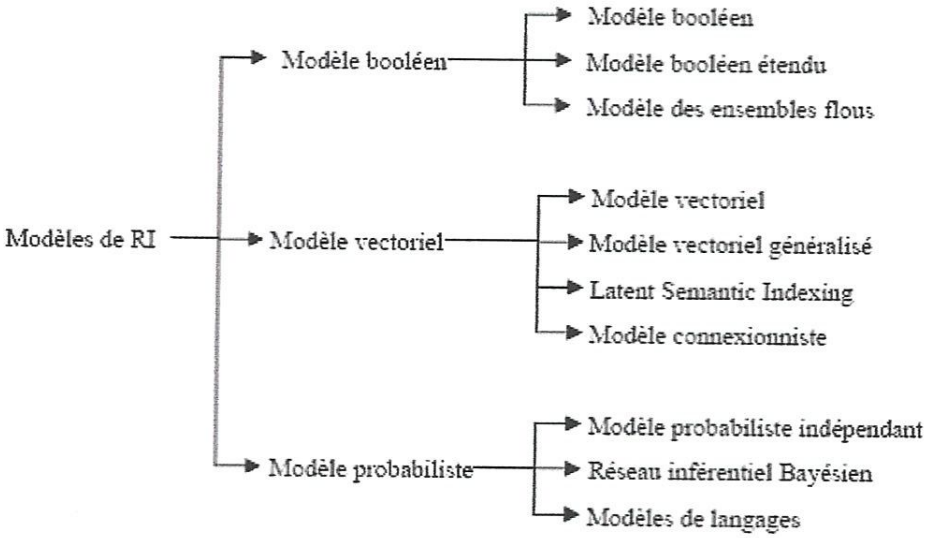


Figure 03: différents modèles de la recherche d'information [Boub, 08]

3.2.1. Les modèles booléens

Les premiers RI développés sont sur le modèle booléen, il est basé sur la théorie des ensembles et l'algèbre de Boole.

Pendant plusieurs années, et ce jusqu'aux années 1990, ces modèles ont constitué le cœur des systèmes de recherche d'information commerciaux.

Ils ont été conçus de façon à répondre exactement aux requêtes formées par des expressions combinant des termes et des operateurs logiques ET, OU et SAUF.

Une façon efficace de traiter ce genre de requête est d'utiliser la structure d'index inversé.

Dans ce cas la recherche booléenne se réduit simplement à parcourir les listes de documents associés à chacun des termes de la requête, et de fusionner ensuite ces listes par rapport aux opérateurs présents dans cette dernière.

Ainsi la requête *information* définit simplement l'ensemble de tous les documents indexés avec le terme *information*.

Les requêtes peuvent être composées de plusieurs termes reliés entre eux par des opérateurs de la logique booléenne.

Une requête combinant deux termes reliés par un ET retrouvera un ensemble de documents inférieur ou égal à l'ensemble des documents restitués par chacun des termes pris séparément.

Par exemple la requête *information* ET *renseignement* retrouvera les documents qui ont été indexés avec les deux termes ; le résultat est donc l'intersection des deux ensembles.

Une requête combinant deux termes reliés par un OU retrouvera un ensemble supérieur ou égal à l'ensemble des documents restitués par chacun des termes pris séparément.

Par exemple la requête *information* OU *renseignement* retrouvera les documents indexés avec *information* ou *renseignement* (ou les deux), le résultat est l'union des deux ensembles.

La figure 04 représente les documents restitués (parties grisés) pour différentes requêtes :

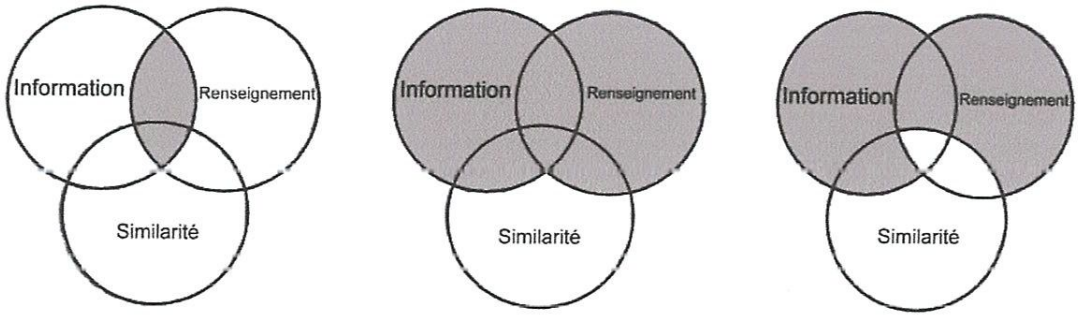


Figure 04: requêtes booléennes sous forme de diagramme dc Vcnn [Champc, 09]

a- Le modèle booléen de base (boolean model)

Basé sur la théorie des ensembles, le document est représenté par un ensemble de termes.

La requête est représentée par un ensemble de mots clés reliés par des opérateurs booléens (ET, OU et SAUF).

Dans ce modèle la recherche booléenne se réduit simplement à parcourir les listes de documents associés à chacun des termes de la requête, et de fusionner ensuite ces listes par rapport aux opérateurs présents dans cette dernière.

Une façon efficace de traiter ce genre de requête est d'utiliser la structure d'index inversé. L'avantage de ce modèle de base est sa simplicité. Par contre les documents qui répondent à la requête sont retournés dans un ordre quelconque.

b- Le Modèle booléen étendu (extended boolean model)

Le modèle booléen étendu a été introduit par Salton [Salton et al.,1983]. C'est une extension du modèle précédent qui vise à tenir compte d'une pondération des termes dans le corpus. Cela permet de pallier les problèmes du modèle de base en ordonnant les documents retournés par le SRI.

La requête demeure une expression booléenne classique. Tandis que les termes d'un document sont maintenant pondérés.

En général, le poids d'un terme dans un document est en fonction du nombre d'occurrences de ce terme dans le document.

L'appariement requête-document est le plus souvent déterminé par les relations introduites dans le modèle p-norm basées sur les p-distances, avec $1 \leq p \leq \infty$. La valeur de p est indiquée au moment de la requête.

c- Modèle flou (fuzzy set model)

Il s'appuie sur la théorie des ensembles flous proposé par Zadeh en (1965) [Zad, 65]. Dans ce modèle, le poids affecté à un terme reflète la mesure dans laquelle ce terme décrit le contenu du document où il apparaît. L'intégration de cette théorie dans la RI a permis de traiter l'ambigüité des requêtes, l'imprécision qui caractérise le processus d'indexation ainsi que la divergence de pertinence entre les documents retournés. L'inconvénient sur les modèles flou est qu'ils ne permettent pas l'ordonnancement des résultats selon leur pertinence.

Les avantages du modèle booléen :

- Facile à implémenter.
- Tout à fait fonctionnel.
- Il permet aux utilisateurs d'exprimer des contraintes structurelles et conceptuelles.
- Il aux utilisateurs l'utilisation de synonymes (grâce à la clause OR) et de groupes de mots (grâce à la clause ET) qui sont utiles pour la formulation de la requête.
- L'approche booléenne possède un grand pouvoir d'expressivité : elle est tout à fait adaptée aux requêtes qui appellent une sélection exhaustive et non ambigüe.

Les inconvénients du modèle booléen :

- Difficile pour un utilisateur non expert de formuler des requêtes adéquates à l'aide d'expressions booléennes

Par exemple l'expression « A et B » laisse penser, dans le langage courant, que ce que représentent A et B est quelque chose de plus que A seul. Or, en logique booléenne le ET logique entre deux ensembles représente leur intersection, c'est-à-dire une partie

commune. Inversement le OU logique (OR) exprime la somme des ensembles alors que le OU de la langue suggère plutôt un choix : l'un ou l'autre.

En plus d'éviter la confusion entre le ET de la langue et le OR logique lors du passage de la requête en langage naturel à la requête booléenne, un utilisateur qui souhaite formuler des requêtes complexes devra se familiariser avec le concept de priorité et l'utilisation de parenthèses et de parenthèses imbriquées.

- Le fonctionnement des opérateurs ET et OU pose des problèmes : le ET logique ne fait aucune différence entre deux cas pourtant distincts : si aucun terme ne satisfait la requête ou si tous sauf un satisfont la requête l'opérateur ne retrouve pas de document (Null Output problem).

Symétriquement, un usage trop fréquent de OR ramène trop de documents (Overload Output problem).

3.2.2. Le modèle vectoriel

En 1957, Luhn [Lhun, 55] a proposé de représenter les textes (documents et information recherchée) sous forme de vecteurs pondérés et de procéder à un calcul statistique.

Au début des années 70, Salton [Sal, 71] [Sal, 75] a développé cette idée, en proposant le modèle SMART (Salton's Magical Automatic Retriever of Text).

Le modèle vectoriel est un modèle algébrique où l'on représente les documents et les requêtes par des vecteurs de poids dans un espace multidimensionnel dont les dimensions sont les termes issus de l'indexation [Salton et al., 1983]

Dans ce modèle, le sens d'un document est donné par les mots qui y figurent. Un coefficient de similarité est calculé pour restituer les documents dont le contenu (termes) correspond au contenu de la requête. Une mesure de distance calcule l'angle entre les deux représentations.

La pertinence d'un document vis à vis d'une requête est définie par des mesures de distances entre vecteurs.

La pondération des termes (L'importance d'un terme dans un document) dépend de trois choses : l'importance du terme dans le document (pondération locale), l'importance du terme dans la collection (pondération globale), l'importance du document (normalisation en fonction de la taille du document)

Plusieurs modèles proposés en RI se basent sur le modèle vectoriel, dont: le modèle connexionniste et le modèle LSI (Latent Semantic Indexing).

a- Le modèle vectoriel de base

Ce modèle est représenté sous forme d'un vecteur dans l'espace vectoriel composé de tous les termes d'indexation. Les coordonnées d'un vecteur document représentent les poids des termes correspondants. Le problème avec ce type de modèle est que les termes qui constituent le vecteur n'ont pas lien entre eux (pas de co-occurrence).

b- L'indexation sémantique latente (Latent Semantic indexing)

Ce modèle apporte une solution au problème d'indépendance de termes repéré dans le modèle vectoriel de base par la réduction de l'espace dimensionnel de représentation des documents en reliant les termes afin de traiter les "concepts" plutôt que les mots simples. Les recherches menées sur le modèle d'indexation sémantique latente (LSI) peuvent renvoyer des documents ne contenant aucun mot de la requête, par contre ils contiennent la même sémantique émergée à partir de la structure globale des documents et les relations entre les termes qu'ils incluent.

Les avantages du modèle vectoriel

- Simplicité conceptuelle et de mise en œuvre.
- Il permet de trier les résultats d'une recherche à travers une mesure de similarité document/requête, en plaçant en tête les documents jugés les plus similaires à la requête.
- Un appariement partiel qui permet de retrouver les documents qui répondent en partie à la requête

Les inconvénients du modèle vectoriel

- Difficulté d'aller plus avant dans le cadre vectoriel (modèle relativement simple)
- Ce modèle ne permet pas de modéliser les associations entre les termes
- Complexité : La similarité requête-document est plus coûteuse

3.2.3. Modèle probabiliste

La recherche d'information traite une information incertaine dans la représentation d'un document, la représentation de la requête, la mise en correspondance de la requête et du document et la pertinence du résultat par rapport à la requête.

Les probabilités sont des outils naturels pour essayer de quantifier l'incertitude, Dans ce cas, l'utilisation des modèles probabilistes s'avère essentielle.

Les modèles probabilistes se basent sur la distribution théorique des mots dans un document. Ils retournent une liste ordonnée de documents et l'ordre des documents est plus important que leur score qui est fonction de sa probabilité de pertinence.

D'autres part, dans ces modèles, la pertinence reste une notion floue : elle est indépendante des jugements portés sur les autres documents, et l'utilité d'un document pertinent ne dépend pas du nombre de documents pertinents que l'utilisateur a déjà obtenus.On utilise la présence ou l'absence d'un terme dans les documents pertinents ou non pertinents.

Pour ce qui est de l'indexation probabiliste (indexation binaire des termes), le choix de retenir ou non un terme d'indexation doit être lié à la probabilité qu'un utilisateur désirant ce document écrive ce terme dans la requête.

Avantages du modèle probabiliste :

- Une notion claire et théoriquement fondée du degré de pertinence
- Sensibilité aux valeurs initiales
- Un cadre théorique clair et bien fondé, facile à implanter et conduisant à de bons résultats
- Facile à étendre à d'autres cadres (recherche d'information multilingue, boucle de pertinence, ...)
- le modèle de recherche probabiliste est plus efficace que le modèle de recherche booléen, mais moins performant que le modèle de recherche vectoriel [Savoy, 93].

Inconvénients du modèle probabiliste :

- Procédure d'estimation certes intéressante mais coûteuse
- Le processus de recherche d'information est un processus itératif qui implique l'utilisateur
- Complexité similaire au modèle vectoriel à chaque itération ; un peu plus complexe en général
- Estimation des nécessite des données d'apprentissage

- Difficulté (conceptuelle) d'une boucle de rétro-pertinence ((pseudo-) relevance feedback)

3.3. Evaluation d'un système de recherche d'information

Depuis la naissance du domaine de la RI, l'évaluation des modèles et méthodes proposés a toujours été un centre d'intérêt important.

Cette étape permet de mesurer les caractéristiques du système en termes de qualité, de service et de facilité d'utilisation.

Cleverdon [Cleverdon, 70] définit six principales mesures de la qualité d'un SRI :

- l'univers du discours de la collection
- le temps de réponse
- la présentation des résultats
- l'effort requis de l'utilisateur pour retrouver, parmi les documents retournés, ceux qui répondent à son besoin
- le taux de rappel du système
- la précision du système.

Les mesures de rappel et de précision sont intrinsèques au modèle de recherche du système et couvrent une pertinence, dépendant de nombreux critères examinés liés au contexte de la recherche, tels que : le degré de correspondance (aboutness), l'utilité (usefulness/ utility), la rentabilité (usability), l'importance, … sur les résultats retournés par rapport aux objectifs, aux intérêts et à la situation intrinsèque du moment.

Ces différents critères ont amené à la catégorisation de la pertinence utilisateur en quatre classes de pertinence qui sont : [Saracevic, 96]

1. la pertinence thématique traduit le degré d'adéquation de l'information retrouvée au thème (et non au contenu) de la requête.

   C'est la pertinence classique telle qu'elle est définie dans le paradigme de Cleverdon [Cleverdon, 70],
2. la pertinence cognitive représente la relation intellectuelle entre le besoin informationnel intrinsèque de l'utilisateur et l'information portée par les documents telle qu'interprétée par l'utilisateur,
3. la pertinence situationnelle est vue comme l'utilité de l'information retrouvée par rapport au but de la recherche tel que par l'utilisateur

4. la pertinence motivationnelle ou affective décrit la relation entre les intentions, les buts, et les motivations de la recherche tels que fixés par l'utilisateur d'une part et les informations retrouvées d'autre part.

La mesure communément utilisée dans les campagnes d'évaluation classique en RI est sans doute la pertinence thématique.

On adopte pour cela, une approche quantitative des SRI qui s'attache à mesurer le degré d'adéquation du document et la requête.

Pour mesurer cette adéquation, le SRI procède à la comparaison de la représentation interne de la requête et de la représentation interne des documents.

Le degré de similitude du document et de la requête mesure la pertinence du document pour cette requête. Il s'agit là de la pertinence système, ou pertinence algorithmique [Saracevic, 96] (ou pertinence logique [Cooper, 71]).

Pour cela des mesures de qualité des systèmes de recherche ainsi que des ensembles de données ont été développés afin de tester ces systèmes sur une base commune.

La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses que l'utilisateur espère. Plus les réponses du système correspondent à celles que l'utilisateur espère, meilleur est le système.

3.3.1. Collection de test

Une collection de tests comprend :

1. Un ensemble de documents (ou collection de documents) à indexer, sur lesquels le système sera évalué,
2. Une liste de requêtes prédéfinies,
3. Les jugements de pertinence, manuellement établis, pour chaque requête.

3.3.2. Mesures d'évaluation

Traditionnellement, la pertinence des résultats d'une recherche se mesure en comparant les réponses du système avec les réponses idéales que l'utilisateur s'attend à recevoir sur la base de certaines mesures. L'étude présentée par [Bac, 10] définit plus de 20 mesures d'évaluation.

Les mesures les plus utilisées sont :

- *Le rappel* mesure la capacité du système à retrouver tous les documents pertinents à une requête.
- *La précision* mesure la capacité du système à ne pas classer les documents non pertinents à une requête.

- *La précision à X documents* qui représente le nombre de documents pertinents sur les X premiers sélectionnés
- *La F-mesure (F-score)* traduisant l'importance relative du rappel et de la précision.

- *La moyenne harmonique du rappel et de précision* qui est calculé par la formule

$$F = \frac{Precision \times Rappel}{Precision + Rappel}$$

Équation 01 : Calcul de moyenne harmonique du rappel et de précision

- *La précision moyenne (ou MAP pour Median Average Precision),* qui représente la moyenne des précisions calculées pour chaque document pertinent dans la liste ordonnée du résultat au rang de ce document [Champc, 09].

4. CONCLUSION

Dans ce chapitre nous avons vu les différents concepts de base de la RI, ainsi que les grandes étapes du processus de la recherche d'information, à savoir l'indexation, les modèles de recherche d'information et l'évaluation.

Dans le prochain chapitre on se focalisera sur l'une des étapes de l'indexation qui est la *radicalisation*. On verra dans ce cadre plusieurs algorithmes qui permettent l'extraction des radicaux.

CHAPITRE II : LES ALGORITHMES DE RACINISATION

CHAPITRE II : LES ALGORITHMES DE RACINISATION

1. INTRODUCTION

La racinisation (dé-suffixation ou stemmatisation) (note de bas de page : stemming en anglais) est le nom donné au procédé qui vise à transformer les flexions en leur radical. Il cherche à rassembler les différentes variantes flexionnelles et dérivationnelles d'un mot autour d'un radical. La racine d'un mot correspond à la partie du mot restante, à savoir son radical, après suppression de son (préfixe, suffixe). Elle est aussi parfois connue sous le nom de *stem* d'un mot. Contrairement au lemme qui correspond à un mot réel de la langue, la racine ou stem ne correspond généralement pas à un mot réel. Par exemple, le mot "chercher" à pour radical ou stem "cherch" qui ne correspond pas à un mot réel. Par contre, dans l'exemple de " "frontal", le radical ou stem est "front" qui lui l'est.

La racinisation est une phase importante pour le processus d'indexation. Des algorithmes de dé-suffixation ont été développés pour traiter efficacement certains problèmes (tels que le manque de précision). Ces algorithmes ont pour but de dégager les racines des mots en passant par un ensemble des règles et de conditions.

2. LES ALGORITHMES DE RACINISATION

Les algorithmes qui extraient le radicale d'un mot sont appelés des algorithmes de racinisation ou encore un racinisateur (en anglais 'stemmer').

Ces algorithmes de racinisation procèdent en deux étapes : un pas de dé-suffixation qui consiste à ôter aux mots des terminaisons prédéfinies les plus longues possibles, et un pas de recodage qui ajoute aux racines obtenues des terminaisons prédéfinies.

L'algorithme de Lovins fait les deux étapes séparés, mais celui de Porter fait les deux étapes en simultané.

Deux principales familles de stemmers sont présentes dans la littérature : les stemmers algorithmiques et ceux utilisant un dictionnaire.

Un stemmer algorithmique va être souvent plus rapide et va permettre d'extraire des racines de mots inconnus (en un sens, tous les mots qu'il rencontre lui sont inconnus). Il va cependant avoir un taux d'crreur plus élevé, groupant ensemble des mots qui ne devraient pas l'être (sur-racinisation, ou over-stemming).

L'approche par dictionnaire quant à elle ne fait pas d'erreur sur les mots connus, mais en produit sur ceux qu'elle ne liste pas. Elle est aussi plus lente, et nécessite malgré tout la suppression de suffixes avant d'aller chercher la racine correspondante dans le dictionnaire.

Parmi ces algorithmes on trouve des algorithmes linguistiques comme PORTER [Porter,80], PAICE [Paice,90], algorithmes automatiques quand ils se basent sur des méthodes statistiques comme les n-grammes [Adamson,74] ou être un mélange des deux comme Krovetz, Paice, Carry [Krovetz,93], [Paice,96]et [carry, 02]. Ils peuvent également se baser sur des lexiques afin de valider ou d'invalider une tentative de transformation d'un mot en radical [Savoy,93].

Les algorithmes d'élimination des affixes (préfixes et suffixes) sont les plus fréquents [Frakes,96].

2.1. Algorithme de Lovins

2.1.1. Définitions :

L'algorithme de lovins est le premier racinisateur qui a été créé par Julie Beth Lovins [lovins, 68] en 1968. La conception de l'algorithme a été beaucoup influencée par le vocabulaire technique utilisé durant son travail .L'algorithme est légèrement limité pour certaines terminaisons communes qui ne sont pas représentés (par exemple 'ements' et 'ents', correspondant aux formes du singulier 'ement' et 'ent'), et aussi dans le traitement des mots courts, ou des mots avec des stems courts.

L'algorithme de Lovins a 294 terminaisons, 29 conditions et 35 règles de transformation et où chaque terminaison est associée à l'une des conditions. Dans la première étape, si la plus longue terminaison trouvée satisfait sa condition qui lui est associée, cette terminaison sera éliminée. Dans la deuxième étape, les 35 règles sont appliquées pour transformer la terminaison. La seconde étape est effectuée si une terminaison est supprimée dans la première étape ou non.

2.1.2. Limites de l'algorithme de lovins

Durant la phase de traitement et d'analyse du corpus ainsi que le résultat fourni par l'algorithme « LOVINS », un ensemble de mots qui ont le même contexte (sens) mais de différents genres, nombres (singulier, pluriel, féminin, masculin) et nature (nom, adjectif, adverbe…), qui doivent avoir les mêmes stems, certains n'ont pas été trouvé par l'algorithme Lovins.

2.2. L'algorithme de Paice et Husk

2.2.1. Définitions :

Cet algorithme a été développé par Chris Paice à l'université de Lancaster dans les années 80. Ensuite, il a été codé en Pascal, C, PERL et Java. Il appartient à la famille des stemmers algorithmiques.

Il se base sur un ensemble de règles pour extraire les racines [paice, 94]. Et de plus stocke ces règles en dehors du code. Ainsi, il est possible de traiter, de la même façon, une nouvelle langue à partir d'un autre ensemble de règles sans réécrire le code, moyennant quelques ajustements (pour chaque langue, la liste des voyelles acceptées et les règles de validité des racines doivent être fournies). L'algorithme est plus facilement portable à la gestion d'une nouvelle langue.

L'implémentation de l'algorithme de Paice et Husk [Paice, 90] est composée d'un ensemble de fonctions qui vont utiliser les règles d'extraction de racines applicables au mot fourni en entrée et vérifier l'acceptabilité de la racine proposé, ainsi que l'ensemble de règles de réécriture.

2.2.2. Limites de l'algorithme de Paice et Husk

Paice et Husk remarquent que les séquences qui introduisent les it impersonnels partagent une forme remarquable (elles commencent par un *it* et se terminent par un délimiteur comme *to, that, whether*...) même si le détail des règles varie d'un délimiteur à l'autre. Les contraintes portent sur le contexte gauche (le pronom ne doit pas être précédé immédiatement par certains mots comme *before, from, to*...), sur le nombre de mots séparant le pronom et le délimiteur (ce nombre doit être inférieur à 25 mots), et enfin sur les éléments lexicaux présents entre le pronom et le délimiteur (la séquence doit contenir ou ne pas contenir certains mots appartenant à des classes connues de mots, par exemple les mots exprimant une modalité propositionnelle, (e.g. *certain, known, unclear*...). Les tests réalisés par Paice montrent que ces règles réalisent un bon score avec 91.4%Acc sur un corpus technique. Cependant leurs performances se dégradent si on les applique à des corpus de natures différentes. On observe que le nombre de Faux Positifs augmente. Cette augmentation est dûe à une erreur d'estimation, les attributs qui sont discriminants sur les corpus techniques ne l'étant plus dès lors que la nature du corpus change [Weiss, 08].

2.3. Algorithme de Krovetz

2.3.1. Définitions :

L'algorithme de Krovetz [Krovetz ,93] qui a été aussi développé par Bob Krovetz, à l'Université du Massachusetts, en 1993. C'est un racinisateur léger, car il fait appel à la morphologie linguistique flexionnelle

L'algorithme de Krovetz supprime précisément les suffixes :

- La conversion d'un pluriel à sa forme singulière (par exemple '*-ies*', '*-es*', '*-s*').
- La conversion de passé au temps présent (par exemple '*-ed*').
- L''élimination des "*in*g" puis à travers un processus de vérification dans un dictionnaire pour tout recodage (aussi bien connaître des exceptions aux règles normales recodage), retourne le stem de mot.

Krovetz est souvent utilisé en conjonction avec d'autres racinisateurs, en utilisant l'avantage de la précision de suppression de suffixes par cet algorithme, qui ajoute ensuite la compression d'un autre Racinisateur comme l'algorithme Paice / Husk ou Porter.

2.3.2. Limites de l'algorithme de Paice et Husk

C'est un algorithme de faible force et très compliqué en raison des processus impliqués dans la morphologie linguistique et sa nature flexionnelle.

Si la taille du document d'entrée est grande, cette source devient faible et ne fonctionne pas très efficacement.

Le défaut majeur et évident de cet algorithme qui basé sur le dictionnaire est sa incapacité à faire face aux mots, qui ne sont pas dans le lexique. En outre, un lexique doit être manuellement Crée à l'avance, ce qui nécessite des efforts considérables. Ce stemmer ne produit pas de bonnes performances de précision et de rappel.

2.4. Algorithme de EDA

EDA (Estimation of Distribution Algorithms) [Nakache, 06] est un algorithme de racinisation de corpus médical qui a été développé en 2006. L'algorithme EDA fonctionne sur deux phases principales : une phase de préparation et d'harmonisation de la forme suivi d'une phase de traitement. La première phase consiste à préparer le mot à raciniser en appliquant quelques modifications :

- Elimination de la casse.
- Séparation des caractères ligaturés ('*cœur*' devient '*coeur*') et des traits d'unions.

- Suppression des signes diacritiques (ex : accents *'dégénéré'* devient *'degenere'*).
- Suppression des lettres doublées.
- Remplacer *'ck'*, *'cqu'*, *'qu'*, et *'c'* par *'k'*, et *'y'* par *'i'*.

Cette première phase permet de nettoyer le terme et le mettre dans une forme 'standard'. La suppression des accents permet de regrouper de nombreux termes qui étaient considérés, au préalable, comme différents. Les trois dernières règles de préparation des données permettent uniquement de corriger les erreurs de frappe et les erreurs induites par le changement de casse. Le remplacement des lettres 'ck', 'cqu', 'qu', et 'c' par la lettre 'k', ainsi que 'y' par 'i' s'explique par le grand nombre de noms propres utilisés en médecine, avec des orthographes différentes.

L'objectif est donc de se rapprocher d'une prononciation phonétique.

La seconde phase consiste à exécuter séquentiellement un ensemble de 37 règles. Il est important de respecter ces règles dans l'ordre, jusqu'à ce que le mot résultant contienne 5 caractères ou, dans le cas contraire, jusqu'à la dernière règle. Chaque règle s'applique sur le résultat obtenu par la règle précédente.

Cet algorithme est très simple à implémenter et ne nécessite aucunc base de données, aucun traitement lourd, ni aucune mise à jour. Il est très rapide à implémenter et en exécution.

2.5. L'algorithme de Portcr

2.5.1. Définitions :

L'algorithme de Porter est l'un des plus célèbres algorithmes de normalisation des mots qui a été développé par martin PORTER [porter, 80] en 1979 au laboratoire informatique au sein de l'université du Cambridge (Angleterre). Il fait partie d'un grand projet dans le domaine de la recherche d'information. Il permet d'éliminer les affixes des mots pour obtenir une forme canonique du mot. Cet algorithme est utilisé pour la langue anglaise, mais son efficacité est limitée pour la langue française où les flexions sont plus importants et plus diverses. Il reste toutefois un algorithme fondamental couramment enseigné en TALN [Kabil, 13].

2.5.2. Principe de l'algorithme de Porter

Cet algorithme se compose d'une cinquantaine de règles de normalisation classées en sept phases successives [Porter, 80] (traitement des pluriels et verbes à la troisième personne du singulier, traitement du passé et du progressif,. . .). Les

mots à analyser passent par tous les stades et, dans le cas où plusieurs règles pourraient leur être appliquées, c'est toujours celle comprenant le suffixe le plus long qui est choisie. La normalisation est accompagnée, dans la même étape, de règles de recodage.

Ainsi, par exemple : "troubled" deviendra "troubl" par enlèvement du suffixe marqueur du progressif "-ed" et sera ensuite transformé en "trouble" par application de la règle "bl" devient "ble".

Cet algorithme comprend aussi cinq règles de contexte, qui indiquent les conditions dans lesquelles un suffixe devra être supprimé.

La terminaison en "ing", par exemple, ne sera enlevée que si le radical comporte au moins une voyelle. De cette manière, "writing" deviendra "writ", alors que "sing" restera "sing".

L'algorithme de Porter fonctionne sur les mots composés à leur plus bas niveau de lettres. L'ensemble des lettres de l'alphabet latin, les voyelles et les consonnes :

- Les voyelles (v) : la liste des voyelles est (A, E, I, O, U et le Y) le "y" on le considère comme une voyelle si son prédécesseur est une consonne.
- Les consonnes (c) : sont tous les lettre de l'alphabet latin sauf les voyelles et le 'y' si son prédécesseur est une consonne.

Quelques notations:

- V : une suite de voyelles.
- C : une suite de consonnes.
- m : un entier est appelée la mesure d'un mot.
- (VC)m : la suite "CV" se répète "m" fois.
- [C]: la suite des consonnes "C" peux être existe et peux être non.
- *S : signifie que le radical se termine par la lettre S
- *v* : signifie que le radical contient une voyelle
- *d : signifie que le radical se termine par deux consonnes
- *o : signifie que le radical se termine par la séquence cvc et que la dernière consonne n'est ni W, ni X et ni Y.

Un mot en anglais peut avoir l'une des 4 formes suivantes :

Forme 1 : CVCVC…C

Forme 2 : CVCVC…V

Forme 3 : VCVCV…C

Forme 4 : VCVCV…V

La formule générale est : $[C]\ (VC)^m\ [V]$.

Quelques exemples:

Si m=0: no, tree, toy, to...

T	R	e	E
C		V	
[C]		[V]	

m=1: in, nine, oats, trees, write, one...

m=2: troubles, private, writing, oaten, orrery …

W	R	i	T	I	N	g
C		V	C	V	C	
[C]		m=1		m=2		

2.5.3. L'algorithme :

- Entrées : fichier contenant l'ensemble des mots
- Sorties : fichier contenant l'ensemble des racines des mots

Quelques notations

- c : consonne
- e : voyelle
- *e : le préfixe se termine par la lettre e
- *v* : le préfixe contient une voyelle
- *d : le préfixe se termine par une consonne doublée
- *o : le préfixe se termine par cvc où le second c n'est ni w, ni x, ni y

DEBUT

Tant que (non fin fichier) faire

Étape 1 :

a. Enlever la forme plurielle

- SSES → SS
- IES → I
- SS → SS
- S →

b. Enlever "ed" et "ing" du verbe

- (m>0) EED → EE
- (*v*) ED →
- (*v*) ING →

c. si (existe voyelle dans le stem) alors

- Y → I

Fin si ;

étape 2 : Enlever la terminaison si (m>0)

- ATIONAL → ATE
- TIONAL → TION
- ENCI → ENCE
- ANCI → ANCE
- IZER→ IZE
- ABLI → ABLE
- ALLI → al
- ENTLI → ENT
- ELI → E
- OUSLI → OUS
- IZATION → IZE
- ATION → ATE
- ATOR → ATE
- ALISM → AL
- IVENESS → IVE
- FILNESS→ FUL
- OUSNESS → OUS
- ALITI → AL
- IVITI → IVE
- BILITI → BLE

étape 3 : si (m>0)

- ICATE → IC
- ATIVE →
- ALIZE → AL
- ICITI → IC
- ICAL → IC

- FUL →
- NESS →

étape 4 : (m>1)

- AL →
- ANCE →
- ENCE →
- ER →
- IC →
- ABLE →
- IBLE →
- ANT →
- EMENT →
- MENT →
- ENT →
- (*S ou *T) ION →
- OU →
- ISM →
- ATE →
- ITI →
- OUS →
- IVE →
- IZE →

étape 5

- (m>1) E →
- (m=1 et not *o) E →
- (m>1 et *d et *L) → lettre non doublée

Retourner (stem)

Fin tant que ;

Fin.

Algorithme 1 : Algorithme de Porter.

- La première étape de l'algorithme permet de supprimer la forme plurielle (Par exemple ; caresses→caress). Elle permet aussi de supprimer la terminaison des verbes 'ed' et 'ing'. Exemple : Plastered→Plaster
- La deuxième et la troisième étape permettent d'éliminer la terminaison si m>0

Exemple

(relational→relate, formaliti→formal)

(hopeful→hope, goodness→ good)

- La quatrième étape permet d'éliminer la terminaison si m>1 (Par exemple allowance→allow)
- La dernière permet d'enlever le 'e' si m>1 ou si (m=1 et le préfixe se termine par cvc où le second c n'est ni w, ni x, ni y). Elle permet aussi d'enlever la dernière lettre doublée si m>1 et si la lettre n'est pas 'L'.

Exemple :

1. **Généralisations :** En appliquant les différentes étapes de l'algorithme de Porter on aura :
   - étape 1: Generalizations → Generalization
   - étape 2: Generali**zation** → Generali**ze**
   - étape 3: Generali**ze** → Gener**al**
   - étape 4: Gener**al** → Gener
2. **Oscillators :** En appliquant les différentes étapes de l'algorithme de Porter on aura :
   - étape 1: Oscillators → Oscillator
   - étape 2: Oscill**ator** → Oscill**ate**
   - étape 4: Oscill**ate** → Oscill
   - étape 5: Oscill → Oscil

2.5.4. Limites de l'algorithme

Même si l'algorithme de Porter est connu comme un algorithme puissant, il a encore pas mal de problèmes, parmi lesquelles :

- les racines fournies par l'algorithme de Porter ne sont pas forcément de véritables morphèmes.
- Ne traite pas les verbes irréguliers.
- De nombreuses exceptions ne sont pas contrôlées : conjugaison des verbes, les noms possessifs, les formes irrégulières de comparatifs et superlatifs.
- Ne tient pas compte des mots composés.
- Plusieurs suffixes sont manquants. Par exemple : –man, -e, -ativistic, etc.

2.6. L'algorithme de carry

2.6.1. Définitions :

L'algorithme de Carry a été proposé pour l'étude de la morphologie de français, c'est la version améliorée de l'algorithme de Porter avec en plus les suffixe du français [carry, 02].

Tout comme celui de Porter, l'algorithme de Carry se déroule en diverse étapes par lesquelles les mots à traiter passent successivement. Selon les règles, quand l'analyseur reconnaît un suffixe de la liste, soit il le supprime, soit il le transforme. C'est ici aussi le suffixe le plus long qui détermine la règle à appliquer.

L'algorithme de Carry étant destiné au français, ils ont voulu l'appliqué aussi sur du texte en anglais afin de comparer les résultats obtenues avec Porter.

Les résultats de leurs tests sont présentés au tableau 01. Un résultat de 1 correspond à un groupement parfait entre les formes de base et les radicaux.

	Anglais (PORTER)	Français (Carry
Recall	0,972	0,917
Precision	0,777	0,905
Adjusted RAND Index	0,606	0,897

Tableau 01 : Recall, Precision et Adjusted RAND Index [carry, 02].

En effet, tout en étant simple et léger, l'algorithme de Carry permet de regrouper les mots liés sémantiquement presque aussi bien que ne le ferait un analyseur morphologique plus lourd et donc plus lent.

2.6.2. Limites de l'algorithme

Les résultats obtenus par l'algorithme Carry sont moins bons que ceux de Porter en termes de rappel mais meilleurs en termes de précision. Cependant, la qualité des deux lexiques utilisés n'étant pas équivalente, il est difficile de comparer les deux algorithmes uniquement sur cette base. Néanmoins, les deux sont proches de 1, c'est-à-dire que les radicaux obtenus par application de cet algorithme correspondent largement aux formes définies par Porter.

Le RAND Index montre cette tendance, c'est-à-dire, que les groupements effectués par Carry et par Porter sont non seulement aussi nombreux, mais encore très similaires.

3. CONCLUSION

Dans ce chapitre on c'est intéressé à l'une des partie les plus importante de l'indexation qui est l'extraction des radicaux. Pour ce faire, une synthèse a était effectuée concernant plusieurs algorithmes parmi lesquels l'algorithme de Porter sur lequel notre choix c'est porté pour la conception de notre système d'indexation parallèle.

CHAPITRE III : CONCEPTION ET IMPLEMENTATION DU SYSTEME

CHAPITRE III : CONCEPTION ET IMPLEMENTATION DU SYSTEME

1. INTRODUCTION :

Le but de ce travail étant le gain de temps lors de l'étape d'indexation. Pour ce faire, on va construire deux systèmes d'indexation des documents en anglais qui incluent les différentes étapes de l'indexation (l'analyse lexical, la sélection, l'extraction des radicaux, calcule du nombre d'occurrence).

La première version qui sera séquentielle qui va nous servir pour la comparaison avec la deuxième version qui sera parallèle. La comparaison des résultats sera on fonction du temps d'exécution dans les mêmes conditions (un même document, la même machine).

2. ARCHITECTURE DU SYSTEME :

Afin de mesurer les performances de notre système d'indexation parallèle, on a construit un autre système d'indexation qui est séquentiel (Figure 05).

2.1. Le système d'indexation sequentiel :

Dans ce système on va appliquer les différentes étapes d'indexation sur un document en anglais entré par l'utilisateur. La première partie du système concerne l'analyse lexicale suivie par un filtrage en utilisant un anti-dictionnaire, ensuite on applique l'algorithme de Porter afin d'obtenir les racines des termes et enfin la pondération de ces termes.

2.2. Le système d'indexation parallèle :

On a remarqué que certaines parties du système pouvaient être parallélisable, c'est-à-dire, qu'il y avait certaines parties qui pouvaient être divisées en sous parties afin que ces dernières puissent s'exécuter en parallèle par plusieurs processeurs en même temps.

Le but de cette parallélisassion est d'obtenir un autre système d'indexation qui va faire le même traitement (c'est-à-dire, obtenir les mêmes résultats) mais dans un temps plus court que celui du système séquentiel.

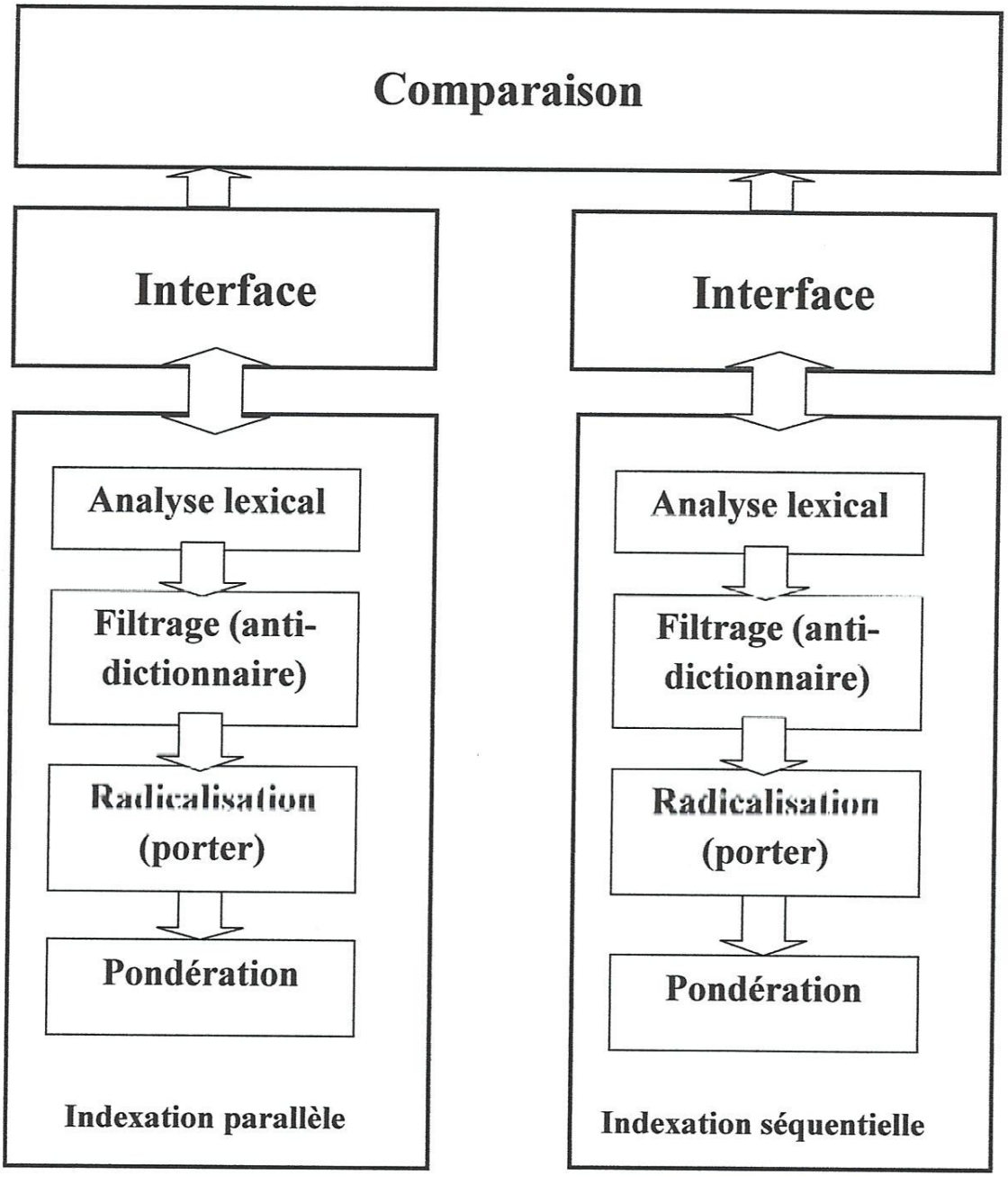


Figure 05 : Architecture globale du système

3. LE PROCESSUS DU SYSTEME :

Tout d'abord, on a développé un système séquentiel, qui a été codé sous *JAVA* en utilisant l'environnement *NetBeans*. La machine utilisée est une machine quadruple cœur pour l'exécution en parallèle.

Le système qu'on a conçu suit les différentes étapes de l'indexation vues dans le premier chapitre, à savoir: l'analyse lexicale, la sélection avec un anti-dictionnaire, la radicalisation et la pondération (voir *Figure 05*).

3.1. L'analyse lexicale :

Cette étape se fait en deux parties ; d'abord on va extraire tous les termes qui composent le document et supprimer tous le reste (la mise en forme, toute forme de ponctuation, la casse,…), ensuite on modifie la casse.

- **Supprimer la ponctuation et les caractères spéciaux :**

  Dans cette étape on va supprimer les signes de ponctuation comme les virgules, les points, les points virgules… et aussi les chiffres et les caractères spéciaux afin d'obtenir qu'une suite de termes.

  Pour éliminer la ponctuation, on a crée une méthode appelée *ponctuation(char c)*, qui va tester si le caractère entrant est un caractère spécial ou non, elle retourne *true* si oui sinon *false*.

  Exemple :

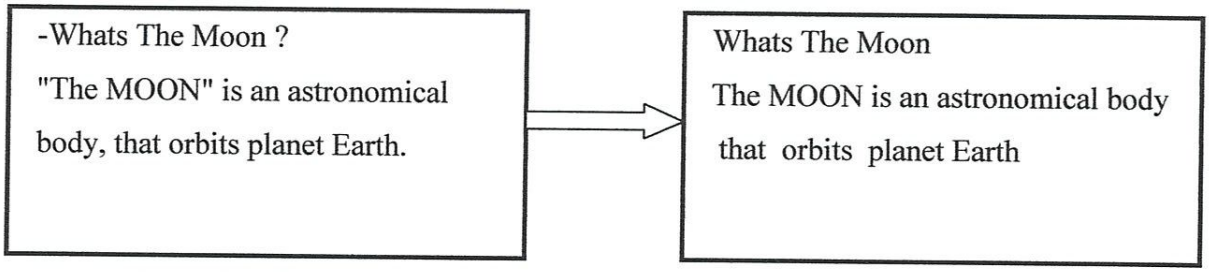


Figure 06 : Exemple sur l'étape élimination de la ponctuation.

- **Modifier la casse :**

  Une fois le mot extrait, on fait appel à une méthode prédéfinie on *JAVA* qui est "*toLowerCase*", qui prend une chaine de caractère en entrée et renvoie comme résultat une autre chaine de caractère en minuscule (c'est-à-dire, elle va modifier la casse de chaque terme pour éliminer la majuscule).

Exemple :

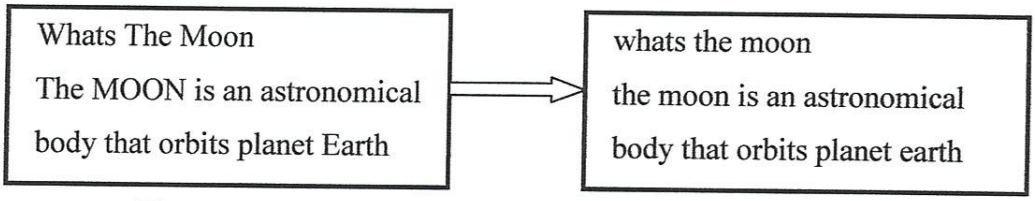


Figure 07 : Exemple sur l'étape de la casse.

3.2. Le filtrage :

Le filtrage est une sélection des mots ayant un sens parmi ce qui non pas (mots vides de sens). Cette étape se fait a l'aide d'un anti-dictionnaire pour la langue anglaise qui contient une liste de mots vides de sens. A la fin de cette étape, il nous restera seulement les mots qu'on va y extraire le radical.

Pour ce faire, on a défini une classe *stopliste* qui va contenir l'anti-dictionnaire, celui la peut être enrichi par d'autre mots vides de sens. Dans cette classe on va tester si le mot est vide ou non.

On prenant le même exemple précédent, on obtiendra:

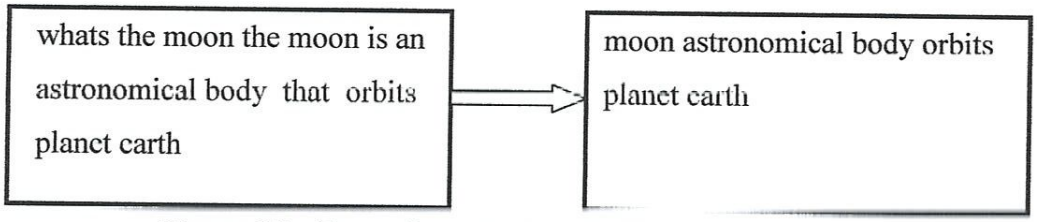


Figure 08 : Exemple sur l'étape le filtrage.

- Le parallélisme :

  On a ajouté du parallélisme au niveau de l'étape de filtrage pour plusieurs raison :

  - Premièrement, parce que la liste des mots vides de sens contient plus de 400 mots.
  - Ensuite, cette étape doit vérifier chacun des termes obtenu à l'étape précédente s'il appartient à la liste ou pas. Étant donné que la plupart des documents contiennent un nombre important de mots, cela prendra beaucoup de temps à faire les vérifications pour tous les termes du document.

  L'utilisation du parallélisme va nous permettre de gagner beaucoup de temps lors de l'indexation des documents. Pour ce faire, on lance plusieurs *Threads* en même temps, chacun d'eux va explorer une partie de l'anti-dictionnaire. Si un *Thread* le repère dans sa portion, alors il prévient les autres *Threads* que c'est un mot vide de sens et ils arrêtent instantanément leurs exécution. Cette procédure est répétée pour chacun des mots du document.

3.3. La radicalisation :

Dans cette étape, et étant donnée qu'on travail sur des documents en anglais, on a choisi l'algorithme de Porter afin d'extraire les radicaux des termes obtenues à l'étape précédente.

L'algorithme de porter expliqué en détail dans le Chapitre II avec ses différentes étapes traite le mot sur 5 étapes. Le passage d'une étape à une autre doit être séquentiel. En effet, chaque étape doit utiliser le résultat obtenu à l'étape précédente.

- La pondération :

  La dernière étape qui est la pondération va affecter à chaque terme une valeur numérique qui est appelée « *poids* ». Dans notre système, cette valeur représente le nombre d'occurrence du terme dans le document.

  Dans l'exemple précédent, le terme "moon" a un poids de « 2 ».

Exemple sur le processus d'indexation :

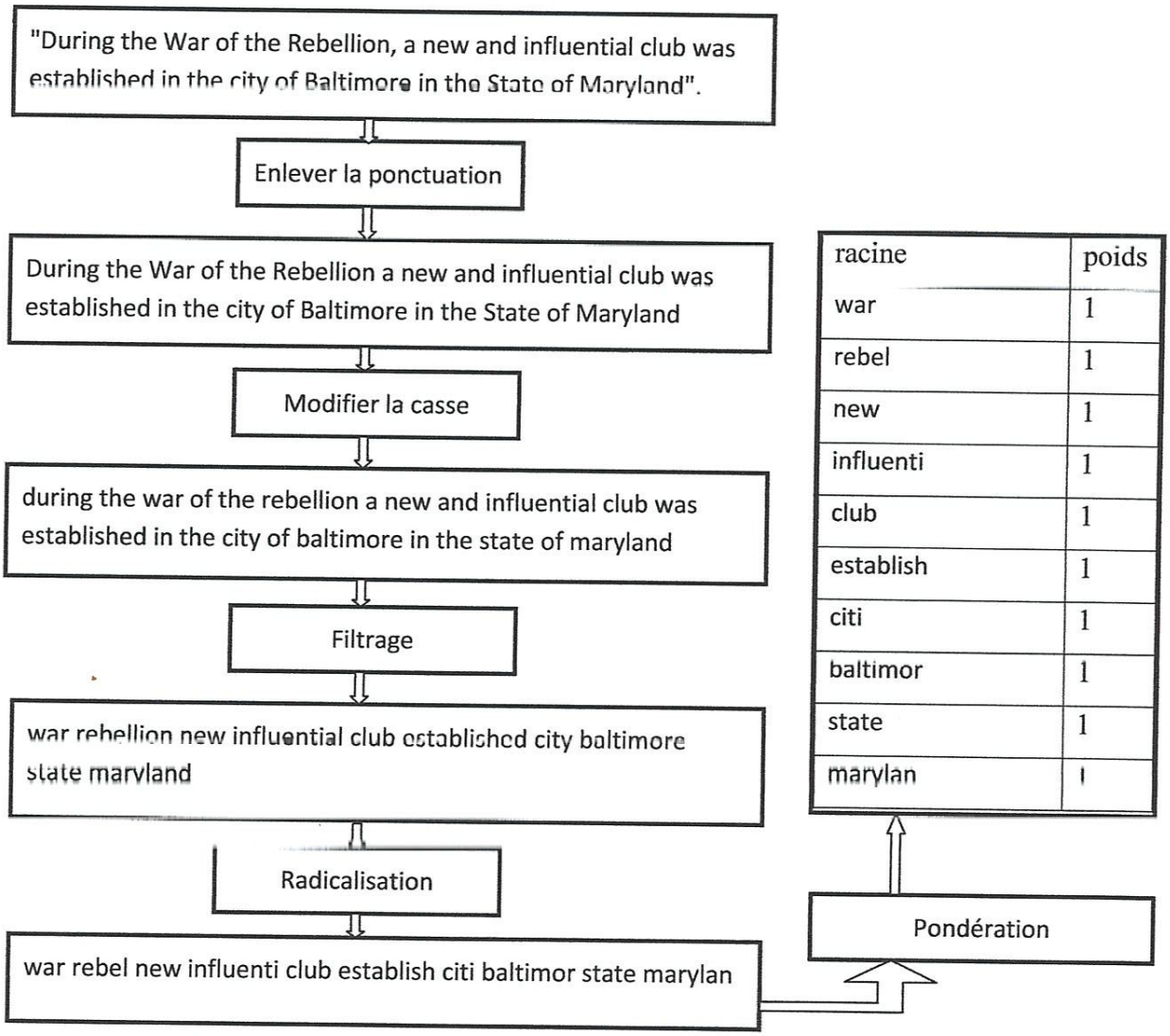


Figure 09 : Exemple d'exécution du système d'indexation

En exécutant notre système sur l'exemple vu précédemment et en appliquant les deux systèmes conçus on aura :

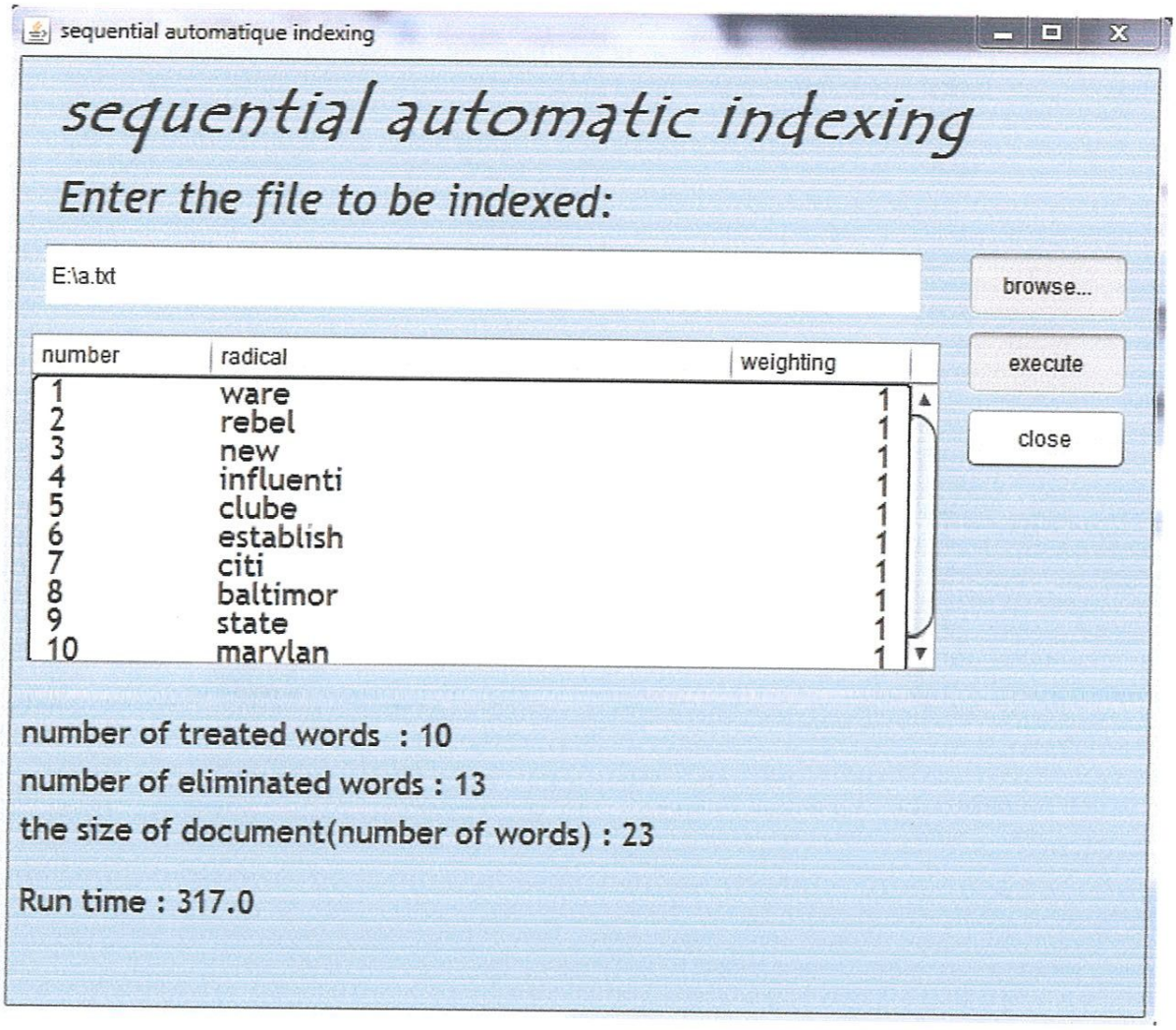


Figure 10 : Exécution du système sur l'exemple précédent avec le système séquentiel.

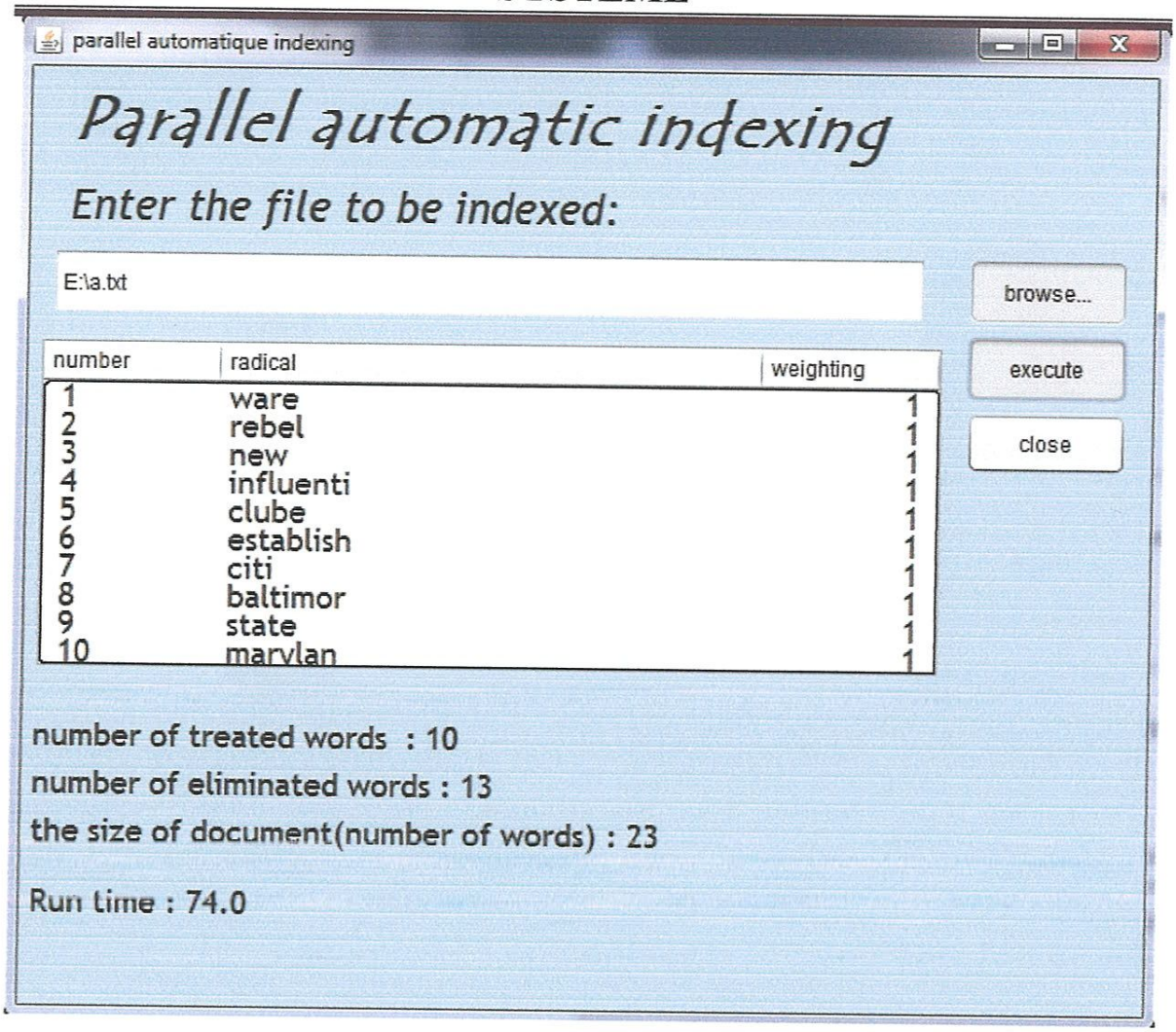


Figure 11 : Exécution du système sur l'exemple précédent avec le système parrallel.

Le système affiche les termes indexés avec leurs poids respectifs. Il affiche aussi le nombre total des mots que contient le document ainsi que le nombre des mots vides de sens qui ont été supprimés.

Le système affiche aussi le temps écoulé en millisecondes nécessaire pour indexer le document.

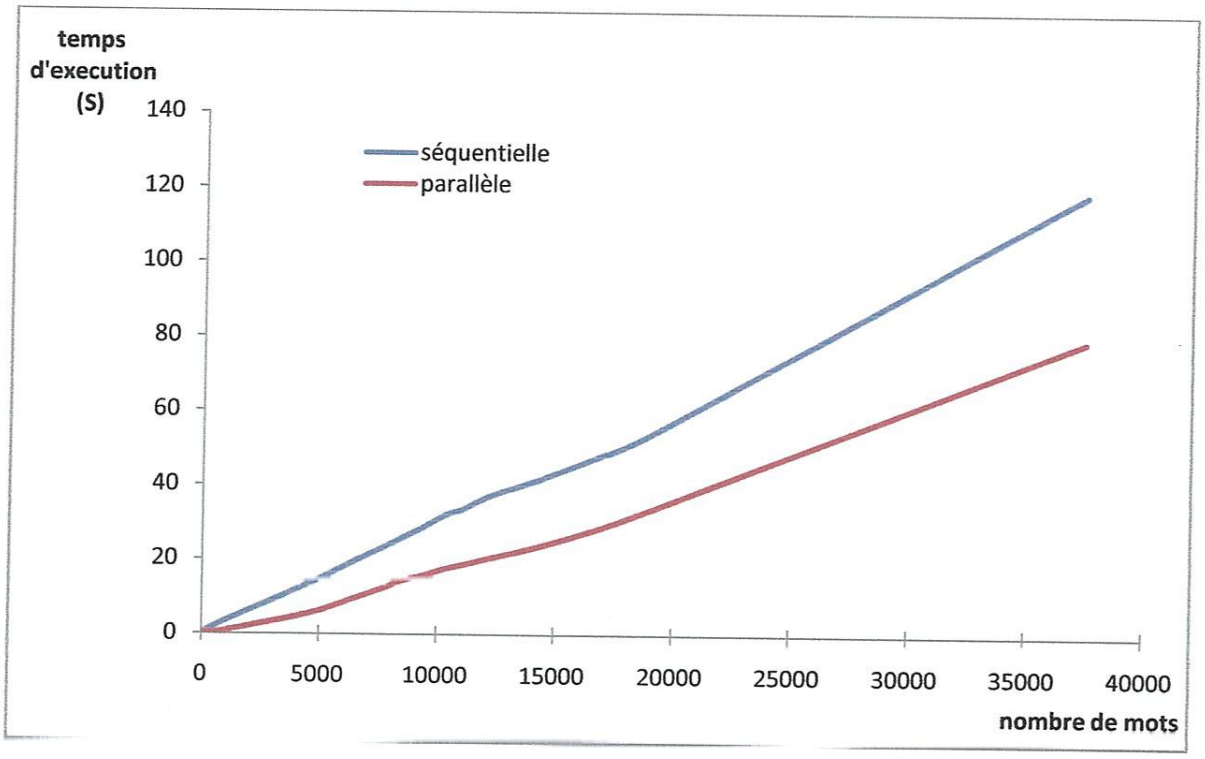
4. COMPARAISON :

Pour tester notre système, on a fait une comparaison entre le temps d'indexation en séquentiel et celui en parallèle. Pour se faire, on a utilisé exactement les mêmes conditions pour les deux exécutions à savoir : la même machine (multi-cœur), les mêmes documents pour les deux systèmes.

Le tableau suivant présente quelques résultats obtenus en millisecondes ; mais, étant donné que la machine, durant l'exécution, peut exécuter des programmes systèmes, on a effectué plusieurs exécutions pour chaque document. Le temps d'exécution de chaque document est représenté par la moyenne de toutes les exécutions.

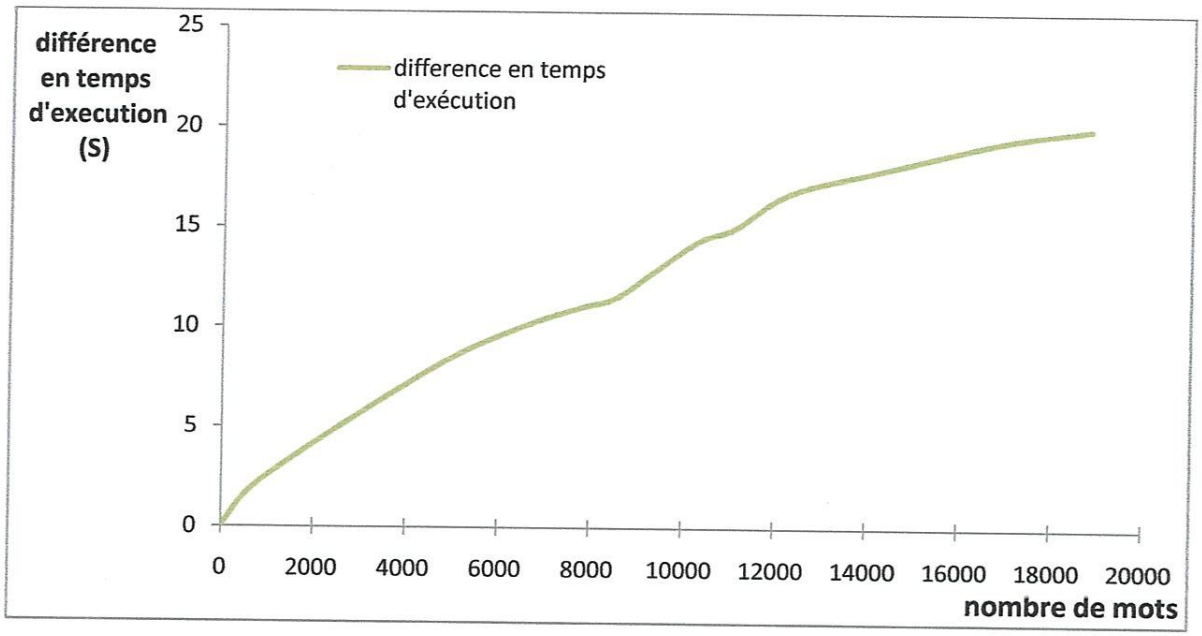
nombre de mots / Temps(s)	37533	18846	16900	14286	12312	10319
Système séquentiel	118,574	53,111	47,857	414,52	37,858	31,902
Système parallèle	78,736	33,100	28,516	235,40	20,548	17,568

Le graphe 01 représente les valeurs obtenues dans cette étude expérimentale :



Graphe 01 : Temps d'exécution avec les deux systèmes utilisés

On remarque que quelque soit la taille du document, le système parallèle fournis toujours un meilleur temps d'exécution que le système séquentiel. On peut aussi voir la différence dans le graphe 02 qui montre la différence en temps d'exécution entre les deux systèmes séquentiel et parallèle.



Graphe 02 : La différence en temps d'exécution avec les deux systèmes utilisés

5. CONCLUSION :

Dans ce chapitre, nous avons présenté les différentes étapes qui constituent notre système d'indexation parallèle. On remarque que le système parallèle est moins couteux en fonction de temps comparant avec le système séquentiel, et avec une différance plus au moins remarquable.

CONCLUSION GENERALE :

Nous avons présenté dans ce travail les concepts fondamentaux de la RI. Le but d'un SRI est de rechercher l'information pertinente pour une requête utilisateur. Son efficacité est mesurée par des paramètres qui reflètent sa capacité à accomplir un tel but.

Nous avons essentiellement présenté d'une manière générale les notions de base de la recherche d'information, mettant l'accent sur le processus de recherche en commençant par l'indexation, les différents modèles de la RI existant dans la littérature en essayant de donner brièvement les avantages et les limites de chacun et enfin l'évaluation d'un SRI.

Et vu que notre travail s'est basé sur le processus d'indexation, on a balayé les différents algorithmes de racinisation les plus répondus et surtout l'algorithme de Porter.

Enfin, on a utilisé la version originale de l'algorithme de Porter (séquentielle) pour développer un nouveau système utilisant le parallélisme et on a remarqué qu'on a eu de bons résultats en temps de réponse de l'étape d'indexation.

En conclusion, on peut dire que l'indexation basée sur un système parallèle, peut réellement améliorer l'efficacité de la RI car elle donne de bonnes résultats en gain de temps.

CONCLUSION GENERALE :

Nous avons essentiellement présenté d'une manière générale les notions de base de la recherche d'information en mettant l'accent sur le processus d'indexation. Nous avons exploré aussi les différents modèles de la RI en essayant de donner brièvement les avantages et les limites de chacun.

Etant donné que ce travail est basé sur le processus d'indexation, on a présenté les différentes étapes qui la constituent en mettant l'accent sur l'étape de racinisation.

Dans ce cadre, on a choisi l'algorithme de Porter (pour l'extraction du radicale) pour développer un nouveau système d'indexation utilisant le parallélisme dont le but est d'améliorer les performances en termes de temps.

Le système réalisé dans ce travail offre une indexation automatique d'un document en exécutant plusieurs tâches en parallèle ce qui nous a fournis un gain de temps assez conséquent en comparant avec la version séquentielle du même système et dans les mêmes conditions.

PERSPECTIVES

L'une des éventuelles améliorations qu'on peut apporter au système d'indexation est l'introduction de la sémantique des termes, c'est-à-dire, qu'il peut arriver dans certain cas que des mots qui se considèrent comme vide sont utilisé pour éviter la répétition. Par exemple « *il est le roi des forets* » ; le « *il* » ici fait référence au « lion » alors on ne peut pas le supprimer.

Dans ce travail, on a pris en considération seulement les documents en anglais ; une éventuelle amélioration serait qu'on améliore le système afin qu'il puisse gérer des documents dans d'autres langues

BIBLIOGRAPHIE

[Moer, 50] Mooers, C. Information retrieval viewed as temporal signaling. Proceedings of the International Congress of Mathematicians.1:572-573, 1950.

[Sal, 88a] Salton, G. Syntactic approaches to automatic book indexing. In Proc. of the annual meeting on Association for Computational Linguis-tics (ACL), Department of Computer Science, Cornell University, Ithaca, New York, pp. 204–210, 1988.

[Sal, 88b] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. Information Processing & Management (IPM) 24, 5, 513–523, 1988.

[Bac, 10] Baccini, A., Déjean, S., Kompaore, N. D., and Mothe, J.. Analyse des critères d'évaluation des systèmes de recherche d'information. Technique et Science Informatiques, 2010.

[Sal, 71] Salton, G. The Smart retrieval system-experiments. Automatic Document Processing, Prentice Hall Inc 1971.

[Sal, 75] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. Communications of the ACM (CACM) 18, 11, 613–620, 1975.

[Champc, 09] Champclaux, Y. Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information, 2009.

[Brini, 05] Brini, A.H , Un Modèle de Recherche d'Information basé sur les Réseaux Possibilistes, 2005.

[Salton et al., 1983] Salton, G. and McGill, M.J. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[Savoy, 93] Savoy, J. Stemming of French words based on grammatical categories Journal of the American Society for Information Science, 44(1), p.1-9, 1993.

[Boub, 08] Boubekeur, F. , Contribution à la définition de modèles de recherche d'information flexibles basées sur les CP-Nets, 2008.

[Clev, 67] Cleverdon, C. The cranfield test on index language devices. In: Aslib, pp. 173–194, 1967.

[Mandl, 08] Mandl, T. : Recent Developments in the Evaluation of Information Retrieval Systems: Moving Towards Diversity and Practical Relevance. Informatica (32): 27–38,2008.

[Schutz et al., 73] Schutz, A. and Luckmann, T. Structures of the Life World. Northwestern University Press, Evanston, Ill., Ed. ACM, New York, Sept. 1990, p. 45-61, 1973.

[Cleverdon, 67] Cleverdon, C. The cranfield tests on index language devices. In Aslib Proceedings, volume 19, pages 173-193, 1967.

[Cleverdon, 70] Cleverdon, C. Progress in documentation. evaluation of information retrieval systems. Journal of Documentation 26, 55–67, 1970.

[Saracevic, 96] Saracevic, T. Relevance Reconsidered '96. In P. Ingwersen, & N.O. Pors (Eds.), Proceedings of CoLIS 2, second international conference on conceptions of library and information science : Integration in perspective, Copenhagen (pp. 201-218). Copenhagen : Royal School of Librarianship, 1996.

[Cooper, 71] Cooper, W. A definition of relevance for information retrieval. Dans Information Storage and Retrieval, 1971.

[Zad, 65] Zadeh, L. A. Fuzzy sets. Information and control, 8, 338-353, 1965.

[Croft et al., 03] Croft, W. B., & Lafferty, J. (Eds.). Language modeling for information retrieval. In No. 13 in Information Retrieval Book Series. Kluwer, 2003.

[Amin.Gauss, 13] Massih-reza AMINI – Eric GAUSSIER recherche d'information :Applications, modèles et algorithmes- Fouille de données, decisionnel et big data,, 2013.

[Porter,80] Porter, M. An algorithm for suffix stripping. Program, 14, 1980.

[Adamson, 74] Adamson, G., Boreham J. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. In Information Storage and Retrieval, 10, p. 253–60, 1974.

[Krovetz, 93] Krovetz, R. Viewing morphology as an inference process. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval p. 191-202, 1993.

[Paice, 96] Paice, C.D. Method for evaluation of stemming algorithms based on error counting. Journal of the American Society for Information Science 47 (8) p. 632-49, 1996.

[Ogilvie et al., 03] Ogilvie, P., and Callan J. Combining Document Representations for Known Item Search, SIGIR, 2003.

[Bough, 14] Boughareb, D. recherche d'information multi critère, Thèse de doctorat en informatique , 2014.

[Zipf, 49] G. Zipf. Human Behaviour and the Principle of Least Effort. Addison-Wesley, 1949.

[Lhun, 55] Luhn, H.P. A new method of recording and searching information. American Documentation vol. 4:1; p. 14-16, 1955.

[Sal, 68] Salton, G. Search and retrieval experiments in real-time information retrieval. IFIP Congress (2) 1968: 1082-1093, 1968.

[Spa, 79] Sparck-Jones, K. Experiments in relevance weighting of search terms. Inf. Process. Manage, 15(3):133–144, 1979.

[Frakes et al., 92] Frakes, W.B. Stemming Algorithms. In: Frakes, W.B., Baeza-Yates, R. (eds.): InformationRetrieval Data Structures and Algorithms. Prentice Hall, New Jersey p. 131-160 , 1992.

[Rod, 10] Rodhain,F., Fallery, B., Girard,A., Desq, S. Une histoire de la recherche en Systèmes d'Information, à travers 30 trente ans de publications. Entreprises et Histoire, Eska, pp.78-97 , 2010.

[Carry ,02] Paternostre, M., Francq, P., Lamoral, J., Wartel, D. , et Saerens, M. : Carry, un algorithme de désuffixation pour le français, 2002.

[Kabil, 13]Kabil, B. : Un Nouvel Algorithme de Stemmatisation pour l'Indexation Automatique de Documents non-structurés : Stemmer SAID , 2013.

Bibliographie

[lovins, 68] lovins,J .B Devlopement of a stemming algorithm. Mechanical Translation and computational linguistics, 1968.

[paice, 90] Paice, C. et Chris, D. «Another stemmer,» SIGIR Forum 24, pp. p 56-61, 1990.

[paice, 94] Paice, C. «An evaluation method for stemming algorithms,» In Proceedings of the 7th, pp. p 42-50, 1994.

[Petitp, 94] Petitpierre, D. et Russel, G. . Mmorph - The Multext Morphology Program Technical Report, 1994.

[Weiss, 08] Weissenbacher, D. . Influence des annotations imparfaites sur les systèmes de Traitement Automatique des Langues, un cadre applicatif: la résolution de l'anaphore pronominale. Informatique et langage [cs.CL], 2008.

[Nakache, 06] D. Nakache, E. Métais et A. Dierstein, « EDA : algorithme de désuffixation du langage médical,» Revue des Nouvelles Technologies de l'Information, pp. p 705-706, 2006.