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 : Systèmes Informatiques

Thème :

Communication inter processus dans les systèmes d'indexation des documents textuels

Encadré Par :
Dr. SOUSSI Hakim

Présenté par :
HENNI Djalal

Septembre 2021

REMERCIEMENT

Au terme de mon travail, je tiens à remercier,

Notre DIEU Miséricordieux qui nous a donné la force, la volonté et la patiente pour achever ce travail.
Ma famille, mes parents, ma sœurs pour leurs sacrifices, aides, encouragements et soutiens.

Dr. Hakim SOUISSI, mon directeur de mémoire, pour son aide, sa patience,
ses encouragements sans cesse et pour la confiance qu'il m'a accordé.

Le président du jury d'avoir accepté de présider mon jury de soutenance.

L'examinateur d'avoir accepté de rapporter ce mémoire.

Ma très chère amie Dr. Manel LATRACHE, ma source de courage pour ses sacrifices et tous les moments qu'on a passé. Merci pour la richesse que m'apport à ma vie.

Mes amis : Hazem BENSALAH, Idriss DRISSI, Ouaasma NACEF, Charaf GUIDOUM, Dr. Mohammed BENZERARA. Merci pour m'avoir apporté l'aide dont j'avais besoin. Je vous suis profondément reconnaissant pour tout ce que vous avez fait pour moi. Une chose est sure : je ne vous oublierais jamais.

Tous les enseignants du département d'informatique de l'université 08 Mai 1945.

.

Toute personne qui m'a aidé de près ou de loin pour l'achèvement de ce travail.

RESUME

Le processus de la Recherche d'Information (RI) est réalisé par des outils, appelés Systèmes de Recherche d'Information (SRI). Ces systèmes ont pour objectif d'aider un utilisateur à trouver des documents susceptibles de l'intéresser. L'utilisateur exprime son intérêt ou ses demandes d'informations via des requêtes, puis le SRI doit être en mesure de fournir à l'utilisateur des documents liés à sa requête. Pour avoir un système de recherche de qualité, il est important que son système d'indexation reflète au mieux le contenu de la collection original. L'extraction des radicaux est généralement effectuée en supprimant tous les suffixes et préfixes attachés aux termes de l'index avant l'affectation réelle du terme à l'index. Étant donné que la racine d'un terme représente un concept plus large que le terme d'origine, le processus de racine augmente finalement le nombre de documents récupérés dans un système de recherche d'information.

Ce travail est consacré à la réalisation d'un système d'indexation parallèle dont le but est d'améliorer l'efficacité en termes de temps. Le système parallèle sera comparé à un autre qui est séquentiel afin de mieux voir les avantages d'utiliser un système parallèle pour réduire le temps d'exécution lors de l'étape d'indexation qui est le but de notre travail.

Mots clés : Parallélisme, indexation, recherche d'information

SOMMAIRE

CHAPITRE II: LES ALGORITHMES DE RADICALISATION

CHAPITRE III: CONCEPTION ET IMPLEMENTATION

LISTE DES FIGURES

LISTE DES TABLEAUX

Introduction générale

INTRODUCTION GENERALE

La recherche à base de mots est une caractéristique importante prise en charge par les systèmes d'indexation et de recherche actuels. L'indexation et la recherche font à leur tour partie des applications de TextMining, des systèmes de traitement du langage naturel et des systèmes de recherche d'informations (SRI).
L'idée principale de ce travail est d'améliorer la mémorisation par une gestion automatique des terminaisons de mots en réduisant les mots à leurs radicaux, au moment de l'indexation afin d'augmenter la précision des documents récupérés. La radicalisation est généralement effectuée en supprimant tous les suffixes et préfixes attachés aux termes. Étant donné que l'extraction du radical d'un terme représente un concept plus large que le terme d'origine, le processus d'extraction des radicaux augmente finalement le nombre de documents récupérés dans un système de recherche d'information.

Ce travail est consacré à la réalisation d'un système d'indexation parallèle (dans la partie extraction des radicaux) dont le but est d'améliorer 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 :
Dans le premier chapitre nous allons voir les concepts de base de la recherche d'information, puis nous allons détailler les différentes étapes d'indexation et les principaux modèles de la recherche d'information.
Dans le deuxième nous allons présenter quelques algorithmes qui traitent la radicalisation. Comme les algorithmes de : Lovins, Porter, Paice/Husk, N-Gram, HMM, YASS, Krovetz et Xerox en se basant sur l'algorithme de Porter que nous avons utilisé dans notre travail.
Le troisième chapitre est consacré à la description des systèmes séquentiel et parallèle ainsi qu'à une comparaison entre les deux afin de mieux voir les avantages d'utiliser le système parallèle. L'implémentation et les résultats obtenus seront présentés et discutés.

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

CHAPITRE I : RECHERCHE D'INFORMATION

1. INTRODUCTION

La Recherche d'Information (RI) est un sous domaine de l'informatique, qui concerne principalement la recherche et la récupération d'informations prédites à partir d'un volume important de documents disponibles. Le défi est de pouvoir trouver les informations qui correspondent aux besoins de l'utilisateur.

Le processus de la Recherche d'Information (RI) est réalisé par des outils, appelés Systèmes de Recherche d'Information (SRI). Ces systèmes ont pour objectif d'aider un utilisateur à trouver, à retrouver ou à découvrir des documents susceptibles de l'intéresser. L'utilisateur exprime son intérêt ou ses demandes d'informations par des requêtes, puis le SRI doit être en mesure de fournir à l'utilisateur des documents liés à sa requête.

Dans ce chapitre nous allons voir les concepts de base de la recherche d'information, puis nous allons détailler les différentes étapes d'indexation et les principaux modèles de la recherche d'information.

2. HISTORIQUE DE LA RECHERCHE D'INFORMATION

Le terme recherche d'information a été utilisée pour la première fois lors du lancement de la conférence RIOA (Recherche d'Information Assistée par Ordinateur) en 1885, à Grenoble. On parlait auparavant de « recherche documentaire » ou « d'information documentaire ».

Le domaine 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 automatiques des informations documentaires, afin de répondre à un besoin d'information (Champclaux, 2009).

On peut distinguer cinq grandes périodes dans l'histoire des systèmes de recherche d'information(RODHAIN, FALLERY, GIRARD, & DESQ., 2010) :

- Période de développement des SRI, avant 1980 : trois études (Ives, Hamilton, Devis 1980,1981, 1982) concluent que la grande majorité des recherches publiées sont alors de nature non empirique (soit conceptuelle, soit technique) sans formulation d'hypothèses claires.
- Période de la théorisation des SRI, 1980-1995 : trois études (Ives et Hamilton 1982, Culnan et Swanson 1984, Culnana1980-1985) montant des progrès importants vers une tradition de recherche cumulative et un abandon des recherches technique :

développement et accumulation de connaissances, travail en équipe, construction de théories, production de références propres à la discipline.

- 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, et du niveau inter-organisationnel. Les études empiriques sur le terrain prennent une place de plus en plus importante.
- Période de la diversification, 1990-2000 : (Cheon et al. 1993, Alavi et Calson 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…).
- 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

3. CONCEPTS DEBASEDE LA RECHERCHED'INFORMATION

3.1. Système de recherche d'information

Un SRI (système de recherche d'information) est un outil qui permet de sélectionner l'information pertinente à partir d'une collection de document sen réponse au besoin exprimé par l'utilisateur à l'aide d'une requête.

3.2. Requête

Une requête exprime le besoin d'information d'un utilisateur. Elle représente le lien entre l'utilisateur et le système, on distingue dans la littérature quatre formalismes de requête (Baziz, 2005): liste de mots clés, langage naturel, langage booléen ou graphique.

3.3. Document

On appelle un document toute unité qui peut constituer l'élément de l'information de base dans un SRI, il peut être un texte, un morceau de texte, page web, image, une bande vidéo, ou une combinaison des objets précédents. L'ensemble de documents dans un SRI s'appelle *Collection de documents* (*corpus*).

3.4. Pertinence

Le principal objectif d'un SRI est de garantir une meilleure pertinence de résultat aux utilisateurs.

La pertinence mesure le degré de similarité entre les documents et une requête pour retourner ceux qui répondent au mieux au besoin de l'utilisateur. Deux types de pertinence ont été distingués dans la littérature (Ben Aouicha, 2009) :

- La pertinence système est définit par les modèles de RI. Elle est souvent traduite par un score qui évaluant la correspondance du contenu d'un document par rapport de la requête d'utilisateur (Cleverdon, 1970). Ce score évalué en fonction du poids des mots de la requête dans le document interrogé. Ce poids représente l'importance du mot dans le document.
- D'autre part, la pertinence de l'utilisateur est associée à l'évaluation par l'utilisateur des informations renvoyées par le système(Harter, 1992). L'évaluation des utilisateurs peuvent également être implicites ou explicites, et elles peuvent être utilisées pour construire leur profil puis l'intégrer dans un processus de RI personnalisé.

Plusieurs pertinences sont possibles entre un document et une requête (Daoud, 2009), parmi lesquelles on note :

- *La pertinence algorithmique :*basé sur le calcule de correspondance entre document et requête et à quel point il a réussi à retrouver les documents pertinents à cette requête.
- *La pertinence thématique :*elle dépend du thème évoqué par une requête de recherche et celui des documents retrouvés.
- *La pertinence contextuelle ou situationnelle :* elle est dynamique et dépond du contexte de recherche de l'utilisateur et de sa perception.
- *La pertinence cognitive :* dite cognitive car elle améliore les connaissances de l'utilisateur à travers le contenu renvoyé lors de la recherche.

4. LE PROCESSUS DE RECHERCHE D'INFORMATION

Il apparaît à l'utilisateur que l'accès à l'information est une tâche simple qui peut être résumée en quelques clics, alors qu'un processus sophistiqué cache derrière cette simplicité.
La figure 1 illustre le processus en *U* de recherche d'information qui comporte deux grandes étapes sont : l'indexation et la comparaison

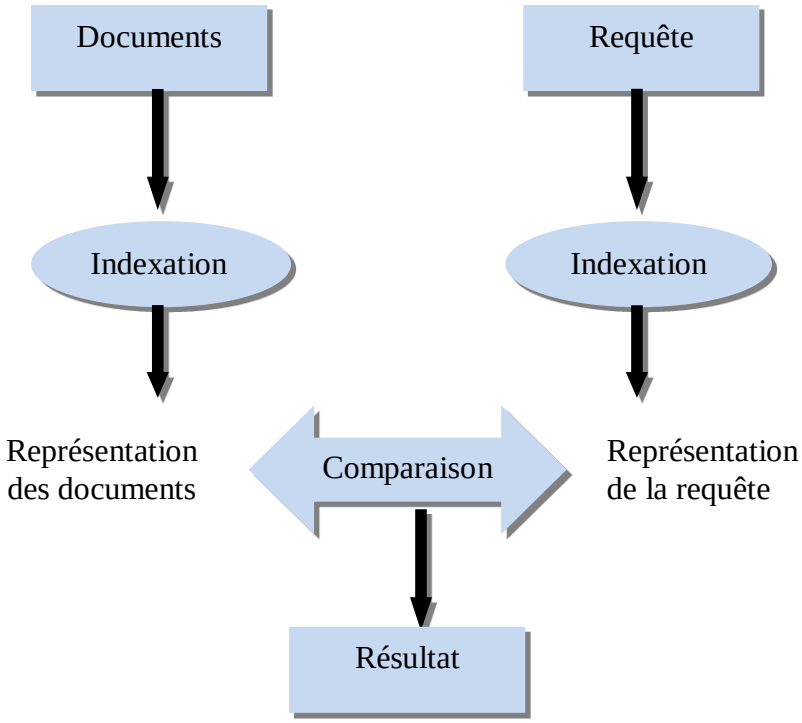


Figure 1 : Processus en *U* de la recherche d'information

4.1. Indexation

L'indexation est une phase très importante dans le processus de RI. Cela inclut l'identification et l'extraction de termes qui représentent le contenu d'un document ou d'une requête, et ces termes doivent couvrir au mieux leur contenu sémantique.
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 de processus d'indexation est l'analyse de la ressource afin d'en extraire les caractéristiques les plus importantes et les structurer par la suit dans l'un des modèles de recherche d'information. L'indexation des documents de la BDD a pour objectif d'éviter au système de les analyser à chaque interrogation.
L'indexation peut être manuelle, semi-automatique, automatique.

4.1.1. Indexation Manuelle

Cette approche permet de retourner un index plus proche du document original car il est fait par un expert du domaine. Cette approche pose problème dans le cas des documents de taille volumineux, et aussi le choix des termes d'indexation dépond de l'indexeur et ses connaissances.

4.1.2. Indexation Semi-automatique:

Elle est une combinaison des deux méthodes (manuelle et automatique). L'indexation semi-automatique consiste en premier temps à indexer automatiquement les documents en s'appuyant sur un vocabulaire contrôlé, ce type requiert le contrôle manuel du processus par un expert du domaine qui intervient pour établir des relations sémantiques entre mots-clés et choisir les termes significatifs.

4.1.3. Indexation Automatique:

Ici l'ordinateur peut aussi bien faire une indexation avec ou sans vocabulaire contrôlé. Cette méthode est plus rapide mais peu fiable par rapport à l'indexation humaine (la méthode manuelle).

Nous nous intéressons particulièrement à l'approche d'indexation automatique dans le cadre de notre travail. L'indexation automatique repose sur les étapes mentionnées ci-dessous (Figure2).

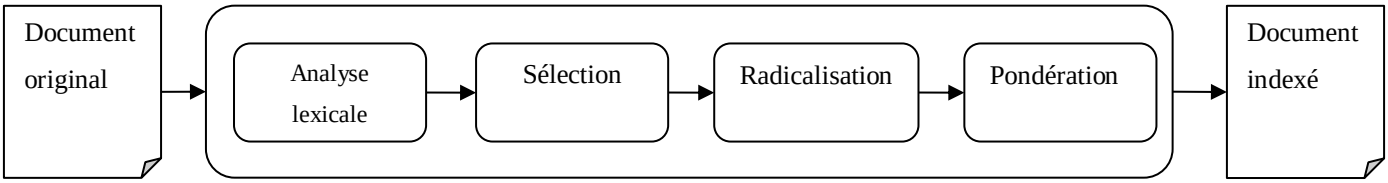


Figure 2: les opérations effectuées sur les données textuelles lors de l'indexation automatique. (Champclaux, 2009)

4.1.3.1. Analyse lexical

C'est la première étape du processus d'indexation automatique qui permet de transformer le document en un ensemble de termes, en d'autres termes : la ponctuation, la case (majuscule et minuscule), et la mise en page sont supprimées.

4.1.3.2. Sélection et suppression des mots vides de sens

La suppression des mots vides de sens à partir de la liste des mots résultants de l'étape d'analyse lexicale, en utilisant un anti-dictionnaire (stoplist). Ces mots peuvent être des pronoms personnels, prépositions, les mots outils et les mots mathématiques, tous ces mots n'ayant pas de réel rapport avec le sujet traité seront éliminés.

L'élimination des mots vides de sens doit être contrôlée car elle influence sur la qualité de la recherche. Le traitement lié à l'anti-dictionnaire est très simple. Lorsqu'un mot est rencontré dans un texte à indexer, s'il apparait dans l'anti-dictionnaire, il n'est pas considéré comme un index.

4.1.3.3. Radicalisation et lemmatisation

La radicalisation et la lemmatisation sont des techniques de normalisation de texte dans le domaine du traitement du langage naturel qui sont utilisées pour préparer le texte, les mots et les documents en faveur d'un traitement ultérieur.

Dans la littérature une différence entre radical connu par le nom « *stem* » et la racine « *lemme* »,la radicalisation et la lemmatisation génèrent toutes les deux le type de base des mots fléchis et donc la seule différence est que le *stem* peut ne pas être un mot réel alors que *lemme* est un mot du langage réel.

La radicalisation coupe essentiellement les lettres de la fin jusqu'à ce que le *stem* soit atteinte. Cela fonctionne dans la plupart des cas, mais malheureusement, l'anglais a des nombreuses exceptions où un processus plus sophistiqué est nécessaire.

Exemple (selon l'algorithme de Porter):

caresses → caress

ponies → poni

fairly → fairli

Contrairement à la radicalisation, la lemmatisation voie au-delà de la réduction des mots et considère le vocabulaire complet d'une langue pour appliquer une analyse morphologique aux mots. Exemple : le *lemme* de « was » et « *be* » et le *lemme* de « *mice* » et « *mouse* »et le *lemme* de «*fairly* » et « *fair* ».La lemmatisation est généralement considérée pour beaucoup plus informative que la simple radicalisation.

La radicalisation suit un algorithme avec des étapes appliqué sur le mot, ce qui le rend plus rapide. Alors que dans la lemmatisation vous devez utiliser le corpus également pour fourni le *lemme*,ce qui la rend plus lente.

L'un des outils de radicalisation les plus courants et les plus efficaces est l'algorithme de Porter développé par Martin Porter 1980 (Porter, 1980). L'algorithme utilise cinq phases de réduction de mots, chacune avec son propre ensemble de règles.

4.1.3.4. Pondération

La pondération consiste à associer un poids à chaque terme d'indexation qui mesure son importance dans le document.

La fréquence relative d'un terme dans un document est représentative du pouvoir de représentation du terme dans la collection est caractéristique du pouvoir de discrimination du terme pour les documents. Les mesures de pondération les plus répondues dans la littérature sont :

- La loi du Zipf(Zipf, 1949), elle considère que les termes des documents s'organisent suivant une loi inversement proportionnelle à leur fréquence d'apparition dans le corpus appelé rang. Elle est donnée par la forme suivante :

$$\text{Fréquence} * \text{rang} \approx \text{constante}$$

- La conjecture de Luhn(Luhn, 1955), 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 rang 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(Salton, 1968)(Sparck Jones, 1979), est le schéma de pondération le plus utilisé, il combine deux facteurs de pondération : le facteur TF (TermFrequency), qui mesure la fréquence d'un terme dans le document ou il apparait et le facteur IDE (Inverse Document Frequency), qui mesure sa fréquence dans tout le corpus. Elle est donnée par l'équation suivante :

$$TF * IDF = log\ (1+TF) * IDF$$

Ou

$$TF\left(t_i, d_j\right) = \frac{n_{(t_i, d_j)}}{\sum_k n_{(t_k, d_j)}}$$

Tel que $TF\ (t_i, d_j)$ représente la fréquence du terme t_i dans le document d_j, $n_{(t_i, d_j)}$ est le nombre d'occurrences de t_i dans d_j et $\sum_k n_{(t_k, d_j)}$ donne la somme des occurrences de chaque terme t_k dans d_j.

$$IDF\left(t_i, d_j\right) = \log \frac{|D|}{\left|\left\{d_j : t_i \in d_j\right\}\right|}$$

Tel que $IDF(t_i, d_j)$ est la fréquence absolue inverse du terme t_i dans le document d_j, $|D|$ est le nombre total de document dans le corpus et $\left|\left\{d_j : t_i \in d_j\right\}\right|$ calcule le nombre de document ou le terme t_i apparait.

Exemple

- Document originale :

Cambridge University Press (www.cambridge.org) is the publishing division of the University of Cambridge, one of the world's leading research institutions and winner of 81 Nobel Prizes.

- Après l'analyse lexicale :

cambridgeuniversity press wwwcambridgeorg is the publishing division of the university of cambridge one of the worlds leading research institutions and winner of 81 nobel prizes

- Après la suppression des mots vides :

cambridge university press wwwcambridgeorg publishing division university cambridge worlds leading research institutions winner 81 nobel prizes

- Après radicalisation avec l'algorithme de Porter :

cambridge univer press wwwcambridgeorg publish divi univer cambrid world lead research institut winner 81 nobel prize

4.2. La comparaison

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 est calculé à partir d'une probabilité ou une similarité appelée en anglais « Retrieval Status » Value RSV (d, q), ou 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 de comparaison constituent les deux éléments essentiels du modèle de recherche (Champclaux, 2009).

5. LESMODELES DERECHERCHE D'INFORMATION

Les modèles de RI aident les utilisateurs à trouver les informations qu'ils cherchent dans une collection de documents.

Le but de ces modèles est de retourner un sous ensemble de documents de la collection de document. Ces derniers sont appelés documents pertinents, les autres étant des documents non pertinents par rapport à la requête.

Un modèle de RI doit accomplir deux rôles :

- Créer une représentation interne pour les documents et la requête basée sur les termes de l'indexation.
- Définir une méthode de comparaison entre une représentation de document et une représentation de requête afin de déterminer leur degré de correspondance (mesure de pertinence).

Les modèles de RI se distinguent par le principe d'appariement : appariement exact et appariement approché.

I. Appariement exact

- La requête spécifie de manière précise les critères recherchés.
- L'ensemble des documents respectant exactement la requête sélectionnés, mais pas ordonné.

II. Appariement approché

- La requête décrit les critères recherchés dans un document.
- Les documents sont sélectionnés selon un degré de pertinence vis-à-vis de la requête et sont ordonnés.

De nombreux modèles existent(Figure3).D'abord il y a le modèle booléen qui est historiquement un des premiers modèles étudiés et qui a servi de point de départ aux recherches dans le domaine de la RI, puis le modèle vectoriel (approche algébrique) et enfin, le modèle probabiliste.

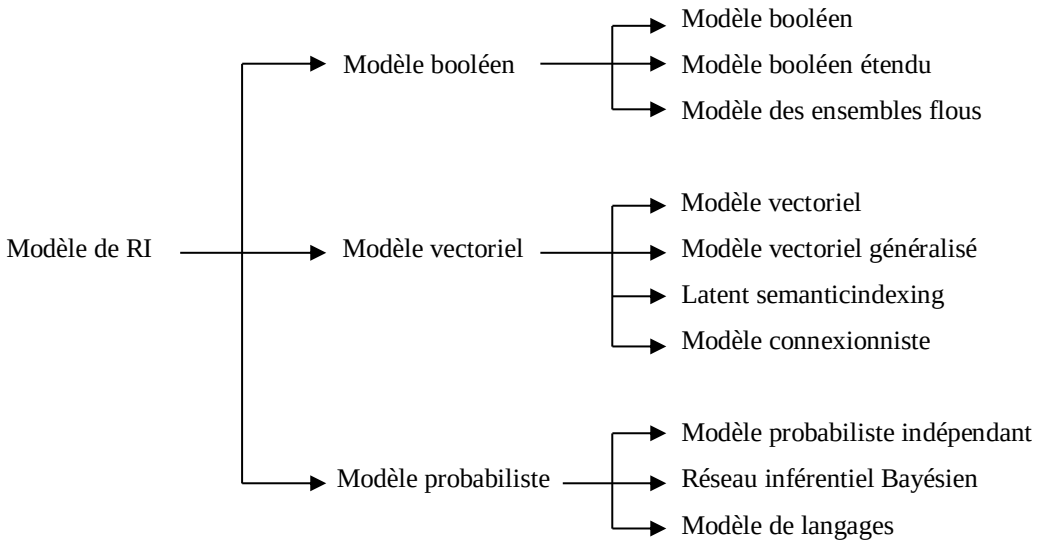


Figure 3 : Les Modèle de recherche d'information(BOUBEKEUR, 2008).

Concernant la représentation interne des documents et de la requête, les principaux modèles utilisent une représentation par mots-clés, et c'est dans la comparaison des représentations que chaque approche a sa propre manière de faire.

5.1. Modèle booléen

Le premier modèle développer dans la RI est le modèle booléen. Il est basé sur la théorie des ensembles (Salton, 1969). Un document est représenté par l'ensemble des termes qui le composent. Les requêtes peuvent être composées de plusieurs termes reliés entre eux par des opérateurs de la logique booléenne OR, AND et NOT (ET, OU NON).

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

Exemple :la requête *Connaissance* ET *Renseignement* retrouvera les documents qui ont été indexés avec les deux termes, alors le résultat c'est l'intersection des deux ensemble.

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

Exemple : la requête *Connaissance* OU *Renseignement* retrouvera les documents indexés avec *Connaissance* ou *Renseignement*(ou les deux), alors le résultat c'est l'union des deux ensemble.

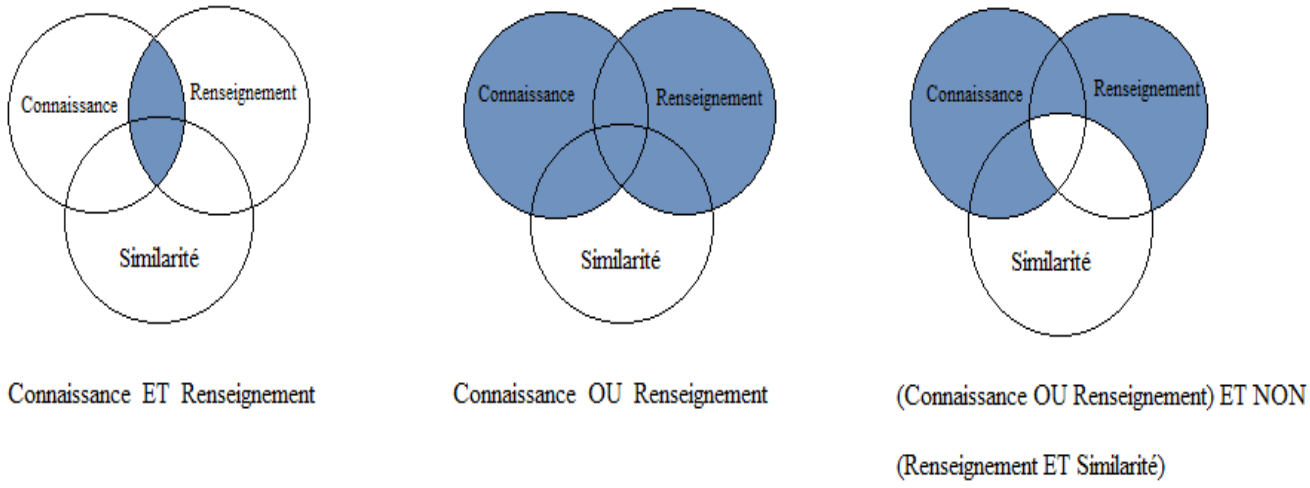


Figure 4 : Combinaison booléennes d'ensembles visualisés sous la forme de diagramme de Venn (Champclaux, 2009).

Un document est représenté par une liste des termes, exemple d : $t_1, t_2, \dots, t_n$.

Une requête est représentée par une expression logique quelconque de termes utilisant les opérateurs and, or et not.

La correspondance RSV(d, q) entre une requête q et un document d est déterminée de la façon suivante:

RSV(d, t_i) = 1 si $t_i \in$ d; 0 sinon.

RSV(d, q_1 AND q_2) = 1 si RSV(d, q_1) = 1 ET RSV(d, q_2) = 1; 0 sinon.

RSV(d, q_1 OR q_2) = 1 si RSV(d, q_1) = 1 OU RSV(d, q_2) = 1; 0 sinon

RSV(d, NOT q_1) = 1 si RSV(d, q_1) = 0 ; 0 sinon.

Ce modèle présente de nombreux avantages et inconvénients :

Les avantages :

- Facile à implémenter.
- Tout à fait fonctionnel.
- Permet aux utilisateurs d'exprimer des contraintes structurelles et conceptuelles(Marcus, 1991).
- Pour la formulation des requetés les utilisateurs trouve que l'utilisation de synonymes (grâce à la clause OR) et de groupe de mots (grâce à la clause ET) très utiles (Cooper, 1988).
- L'approche booléenne possède un grand pouvoir d'expressivité : adaptée aux requêtes qui appellent une section exhaustive et nom ambigüe car dans le cas d'une requête

ambiguë, le système va avoir plus de difficultés à identifier le besoin informationnel sous-jacent à la requête.

Les inconvénients :

- Difficile aux utilisateurs non experts de formuler des requêtes adéquates à l'aide d'expressions booléennes(Edward A & Matthew B, 1988)(Belkin & Croft, 1992).
- les résultats de recherche ne sont pas triés, leur nombre est difficile à contrôler.
- Le classement des documents extraits par ordre de pertinence est difficile.

5.2. Modèle vectoriel

Le modèle vectoriel a été créé par Gérard Salton en 1971(Gerard, 1971). Le modèle consiste à représenter les documents et les requêtes dans un espace multidimensionnel, ou chaque dimension correspond à un terme d'indexation (Salton & McGill, 1983).

Chaque terme du même document ou de la même requête est pondéré selon son degré d'importance par rapport aux autres termes. Le modèle vectoriel repose sur le calcul de la similarité entre le vecteur document et le vecteur requête, pour déterminer les documents pertinents vis-à-vis une requête donnée. Plus les deux vecteurs sont proches plus leur contenu est semblable.

Le modèle vectoriel exprime le degré de similitude entre le document et la requête par l'unedes mesures suivant :

- Le produit scalaire de la requête Q et du document D représentés par leur vecteur de terme$\vec{Q},\vec{D}$ respectivement défini par l'équation suivante :

$$D_E\left(\vec{Q},\vec{D}\right)=\left(\sum_{i=1}^{n}\left|w_{i,Q}-w_{i,D}\right|^2\right)^{\frac{1}{2}}$$

- Le cosinus de l'angle, plus les vecteurs sont similaires, plus l'angle formé est petit, et plus le cosinus de cet angle est grand.

$$\cos(\vec{Q},\vec{D})=\frac{\vec{Q}.\vec{D}}{|\vec{Q}|\times|\vec{D}|}=\frac{\sum_{i=1}^{n}w_{i,Q}\times w_{i,D}}{(\sum w_{i,Q}^2)^{1/2}\times(\sum w_{i,D}^2)^{1/2}}$$

- La mesure de Jaccard, la similarité de Jaccard entre la requête *Q* et le document *D* défini par l'équation suivante :

$$J(Q,D)=\frac{|Q\cap D|}{|Q\cup D|}=\frac{\sum_{i=1}^{n} w_{i,Q}\times w_{i,D}}{\sum w_{i,Q}+\sum w_{i,D}-\sum w_{i,Q}\times w_{i,D}}$$

- La mesure de Dice :

$$sim(d_i,Q)=2*\frac{\sum_{j=1}^{n} w_{qj}*w_{ij}}{\left(\sum_{j=1}^{n} {w_{qj}}^2\right)+\left(\sum_{j=1}^{n} {w_{ij}}^2\right)}$$

- La mesure de Superposition :

$$sim(d_i,Q)=\frac{\sum_{j=1}^{n} w_{qj}*w_{ij}}{\min\left(\left(\sum_{j=1}^{n} {w_{qj}}^2\right),\left(\sum_{j=1}^{n} {w_{ij}}^2\right)\right)}$$

Ce modèle présente de nombreux avantages et inconvénients :

Les avantages :

- Il est possible d'assigner une pondération aux termes d'une requête. Ainsi que le coefficient de similarité permet de classer les documents par ordre de pertinence.
- Ce modèle simple à appréhender (algèbre linière) et est facile à implémenter.

Les inconvénients :

Ce modèle ne permet pas de modéliser les associations entre les termes.

5.3. Modèle probabiliste

Ce modèle est basé sur le calcul de la probabilité de pertinence d'un document pour une requête (Stephen & Karen, 1976). C'est-à-dire le modèle estime la probabilité que le document *D* appartient à la classe des documents pertinents, ou non pertinents.

Le modèle probabiliste classifie des documents en fonction de probabilité de pertinence pour une requête en deux classes :P(R/*d*) le document *d* inclus l'information pertinente pour la requête *q* et P (NR/*d*) le document *d* n'inclus pas l'information pertinente.

Le score d'appariement entre un document et une requête est donné par :

$$RSV(D,Q) = \frac{p\left(\frac{R}{D}\right)}{p\left(\frac{NR}{D}\right)}$$

Les avantages de modèle probabiliste :

- La fonction d'appariement permet de trier les documents
- Apprentissage du besoin d'information tente d'estimer la probabilité d'observer des événements liés au document et à la requête.
- Plus efficace que le modèle booléen, mais moins performant que le modèle vectoriel(Savoy, 1993).

Les inconvénients de modèle probabiliste :

- Ce modèle est coûteux à implémenter
- La complexité augmente rapidement avec la taille des collections de documents.
- Estimation des nécessites des données d'apprentissage.

6. Conclusion

Dans ce chapitre, les concepts de base de la recherche d'information ont été présenté ainsi que les grands étapes du processus de la RI, à savoir l'indexation, les modèles de recherche d'information.

Dans le prochain chapitre on s'est intéressé sur l'une des étapes de l'indexation automatique qui est la radicalisation.

CHAPITRE II : LES ALGORITHMESDE RADICALISATION

1. INTRODUCTION

Avec l'expansion importante des données textuelles disponibles, l'exploration de texte est devenue d'un intérêt particulier. En raison de leur nature non structurée, ces données nécessitent d'importantes étapes de prétraitement. Parmi eux la radicalisation qui est une phase importante pour le processus d'indexation. Certains chercheurs comme Porter (Porter, 1980), Paice et Husk(Paice, 1990)et Krovetz (Krovetz, 1993)ont développés des algorithmes afin de raffiner la tâche de radicalisation (stemming en anglais) après la publication du premier algorithme Lovins 1968 (Lovins, 1968), Ces algorithmes ont pour but de dégager les racines des mots en passent par un ensemble des règles et des conditions.

2. LES ALGORITHMES DE RADICALISATION

La racinisation est utilisé pour produire des termes significatifs en suppriment des caractères qui aboutissent finalement à des résultats précis et plus pertinentes. L'objectif principal de l'algorithme de radicalisation est d'obtenir des termes utiles et de réduire les formes grammaticales dans la structure morphologique de certaine langue.

Le travail d'un stemmer est également très important dans la phase d'indexation. Il est courant que la plupart des mots qui ont une sémantique similaire doivent être considéré comme identique pour les systèmes de recherche d'informations. Le stemmer supprime les termes pour en faire un terme racine (unique).Différents types de stemmers sont utilisés afin que la recherche est effectuée sur des termes stemmed au lieu de terme réel pour obtenir des résultats plus authentiques à partir du grand corpus de documents en réduisant le temps de traitement.

Les algorithmes de radicalisation peuvent être classés en trois groupes, ces groupes sont représentés dans la figure 4. Chacun de ces groupes a une façon de trouver le radical du mot.

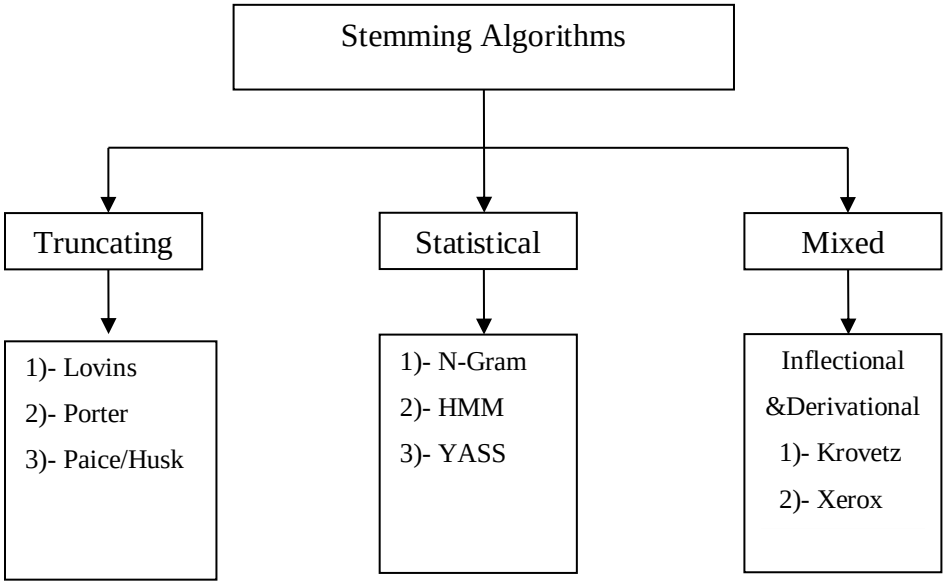


Figure 5: Types des algorithmes de radicalisation(Jivani, 2011).

2.1. Méthodes de troncature (suppression des affixes)

Comme son nom l'indique clairement, ces méthodes sont liées à la suppression des suffixes ou préfixes (communément appelés affixes) d'un mot. Le stemmer le plus basique était le stemmer Trancate (n) qui coupe un mot au n-ième symbole, c'est-à-dire garder n lettres et supprimer le reste. Dans cette méthode les mots les plus courts que n sont conservés sans le moindre changement.

Une autre approche simple qui est le S-stemmer, un algorithme confondant les formes singulières et plurielles des noms anglais. Cet algorithme a été proposé par Donna Harman.

L'algorithme a des règles pour supprimer les suffixes au pluriel afin de les convertir en formes singulières (Harman, 1991).

2.1.1. Algorithme de Lovins

Le premier stemmer populaire et efficace proposé par Lovins (Lovins, 1968) en 1968. Il effectue une recherche sur une table de 294 terminaisons, 29 conditions et 35 règles de transformation, qui ont été organisées sur un principe de correspondance le plus long. Le stemmer Lovins supprime le suffixe le plus long d'un mot. Une fois la fin supprimée, le mot est recodé à l'aide d'un tableau différent avec divers ajustements pour convertir ces stems en mots valides.

Il supprime toujours un maximum de suffixe, en raison de sa nature en tant qu'algorithme à passage unique.

L'un des avantages de cet algorithme est qu'il est rapide et peut gérer la suppression des lettres doubles dans des mots comme « getting » transformé en « get » et parmi les inconvénients de l'approche Lovins qu'elle prend beaucoup des données ce qui influent sur le temps de l'exécution En outre, de nombreux suffixes ne sont pas disponibles dans le tableau des terminaisons. Il est parfois peu fiable et ne parvient souvent pas à former des mots à partir des stems ou à correspondre aux stems de mot de sens similaire. La raison étant le vocabulaire technique utilisé par l'auteur.

Voici les codes des règles contextuelles associées à certaines terminaisons et leurs significations (* signifie tous les lettres)

Code	Significations
A	aucune restriction sur le stem
B	longueur minimale de stem = 3
C	longueur minimale de stem = 4
D	longueur minimale de stem = 5
E	ne supprimer pas la terminaison après e
F	longueur minimale de stem = 3 et ne supprimer pas la terminaison après e
G	longueur minimale de stem = 3 et supprimer la terminaison seulement après f
H	supprimer la terminaison seulement après t ou ll
I	ne supprimer pas la terminaison après o ou e
J	ne supprimer pas la terminaison après a ou e
K	longueur minimale de stem = 3 et supprimer la terminaison seulement après l, i ou u*e
L	ne supprimer pas la terminaison après u, x ou s, sauf s suit de o

M	ne supprimer pas la terminaison après a, c, e ou m
N	longueur minimale de stem = 4 après s*, autre part = 3
O	supprimer la terminaison seulement après l ou i
P	ne supprimer pas la terminaison après c
Q	longueur minimale de stem = 3 ne supprimer pas la terminaison l ou n
R	supprimer la terminaison seulement après n ou r
S	supprimer la terminaison seulement après dr ou t, sauf t suit de t
T	supprimer la terminaison seulement après s ou t, sauf t suit o
U	supprimer la terminaison seulement après l, m, n ou r
V	supprimer la terminaison seulement après c
W	ne supprimer pas la terminaison après s ou u
X	supprimer la terminaison seulement après l, i ou u*e
Y	supprimer la terminaison seulement après in
Z	ne supprimer pas la terminaison après f
AA	supprimer la terminaison seulement après d, f, ph, th, l, er, or, es ou t
BB	longueur minimale de stem = 3 et ne supprimer pas la terminaison après met ou ryst
CC	supprimer la terminaison seulement après l

L'algorithme :

Algorithme de Lovins

Entrées : Document texte contient ensemble des mots

Sorties : Document texte contient les mots indexés

Début

Tant que (non fin fichier) **faire**

Procédure de racinisation :

si (terminaison trouvée) alors

Appliquer la règle

Fin si

Procédure de recodage :

Si(existe transformation) alors

Recoder stem selon règle

Fin si

Retourner (stem) ;

Fin Tant que

Fin

2.1.2. Algorithme de Porter

L'algorithme de Porter est l'un des algorithmes de normalisation de mots les plus célèbres, qui a été développé par Martin PORTER au Computer Laboratory de l'Université de Cambridge en 1979. Cet algorithme est utilisé dans la langue anglaise, et il est basé sur l'idée que les suffixes de la langue anglaise sont principalement composés de combinaisons de suffixes plus petites et plus simples(Porter, 1980).

Principe de l'algorithme de Porter :

Le stemmer est divisé en un certain nombre d'étapes linéaires (5 étapes), qui sont utilisées pour produire le stem finals. Dans chaque étape des règles sont appliquées jusqu'à ce que l'une d'elle passe les conditions, si une règle est acceptée le suffixe est supprimé en conséquence et l'étape suivante est effectuée. Le stem résultant à la fin de la cinquième étape est retournée.

Une liste de consonne supérieure ou égale à la langueur sera indiquée par un C et une liste similaire de voyelles par un V.

On note :

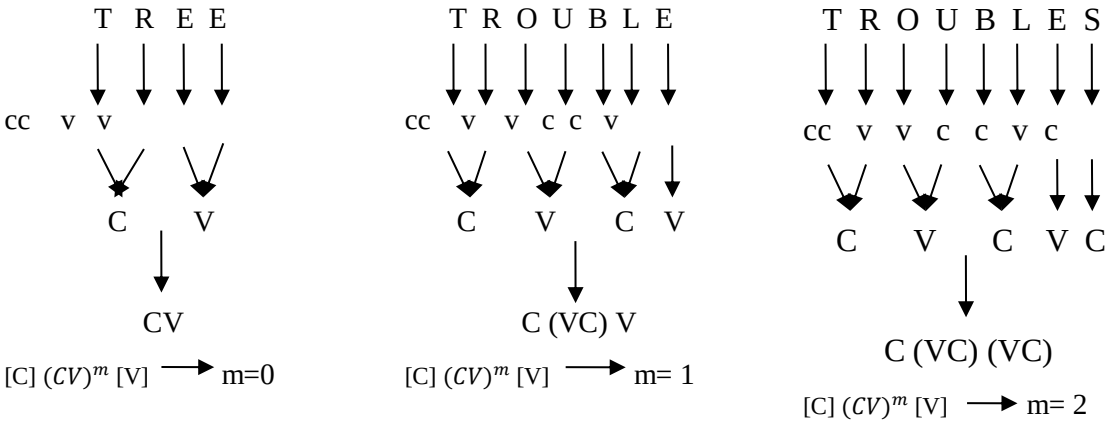
- v (voyelles) : a, e, i, o, u et y quand il est précédé d'une consonne ;
- c (consonne) : toutes les autres lettres et y quand il est précédé d'une voyelle ;
- V représente une suite de voyelles ;
- C représente une suite de consonne.

Un mot en anglais peut être de l'une des formes suivantes :

- CVCV… C
- CVCV… V
- VCVC… C
- VCVC… V

N'importe quel mot peut donc être représenté par la forme unique : [C] $(CV)^m$ [V]

Lorsque le m désigne les répétitions de VC et les supports carrés [ ] désignent la présence facultative de leur contenu. La valeur *m* est appelée mesure d'un mot et peut prendre toute valeur supérieure ou égale à zéro, elle est utilisée pour décider si un suffixe donné doit être supprimé.



Toutes ces règles sont de la forme : (condition) S1 -> S2

- (condition) est une condition que le radical doit vérifier pour que la règle soit appliquée ;
- S1 est le suffixe que nous retirons au mot afin d'obtenir le radical ;
- S2 est le suffixe que nous ajoutons au radical si la condition est remplie.

Ce qui signifie que le suffixe S1 est remplacé par S2 si les lettres restantes de S1 satisfont à la condition.

Par exemple, une règle (m>0) eed → ee signifie « si le mot à au moins une voyelle et une consonne plus la terminaison eed, changer la terminaison en ee ». Ainsi « agreed » devient « agree » tandis que « feed » reste inchangé.

Il y a un certain nombre de notations spécifiques à certaines conditions qui s'applique sur le mot considéré privé de S1 :

- ∗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 ou le second c n'est ni w, ni x, ni y

Etape 1a	
1- SSES → SS 2- IES → I 3- SS → SS 4- S →	**c**aresses→caress ties → ti caress →caress cats →cat
Etape 1b	
1- (m>0) EED → EE 2- (* v *) ED → 3- (* v *) ING →	feed → feed palastered→ palaster motoring→motor sing → sing

Si la deuxième ou la troisième des règles de l'étape 1b réussit, les opérations suivantes sont effectuées.

AT → ATE BL → BLE IZ → IZE (* d and not (*L or *S or *Z)) →lettre unique (m=1 and *o) → E	conflat(ed) →conflate troubl(ed) → trouble siz(ed) → size hopp(ing) → hop fall(ing) →fall hiss(ing) →hiss fail(ing) →fail
Etape 2	
(* v *) Y → I	happy →happi sky→sky
Etape 3	
(m>0) ATIONAL →ATE (m>0) TIONAL →TION (m>0) ENCI → ENCE (m>0) ANCI →ANCE (m>0) IZER → IZE (m>0) ABLI → ABLE (m>0) ALLI → AL (m>0) ENTLI→ENT (m>0) ELI →E (m>0) OUSLI→OUS (m>0) IZATION →IZE (m>0) ATION →ATE (m>0) ATOR →ATE (m>0) ALISM →AL (m>0) IVENESS →IVE (m>0) FULNESS →FUL (m>0) OUSNESS →OUS (m>0) ALITI →AL (m>0) IVITI →IVE (m>0) BILITI→BLE (m>0) ICATE →IC (m>0) ATIVE → (m>0) ALIZE →AL (m>0) ICITI → IC (m>0) ICAL →IC (m>0) FUL → (m>0) NESS →	relational→relate conditional→ condition rational → rational valenci→ valence hesitanci→ hesitance digitizer→ digitize conformabli→conformable radicalli→ radical differentli→ different vileli → vile analogousli→ analogous vietnamization→vietnamize predication→ predicate operator→ operate feudalism→ feudal decisiveness→ decisive hopefulness→hopeful callousness→ callous formaliti→ formal sensitiviti→ sensitive sensibiliti→ sensible triplicate→ triplic formative → form formalize→formal electriciti→electric electrical→electric hopeful →hope goodness→good
Etape 4	

(m>1) AL→	revival →reviv
(m>1) ANCE→	allowance→allow
(m>1) ENCE→	inference→infer
(m>1) ER→	airliner→airlin
(m>1) IC→	gyroscopic→gyroscop
(m>1) ABLE→	adjustable→adjust
(m>1) IBLE→	defensible→defens
(m>1) ANT→	irritant→irrit
(m>1) EMENT→	replacement→replac
(m>1) MENT→	adjustment→adjust
(m>1) ENT→	dependent→depend
(m>1 and (*S or *T)) ION→	adoption→adopt
(m>1) OU→	homologou→homolog
(m>1) ISM→	communism→commun
(m>1) ATE→	activate→activ
(m>1) ITI→	angulariti→angular
(m>1) OUS→	homologous→homolog
(m>1) IVE→	effective→effect
(m>1) IZE→	bowdlerize→bowdle
Etape 5	
(m>1) E→	probate →probat rate →rate
(m=1 and not *o) E→	cease→ceas
(m > 1 and *d and *L)→lettreunique	controll→control roll→roll

Tableau 1 :les conditions des terminaisons avec des exemples (Porter, 1980).

L'algorithme :

Algorithme : PORTER

Entrées : fichier contenant l'ensemble des mots

Sorties : fichier contenant l'ensemble des mots indexés

Début

Tant que (non fin fichier) **faire**

/* Étape 1 */

Enlever et recoder forme pluriel

Enlever « ed » et « ing » du verbe

/* Étape 2 */

Si (existe voyelle dans le stem) alors

Transformer y en i

Fin si

/* Étape 3 */

Indexer la lettre avant dernière

Enlever le doublement s'il existe

/* Étape 4 */

Indexer la lettre finale du stem

Enlever la terminaison indiquée si possible

/* Étape 5 */

Si (stem a la forme <c>vcvc<v>) alors

Enlever la terminaison

Fin si

/* étape 5a */

Si (plus qu'une séquence de consonne dans le stem) alors

Enlever terminaison

Fin si

Retourner (stem)

Fin Tant que

Fin

En se fondant sur les erreurs de radicalisation, Paice est arrivé à la conclusion que le stemmer Porter produit moins de taux d'erreur que le stemmer Lovins. Cependant, il a été noté que le stemmer Lovins est un stemmer plus lourd qui produit une meilleure réduction des données (Eiman Tamah, 2008).

L'algorithme de Lovins est sensiblement plus grand que l'algorithme de Porter, en raison de sa liste de terminaisons très étendue. Mais d'une certaine manière, c'est utilisé à l'avantage : c'est plus rapide, il n'a besoin que deux étapes majeures pour supprimer un suffixe, par rapport aux cinq étapes de l'algorithme de Porter.

2.1.3. Algorithme de Paice/Husk

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.

Le stemmer Paice/Husk est un algorithme itératif avec une table contenant environ 120 règles indexées par la dernière lettre d'un suffixe (Paice, 1990). A chaque itération, il tente de trouver une règle applicable par le dernier caractère du mot. Chaque règle spécifie la suppression ou le remplacement d'une fin. En l'absence d'une telle règle, elle prend fin. Il se termine également si un mot commence par une consonne et qu'il ne reste que trois caractères. Si non, la règle est appliquée et le processus se répète.

Parmi ces avantages est sa forme simple et chaque itération prenant soin à la fois de la suppression et du remplacement selon la règle appliquée. L'algorithme est plus facilement portable à la gestion d'une nouvelle langue. L'inconvénient est qu'il s'agit d'un algorithme très lourd.

.

2.2. Méthode statistique

Ce sont les stemmers qui sont basés sur l'analyse statistique et les techniques des études statistiques tel que N-Gram stemmer, HMM stemmers et YASS stemmers. La plupart des méthodes suppriment les affixes, mais après la mise en œuvre d'une procédure statistique.

2.2.1. N-Gram stemmer

C'est une méthode très intéressante et indépendante du langage. Ici, l'approche de similarité de la chaîne est utilisée pour convertir l'inflation des mots en son radical. Un n-gram est une chaîne de n caractères, généralement adjacents, extraits d'une section de texte continu. Pour être précis, un n-gram est un ensemble de n caractères consécutifs extraits d'un mot. L'idée principale derrière cette approche est que des mots similaires auront une forte proportion de n-grammes en commun. Pour n est égal à 2 ou 3, les mots extraits sont appelés respectivement digrammes ou trigrammes. Par exemple, le mot « INTRODUCTIONS » entraîne la génération des digrammes:

*I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S**

Et les trigrammes :

I, *IN, INT, NTR, TRO, ROD, ODU, DUC, UCT, CTI, TIO, ION, ONS, NS*, S

Où '*' indique un espace de remplissage.

La plupart des stemmers sont spécifiques à la langue. En général, une valeur de 4 ou 5 est sélectionnée pour n. Après cela, une donnée textuelle ou un document est analysé pour tous les n-grammes. Il est évident qu'une racine de mot se produit généralement moins fréquemment que sa forme morphologique. Cela signifie qu'un mot a généralement un affixe qui lui est associé. Une analyse statistique typique basée sur la fréquence inverse du document (IDF) peut être utilisée pour les identifier.

Ce stemmer a l'avantage d'être indépendant du langage et donc très utile dans de nombreuses applications. L'inconvénient est qu'il nécessite une quantité importante de mémoire de stockage pour créer et stocker les N-Gram et les index, donc ce n'est pas une approche très pratique.

2.2.2. HMM stemmer

Ce stemmer est basé sur le concept du modèle de Markov caché (HMMs) qui sont des automates à états finis ou les transitions entre états réagissent par des fonctions de probabilité. A chaque transition, le nouvel état émet un symbole avec une probabilité donnée. Ce modèle a été proposé par Melucci et Orio (Melucci & Orio, 2003).

Cette méthode est basée sur un apprentissage non supervisé et n'a pas besoin d'une connaissance linguistique préalable de l'ensemble de données. Dans cette méthode, la probabilité de chaque chemin peut être calculée et le chemin le plus probable est en utilisant le codage de Viterbi dans le graphe des automates.

Afin d'appliquer HMMs à la recherche de radical, une séquence de lettres qui forme un mot peut être considérée comme le résultat d'une concaténation de deux sous-séquences : un préfixe et un suffixe. Une façon de modéliser ce processus est par le biais d'un HMM ou les états sont divisés en deux ensembles disjoints : initial peut être les stems uniquement et le dernier peut être les stems ou les suffixes. Les transitions entre les états définissent le processus de création de mots. Certaines hypothèses peuvent être formulées dans cette méthode (Jivani, 2011):

- Les états initiaux n'appartiennent qu'à l'ensemble de stem, un mot commence toujours par un radical.
- Les transitions des états de l'ensemble de suffixes aux états de l'ensemble de stem ont toujours une probabilité nulle, un mot ne peut être qu'une concaténation d'un radical et d'un suffixe.
- Les états finaux appartiennent aux deux ensembles, un radical peut avoir un certain nombre de dérivations différentes, mais il peut également n'avoir aucun suffixe.

Pour un mot donné, le chemin le plus probable de l'état initial à l'état final produira le point de fractionnement (une transition des racines aux suffixes). Ensuite, la séquence de caractère avant ce point peut être considéré comme un radical.

L'avantage de cette méthode est qu'elle n'est pas supervisée et que la connaissance de la langue n'est donc pas requise.

L'inconvénient est qu'il est un peu complexe et peut parfois sur-endiguer les mots :le radical peut coïncider avec un terme de vocabulaire ordinaire. C'est par exemple le cas de comme frontal qui donne la racine front.

2.2.3. YASS stemmer

Le nom est un acronyme pour Yet Another Suffix Striper. Ce stemmer a été proposé par Prasenjit Majumder et. al (Prasenjit, et al., 2007). Selon les auteurs, la performance d'un stemmer généré par le regroupement d'un lexique sans aucune entrée linguistique est comparable à celle obtenue à l'aide de stemmers standard basés sur des règles telles que Porter.

Ce stemmer entre dans la catégorie des statistiques ainsi que du corpus. Il ne s'appuie pas sur l'expertise linguistique. Les expériences de récupération menées par les auteurs sur des ensembles de données en Anglais, en Français montrent que

l'approche proposée est efficace pour les langues qui sont principalement de nature suffixé (Jivani, 2011).

2.3. Méthodes flexionnelles et dérivationnelles

Il s'agit d'une autre approche de radicalisation et elle implique à la fois l'analyse de morphologie flexionnelle et dérivationnelle. Le corpus doit être très grand pour développer ces types de stemmers et donc ils font également partie des stemmers de base du corpus. En cas de flexion, les variantes de mots sont liées aux variations syntaxiques spécifiques à la langue comme le pluriel, le genre, le cas, etc. tandis que dans la dérivationnelle, les variantes de mots sont liées à la POS (Part Of Speech) d'une phrase où le mot se produit.

2.3.1. Algorithme de Krovetz

Le stemmer Krovetz a été présenté en 1993 par Robert Krovetz (Krovetz, 1993), et est un stemmer de validation lexical linguistique. Comme il est basé sur la propriété flexionnelle des mots et la syntaxe du langage, il est de nature très compliquée. Il supprime efficacement et avec précision les suffixes flexionnels en trois étapes :

- Transformer les pluriels d'un mot en sa forme singulière.
- Conversion de passé au temps présent.
- Suppression du suffixe « ing ».

Le processus de conversion supprime d'abord le suffixe, puis, tout au long du processus de vérification dans un dictionnaire pour tout recodage, retourne le radical à un mot. La recherche dans le dictionnaire effectue également toutes les transformations requises en raison de l'exception d'orthographe et convertit également tout radical produit en un mot réel, dont la signification peut être comprise.

La force de l'analyse dérivationnelle et flexionnelle réside dans leur capacité à produire des radicaux morphologiquement corrects, à faire face aux exceptions, à traiter les préfixes ainsi que les suffixes. Étant donné que ce stemmer ne trouve pas les stems pour toutes les variantes de mots, il peut être utilisé comme pré-stemmer avant d'appliquer réellement un algorithme de radicalisation. Cela augmenterait la vitesse et l'efficacité du l'algorithme principal. Par rapport à Porter et Paice / Husk, c'est un stemmer très léger.

Le stemmer Krovetz tente d'augmenter la précision et la robustesse en traitant les fautes d'orthographe et les stems dénués de sens.

Si la taille du document d'entrée est grande, ce générateur de stem devient faible et ne fonctionne pas très efficacement. Le défaut majeur et évident des algorithmes basés sur des dictionnaires est leur incapacité à faire face aux mots, qui ne sont pas dans le lexique. En outre, un lexique doit être créé manuellement à l'avance, ce qui nécessite des efforts importants. Ce stemmer ne produit pas systématiquement un bon rappel et une bonne performance de précision

2.3.2. Xerox Analyseur flexionnel et dérivationnel

Les linguistes de Xerox ont développé une base de données lexicale pour l'anglais et d'autres langues également qui peuvent analyser et générer une morphologie flexionnelle et dérivationnelle. La base de données flexionnelle réduit chaque mot de surface à la forme que l'on peut trouver dans le dictionnaire, comme suit (Hull & Gregory, 1996) :

- Nom singulier (ex : Children→ Child)
- Verbe a l'infinitif (ex :understood→understand)
- Adjectif forme positive (ex : best→good)
- Pronom nominatif (ex : whom→ ho)

La base de données dérivationnelle réduit les formes de surface à des stems qui sont liées à l'original à la fois dans la forme et la sémantique. Par exemple, « government» devient« govern» tandis que « department» n'est pas réduit à « depart» puisque les deux formes ont des significations différentes. Tous les stems sont des termes anglais valides et les formulaires irréguliers sont traités correctement.

Le processus dérivationnel utilise à la fois la suppression de suffixe et de préfixe, contrairement à la plupart des algorithmes de radicalisation qui reposent uniquement sur la suppression de suffixe. Un exemple des suffixes et préfixes qui sont supprimés est donné ci-dessous (Hull & Gregory, 1996) :

- Suffixes: ly, ness, ion, ize, ant, ent, ic, al, Ic, ical, able, ance, ary, ate, ce, y, dom, ee, eer, ence, ency, ery, ess, ful, hood, ible, icity, ify, ing, ish, ism, ist, istic, ity, ive, less, let, like, ment, ory, ous, ty, ship, some, ure
- Prefixes: anti, bi, co, contra, counter, de, di, dis, en, extra, in, inter, intra, micro, mid, mini, multi, non, over, para, poly, post, pre, pro, re, semi, sub, super, supra, sur, trans, tn, ultra, un

Les avantages de ce stemmer est qu'il fonctionne bien avec les vastes documents et supprime également les préfixes le cas échéant. Tous les stems sont des mots valides puisqu'une base de données lexicale qui fournit une analyse morphologique de n'importe quel mot dans le lexique est disponible pour le radical.

Il s'est avéré qu'il fonctionne mieux que le stemmer Krovetz pour un grand corpus.

L'inconvénient est que la sortie dépend de la base de données lexicale qui peut ne pas être exhaustive. Étant donné que cette méthode est basée sur un lexique, elle ne peut pas endiguer correctement les mots qui ne font pas partie du lexique. Ce générateur de stem n'a pas été implémenté avec succès sur de nombreuses autres langues. La dépendance au lexique en fait un stemmer dépendant de la langue.

3. COMPARAISON ENTRE LES ALGORITHME

LOVENS	
Avantage	Inconvénient
1. Rapide - algorithme à passage unique. 2. Gère la suppression des lettres doubles dans des mots comme « getting» en cours de transformation en « get». 3. Gère de nombreux pluriels irréguliers comme - mouse et mice , etc.	1. Consommer beaucoup du temps 2. Tous les suffixes ne sont pas disponibles. 3. Pas très fiable et échoue fréquemment à former des mots à partir des tiges. 4. Dépend du vocabulaire technique utilisé par l'auteur.
PORTER	
Avantage	Inconvénient
1. Produit le meilleur rendement par rapport aux autres stemmers. 2. Moins de taux d'erreur. 3. Par rapport à Lovins, c'est un stemmer léger. 4. Le cadre de l'algorithme« Snowball » conçu par Porter est une approche indépendante du langage pour le radical.	1. Les stems produits ne sont pas toujours de vrais mots. 2. Il a au moins cinq étapes et soixante règles et prend donc du temps.

PAICE / HUSK	
Avantage	Inconvénient
1. Forme simple. 2. Chaque itération s'occupe de la suppression et du remplacement.	1. Algorithme lourd. 2. Un sur-stemme peut se produire.
N-GRAM	
Avantage	Inconvénient
1. Basé sur le concept de n-gram et de comparaisons de chaînes. 2. Indépendant de la langue.	1. Pas de temps efficace. 2. Nécessite une quantité importante d'espace pour créer et indexer les n-gram.
HMM	
Avantage	Inconvénient
1. Basé sur le concept de modèle de Markov caché. 2. Méthode non supervisée et est donc indépendant de la langue.	1. Une méthode complexe de mise en œuvre. 2. Un sur-stemme peut se produire dans cette méthode.
YASS	
Avantage	Inconvénient
1. Basé sur l'approche de regroupement hiérarchique et les mesures de distance. 2. C'est aussi une méthode basée sur un corpus. 3. Peut être utilisé pour n'importe quelle langue sans connaître sa morphologie.	1. Difficile de décider d'un seuil pour la création de clusters. 2. Nécessite une puissance de calcul importante.
KROVETZ	
Avantage	Inconvénient

1. C'est un stemmer léger. 2. Peut être utilisé comme pré-stemmer pour d'autres stemmers.	1. Pour les documents de grande taille, ce stemmer n'est pas efficace. 2. Incapacité à faire face aux mots en dehors du lexique. 3. Ne produit pas toujours un bon rappel et une bonne précision.. 4. Un lexique doit être créé à l'avance.
XEROX	
Avantage	Inconvénient
1. Fonctionne bien pour un grand document. 2. Tous les stems sont des mots valides. 3. Supprime les préfixes s'ils existent.	1. Incapacité à gérer avec des mots en dehors du lexique. 2. La dépendance au lexique le rend dépendant de la langue. 3. Non implémenté avec succès avec un autre langage que l'anglais.

Tableau 2 :Comparaison entre les algorithmes de radicalisation

4. CONCLUSION

Dans ce chapitre une synthèse bibliographique a été établie sur plusieurs algorithmes qui permettent l'extraction des acines ou des radicaux. Pour notre travail, notre choix c'est porté sur l'algorithme de porter afin de dégager les racines des mots qui produit le meilleur rendement par rapport aux autres stemmers avec moins de taux d'erreur en passent par un ensemble des règles et des conditions.

CHAPITRE III : CONCEPTION ET IMPLEMENTATION

1. INTRODUCTION

Dans ce chapitre deux systèmes d'indexation pour les documents en anglais ont été construit qui incluent les différentes étapes d'indexation telle que l'analyse lexicale, la sélection, l'extraction des radicaux et la pondération. Il s'agit du système séquentiel qui va nous servir d'outil de comparaison afin de mieux voir les avantages d'utiliser un système parallèle afin réduire le temps d'exécution lors de l'étape d'indexation qui est le but de notre travail.

2. ARCHITECTURE DE SYSTEME

Afin de mesurer les performances de notre système d'indexation parallèle, on a construit un autre système d'indexation qui est séquentiel pour permettre une bonne comparaison entre les deux systèmes (Figure 6).

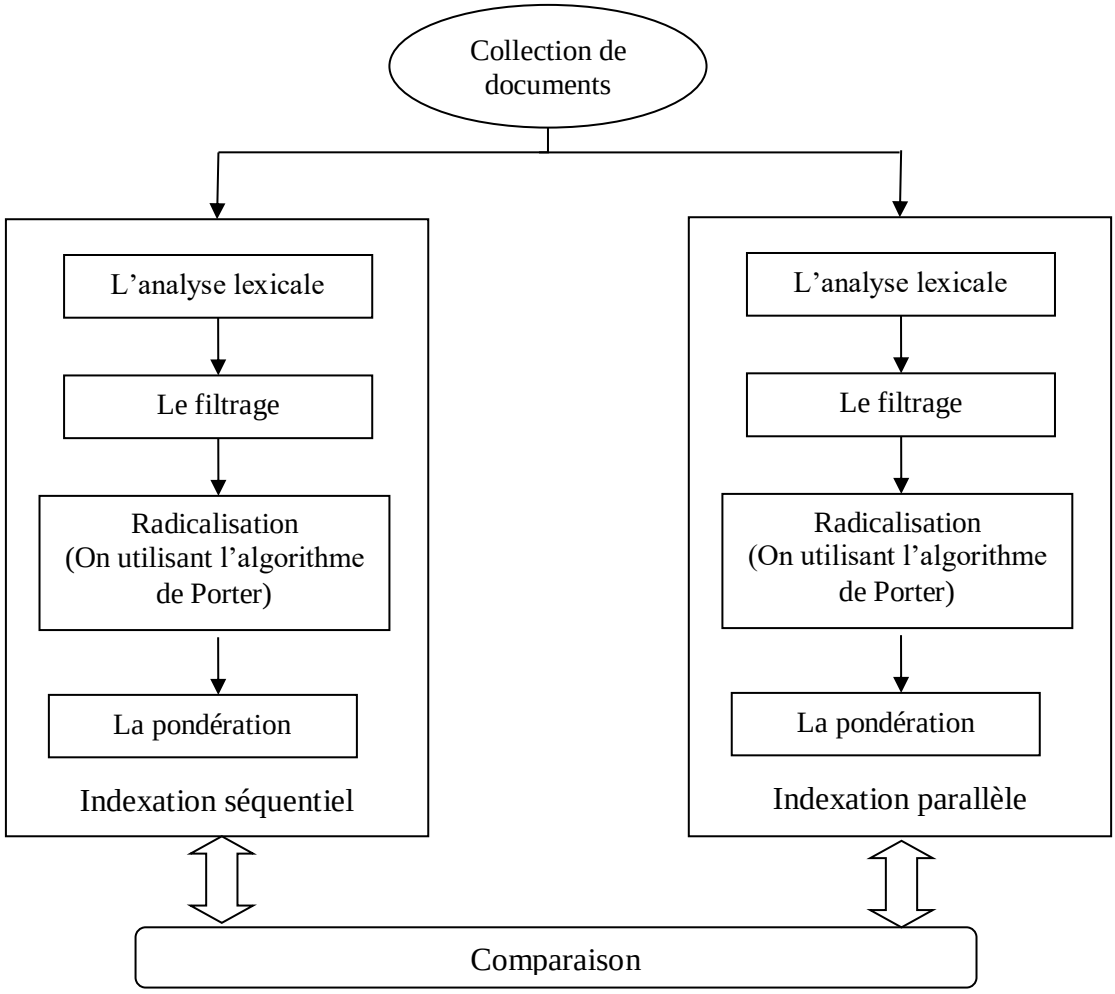


Figure 6: Architecture du système d'indexation.

2.1. Le système d'indexation séquentiel

Dans ce système, différentes étapes d'indexation seront appliquées aux documents en anglais. La première partie du système implique une analyse lexicale, suivid'un filtrage avec un anti-dictionnaire, puis nous appliquons l'algorithme de Porter pour obtenir les radicaux des termes, et enfin affecter un poids à chaque termes.

2.2. Le système d'indexation parallèle

Certaines parties du système peuvent être divisées en sous-parties afin que ces dernières puissent être exécutées en parallèle par plusieurs processeurs en même temps.

Le but de ces parallélisassions est d'obtenir un autre système d'indexation qui effectuera le même traitement (obtiendra les mêmes résultats) mais dans un temps plus court que le système séquentiel.

3. LE PROCESSUS DE SYSTEME

Dans la première étape, un système séquentiel a été développé, qui a été codé en *JAVA* à l'aide de l'environnement *eclipse* en utilisant une machine quadruple cœur pour l'exécution en parallèle.

Le système qui a été conçu suit les déférentes étapes de l'indexation : l'analyse lexicale, le filtrage, la radicalisation et la pondération.

3.1. Analyse lexicale

Dans cette étape un document textuel est transformé en un ensemble de termes en éliminant les signes de ponctuations (virgule, point, les points virgules, …), les caractères spéciaux, la numérotation, la mise en page, et la casse (majuscule en minuscule).

.

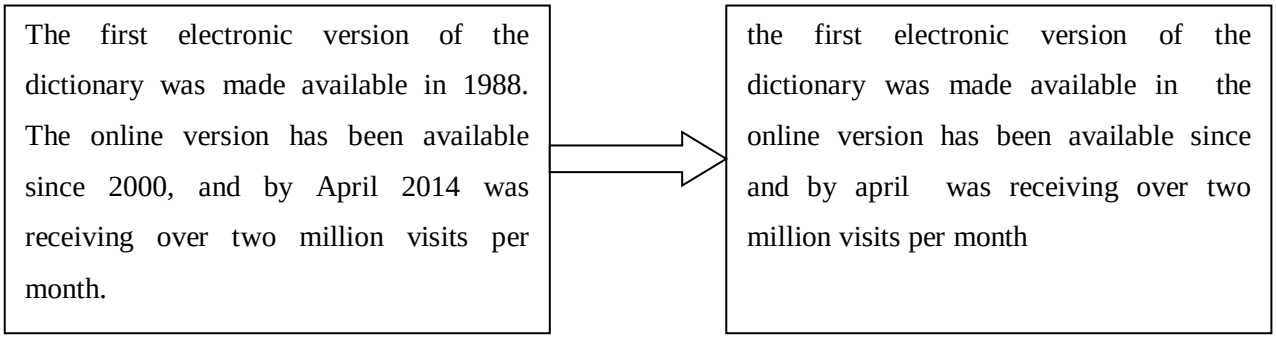


Figure 7: Exemple sur l'étape d'analyse lexicale

3.2. Le filtrage

La deuxième sous étape est la normalisation qui a pour objectif d'éliminer les mots vides de sens comme : about, always, good, before, …etc. Pour ce faire un anti-dictionnaire (*stopwords*) a été utilisé qui contient une liste des mots vides de sens pour la langue anglaise. A la fin de cette étape, il reste que les mots qu'on va y extraire le radical.

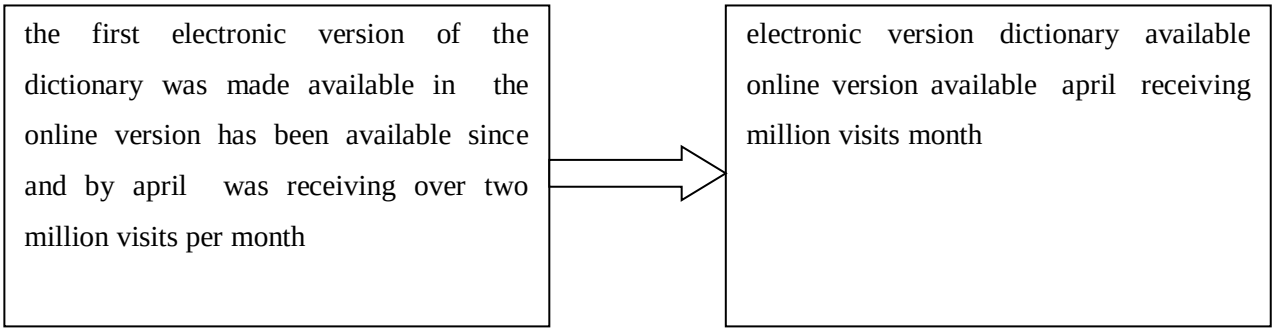


Figure 8: Exemple sur l'étape de filtrage.

3.3. L'extraction des radicaux

Dans cette étape, l'algorithme utilisé est l'algorithme de Porter. Cet algorithme permet une réduction significative de la complexité des règles associées à la suppression des suffixes en utilisant une approche unique et unifiée de la gestion du contexte.

Après l'étape de normalisation, il reste que les mots qui ne sont pas vide de sens dans le document. Alors le système prend un mot à la fois et applique sur lui l'algorithme de PORTER qui a pour objectif l'extraction du radical.

- Le parallélisme :

Le parallélisme ici consiste à restructurer l'algorithme de Porter qui est séquentiel (c'est-à-dire, le mot passe par les cinq étapes ; de la première étape jusqu'à la cinquième) afin de le rendre parallèle (c'est-à-dire, le mot s'exécute sur les tous les étapes en même temps).

Cinq threads ont été créé qui sont contrôlés par le thread Manager. On note que pour chaque thread dispose d'un *IDthread* initialisé respectivement de 0 à 4.

Principe de la restructuration de l'algorithme de Porter :

Le thread Manger déclenche tous les threads en même temps avec l'envoi du mot (*Word*). Dans notre système on utilise un tableau de chaine contenant 05 cases. Apres la fin de l'exécution des 05 threads, chaque thread a un *NewWord* comme résultat et qui sera placé dans la case du tableau appropriée. Une comparaison du *NewWord* avec le *Word* (mot original) sera effectuée par le système (figure 9).

Si le mot original est changé le *IDthread* sera changé. Par la suite, le thread Manager déclenche une autre exécution avec le nouveau mot à partir de *IDthread +1*. Sinon, le système affiche le mot original comme résultat dans le cas où le *IDthread* n'est pas changé (c'est-à-dire, le mot original est son propre radical) (figure 9).

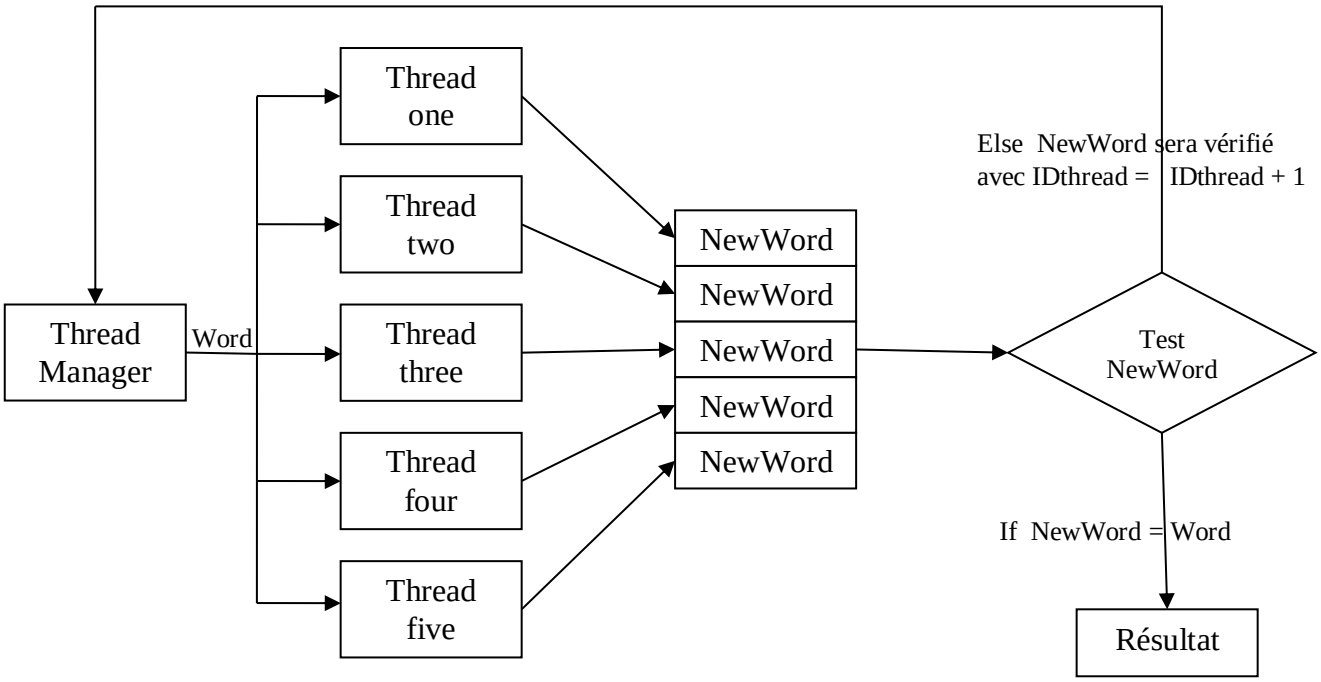


Figure 9: Algorithme de Porter parallèle pour l'extraction de radicale

Algorithme de Porter parallèle :

Algorithme de Porter parallèle

Entrées : fichier contenant l'ensemble des mots

Sorties : fichier contenant l'ensemble des mots indexés

Début

Initialisation :

IDthread entier égale à 5 ;

Tableau contient 5 case ;

Fonction Thread one (Word)

New Word = Step1(Word) ;

Si (New Word != Word)

    IDthread = 0 ;

Fin si

Tab [0] = New Word

Fonction Thread Two (Word)

New Word = Step2(Word) ;

Si (New Word != Word)

    IDthread =1 ;

Fin si

Tab[1] = New Word

Fonction Thread three (Word)

New Word = Step3(Word) ;

Si (New Word != Word)

    IDthread = 2 ;

Fin si

Tab[2] = New Word

Fonction Thread four (Word)

New Word = Step4(Word) ;

Si (New Word != Word)

    IDthread = 3 ;

Fin si

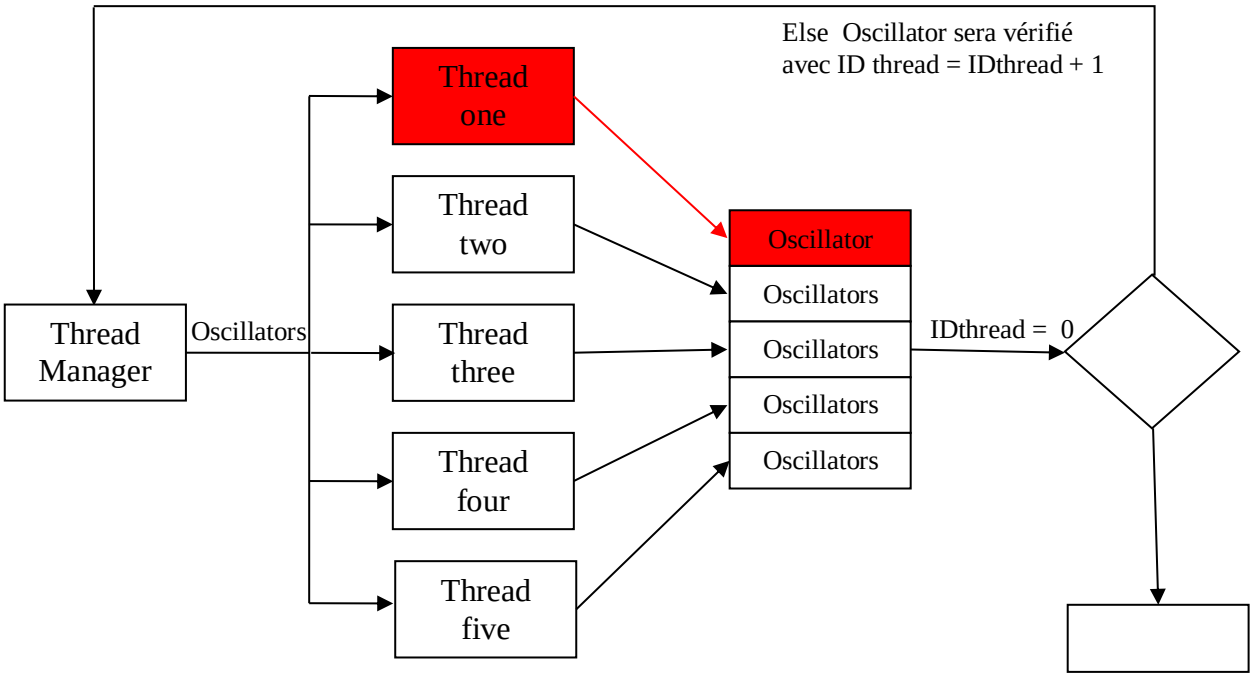
Tab[3] = New Word

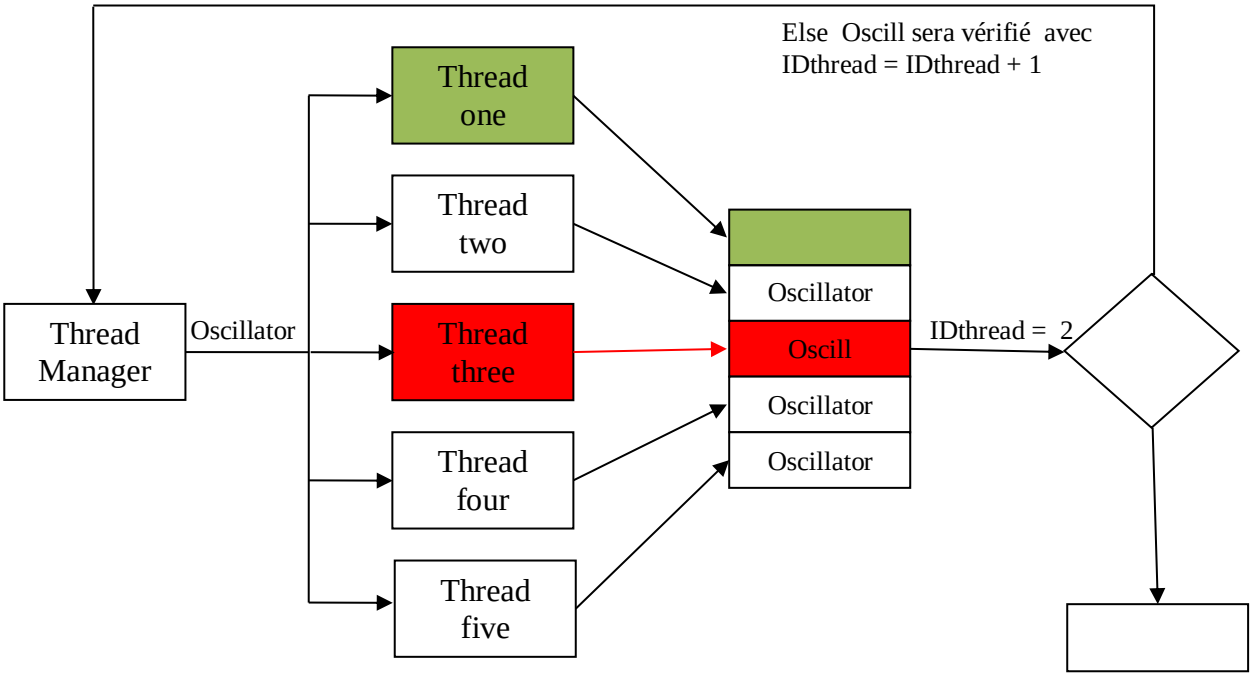
```
Fonction Thread five (Word)
New Word = Step5(Word) ;
Si (New Word != Word)
    IDthread = 4 ;
Fin si
Tab[ 4 ] = New Word
 Tant que (non fin fichier) faire
  Déclencher tous les threads aux  même temps
    Tant que (IDthread != 5 ou IDthread != 4) faire
      Déclencher les threads avec
      IDthread = IDthread + 1
     Fin
   Retourne New Word ;
 Fin
Fin
```

Exemples sur les cas que nous pouvons rencontrer :

*Le pire de cas :*en prend le terme « Oscillators » comme exemple ; le système déclenche tous les threads au même moment. La terminaison de ce terme est « s » donc l'étape 1 va se déclencher (voir le tableau des conditions dans chapitre 2)qui va supprimer cette terminaison. Ensuite, le système fait une comparaison entre le terme originale et le nouveau terme. Et comme les deux termes ne sont pas égaux ; le système déclenche à nouveau les Threads ayants un *IDthread* supérieur à celui qui viens de se déclencher.

La terminaison de nouveau terme « Oscillator » est « ator », alors l'étape qui va se déclencher pour éliminer cette terminaison est l'étape 4 (voir Tableau 1). Par la suite, on obtient un nouveau terme « Oscill », le système fait une autre comparaison entre le terme original « Oscillator » et le nouveau terme « Oscill », correspondant au *IDthread 3*. Comme les deux termes ne sont pas égaux ; le système déclenche à nouveau les Threads ayants un *IDthread* supérieur à celui qui viens de se déclenche. Dans ce cas, les Threads 1, 2, 3 et 4 sont éliminés. Enfin, le terme « Oscill » sera testé dans la dernière étape, qui sera déclenchée car il y a une règle qui a été satisfaite (Tableau 1), donc le terme sera changé à « Oscil » voir Figure 10.


Else Oscillator sera vérifié
avec ID thread = IDthread + 1
Thread one
Thread two
Thread three
Thread four
Thread five
Thread Manager
Oscillators
Oscillator
Oscillators
Oscillators
Oscillators
Oscillators
IDthread = 0


Else Oscill sera vérifié avec
IDthread = IDthread + 1
Thread one
Thread two
Thread three
Thread four
Thread five
Thread Manager
Oscillator
Oscillator
Oscill
Oscillator
Oscillator
IDthread = 2

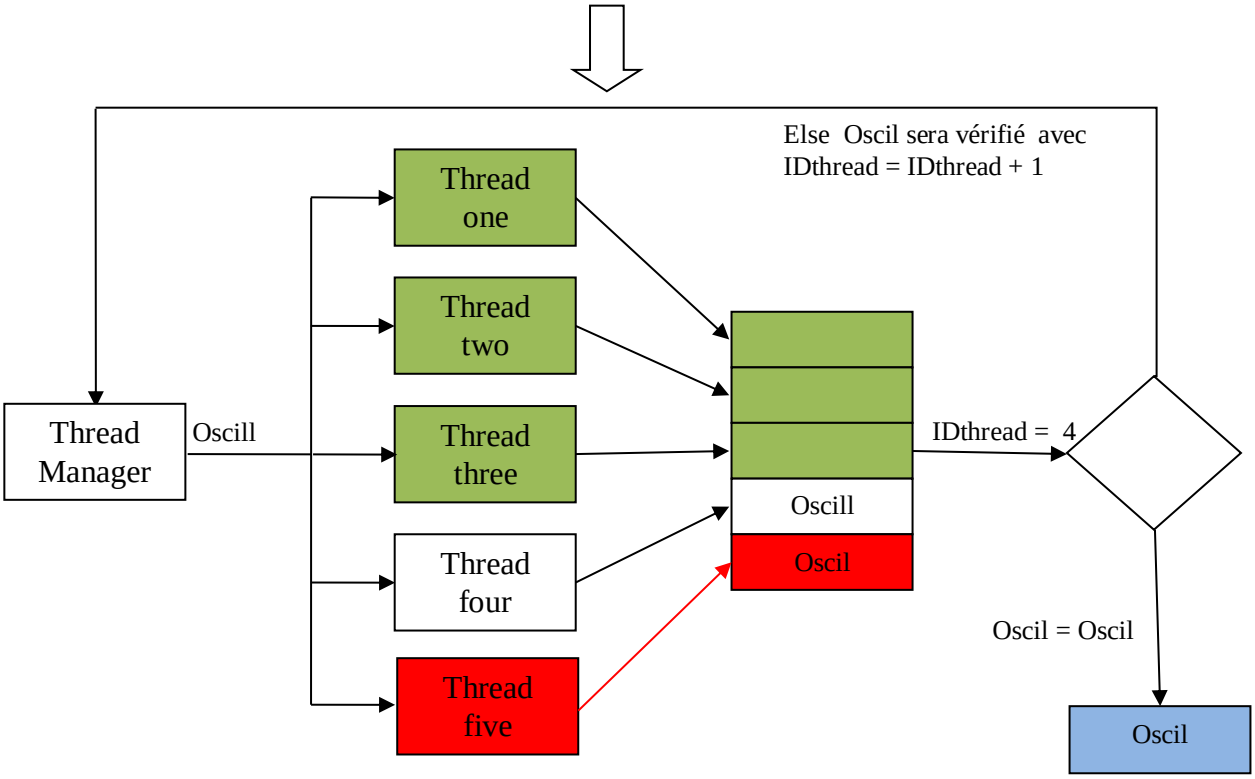


Figure 10: Exemple de pire de cas de radicalisation du mot Oscillators.

En comparaison entre le système séquentiel et le système parallèle dont les étapes se déclenches 3 fois (dans notre cas), ce dernier il reste bénéfique car les étapes dans le système séquentiel se déclenches séquentiellement (5 fois l'une après l'autre). Beaucoup de teste ont été effectués et le maximum d'étapes concernées par le radical sur un même mot est de 3 au plus.

Le cas normal :

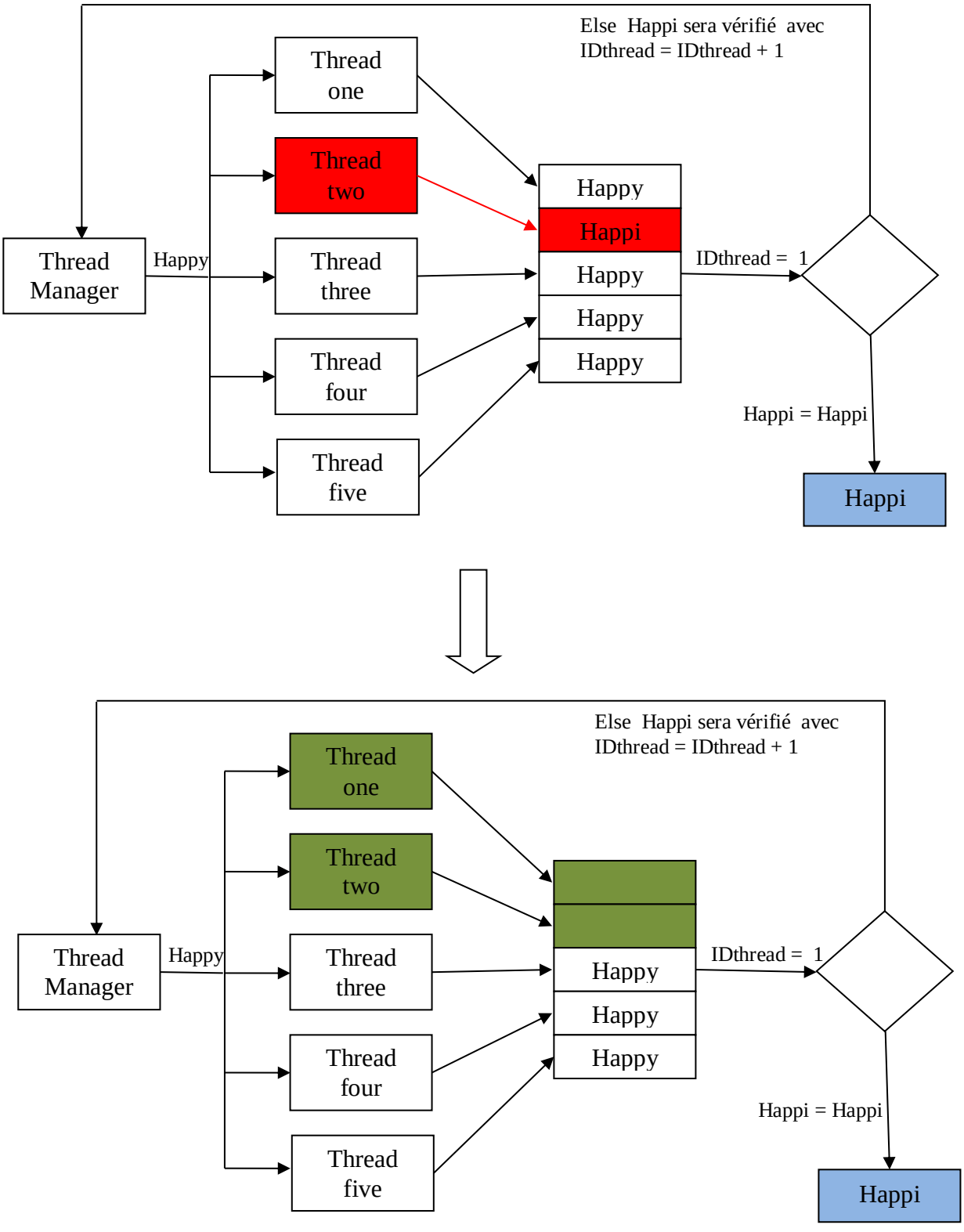


Figure 11: Exemple du cas normal de radicalisation du mot Happy.

Le meilleur cas :

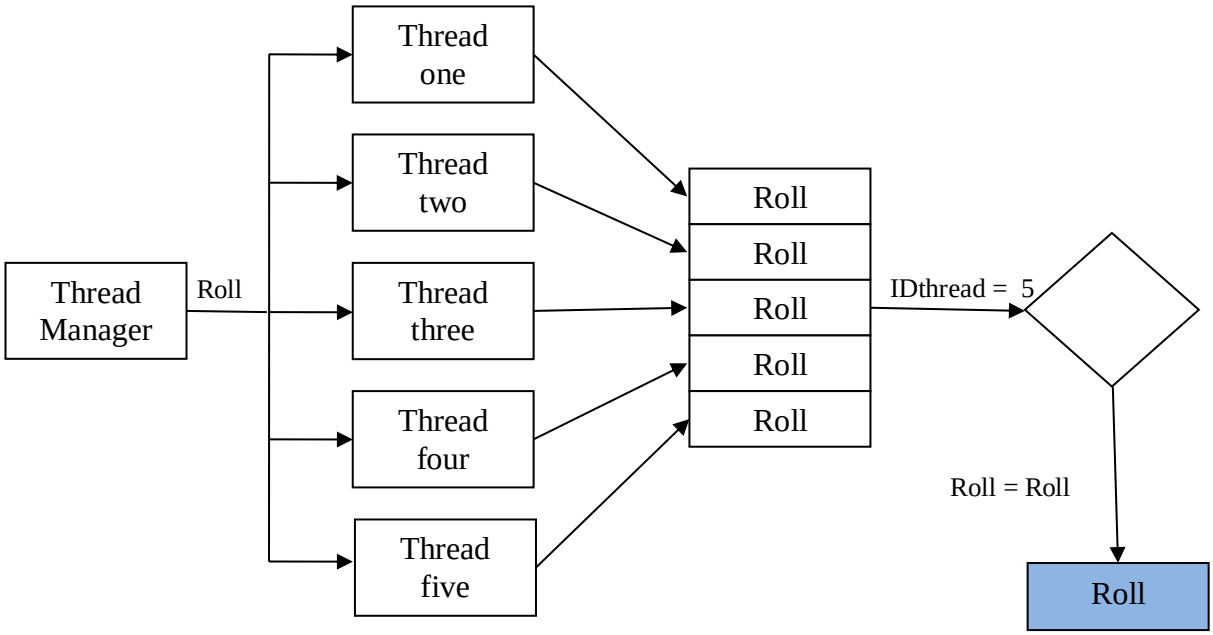


Figure 12: Exemple de meilleur de cas de radicalisation du mot Roll

3.4. La pondération

Dans ce système, le poids représente le nombre d'occurrence du mot dans le document. Dans l'exemple précédent, le mot « available » à un poids de « 2 » (figure 07).

Exemple sur le processus d'indexation :

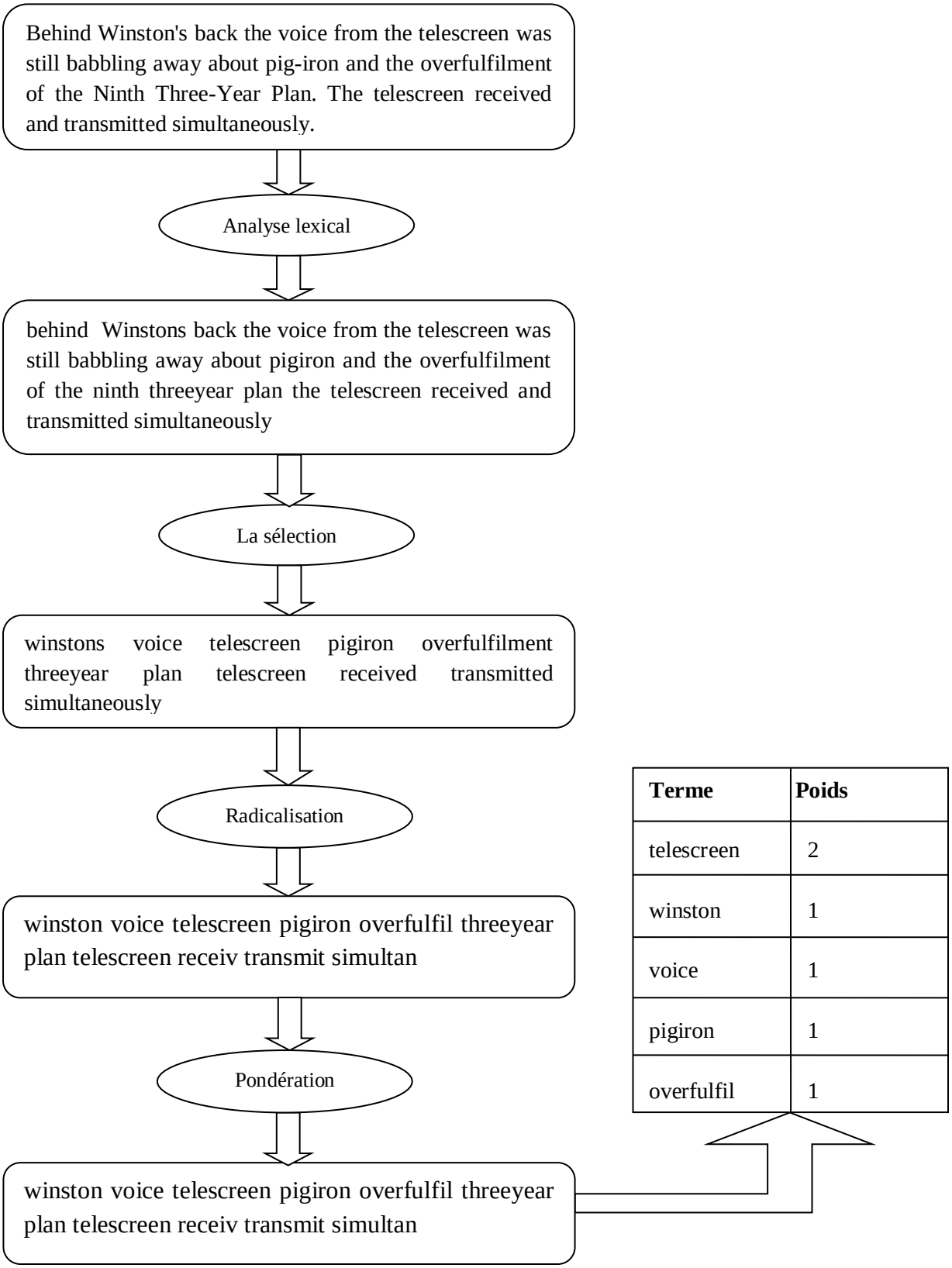


Figure 13: Exemple sur le processus d'indexation.

En exécutant notre système sur un exemple et dans les mêmes conditions (la même machine et le même document) en appliquant les deux systèmes conçus, on obtient :

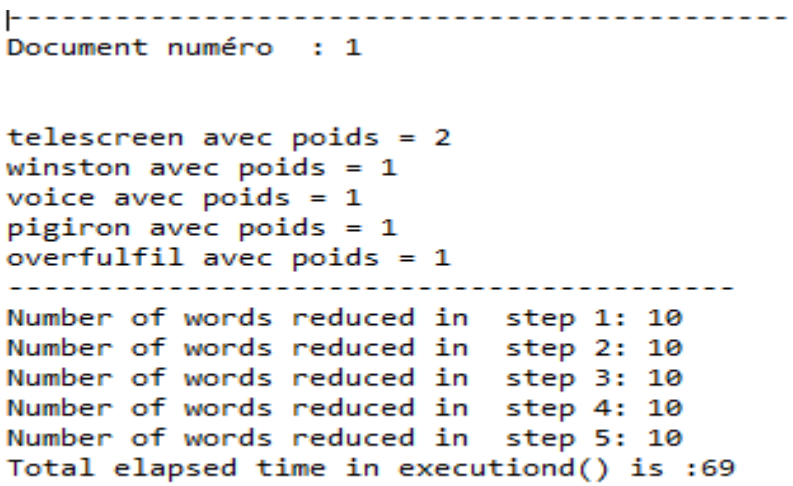


Figure 14 : Exécution du système séquentiel.

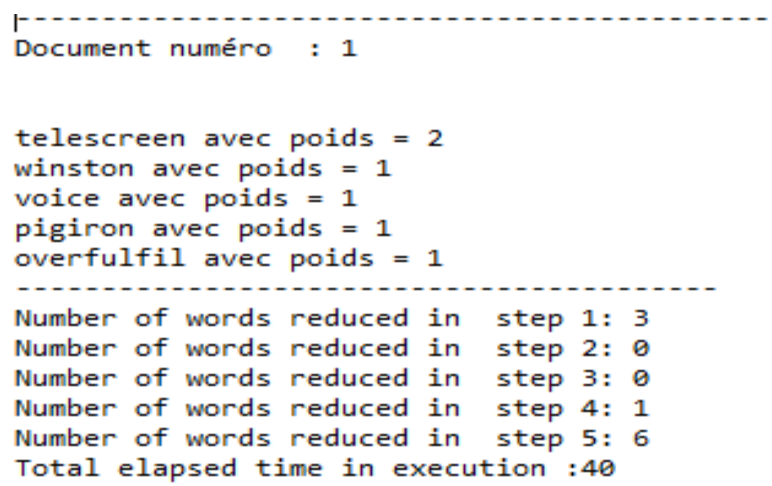


Figure 15 : Exécution du système parallèle.

Le système affiche les termes indexés avec leurs poids respectifs. Le système affiche aussi le temps écoulé en millisecondes nécessaire pour indexer le document. Il est constaté que le système parallèle est moins couteux en fonction de temps en comparaison avec le système séquentiel, et avec une différence plus au moins remarquable.

4. Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes qui constituent notre système d'indexation parallèle. A partir de tous ces résultats, nous pouvons conclure que le système parallèle est plus efficace que le système séquentiel et son efficacité est remarquable pour une taille de données non excessives. Par contre lorsqu'il s'agit d'une taille de données très volumineuse le système se bloque à cause de la machine utilisée qui n'est pas adaptée à ce type d'exécution parallèle.

CONCLUSION GENERALE

CONCLUSION GENERALE

Dans ce mémoire nous avons présenté les notions de base de la recherche d'information de façon générale. Ainsi que les déférents modèles de la recherche d'information en décrivant brièvement les avantages et les limites de chaque modèle.

Dans ce travail, deux systèmes d'indexations ont été développés ; le premier séquentiel qui va nous servir lors de la comparaison avec le deuxième système qui est parallèle. Ce dernier est basé sur la parallélisassions le l'algorithme de Porter pour l'extraction des radicaux. Cette transformation de l'algorithme de Porter va nous permettre d'améliorer les performances en termes de temps.

Il existe plusieurs algorithmes qui traitent la radicalisation. Comme les algorithmes de : Lovins, Porter, Paice/Husk, N-Gram, HMM, YASS, Krovetz et Xerox. Dans notre mémoire nous somme basé sur l'algorithme de Porter qui a été choisi dans notre travail, de manière à développer un système parallèle dans le but d'améliorer les performances en termes de temps.

En comparaison avec la version séquentiel du même système et dans les mêmes conditions, le système réalisé nous a apporté un gain de temps assez considérable.

A partir des résultats trouvés, nous pouvons conclure que le système parallèle est plus efficace que le système séquentiel et son efficacité est remarquable pour une taille de données assez volumineuse.

Dans notre travail, nous avons rencontré des plusieurs difficultés à cause de notre machine utilisée pour exécuter notre système parallèle. Une machine ou un réseau de machines parallèle plus performant est plus que nécessaire pour pouvoir traiter une taille de données plus grandes (qui parfois se bloc à cause des limite de notre matériel).

PERSPECTIVES

PERSPECTIVES

Nous proposons de paralléliser d'autres algorithmes de radicalisation et d'autres étapes d'indexation.

Aussi, 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'autre langues.

Le système peut traiter non seulement les documents numériques textuels mais aussi les images et les vidéos.

REFERENCES BIBLIOGRAPHIQUES

REFERENCES BIBLIOGRAPHIQUES

Baziz, M. (2005). Indexation conceptuelle guidée par ontologie pour la recherche d'information. *Thèse de doctorat*. Universit de Toulouse III- Paul Sabatier, Toulouse, France.

Belkin, N. J., & Croft, W. B. (1992). Information Filtering and Information Retrieval: Two Sides of the Same Coin. *Communications of the ACM*, 29-38.

Ben Aouicha, M. (2009). Une approche algébrique pour la recherche d'information structurée. *Thèse de doctorat. Universit de Toulouse III- Paul Sabatier*. Toulouse, France.

BOUBEKEUR, F. (2008). Contribution à la définition de modèles de recherche d'information flexibles basés. *Thèse de doctorat. Université Toulouse III - Paul Sabatier*. Toulouse, France.

Champclaux, Y. (2009). un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amelioration du processus de recherche d'information. *Thèse de doctorat. Université Toulouse III - Paul Sabatier*. Toulouse, France.

Cleverdon, C. (1970). Evaluation tests of information retrieval systems. *Journal of Documentation*, 55-64.

Cooper, W. (1988). Getting Beyond Boole. *Information Processing & Management*, 24-30.

Daoud, M. (2009). Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche. *Thèse de doctorat. Université Toulouse III - Paul Sabatier*. France.

Edward A, F., & Matthew B, K. (1988). Partial Enhanced Boolean Retrieval: Experiments with the SMART and SIRE Systems. *Information Processing & Management*, 24:3.

Eiman Tamah, A.-S. (2008). Towards an Error-Free Stemming. *IADIS European Conference on Data Mining*, (pp. 160-163). Amsterdam, Pays-Bas.

Gerard, S. (1971). *The SMART retrieval system: Experiments in automatic document processing.* Englewood Cliffs, N.J.: Prentice-Hall.

Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science.*, 42, 7-15 .

Harter, S. (1992). Psychological relevance and information science. *Journal of the Association for Information Science and Technology*, 602-615.

Hull, D. A., & Gregory, G. (1996). A detailed analysis of English Stemming Algorithms. France: Rank Xeros Research Center.

Jivani, A. G. (2011). A Comparative Study of Stemming Algorithms. *International Journal of Computer Applications in Technology, 2*(6), 1930-1938.

Krovetz, R. (1993). Viewing morphology as an inference process. *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 191-202).

Lovins, J. B. (1968). Development of a Stemming Algorithm. *11*. Mechanical Translation and Computational Linguistics.

Luhn, H. (1955). A New Method of Recording and Searching Information american documentation. *American documentation*, 14-16.

Marcus, R. S. (1991). Computer and Human Understanding in Intelligent Retrieval Assistance. *American Society for Information Science*, 28.

Melucci, M., & Orio, N. (2003). A novel method for stemmer generation based on hidden Markov models. *Proceedings of the twelfth international conference on Information and knowledge management*, (pp. 131-138).

Paice, C. D. (1990). Another stemmer. *24*, pp. 56-61. ACM SIGIR Forum.

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14. 3*, 130-137.

Prasenjit, M., Mandar, M., Swapan, K. P., Gobinda, K., Pabitra, M., & Kalyankumar, D. (2007). "YASS: Yet another suffix stripper". *ACM Transactions on Information Systems, 25*(18).

RODHAIN, F., FALLERY, B., GIRARD, A., & DESQ., S. (2010). Une histoire de la recherche en systèmes d'information à travers 30 ans de publications. *(60)*, 78-97. ESKA Entreprise et histoir.

Salton, G. (1968). Search and retrieval experiments in real-time information retrieval. (pp. 1082-1093). IFTP Congress.

Salton, G. (1969). A comparison between manual and automatic indexing methods. 61-71.

Salton, G., & McGill, M. (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company.

Savoy, J. (1993). Stemming of French Words Based on Grammatical Categories. *Journal of the American Society for Information Science,* , 44 (1), p1-9 .

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

Stephen, R., & Karen, S. J. (1976). Relevance weighting for search terms. *Journal of The American Society for Information Science*, 27(3) :129–146.

Zipf, G. K. (1949). human behavior and the principle of least effort. Addison-Wesley.