République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou
Faculté de Génie Electrique et de l'Informatique
Département informatique

Mémoire de fin d'étude de master Académique

Domaine: Mathématique et informatique

Filière: Informatique

Spécialité: Systèmes Informatiques

Thème :

Extension d'une approche d'expansion de requêtes basée sur le Word Embedding

Présenté par :

M^{elle} : AMARI Djouher

M^{elle} : LIMANI Ania

Devant les jurys composé de :

Président : Mr RADJA Hakim
Encadreur : Mr HAMMACHE Arezki
Examinateur : Mr SAIDANI Fayçal
Examinateur : M^{me} AIT YAKOUB Zina

Promotion : 2018/2019

Remerciements

Tout d'abord on remercie « DIEU » pour nous avoir donné la force, capacité, volonté et courage afin de mener à bien et à terme ce travail

Nous adressons également nos sincères remerciements à :

-Notre promoteur Mr HAMMACHE Arezki pour ses précieux conseils, son dévouement, pour son suivi et pour nous avoir aussi bien encadrées tout au long de la réalisation de ce projet. Nous tenons aussi à lui adresser notre gratitude pour tout le temps qu'il nous a consacrée, sa disponibilité ainsi que ses encouragements.

-Aux membres du jury pour avoir accepter de bien vouloir lire notre travail, l'examiner, l'évaluer et nous corriger.

-Aux parents, et familles qui nous ont toujours été là pour nous, nous avoir soutenues et encouragées tout au long de nos années d'études.

Dédicaces

Je dédie ce travail à :

Mes chers parents que j'aime très fort, pour leurs sacrifices, et précieux conseils, que Dieu les garde et les protège

A mes sœurs : Chimci , son mari Aghiles, katia et ouardia qui m'ont toujours soutenue.

A toutes mes tantes plus particulièrement Khalti Meriem qui prie toujours pour que je réussisse.

A ma très chère binôme Djouher « Djoudjou » et toute sa famille

A tous mes collègues de la promotion 2018/2019

A toute personne qui a contribué de prés ou de loin à l'aboutissement de ce travail

ANIA

Dédicaces

Je souhaite dédier ce travail :

A la mémoire de mon père, qui a tant fait pour moi et sans qui ce travail n'aurais peut être pas vu le jour.

A ma très chère mère que j'aime très fort, pour son soutien et son encouragement tout au long de mes années d'études.

A mon petit ange, mon frère Sadi que le dieu le garde pour moi.

A mon grand frère Mohand Oublaid ainsi que sa femme Sabrina et ses filles Farah et Iliana.

A mes chères sœurs Mina et Dabi qui ont été toujours a mes cotés.

Ma sœur Fafouche ainsi son marie Smail et ses deux anges Faroudja et Dehbia.

Mes oncles, mes tantes, mes cousins et cousines.

A mon très cher binôme Ania, pour les merveilleux moments passés ensemble, et à sa famille.

A tous mes collègues de la promotion 2018/2019.

Mes amies et a toutes personnes qui m'ont aidé de prés ou de loin.

DJOUHER

Sommaire

Introduction générale 1

Chapitre I : La recherche d'information

I.1 Introduction 3

I.2 Définition de la recherche d'information(RI) 3

I.3 Système de recherche d'information (SRI) 3

I.4 Concepts de base de la recherche d'information 4

I.4.1 La requête 4

I.4.2 Document et collection de documents 4

I.4.3 La pertinence 5

I.5 Les processus de recherche d'information 5

I.5.1 L'indexation 6

I.5.1.1 L'analyse lexicale 8

I.5.1.2 L'élimination de mots vides 8

I.5.1.3 La normalisation des termes 9

I.5.1.4 La pondération 9

I.5.2 L'appariement document-requête 10

I.5.3 La reformulation de la requête 11

I.6 Les modèles de la recherche d'information 12

I.6.1 Le modèle ensembliste 12

I.6.2 Les modèles algébriques 14

I.6.3 Les modèles probabilistes. 15

I.6.3.1 Le modèle de langue. 16

I.7 Evaluation des systèmes de recherche d'information 16

I.7.1 Les mesures d'évaluation de SRI 17

I.7.2 Les collections de tests 20

I.7.2.1 Les collections TREC. 20

I.8 Conclusion 21

Chapitre II : Expansion de requêtes et le Word Embedding

I.1 Introduction..... 22

II.2 Expansion de requête..... 22

II.2.1 Définition..... 22

II.2.2 Classification des approches de l'expansion de requête..... 22

II.2.2.1 Approches statistiques..... 22

II.2.2.2 Approches à partir de logs..... 23

II.2.2.3 Approches linguistiques..... 23

II.2.3 Les étapes de l'expansion de requête..... 24

II.2.3.1 Prétraitement de la source de données..... 25

II.2.3.2 Pondération et classement des termes d'expansion des requêtes..... 25

II.2.3.3 Sélection des conditions d'expansion de requêtes..... 27

II.2.3.4 Reformulation de la requête..... 27

II.3 Le Word Embedding..... 28

II.3.1 Définition..... 28

II.3.2 Les réseaux de neurones..... 30

II.3.3 Les modèles du Word Embedding..... 33

II.3.3.1 La modélisation linguistique..... 34

II.3.3.2 Le modèle classique de langage neuronal..... 34

II.3.3.3 Le modèle C&W (Collobert et weston)..... 35

II.3.3.4 Le modèle Word2Vec..... 36

II.3.3.5 Le modèle Glove..... 38

II.4 Le Word Embedding en recherche d'information..... 38

II.4.1 Expansion de requête en utilisant le word embedding..... 39

II.5 Conclusion..... 41

Chapitre III : Approche proposée, implémentation et résultats

III.1 Introduction 42

III.2 L'approche de Roy 42

III.2.1 Objectif de l'approche 42
III.2.2 Les versions de l'approche 42

III.3 Présentation de l'approche proposée 45
III.3.1 Principe de l'approche proposée et sa formalisation 45
III.3.2 La mise en œuvre de l'approche 46
III.4 L'environnement de développement 49

III.4.1 Lucene 49
III.4.1.1 L'architecture de Lucene 50

III.4.2 Les API developpées par Roy 52

III.4.3 TREC_EVAL 53

III.4.4 Le langage de programmation java 53
III.4.4.1 Caractéristiques 53

III.4.4 Netbeans 54

III.5 Evaluation et résultats 55

III.5.1 La collection de test et les requêtes utilisées 55

III.5.2 Présentation des résultats obtenus 56

III.5.2.1 Résultats obtenus avant et après notre extension 56

III.6 Conclusion 59

Conclusion générale 59

Bibliographie

Webographie

Liste des figures

Figure I.1 : Processus en U de la recherche d'information. 6

Figure I.2 : Les étapes de l'indexation. 8

Figure I.3 : Taxonomie des modèles de RI 12

Figure I.4 : Courbe Rappel-précision. 19

Figure II.1 : Taxonomie des approches de l'expansion de requête. 22

Figure II.2 : Les étapes d'expansion automatique de la requête 24

Figure II.3 : Word embedding obtenu avec les modèle Word2vec 29

Figure II.4 : Différence constante mal-femelle 30

Figure II.5 : Schéma générale d'un réseau de neurones 31

Figure II.6 : Analogie entre neurone biologique/neurone artificiel 31

Figure II.7 : Réseau de neurones monocouche 33

Figure II.8 : Réseau de neurones multicouche 33

Figure II.9 Architecture du modèle C&W 36

Figure II.10 : Comparaison entre le modèle CBOW et le modèle Skip-gram 37

Figure III.1 : Extrait d'un fichier résultats de la première recherche 47

Figure III.2 : Extrait d'un fichier qui contient les Word Embedding du terme « notional » 48

Figure III.3 :L'architecture de lucene 50

Figure III.4 : La classe indexation 50

Figure III.5 : La classe recherche 51

Figure III.6 : Les 1000 documents retournés 52

Figure III.7 : Environnement de développement Netbeans 55

Figure III. 8 : Graphe illustrant les résultats de la MAP requête par requête 57

Liste des tableaux

Tableau I.1 : Exemple de calcul de rappel et de précision pour une requête. 18

Tableau III.1: Tableau explicatif de l'approche Pre-retrieval incremental kNN 44

Tableau III.2 : Résultats de la MAP globale 56

Tableau III.3 : Résultats de la MAP des requêtes améliorées 58

INTRODUCTION GENERALE

Introduction générale

Avec l'avènement d'internet, le volume de données ne cesse d'augmenter, chaque jour un grand nombre de données est utilisées et produites sous tous les formats : texte, images, vidéos et audio. Aussi, des milliers de requêtes sont lancées sur des navigateurs, mais à cause des grandes quantités d'informations présentes sur internet, il est devenu très difficile de trouver la donnée ou le document qui satisfait le besoin en information de l'utilisateur.

Pour remédier à cette difficulté de recherche, on a du envisager des outils automatiques qui permettent d'accéder et cibler l'information pertinente à retourner. Ce qui a donné naissance à une nouvelle discipline appelée « La Recherche d'Information RI».

L'opération de la RI est réalisée par des outils appelés systèmes de recherche d'information SRI. Un système de recherche d'information est un système informatique qui permet de retourner à partir d'un ensemble de documents, ceux dont le contenu correspond le mieux à un besoin en information d'un utilisateur, exprimé à l'aide d'une requête, en mettant en correspondance une représentation du besoin de l'utilisateur avec celle du contenu des documents. Cependant, les utilisateurs n'emploient pas le même vocabulaire que celui des auteurs des documents. Par conséquent, les résultats retournés ne correspondent toujours pas au besoin en information de l'utilisateur. Pour y remédier l'expansion de requête est utilisée. L'expansion de requête consiste à étendre la requête initiale avec des mots liés sémantiquement aux termes de la requête. Parmi les méthodes utilisées récemment pour sélectionner les termes d'expansion : le Word Embedding. Ce dernier est une technique qui permet de représenter les mots avec des vecteurs de nombre de faible dimension. Grâce à cette représentation les mots qui sont sémantiquement similaires sont représentés avec des vecteurs similaires.

Notre travail s'insère dans le domaine de la recherche d'information(RI). Plus précisément, il consiste en l'extension d'une approche pour l'expansion de requête basée sur le Word Embedding (WE) proposée dans **[46]**.

Pour atteindre cet objectif, le présent mémoire, comporte outre l'introduction générale, la conclusion et la bibliographie, les trois chapitres suivants :

Chapitre I « La recherche d'information » : L'objectif de ce chapitre est de présenter le domaine de la recherche d'information. Dans un premier temps, nous présenterons les concepts de base de la recherche d'information puis nous passerons aux modèles de la recherche d'information et nous finirons par traiter de l'évaluation des systèmes de recherche d'information (SRI).

Chapitre II « L'expansion de requête et le Word Embedding » : Dans ce chapitre nous présenterons l'expansion de requête et le Word Embedding. Dans un premier temps nous allons définir l'expansion de requête et ses étapes, puis nous allons introduire le concept de Word Embedding. Enfin nous présenterons des travaux en recherche d'information exploitant le Word Embedding.

Chapitre III « L'approche proposée, L'implémentation et résultats » : ce chapitre est dédié à décrire l'approche de notre travail, son implémentation ainsi que les différentes outils utilisés pour la réaliser et nous terminerons avec la présentation et la discussion des résultats obtenus.

Enfin, nous concluons notre mémoire avec une conclusion générale et quelques perspectives.

CHAPITRE I

LA RECHERCHE D'INFORMATION

I.1 Introduction

Avec l'invention du web au début des années quatre vingt dix et les quantités immenses d'informations récoltées quotidiennement, le besoin de la recherche d'information s'est vite fait ressentir. La recherche d'information est la science de la recherche d'information dans des documents, qu'ils soient dans une basede données, dans une base documentaire ou sur le web.

L'opération de la RI est réalisée par des outils informatiques appelés Systèmes de Recherche d'Information (SRI). Ces systèmes ont pour but de mettre en correspondance une représentation du besoin en information de l'utilisateur (une requête) avec une représentation du contenu des documents au moyen d'une fonction de comparaison. L'essor du web a mis la RI face à de nouveaux défis d'accès à l'information, il s'agit cette fois de retrouver une information pertinente dans un espace diversifié et de taille considérable.

L'objectif de ce premier chapitre est de présenter le domaine de la recherche d'information. Dans un premier temps, nous présenterons les concepts de base de la recherche d'information puis nous passerons aux modèles de la recherche d'information et nous finirons par traiter de l'évaluation des systèmes de la recherche d'information.

I.2 Définition de la recherche d'information (RI)

Plusieurs définitions de la Recherche d'Information ont été citées dans la littérature, celles-ci se ressemblent et se rejoignent sur plusieurs points, la plus générale est la suivante :

La recherche d'information (RI) traite de la représentation, du stockage, de l'organisation et de l'accès à l'information. La RI fournit donc les techniques et outils pour permettre de représenter, stocker, organiser, rechercher et retrouver, dans une masse documentaire existante, les documents contenant l'information qui répond aux besoins informationnels exprimés par l'utilisateur sous forme de requête **[1].**

Pour assurer l'ensemble des fonctions nécessaires à la recherche d'information on fait appel à un système de recherche d'information (SRI)

I.3 Système de la recherche d'information (SRI)

Pour répondre au mieux aux requêtes des utilisateurs, un Système de Recherche d'Information offre un ensemble de programmes informatiques qui permettent d'établir une correspondance

entre la requête de l'utilisateur et collection de documents, afin de retourner les documents les plus pertinents **[1].**

I.4 Les concepts de base de la recherche d'information

La définition précédente du Système de Recherche d'Information fait ressortir trois notions très importantes dans la RI : Requête, Documents (collection de documents), et Pertinence.

I.4.1 La requête

La requête consiste en l'expression des besoins en informations de l'utilisateur. Elle est l'intermédiaire entre le SRI et l'utilisateur.

La requête peut être exprimée **[1]** :

- Comme une liste de mots clés ;
- En langage naturel ;
- En langage booléen.

Il existe 3 types de requête :

- Les requêtes informationnelles qui ont pour objectifs de chercher de l'information en rapport à un sujet donné, sans aucun a priori sur la source d'information.
- Les requêtes navigationelles qui consistent à atteindre une page web particulière, connue par l'utilisateur.
- Les requêtes transactionnelles qui souhaitent réaliser des transactions ou bénéficier d'un service en ligne.

I.4.2 Document et collection de documents

- **Document :**

Le document représente le conteneur élémentaire d'information, exploitable et accessible par le Système de Recherche d'Information. Le document peut être un texte, une page web, une image, une bande vidéo.

En résumé, un document est une unité qui peut constituer une réponse à un besoin en information exprimé par un utilisateur.

- **Collection de documents :**

Elle constitue l'ensemble des informations exploitables et accessibles. Elle est constituée d'un ensemble de documents. Dans le cas général et pour un souci d'optimalité, la base constitue des représentations simplifiées mais suffisantes pour ces documents **[5].**

I.4.3 Pertinence

La pertinence est la notion centrale dans la RI car toutes les évaluations s'articulent autour de cette notion. On y trouve plusieurs définitions **[6]**:

- La correspondance entre un document et une requête, une mesure d'informativité du document à la requête.
- Un degré de relation (chevauchement, relativité, etc.) entre le document et la requête.
- Une mesure d'utilité du document pour l'utilisateur

On y distingue deux types de pertinence **[1]** :

- **Pertinence système :**

C'est une évaluation objective, elle est défini à travers des modèles de recherche d'information et est traduite par un score exprimant l'adéquation entre le document et la requête.

- **Pertinence utilisateur :**

C'est une évaluation subjective car elle est liée à l'avis de l'utilisateur sur l'information renvoyée par le système en réponse à sa requête, cette évaluation est variable puisqu'une même information jugée non pertinente à un instant **t** pour une requête peut être jugée pertinente à un instant **t+1**. Elle exprime l'évaluation de l'utilisateur, de la pertinence, vis-à-vis de son besoin en information, des documents retrouvés par le SRI.

Pour répondre aux besoins informationnels de l'utilisateur, le système de recherche d'information implémente les processus suivants : l'indexation, l'appariement document-requête et la reformulation de la requête. La section suivante décrit en détails ces processus.

I.5 Les processus de la recherche d'information

Les différentes étapes du processus de RI, sont représentées schématiquement par le processus en **U.** Il est décomposé en trois principales étapes : l'indexation, l'appariement document-requête et la reformulation de la requête.

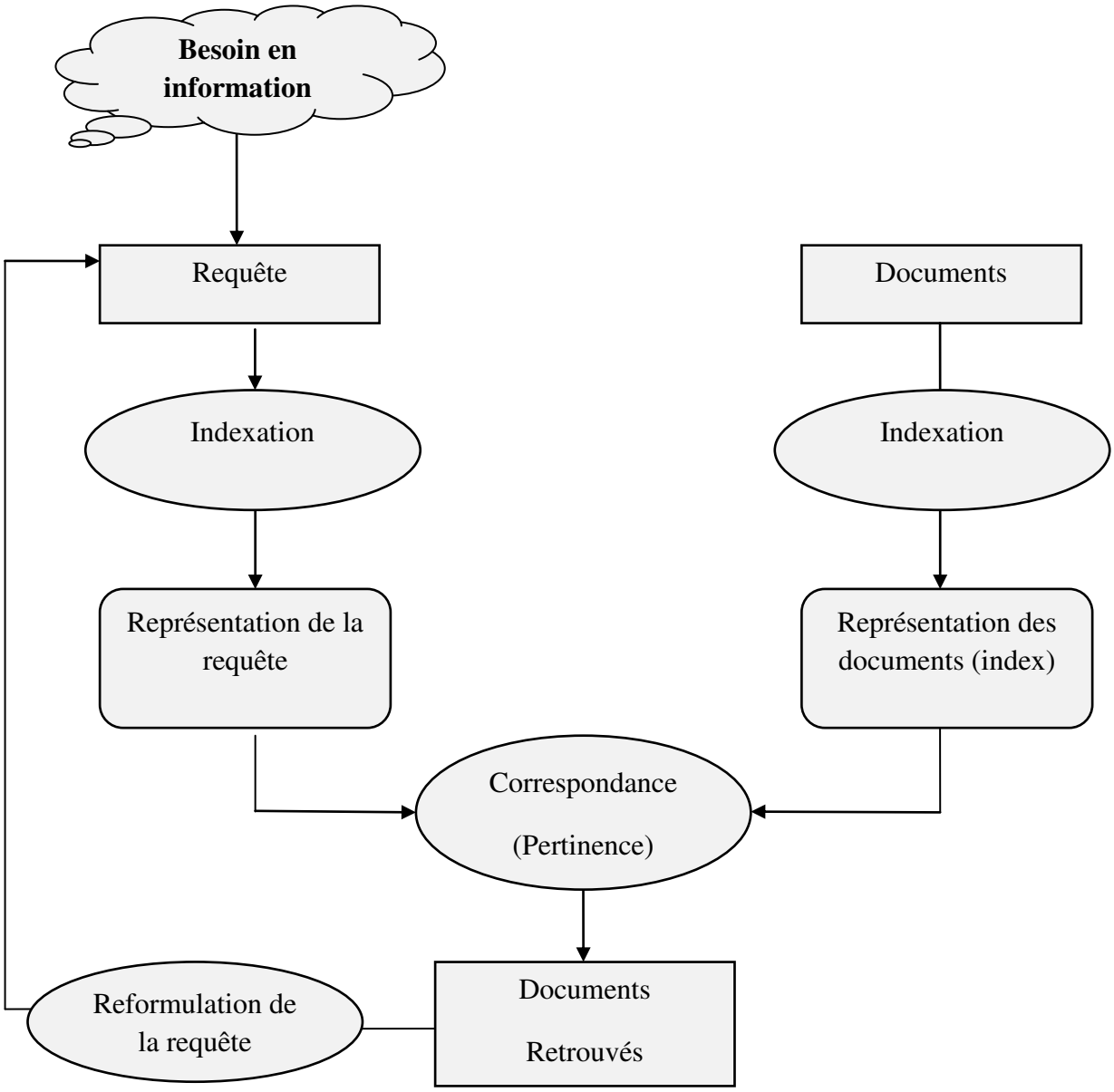


Figure I.1 : Processus en U de recherche d'information.

I.5.1 L'indexation

Un SRI gère les différentes collections de documents en les organisant sous forme d'une représentation intermédiaire permettant de refléter aussi fidèlement que possible leur contenu sémantique. L'interrogation de ce fond documentaire à l'aide d'une requête nécessite également la représentation de cette dernière sous une forme compatible avec celle des documents. Ce processus de conversion est appelé « indexation ».

En d'autres termes, l'indexation consiste à déterminer et extraire les termes représentatifs du contenu du document ou d'une requête, qui couvre au mieux leur contenu sémantique, généralement assortis de poids représentant leur degré de représentativité du contenu sémantique de l'unité qu'ils décrivent **[5].**

Il existe 03 types d'indexation : manuelle, semi-automatique et automatique **[1]**.

- **Indexation manuelle :** c'est un indexeur humain qui se charge de définir les descripteurs (mots clés) représentatifs du contenu du document, l'indexation manuelle assure meilleure précision dans les documents restitués par le SRI en réponse aux requêtes des utilisateurs, néanmoins cette indexation représente un certain nombre d'inconvénients liés notamment à l'effort humain et le prix qu'elle exige(en temps et en nombre de personnes). De plus cette indexation est subjective, liés aux facteurs humains, différents spécialistes peuvent indexer un document avec des termes différents. Pratiquement inapplicable aux corpus de textes volumineux.
- **Indexation semi-automatique:** L'indexation est réalisée par un programme informatique et un indexeur humain. Le choix final d'indexation à partir du vocabulaire fourni, est laissé à l'indexeur humain (généralement spécialiste du domaine).
- **Indexation automatique :** C'est un processus complètement automatisé, elle est réalisée par un programme informatique, elle se charge d'extraire les termes caractéristiques du document**.**

  Cette dernière est particulièrement adaptée aux corpus volumineux, elle passe par un ensemble d'étapes pour créer l'index final, comme le montre la figure suivante.

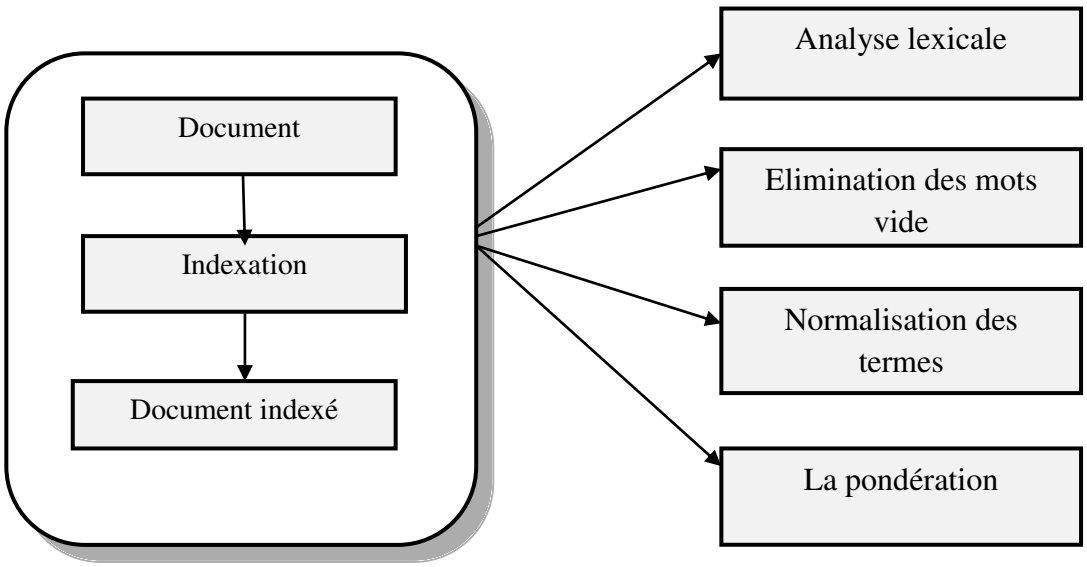


Figure I.2 : Les étapes de l'indexation automatique

I.5.1.1 L'analyse lexicale (Tokenisation)

L'analyse lexicale est l'étape indispensable pour l'identification des unités lexicales du texte puisqu'elle consiste en un processus de conversion d'une chaine de caractères (le texte) en une séquence d'unités lexicales élémentaires appelées « Token », ces dernières sont candidates à l'indexation.

L'analyse lexicale inclut les étapes suivantes :

- La conversion de la casse (majuscule en minuscule)
- L'élimination des accents.
- La tokenisation.

I.5.1.2 L'élimination des mots vides

En recherche d'information, un mot vide (stop Word en anglais) est un mot qui est tellement commun qu'il est inutile de l'indexer ou de l'utiliser dans la recherche car il n'est pas représentatif du contenu du document **[8].**

Les mots vides comme des articles et des propositions ainsi que les caractères non alphabétiques (chiffres, signes de ponctuation, trait d'union etc.) ne sont pas pris en compte comme des éléments d'indexation**[9].**

On distingue deux techniques pour éliminer les mots vides :

- L'utilisation d'une liste de mots vides (aussi appelé anti-dictionnaire ou stop-list)

- L'élimination des mots dépassants un certain nombre d'occurrences dans la collection.

I.5.1.3 La normalisation des termes

La normalisation des mots, aussi appelée « **Lemmatisation** » (ou encore radicalisation, racinisation, ou stemming), cette étape est un processus morphologique permettant de regrouper les différentes variantes d'un mot par un format unique appelé **Lemme** afin de réduire la taille de l'index.

Exemple : économie, économiquement, économiste⟶économie

Pour ce faire, on utilise des règles de transformation :

- Règle de type : condition ⟶action

  Si un mot se termine par 's'alors supprimer la terminaison

  L'algorithme le plus connu est l'algorithme de (**Porter** élimination des affixes)[52].
- Analyse grammaticale **[10]** :

- Utilisation de lexique (dictionnaire)
- Tree-tagger.

L'inconvénient de cette opération, est que dans certain cas elle supprime la sémantique des termes originaux.

I.5.1.4 La pondération [1]

La pondération est une fonction fondamentale en RI. Tous les modèles de recherche, excepté le modèle booléen, se basent sur la pondération des termes.

Le principe de la pondération est d'affecter à chaque terme **t** d'un document **d** ou d'une requête **q**, un poids numérique sensé le caractériser dans le document ou la requête, les poids des termes de la requête et du document peuvent avoir des sémantiques différentes, le poids est donc une mesure statistique de l'importance du terme dans le document (plus un terme est important dans le document plus son poids dans ce document doit être élevé) **[1].**

Parmi les mesures de pondération utilisées, nous avons la mesure $\boldsymbol{tf_{t,d}} * \boldsymbol{idf_t}$

Notant : $\boldsymbol{tf_{t,d}}$(term frquency) : la fréquence d'occurrence du terme **t** dans le document **d**.

Cette mesure est proportionnelle à la fréquence du terme dans le document. Plus un terme est fréquent dans le document, plus il est important dans la description de ce document.

$\boldsymbol{idf_t}$ (Inverse of document frequency) : la fréquence documentaire inverse du terme **t** c'est une mesure de l'importance d'un terme dans toute la collection. L'idée sous-jacente est que les termes qui apparaissent dans peu de documents de la collection sont plus

représentatifs du contenu de ces documents que ceux qui apparaissent dans tous les documents de la collection.

Donc le poids $w_{t,d}$ du terme **t** dans un document **d** est définit alors comme suit :

$$w_{t,d} = tf_{t,d} * idf_t \qquad \text{I.1}$$

I.5.2 L'appariement document-requête

Les SRI intègrent un processus de recherche/décision qui permet de sélectionner l'information jugée pertinente pour l'utilisateur, action possible grâce à la pondération qui est une fonction fondamentale en RI puisqu'elle permet d'affecter à chaque terme d'indexation une valeur qui mesure son importance dans le document où il apparait. Les poids des termes dans la requête et dans le document sont alors combinés lors de la recherche, dans un score de pertinence associé au document et permettant de l'ordonnancer dans l'ensemble final des résultats du SRI pour cette requête. Autrement dit, une mesure de similitude (correspondance) entre la requête indexée et les descripteurs des documents de la collection est calculée. Seuls les documents dont la similitude dépasse un seuil prédéfini sont sélectionnés par le SRI **[5].**

La fonction de correspondance est un élément clé d'un SRI, car la qualité des résultats dépend de l'aptitude du système à calculer une pertinence des documents le plus proche possible du jugement de pertinence de l'utilisateur, ce calcul est fait grâce une fonction de correspondance :

- **La fonction de correspondance :**

Tout système de recherche d'information s'appuie sur un modèle de recherche d'information. Ce modèle se base sur une fonction de correspondance qui met en relation les termes d'un document avec ceux d'une requête en établissant une relation d'égalité entre ces termes, cette dernière représente la base de la fonction de correspondance et, par la même, du système de recherche d'information. Cette fonction notée : ***RSV(d,q)*** (**R**etrieval**S**tatut **V**alue), où

d : un document de la collection

q : la requête.

Il existe deux types d'appariement **[5]** :

- **Appariement exact : (exact match retrieval) :**

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

- **Appariement approché : (best match retrieval) :**

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

Après avoir retourner les résultats, le système de recherche suggère à l'utilisateur de reformuler sa requête, et cela selon la satisfaction des besoins en information de l'utilisateur.

I.5.3 La reformulation de requête

Les utilisateurs des moteurs de recherche, ne sont pas des professionnels de la documentation. L'utilisateur ne sait pas toujours choisir les bons termes qui expriment ses besoins en information. En introduisant la reformulation de requête, la RI est alors envisagée comme une suite de formulations et de reformulations de requêtes jusqu'à la satisfaction du besoin en information d'un utilisateur. La requête initiale permettant rarement d'aboutir à un résultat qui satisfait ce dernier .Il s'agit en particulier d'ajouter des termes à la requête initiale de l'utilisateur et on parle alors d'expansion de la requête de l'utilisateur.

On distingue trois types de reformulation **[7]** :

- **La reformulation manuelle** : cette approche est associée aux systèmes de recherche booléens. On peut procéder à la reformulation de requête en utilisant un vocabulaire contrôlé (thesaurus ou classification) pour permettre à l'utilisateur de trouver les bons termes pour compléter sa requête.
- **La reformulation automatique** : lorsque le feedback de pertinence s'accompagne d'une adjonction (et/ou) suppression de termes, on parle de reformulation automatique. La requête de l'utilisateur est remaniée automatiquement, pour intégrer les descripteurs de documents jugés pertinents ou rejetés.

  On trouve différentes variantes de cette technique : celles qui sont utilisées automatiquement pour reformuler la requête en augmentant le poids des termes présents dans les documents jugés pertinents et inversement pour diminuer les poids des termes jugés non pertinents.

  Le problème avec la reformulation automatique est l'estimation des « bons » termes qui peuvent conduire effectivement à une amélioration du processus de recherche car l'introduction des termes inappropriés peut entrainer un silence ou un bruit.
- **La reformulation interactive** : Dans la reformulation interactive, l'utilisateur joue un rôle actif. A l'inverse de la reformulation automatique, ici, ce sont le système et l'utilisateur qui sont, ensemble, responsables de la détermination et du choix des termes candidats à la reformulation. Le système joue un grand rôle dans la suggestion

des termes, le calcul des poids des termes et l'affichage à l'écran de la liste ordonnée des termes. L'utilisateur examine cette liste et décide du choix des termes à ajouter dans la requête. C'est donc l'utilisateur qui prend la décision ultime dans la sélection des termes.

I.6 Les modèles de recherche d'information

Un modèle de RI a pour objectif d'identifier et d'ordonner les documents pertinents en rapport à un besoin en information de l'utilisateur exprimé avec une requête **[3].**

Plusieurs modèles de recherche d'information ont été proposés. Ils se déclinent en trois grandes catégories qui sont : Les modèles ensemblistes, les modèles algébriques et les modèles probabilistes, comme le montre la figure ci-dessus.

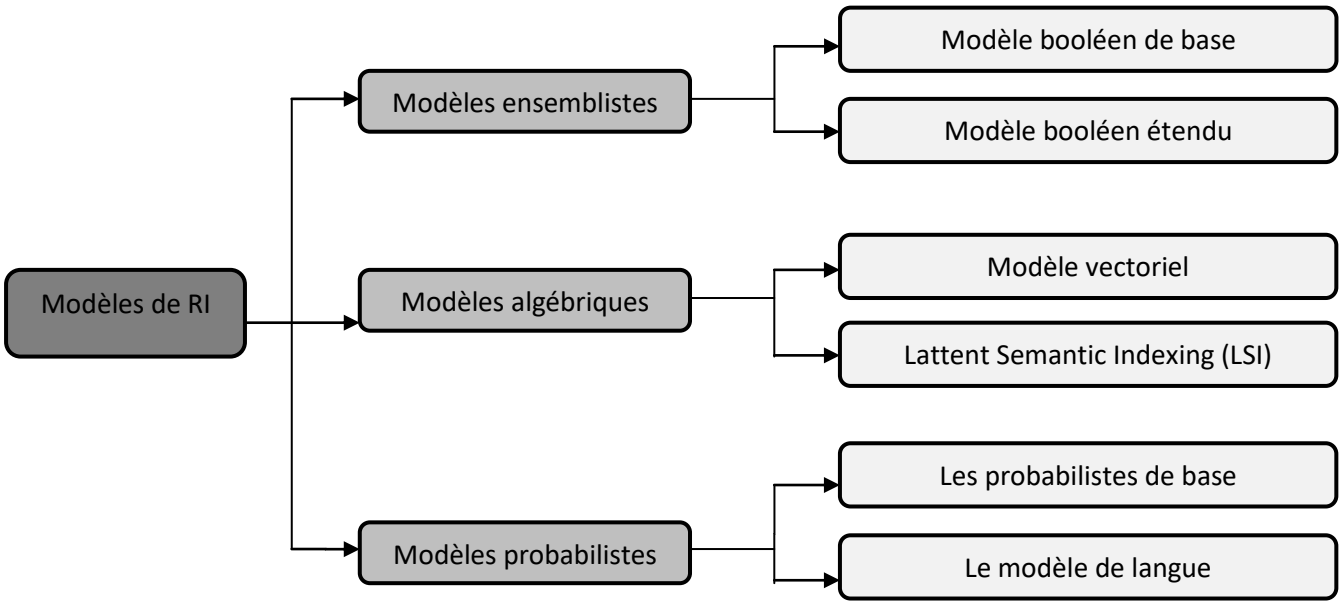


Figure I.3: Taxonomie des modèles de RI [3]

I.6.1 Les modèles ensemblistes

Ce sont les modèles basés sur la théorie des ensembles et l'algèbre de bool, dont le représentant le plus connu est le modèle booléen que nous présentons ci-dessous :

Le modèle booléen est le premier modèle utilisé dans le domaine de la RI basé sur la théorie des ensembles. Dans ce modèle, le document est représenté par un ensemble de termes. Par contre la requête est représentée par sous forme d'une expression logique formée de termes d'indexation reliés par des opérateurs booléens **AND**, **OR** et **NOT**. **[4].** L'appariement

requête-document est strict et se base sur des opérations ensemblistes selon les règles suivantes :

$$RSV(d, t_i) = \begin{cases} 1 & si\ t_i \in d \\ 0 & sinon \end{cases} \quad \text{I.2}$$

$$RSV(d, t_i\ AND\ t_j\ ) = \begin{cases} 1 & si\ t_i \in d\ et\ t_j\ \in d \\ 0 & sinon \end{cases} \quad \text{I.3}$$

$$RSV(d, t_i\ OR\ t_j\ ) = \begin{cases} 1 & si\ t_i \in d\ ou\ t_j\ \in d \\ 0 & sinon \end{cases} \quad \text{I.4}$$

$$RSV(d, NOT\ t_i\ ) = \begin{cases} 1 & si\ t_i \notin d \\ 0 & sinon \end{cases} \quad \text{I.5}$$

Avec :

q : Requête ;

d : Document ;

t_i : Terme ;

RSV (d,q) (**R**etrieval**S**tatut **V**alue) : fonction de correspondance entre document d et différentes forme de requêtes q.

La simplicité du modèle le rend plus compréhensible pour un utilisateur, et facile a mettre en œuvre, mais il présente un certain nombre de faiblesses :

- La sélection d'un document est basée sur une décision binaire
- Pas d'ordre pour les documents sélectionnés
- Formulation de la requête difficile et pas toujours évidente pour beaucoup d'utilisateurs
- Problème de collections volumineuses : le nombre de documents retournés peut être considérable

Et pour résoudre ces problèmes, des extensions ont été proposées parmi elles on trouve : le modèle booléen étendu **[1] [4].**

I.6.2 Les modèles algébriques

Ce sont les modèles basés sur la théorie d'algèbre, comme le modèle vectoriel.

Le modèle vectoriel est le plus populaire en RI. Un document $\mathbf{d_i}$ est représenté par un vecteur de poids $\boldsymbol{w_{ij}}$ de dimension **n**, dans l'espace vectoriel composé de tous les termes d'indexation

$$\boldsymbol{d_i = w_{i1}, w_{i2} \dots, w_{in}} \qquad \textbf{I.6}$$

Une requête **Q** est aussi représentée par un vecteur de poids $\boldsymbol{w_Q}$ défini dans le même espace vectoriel que le document.$\boldsymbol{Q = w_{Q1}, w_{Q2} \dots, w_{Qn}}$ **I.7**

Où $\boldsymbol{w_{Qj}}$ est le poids de terme $\mathbf{t_j}$ dans la requête **Q**, et $\boldsymbol{w_{ij}}$ son poids dans le document$\boldsymbol{d_i}$. Ce poids peut être soit une forme de $\boldsymbol{tf_{t,d} * idf_t}$, soit un poids attribué manuellement par l'utilisateur. **[1]**

La pertinence du document $\mathbf{d_i}$vis-à-vis de la requête **Q** est mesurée comme le degré de corrélation des vecteurs correspondants. Cette corrélation peut être exprimée par l'une des mesures suivantes:

- **Le produit scalaire :** $\boldsymbol{RSV(d_i, Q) = \sum_{j=1}^{n}(wQj * wij)}$ **I.8**
- **La mesure du cosinus :** $\boldsymbol{RSV(d_i, Q) = \frac{\sum_{j=1}^{n} w_{Qj} * w_{ij}}{(\sum_{j=1}^{n} {w_{Qj}}^2)^{\frac{1}{2}} * (\sum_{j=1}^{n} {w_{ij}}^2)^{\frac{1}{2}}}}$ **I.9**
- **La mesure de Dice :** $\boldsymbol{RSV(d_i, Q) = \frac{2 * \sum_{j=1}^{n} w_{ij} * w_{Qj}}{\sum_{j=1}^{n} {w_{Qj}}^2 + \sum_{j=1}^{n} {w_{ij}}^2}}$ **I.10**
- **La mesure de Jaccard** $\boldsymbol{RSV(d_i, Q) = \frac{\sum_{j=1}^{n} w_{ij} * w_{Qj}}{\sum_{1}^{n} {w_{Qj}}^2 + \sum_{1}^{n} {w_{ij}}^2 - \sum_{j=1}^{n} w_{ij} * w_{Qj}}}$ **I.11**
- **Coefficient de superposition :** $\boldsymbol{RSV(d_i, Q) = \frac{\sum_{j=1}^{n} w_{ij} * w_{Qj}}{\min(\sum_{j=1}^{n} {w_{Qj}}^2, \sum_{j=1}^{n} {w_{ij}}^2)}}$ **I.12**

L'un des avantages du modèle vectoriel réside dans sa simplicité conceptuelle et de mise en œuvre. En outre, 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. Cependant, ce modèle présente l'inconvénient de reposer sur l'hypothèse d'indépendance des termes (bag of words) alors que ce sont parfois les expressions ou les groupes de mots qui enrichissent la sémantique du document **[3]**.

I.6.3 Les modèles probabilistes

Ces modèles reposent sur la théorie des probabilités. Dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est comme une probabilité de pertinence document/requête
Le principe de base consiste à présenter les résultats d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête.

Etant donné une requête utilisateur notée **Q** et un document **d**, le modèle probabiliste tente d'estimer la probabilité que le document **d** appartienne à la classe des documents pertinents (ou non pertinents). Un document est sélectionné si la probabilité qu'il soit pertinent pour **Q**notée $\boldsymbol{P(R|d)}$ est supérieure à la probabilité qu'il soit non pertinent pour **Q** notée$\boldsymbol{P(NR|d)}$.

RSV (d,Q) est donné alors par :

$$\boldsymbol{RSV(d_i, Q) = \frac{P(R|d)}{P(NR|d)}} \qquad \textbf{I.13}$$

En utilisant la formule de Bayes et en simplifiant on obtient :

$$\boldsymbol{RSV(d_i, Q) = \frac{P(d|R)}{P(d|NR)}} \qquad \textbf{I.14}$$

$\boldsymbol{P(d|R)}$(Respectivement($\boldsymbol{P(d|NR)}$) est la probabilité que le document appartienne à l'ensemble **R** des documents pertinents (respectivement à l'ensemble **NR** des documents non pertinents).

Il existe des méthodes pour estimer ces probabilités comme dans le modèle **BIR**(**B**inary**I**ndependance**R**etrieval). Les probabilités $\boldsymbol{P(d|R)}$et $\boldsymbol{P(d|NR)}$sont données par :

$$\boldsymbol{P(d|R) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3 \ldots |R) = \prod_i p(t_i = x_i/R) = \prod_i p(t_i = 1/R)^{x_i} * p(t_i = 0/R)^{1-x_i}} \qquad \textbf{I.15}$$

$$\boldsymbol{P(d|NR) = P(t_1 = x_1, t_2 = x_2, t_3 = x_3 \ldots |NR = \prod_i p(t_i = x_i/NR) = \prod_i p(t_i = 1/NR)^{x_i} * p(t_i = 0/NR)^{1-x_i}} \qquad \textbf{I.16}$$

$\mathbf{t_i}$ est le ième terme utilisé pour décrire le document **d**, et $\mathbf{x_i}$ est sa valeur **0** si le terme est absent et **1** si le terme est présent dans le document **[1].**

1.6.3.1 Le modèle de langue

L'approche à base de modèles de langue(ou bien le modèle statistique de langue) pour la recherche de documents textuels a été proposée dès la fin des années 90 (Ponte et al. 1998) **[12]**.

Le principe de ce modèle consiste à construire un modèle de langue pour chaque document, soit $\mathbf{M_d}$, puis de calculer la probabilité qu'une requête **q** puisse être générée par le modèle de la langue du document, soit$\boldsymbol{P(q|M_d)}$**.**

Cette probabilité est exprimée comme suit :

$$\boldsymbol{RSV(d_i, q)= P(q = (t_1, t_2 \dots, t_n)/\mathbf{M_d})=\prod_i p(t_i/M_d)} \qquad \textbf{I.17}$$

$\boldsymbol{P(t|d)}$ est donné par :

$$\boldsymbol{P(t_i|M_d)=\frac{tf(t_i,d)}{\sum_t tf(t,d)}} \qquad \textbf{I.18}$$

Ou $\boldsymbol{tf(t_i, d)}$ est la fréquence d'occurrence du n-gramme (du mot) $\mathbf{t_i}$ dans le document **d** et $\boldsymbol{\sum_t tf(t, d)}$correspond à la taille du document (c'est-à-dire le nombre d'occurrence de n-grammes).

Pour remédier au problème posé pas des mots de la requête absents dans le document, qui ont pour effet d'avoir la probabilité $\boldsymbol{P(q|M_d)}$nulle ; des techniques de lissage (smothing) sont utilisées.

Le lissage consiste à assigner des probabilités non nulles aux termes, qui n'apparaissent pas dans les documents. Différentes techniques de lissage existent dont le lissage de Laplace, le lissage de Good-turing, le lissage Backoff, le lissage par interpolation **[1] [2] [4] [14].**

I.7 Evaluation des systèmes de recherche d'information

L'évaluation constitue une étape très importante dans la mise en œuvre d'un SRI, car elle permet de mesurer les caractéristiques du système en termes de qualité de service et de facilité d'utilisation, elle permet également de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et de fournir des éléments de comparaison entre modèles. Cleverdon en 1970 avait défini plusieurs mesures de la qualité d'un SRI, dont : 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 autrement dit la pertinence qui est évaluée grâce aux deux facteurs : le **taux de rappel du système** et la **précision du système**.

I.7.1 Les mesures d'évaluation d'un SRI

Il existe plusieurs mesures d'évaluation en RI. Nous présentons les plus importantes ci-dessous.

- **Rappel :**

Le rappel est le rapport de documents pertinents restitués par le système sur l'ensemble des documents pertinents contenus dans la base documentaire. Elle mesure la capacité du système de retrouver tous les documents pertinents répondant à la requête.

$$\boldsymbol{Rappel = \frac{nombre\ de\ documents\ pertinents\ trouvés}{nombre\ de\ documents\ pertinents}} \qquad \textbf{I.19}$$

- **Précision :**

La précision est le rapport de documents pertinents trouvés sur l'ensemble des documents restitués par le système. Elle mesure la capacité du système à rejeter tous les documents non pertinents à une requête donnée.

$$\boldsymbol{Précision = \frac{nombre\ de\ documents\ pertinents\ trouvés}{nombre\ de\ documents\ trouvés}} \qquad \textbf{I.20}$$

Idéalement, on souhaiterait qu'un système donne de bons taux de précision et de rappel en même temps. Un système qui aurait 100% pour la précision et pour le rappel signifie qu'il trouve tous les documents pertinents, et rien que les documents pertinents. En pratique, cette situation n'arrive pas. Plus souvent, on peut obtenir un taux de précision et de rappel aux alentours de 30% **[6].**

Les deux métriques ne sont pas indépendantes. Il y a une forte relation entre elles : quand l'une augmente, l'autre diminue. Il ne signifie rien de parler de la qualité d'un système en utilisant seulement une métrique. En effet, il est facile d'avoir 100% de rappel : il suffirait de donner toute la base comme une réponse à chaque requête. Cependant, la précision dans ce cas-ci serait très basse. De même, on peut augmenter la précision en donnant très peu de

documents en réponse, mais le rappel souffrira. Il faut donc utiliser les deux métriques ensemble pour évaluer un SRI.

Les mesures de précision-rappel ne sont pas statiques non plus (c'est-à-dire qu'un système n'a pas qu'une mesure de précision et de rappel). Le comportement d'un système peut varier en faveur de précision ou en faveur de rappel(en détriment de l'autre métrique).

Nous considérons par exemple une requête pour laquelle il existe six (6) documents pertinents dans le corpus. Le Tableau **I.1** illustre le calcul de la précision et de rappel pour les dix (10) premiers documents retournés par un SRI. La lettre (P) précise que le document est pertinent.

Rang du document renvoyé	Pertinence	Rappel	Précision
Document 1	**P**	**0.14**	**1**
Document 2		**0.14**	**0.62**
Document 3	**P**	**0.35**	**0.66**
Document 4	**P**	**0.42**	**0.75**
Document 5		**0.5**	**0.6**
Document 6	**P**	**0.55**	**0.6**
Document 7		**0.57**	**0.57**
Document 8		**0.57**	**0.5**
Document 9	**P**	**0.7**	**0.55**
Document 10	**P**	**0.85**	**0.57**

Tableau I.1 : Exemple de calcul de rappel et de précision pour une requête.

La figure **I.4** suivante illustre la courbe de rappel et précision correspondante aux résultats du **tableau I.1**. Pour rendre la courbe lisible on ne garde que la précision calculée à chaque point de rappel (c'est à dire à chaque document pertinent restitué).

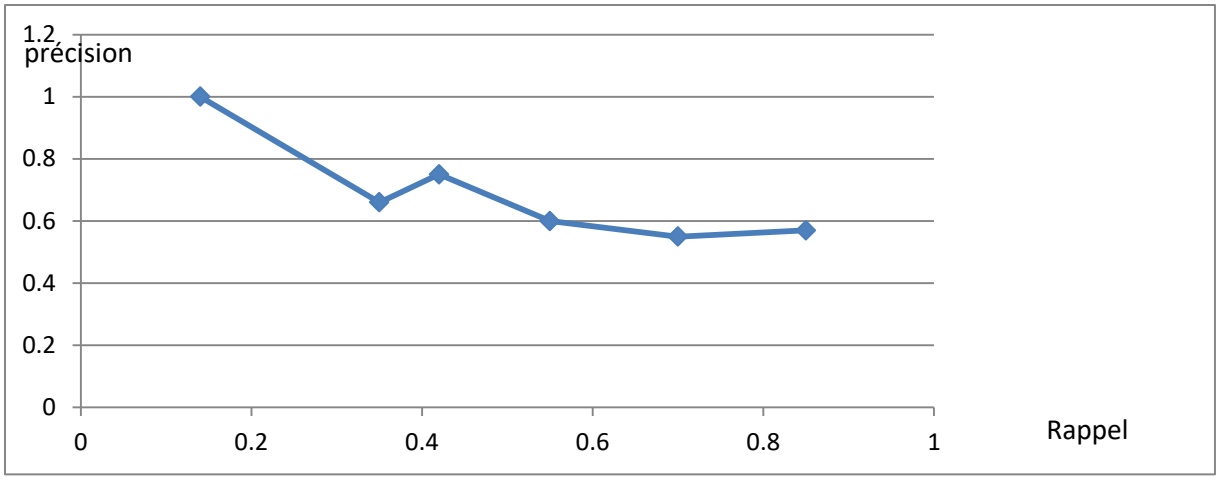


Figure I.4 Courbe de rappel et précision

Le rappel et la précision permettent aussi de définir **le silence documentaire** et **le bruit documentaire** qui représentent respectivement le nombre des documents pertinents non retrouvés et le nombre de documents non pertinents retrouvés.

- **Les mesures alternatives :** plusieurs mesures alternatives ont été proposées, parmi elles nous trouvons :

  a. **R-précision** : elle mesure la précision ou le rappel après que R document ont été restitués pour la requête. R représente le nombre total de documents pertinents pour la requête. Cette mesure compense les limites des mesures de haute précision quand la précision est calculée pour X documents et que le nombre total de document |P| est inferieur à X. Si la valeur e R est plus grande que le nombre total de documents retrouvés, tous les documents non retrouvés sont alors considérés comme non pertinents **[13].**

  b. **Précision moyenne (MeanAveragePrecision-MAP**) :

  Cette mesure calcule la moyenne des valeurs de précision moyenne non interpolées sur l'ensemble des documents pertinents. La formule suivante donne la méthode de calcul de la MAP :

$$MAP = \frac{\sum_{q \in Q} APq}{|Q|} \qquad \text{I.21}$$

Avec APqest la précision moyenne d'une requête **q**, **Q** est l'ensemble des requêtes et $|Q|$est le nombre de requête.

I.7.2 Les collections de tests

Une collection (ou un corpus) de test constitue un moyen d'évaluation des SRI. Elle comprend un ensemble de documents à indexer sur le système qui sera évaluée, une liste de requêtes prédéfinies ainsi que les jugements de pertinence, manuellement établis pour chaque requête (liste de documents jugés pertinents pour cette requête). L'évaluation d'un SRI consiste à comparer les résultats retournés par ce dernier par rapport aux jugements de pertinence. Les collections de test sont les résultats de projets d'évaluation qui se sont multipliés depuis les années 1970.

On peut citer quelques collections de test, parmi elles **[1]** :

- ***adi*** , 82 résumés d'articles présentés à une rencontre de l'American Documentation Institute en 1963, domaine des sciences de l'information.
- ***cacm*** , 3204 documents avec le titre et les liens de citations bibliographiques, domaine de l'informatique.
- ***cisi*** , domaine des sciences de l'information.
- ***Méline*** , domaine médical.
- ***time***, articles du magazine Time.

Différentes collections de test sont utilisées en recherche d'information. Parmi elles TREC et CLEF, dans ce qui suit nous détaillerons la collection TREC :

I.7.2.1 Les collections TREC

Le projet TREC est un programme qui financé par la DARPA (Defense Advanced ResearchProjects Agency) et le NIST(National Institute of Standards and Technology). La première compagne de TREC voit le jour en 1992 avec 25 participants issus du monde académique.

Ce programme met à la disposition des participants un ensemble de documents er de requêtes. Pour chaque requête, l'ensemble des documents pertinents est déterminé par de juges humains.

Dans ce qui suit, nous allons définir les différents éléments qui constituent le projet TREC :

- **Tâches** : l'objectif est de permettre l'évaluation d'approches spécifiques en recherche d'information concentrant le filtrage, le croisement de langues et la recherche.
- **Les participants** : 25 groupes ont participé à la première édition de TREC et 66 groupes à TREC8.

- **Source d'informations** : les documents de la collection sont issus de la presse écrite en 1999.
- **Structure et principe de construction de la** collection : un document TREC est identifié par un numéro et décrit par un auteur, une date de production et un contenu textuel. Une requête est également identifiée par un numéro et décrite par un sujet générique **[13].**

I.8 Conclusion

Dans ce chapitre, nous avons commencé par définir la notion de la recherche d'information et le système de recherche d'information, puis nous avons présenté les notions de bases de la RI dont : la requête, le document et collection de document et la pertinence. Nous avons également décrit les différentes étapes de la RI : l'indexation, l'appariement et la reformulation de la requête. Par la suite, nous avons cité quelques modèles de la RI. Enfin nous avons traité de l'évaluation des systèmes de recherche d'information. Le chapitre suivant est consacré à la présentation de l'expansion de requête et le Word Embedding.

CHAPITRE II

EXPANSION DE REQUETE ET LE WORD EMBEDDING

II.1. Introduction

Dans ce chapitre nous présentons l'expansion de requête et le Word Embedding. Dans un premier temps nous allons définir l'expansion de requête et ses étapes, puis nous allons présenter le concept de Word Embedding qui est basé sur les réseaux de neurones. Enfin nous citons quelques travaux qui utilisent le Word Embedding en recherche d'information et particulièrement au niveau de l'expansion de requête.

II.2. Expansion de requêtes

II.2.1.Définition

L'expansion de la requête est la procédure qui modifie une requête par l'ajout de nouveaux termes afin d'améliorer l'efficacité de la recherche d'information **[15].**

II.2.2.Classification des approches de l'expansion de requête [55]

Nous pouvons classifier les approches d'expansion de requêtes selon les sources de données utilisées en trois catégories: approches statistiques, approche à partir de logs et approches linguistiques. **[16]**

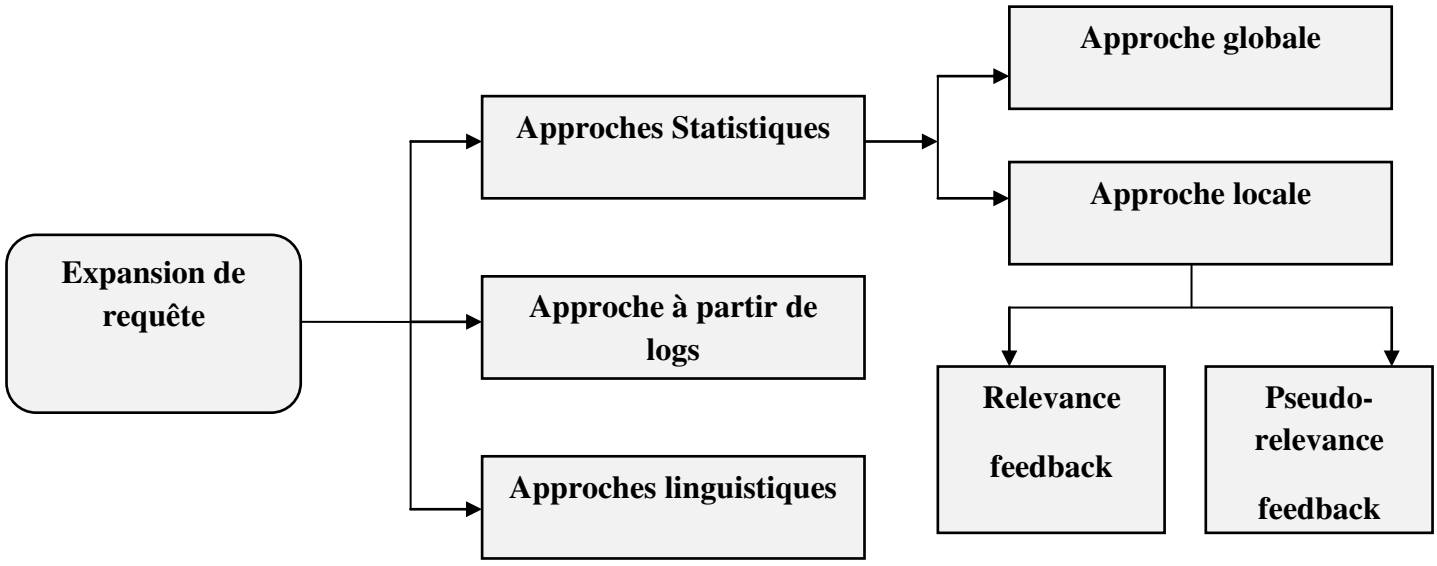


Figure II.1 : Taxonomie des approches d'expansion de requêtes [55]

II.2.2.1 Approches statistiques [55]

Cette approche est une corrélation entre les mots basée sur leurs co-occurrences, elle se décompose en deux catégories :

a. Approches locale

L'approche locale inclut des techniques d'expansion de requête qui sélectionnent les termes d'expansion à partir de l'ensemble documents récupérés en réponse à la requête initiale. Il existe deux façons d'étendre la requête initiale de l'utilisateur en utilisant cette approche : Relevance feedback et Pseudo-relevance feedback.

- **Relevance feedback**

L'utilisateur fait une première requête, et indique parmi les documents retournés lesquels sont les plus pertinents; à partir de ceux-ci des mots clés sont extraits afin de mener une deuxième requête "la requête étendue".

- **Pseudo- relevance feedback**

Le principe reste le même avec l'approche relevance feedback, mais la pertinence est basée sur le classement du moteur de recherche en utilisant directement les documents les mieux classés, récupérés en réponse à la requête initiale (d'où le terme "aveugle").

b. Approches globale

Dans l'approche globale, les techniques d'expansion sélectionnent les termes d'expansion de requête à partir de l'ensemble des documents indexés pour reformuler et développer la requête initiale.

II.2.2.2 Approches à partir de logs

Cette approche utilise plusieurs méthodes:

- Calcul de similarité entre la requête courante et les anciennes requêtes, puis approche statistique
- Association entre les requêtes et les documents retournés.
- Considérer les requêtes précédentes comme des documents, puis techniques statistiques de relevance feedback ou d'expansion aveugle

II.2.2.3 Approches linguistiques [16]

Les approches de cette catégorie analysent les caractéristiques d'expansion de requête à partir des ressources sémantiques (WordNet, ConceptNet) et lexico-syntaxiques (lemmes) afin de

reformuler et développer la requête initiale. Pour se faire, Cette technique est elle-même devisée en sous-types : Word stemming, analyse sémantique et l'analyse syntaxique.

- Le Word stemming est l'une des premières approches les plus connues d'expansion de requête basée sur la linguistique. Cet algorithme peut être utilisé soit au moment de l'extraction, ou de l'indexation.
- L'analyse sémantique et contextuelle est une autre méthode d'expansion de requêtes liée à la sémantique. Il comprend des sources de données telles que les ontologies, le Cloud, les dictionnaires et les thésaurus.
- L'analyse syntaxique est la troisième approche qui fournit des informations linguistiques supplémentaires à la requête initiale. L'objectif est d'extraire des relations entre les termes de la requête, qui peuvent ensuite être utilisés pour identifier les termes d'expansion qui apparaissent dans les relations associées**.**

II.2.3.Les étapes de l'expansion de requête

Le processus de génération d'extension de requête comprend principalement quatre étapes illustrées dans la **figureII.2** qui sont : le prétraitement de la source de données, pondération et classement des termes, sélection des termes et reformulation des requêtes [16] [17]. Chaque étape est abordée dans les sections suivantes.

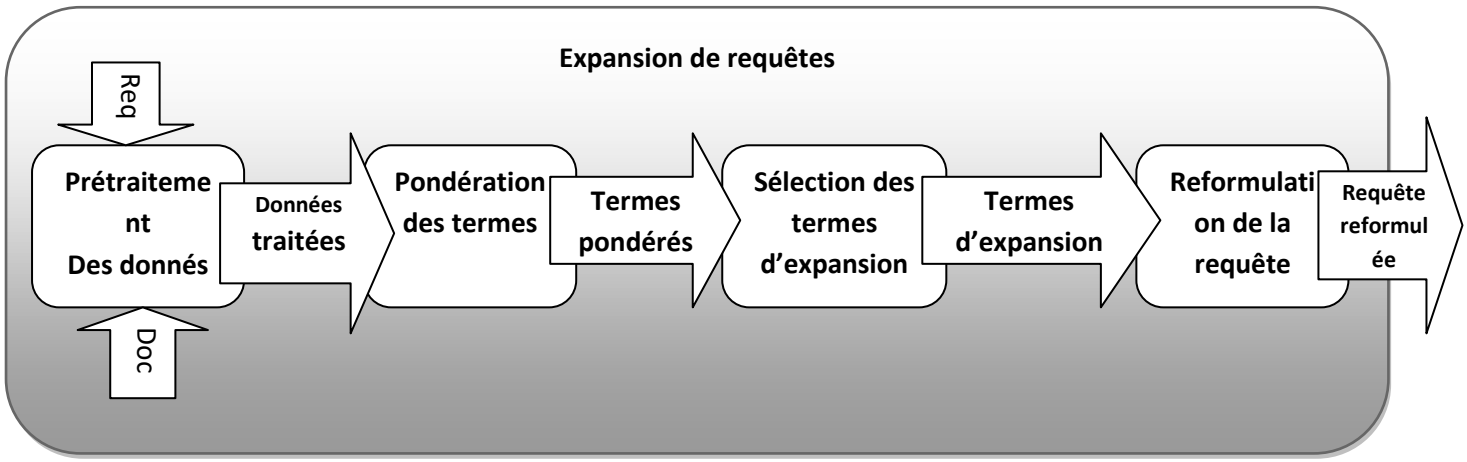


Figure II.2:Les étapes de l'expansion automatique des requêtes [16].

II.2.3.1.Prétraitement de la source de données

Cette étape dépend des sources de données et des approches utilisées pour l'expansion de la requête, au lieu de la requête de l'utilisateur. L'objectif principal est d'extraire un ensemble de termes provenant de sources de données qui augmentent de manière significative la requête initiale de l'utilisateur. Il se compose des cinq sous-étapes suivantes:

- Extraction de texte à partir de sources de données spécifiques utilisées pour l'expansion de la requête (des documents tels que HTML et PDF).
- Tokenization (extraction de mots individuels, en ignorant la ponctuation et la casse).
- suppression des mots fréquemment utilisés, par exemple : articles, adjectif, prépositions
- L'origine des mots (réduction des mots dérivés en leur forme racine.

Un grand nombre de sources de données ont été utilisées pour l'expansion de requête dans la littérature (citées dans la section précédente II.2.2) Toutes ces sources peuvent être classées

II.2.3.2.Pondération et classement des termes d'extension de requêtes

Dans la deuxième étape de l'approche d'expansion de la requête, le système classe les termes d'extension par l'attribution des poids et des rangs, ce classement est très important car la plupart des méthodes d'extension de requête ne choisiront qu'une petite proportion des termes d'expansion possibles à ajouter à la requête initiale, pour se faire , le système utilise la requête de l'utilisateur ainsi que les données traitées dans la première étape.

Les techniques utilisées pour réaliser la pondération et le classement des termes d'expansion générées et les termes sont classées en quatre catégories **[16]** :

- **One-to-One Association**

C'est l'approche de base et la plus simple de génération et de classement des termes d'expansion en fonction de la relation un-à-un, qui est basé sur des associations individuelles entre les termes d'extension et les termes de la requête, c'est-à-dire chaque terme d'expansion est lié a un terme de requête unique, et des poids sont attribués pour chaque terme de requête en utilisant plusieurs techniques. L'une des techniques les plus importante est de s'appuyer sur des associations linguistiques, telles que l'utilisation d'un algorithme d'extraction (tel que l'algorithme de Porter) pour réduire les différentes formes des mots (formes plurielles, temps de verbes ou formes dérivées) à la même racine.

Une autre approche linguistique typique est l'utilisation du thésaurus. L'un des plus utilisé des dictionnaires des synonymes est WordNet pour trouver des synonymes et des termes similaires pour les termes de la requête.

- **One-to-Many Associations**

L'association un-à-un s'applique à ajouter un terme quand il est fortement lié à l'un des termes de la requête. Cependant, cela peut ne pas refléter avec précision les relations du terme d'expansion à la requête dans son ensemble. Ce problème a été analysé par Bai et al. [2007]

C'est pour cela une approche simple est proposée qui est l'approche one-to-many (association un-à-plusieurs) pour enrichir l'association un-à-un.

Dans cette approche plusieurs termes de requête peuvent être développés ensemble comme une unité, l'idée donc est que si un terme d'extension est corrélé à plusieurs termes de requête, il est alors corrélé à la requête dans son ensemble.

- **Distribution des fonctions des documents les mieux classés**

Les techniques décrites dans cette section sont entièrement distincts des approches décrites dans les sections précédentes, car elles n'essaient pas de rechercher des caractéristiques directement associées aux termes de la requête, qu'elles soient simples ou multiples.

L'idée est d'utiliser les premiers documents récupérés en réponse à la requête initiale en tant que description plus détaillée du sujet de la requête, à partir duquel on peut extraire les termes les plus importants à utiliser comme termes d'expansion. En un sens, ces derniers sont liés à la signification complète de la requête car les termes extraits sont ceux qui caractérisent le mieux les documents pseudo-pertinents dans leur ensemble, mais leur association avec les termes de la requête n'est pas analysée explicitement.

De telles approches d'expansion de requêtes présentent de meilleurs résultats par rapport aux approches citées auparavant. Elles peuvent être subdivisées en deux catégories:

- Extension de la requête via le retour de pertinence (Relevance feedback). Les termes d'expansion de la requête sont extraits des documents récupérés en réponse à la requête initiale et l'utilisateur décide de la pertinence des résultats.
- Extension de la requête par Pseudo-relevance feedback. Les termes d'extension de la requête sont extraits des documents les mieux classés en réponse à la requête initiale.

II.2.3.3. Sélection des termes d'extension

Dans cette étape, seul un nombre limité de termes sont sélectionnés pour l'expansion, parce que la requête résultante peut être traitée plus rapidement, vu que l'efficacité de la recherche d'un petit ensemble de termes valables n'est pas nécessairement moins efficace que l'ajout de tous les termes [16].

Quelques travaux de recherche ont été menés déterminer le nombre optimal de termes d'extension à inclure dans la requête initiale. Toutefois, cet optimum suggéré peut varier de cinq à dix termes à quelques centaines de termes [16]. Par contre, d'autres travaux montrent que le nombre de termes utilisés pour l'expansion des requêtes est moins important que la qualité des termes choisis. Il a été généralement démontré que l'efficacité de l'extension de la requête diminue de manière minime avec le nombre non optimal de termes d'expansion [16].

La plupart des études expérimentales s'accordent pour dire que le nombre de termes d'expansion est peu pertinent et qu'il varie d'une requête à une autre. Il a été observé que l'efficacité du développement des requêtes (mesurée en tant que précision moyenne) diminue lorsque nous considérons moins de 20 termes de développement donc 20 à 40 termes constituent le meilleur choix pour l'extension de la requête. Lorsque les scores des caractéristiques peuvent être interprétés comme des probabilités, on ne peut sélectionner que les termes ayant une probabilité supérieure à un certain seuil, par exemple : p = 0,001 comme dans Zhai et Lafferty **[16].**

II.2.3.4.La reformulation de la requête

La dernière étape de l'approche d'expansion de la requête est la reformulation de la requête, où la requête est reformulée pour obtenir la requête étendue qui sera soumise au système de la recherche d'informations. Cette approche est effectuée sur la base des pondérations attribuées aux termes de la requête développée appelée repondération de la requête.

La technique de repondération de requête la plus populaire est celle basée sur la formule de Rocchio **[16].**

Cette formule est décrite comme suit :

$$\boldsymbol{w'_{t,q'} = (1 - \lambda) \, . \, w_{t,q} + \lambda \, . \, \mathbf{score_t}} \qquad \textbf{(II.1)}$$

Où $\mathbf{q'}$ est la requête étendue, $\mathbf{q}$ est la requête initiale, $\boldsymbol{\lambda}$ est un paramètre permettant de pondérer la contribution relative des termes de la requête initiale et des termes d'expansion,

$\mathbf{score_t}$ est une pondération attribuée au terme d'expansion **t**, $\mathbf{w'_{t,q'}}$est la nouvelle pondération du terme **t** de la requête étendue $\mathbf{q'}$ et $\mathbf{w_{t,q}}$est une pondération attribuée au terme d'expansion **t** de la requête initiale **q**.

Lorsque les termes d'extension sont extraits des documents pseudo-pertinents et que leur score est calculé à l'aide des documents ou pondérations de Rocchio, on peut observer que le vecteur de requête étendue mesuré par l'équation précédente est pertinent pour les pseudo-documents pertinents. Cela réduit la différence de distribution des termes entre les documents de pseudo-pertinence et les documents ayant des termes d'expansion lorsque les termes sont pondérés à nouveau par le schéma de pondération de Rochhio.

La valeur de **λ** dans **la formule (II.1)** peut être ajustée de manière appropriée pour améliorer l'efficacité de l'extraction. Un choix par défaut typique est de donner plus d'importance aux termes de la requête initiale.

II.3 Le Word Embedding

II.3.1 Définition :

Le Word Embedding (en français : incorporation des mots= plongement des mots), fut utilisé pour la première fois en 2003 par Bengio et al. appelé aussi modèles sémantiques distribués où représentation distribuée est une méthode d'apprentissage automatique qui vise à mapper des mots à partir d'un vocabulaire en vecteurs de nombres réels dans un espace de faible dimension. En tirant parti des grands corpus de texte, de telles représentations d'espace continues peuvent être calculées pour capturer l'information syntaxique et sémantique sur les mots. Ces Words Embedding, lorsqu'ils sont ensuite utilisés comme données d'entrées, se sont révélés être un grand atout pour une grande variété de tâches en traitement automatique du langage naturel (TALN) et de recherche d'information **[20] [19] .**

Le Word Embedding utilise les modèles de représentation des mots comme Glove et Word2vec qui se base sur les réseaux de neurones. La figure suivante représente un exemple des Words Embedding obtenus à partir du modèle Word2vec.

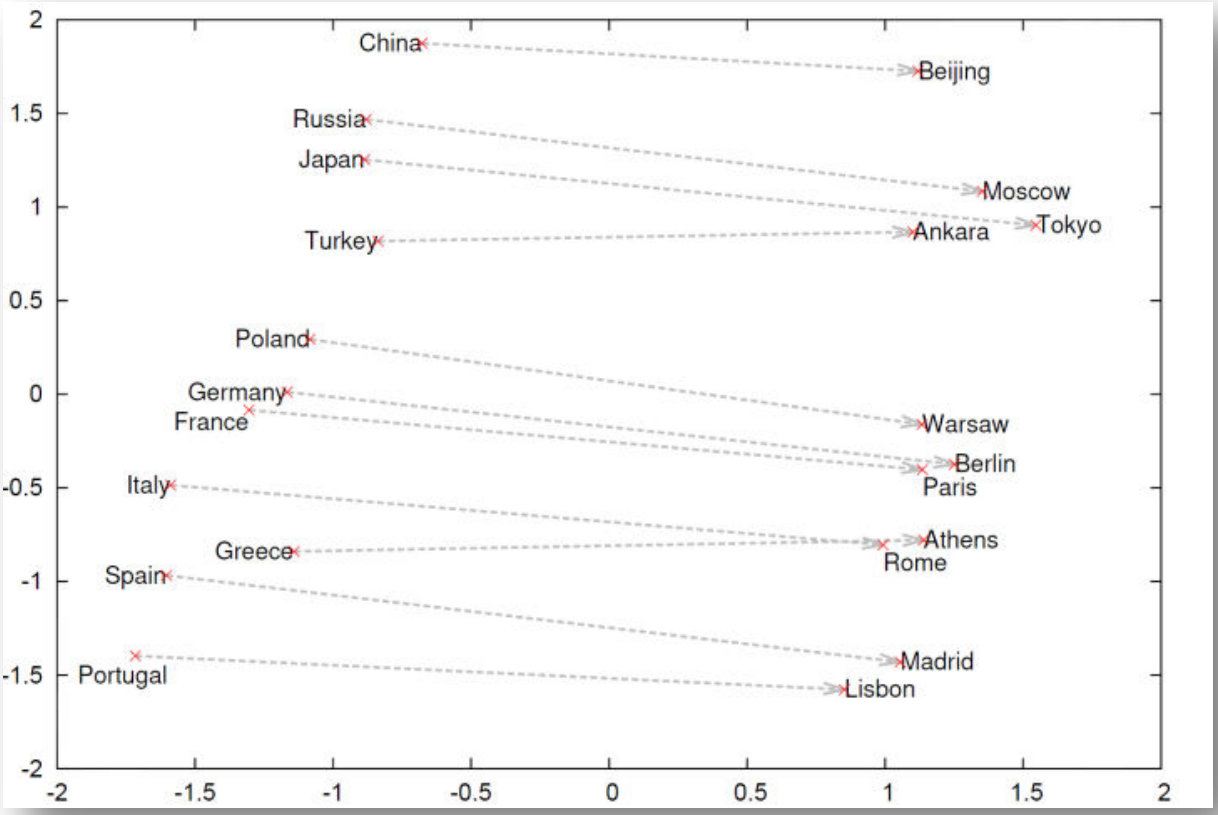


Figure II.3 : Word Embedding obtenu avec le modèle word2vec [21]

Ces représentations semblent exhiber même d'autres propriétés intéressantes : les analogies entre les mots semblent également encodées dans la différence entre leurs vecteurs. Par exemple il semble qu'il y ait une différence constante male-femelle :

W (woman)-W (man)≅W (queen)-W (king)

W (woman)-W (man)≅W (princess)-W (prince)

Tel que W(t) est le vecteur du terme t.

Ceci est montré par la figure suivante

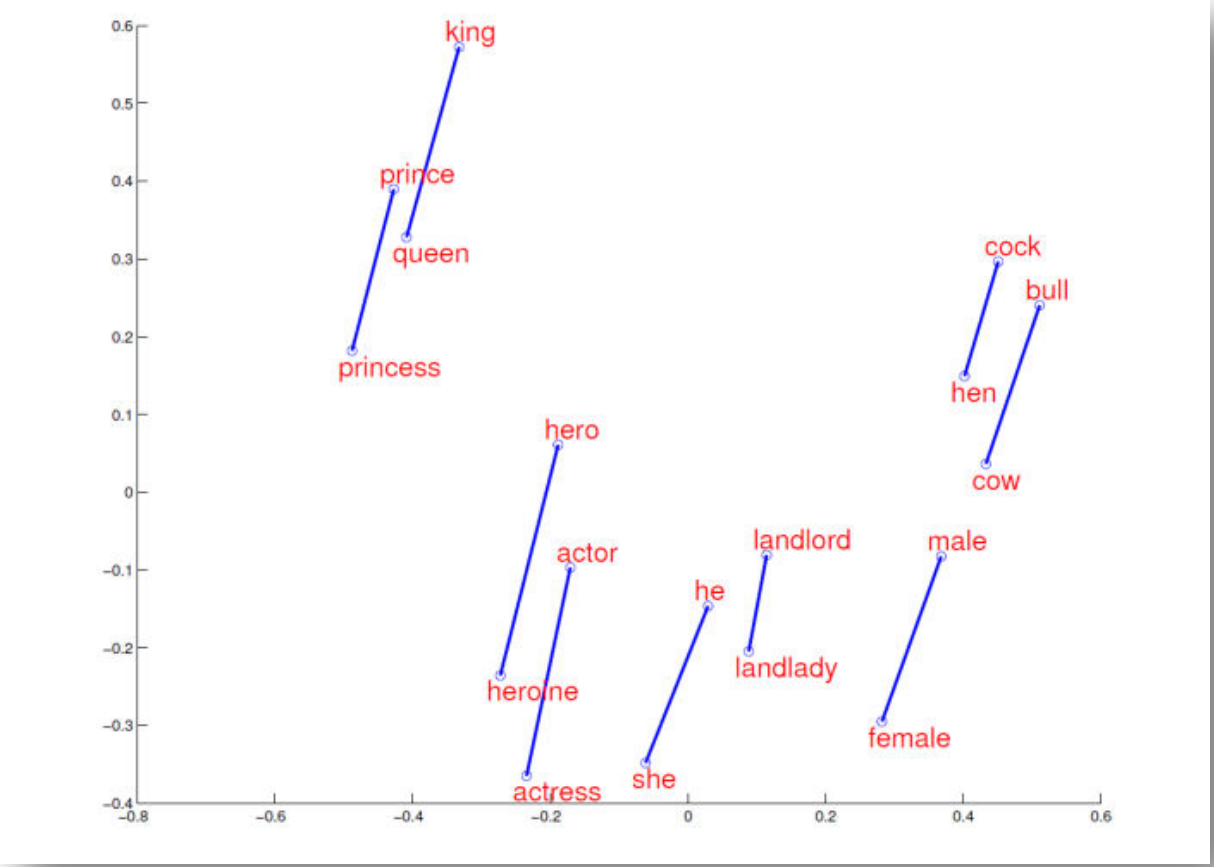


Figure II.4 : Différence constante mal-femelle. [22]

L'architecture des modèles du Word Embedding se base sur des réseaux de neurones, ce concept est détaillé dans la section suivante.

II.3.2 Les réseaux de neurones

Les réseaux de neurones (réseaux neuromimétiques) artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau. **[23]**

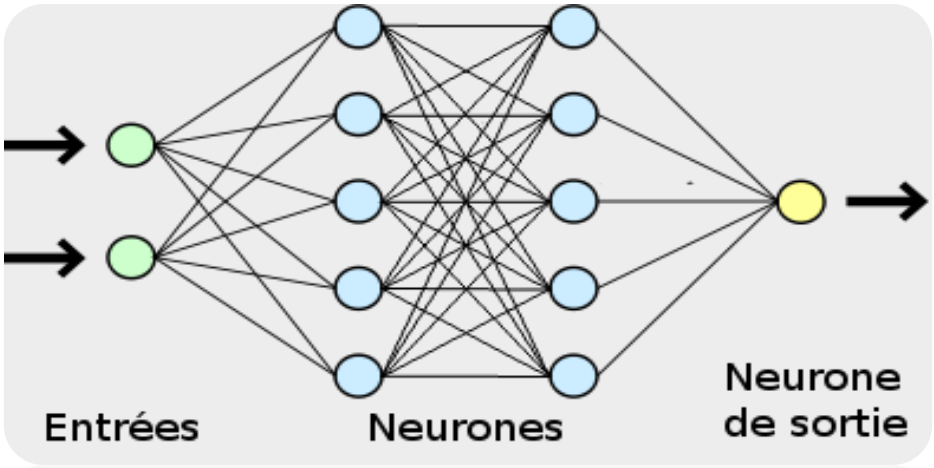


Figure II.5 : Schéma générale d'un réseau de neurones

Pour modéliser l'architecteur d'un neurone artificiel (formel), les chercheurs se sont inspirés de celle d'un neurone biologique.

Un neurone formel est donc une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique. Un modèle et non une copie du neurone biologique [**24].** La figure suivante montre l'analogie entre un neurone artificiel et un neurone biologique.

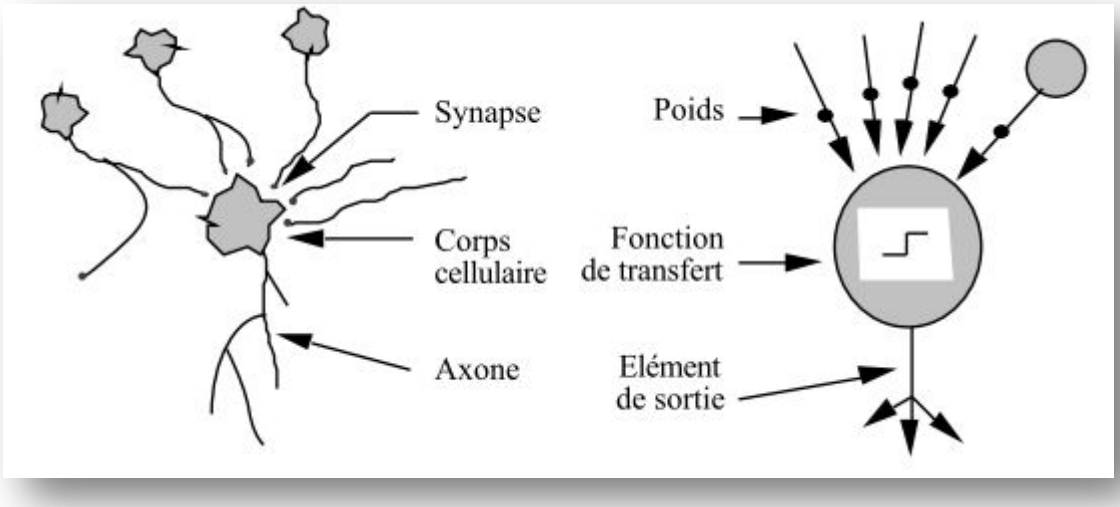


Figure II.6 : **Analogie entre un neurone biologique et un neurone artificiel [23]**

Chaque neurone artificiel est un processeur élémentaire, il reçoit un nombre variable d'entrées en provenance de neurones amonts. A chacune de ces entrées est associé un poids

w abréviation représentatif de la force de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals **[23].**

Afin qu'un réseau de neurone remplisse au mieux la tâche qui lui affectée, il doit passer par la procédure d'apprentissage qui est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré**,** il consiste à déterminer les valeurs des poids permettant à la sortie du réseau de neurone d'être aussi proche que possible de l'objectif fixé.

Deux grandes classes principales de l'apprentissage sont définies : « supervisé » et « non supervisé » **[25] :**

- **Apprentissage supervisé** : est une méthode qui utilise une fonction qui, à partir d'un échantillon de données et des résultats souhaités, se rapproche le mieux de la relation entre entrée et sortie observable dans les données.
- **Apprentissage non supervisé :** est une méthode qui n'a pas de résultats étiquetés. Son objectif est donc de déduire la structure naturelle présente dans un ensemble de données**.**

L'architecture d'un réseau de neurone est définie par : un ensemble d'entrées et de sorties, les neurones qui sont connectés les uns aux autres et qui sont capables d'échanger des informations au moyen des connexions qui les relient **[26].**

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle parmi elles on trouve :

- **Réseau de neurone monocouche :** c'est le type de réseaux neuronal le plus simple appelé aussi le Perceptron, il se décompose de deux couche de neurones : la première composé de cellules d'entrés, la deuxième couche fournit la réponse comme si il dispose d'un unique neurone de sortie [26]. La figure suivante illustre ce type de modèle.

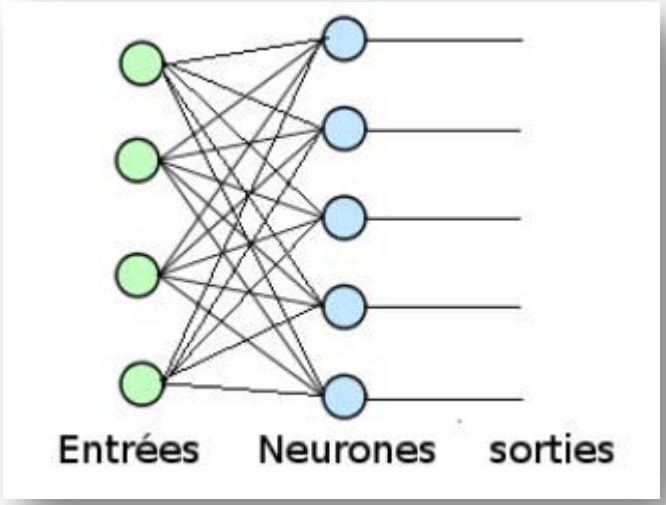


Figure II.7 : Réseau de neurone monocouche [26]

- **Réseau de neurone multicouche :** Dans ce réseau les neurones sont arrangés par couche tel que les neurones de la première couche reçoivent toutes les informations entrées puis ceux de la deuxième reçoivent toutes les sorties des neurones de la première couche, et ainsi de suite jusqu'au neurone de sortie qui reçoit celles de la dernière couche et produit un résultat [26] [23]. Ce modèle est illustré par la figure suivante :

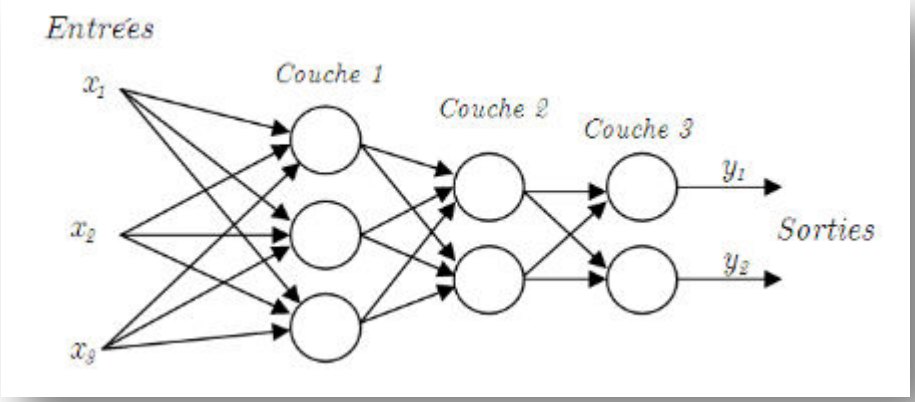


Figure II.8 : Réseau de neurone multicouche [26]

II.3.3: Les modèles du Word Embedding

Il existe de nombreux modèles de Word Embedding, chaque modèle a ses propres méthodes et ils sont liés avec les modèles de langue. Parmi ces modèles nous distinguons : le modèle Word2vec, C & W et Glover.

II.3.3.1 La modélisation linguistique

Les modèles des Words Embedding sont fortement liés avec les modèles de langue, ces derniers évaluent une fonction de probabilité () **P** d'un mot ou une séquence de mots dans une source de donnée, c'est-à-dire

$$\mathbf{P(s) = P(w_1, \dots\dots, w_k)} \qquad \textbf{II.2}$$

Nous pouvons calculer le produit d'une phrase ou d'un document en fonction des probabilités de chaque mot étant donné ses n mots précédents :

$$\boldsymbol{P(s) = \prod_{i=1}^{k} P(w_i | w_{i-1+n}, \dots\dots, w_{i-1})} \qquad \textbf{II.3}$$

Dans cette formulation le terme $\boldsymbol{w_i}$ ne dépend que de ses **n-1** prédécesseurs immédiats, c'est une simplification qui est faite pour réduire le nombre de paramètres à estimer, dans ce cas le modèle est dit modèle **n-gramme.**

Quand n=2 le modèle de langue est dit **bi-gramme,** tandis que n=3 conduits à des modèles de **tri-gramme** qui se sont révélés être une base de référence forte pour la modélisation de langue. **[22][2].**

II.3.3.2 Le modèle classique de langage neural

Le modèle classique de langage neuronal fut proposé par Bengion et al. en 2003(A neuronal probalistic Language model) **[18][19].** L'idée générale de ce modèle est de représenter chaque mot par un vecteur, ces derniers sont des paramètres qui seront appris en même temps que le reste du réseau **[42].** En appliquant la fonction softmax à la sortie de la couche du réseau avec des paramètres Θ, la distribution conditionnelle correspondant au contexte ct, $P(w|c_t)$, est défini comme :

$$\mathbf{P_\Theta(w_t|c_t)} = \frac{\exp(\boldsymbol{\Phi}_\Theta(\boldsymbol{w_t}, \boldsymbol{c_t}))}{\sum_{i=1}^{|w|} \exp(\boldsymbol{\Phi}_\Theta(\boldsymbol{w_i}, \boldsymbol{c_t}))} = \frac{\exp(\boldsymbol{\Phi}_\Theta(\boldsymbol{w_t}, \boldsymbol{c_t}))}{\boldsymbol{Z}_\Theta(\boldsymbol{c_t})}, \boldsymbol{\Phi}_\Theta \in \boldsymbol{R}^{|w|} \qquad \textbf{II.4}$$

Où $\boldsymbol{\Phi_\theta(w_t, c_t)}$est le score de sortie qui quantifie la compatibilité du mot w_t avec le contexte$\boldsymbol{tc_t}$ et $\boldsymbol{z_\theta(c_t)}$est la fonction de partitionnement qui le normalise en une distribution de probabilité.

L'architecture du modèle de langue classique est Composée de **[27]:**

- **Couche Embedding (couche d'intégration) :** une couche qui génère des Word Embedding en multipliant un vecteur d'index par une matrice du Word Embedding.
- **Couches intermédiaires :** une ou plusieurs couches qui produisent une représentation intermédiaire de l'entrée.
- **Couche softmax :** la couche finale qui produit une distribution conditionnelle sur les mots.

II.3.3.3 Le modèle C& W (Collobert et Weston)

Après les premiers pas de Bengio et al **[18] [19]** dans les modèles de langage neuronal, la recherche sur l'intégration de mots stagnait sous la contrainte de la puissance de calcul et d'algorithmes qui ne permettait pas encore la formation d'un vocabulaire étendu. Pour cela, Collobert et Weston (ou C & W) **[26]** ont démontré en 2008 que les Words Embeddding sont formés sur un ensemble de données suffisamment volumineux qui comportent une signification syntaxique et sémantique **[26]**

Leur solution pour éviter de calculer le softmax coûteux qui consiste à utiliser une fonction différente que celle de Bengio et al, **[27].** Collobert et Weston forment un réseau qui produit à la sortie un score plus élevé f_θ pour une séquence de mots correcte (une séquence de mots probable dans le modèle de Bengio) que pour une séquence incorrecte **[27].**

A cet effet, ils illustrent un critère de classement par paire, qui ressemble à ceci :

$$j_\theta = \sum_{x \in X} \sum_{w \in V} \max\{0, 1 - f_\theta(x) + f_\theta(x^{(w)})\} \qquad \text{II.5}$$

Ce modèle consiste à échantillonner les fenêtres **x** contenant **n** mots de l'ensemble des fenêtres **X** possibles dans leur corpus. Pour chaque fenêtre **x**, on produit ensuite une version incorrecte$\mathbf{x}^{(w)}$ corrompue en remplaçant le mot central de **x** par un autre mot **w** de **V.** En réduisant l'objectif, le modèle va attribuer un score supérieur à la fenêtre correcte qu'à la fenêtre incorrecte d'au moins une marge de 1. Leur architecture de modèle est illustrée à la figure II.8 suivante :

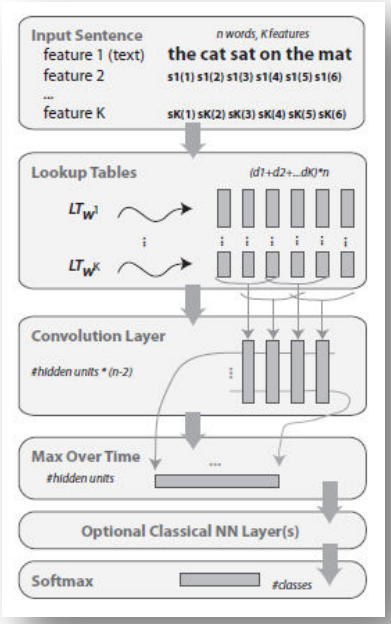


Figure II.9 : **Architecteur du modèle C & W [32]**

II.3.3.4 Le modèle Word2Vec

Le modèle Word2vec est sans doute le modèle le plus populaire. Il fut lancé par Mikolov et al. en 2013 **[29].** Il a pour objectif de représenter les mots en vecteurs. Sachant que l'incorporation de mots est un élément fondamental des modèles d'apprentissage en profondeur pour la PNL (Programmation neuro-linguistique), on suppose souvent que word2vec appartient au même groupe. Toutefois, techniquement, word2vec n'est pas considéré comme faisant partie de l'apprentissage en profondeur, car son architecture n'est ni profonde, ni utilise des non-linéarités. Ce modèle permet d'éliminer la couche cachée coûteuse et offre au modèle de langage un moyen de prendre en compte un contexte supplémentaire **[27] [30].**

Deux architectures furent proposées par Mikolov et al. : Le modèle CBOW et le modèle Skip-gram [**28] [27].**

- **Le modèle CBOW : Continuous bag-of-words**

Le problème rencontré dans le modèle linguistique est le fait que ce dernier ne peut que rechercher les prédictions dans les mots précédents. Mikolov et al. ont mis en place un modèle pouvant utiliser les mots avant et après le mot cible Wt**[27]**. Ils appellent ce type « sac de

mots continu » (CBOW), car il utilise des représentations continues dont l'ordre n'a aucune importance.

Au lieu d'introduire n mots précédents dans le modèle, ce dernier reçoit une fenêtre de n mots autour du mot cible w_t à chaque pas de temps t **[27].**

- **Le modèle skip-gram**

Le modèle Skip-gram emprunte le chemin inverse du modèle CBOW, puisque ce dernier utilise les mots environnants pour prédire le mot central, contrairement au modèle skip-gram qui utilise le mot central pour prédire les mots environnants **[27][29].**

L'objectif skip-gram est d'ajouter des n mots environnants à gauche et à droite du mot cible w_t.

Tel qu'à chaque mot, sont associés deux vecteurs à apprendre :Uw (vecteur d'entrée :input) et Vw (vecteur de sortie :output), puis calculer ainsi la probabilité reliant chaque mot aux mots du vocabulaire par l'équation :

$$\mathbf{P(w_i|w_j) = \frac{\exp(U_{wi}^T V_{wj})}{\sum_{l-1}^{V} \exp(U_j^T V_{wj})}} \qquad \mathbf{II.6}$$

Où V est la taille du vocabulaire.

Dans ce qui suit, une figure montrant la principale différence entre le modèle CBOW et Skip-gram

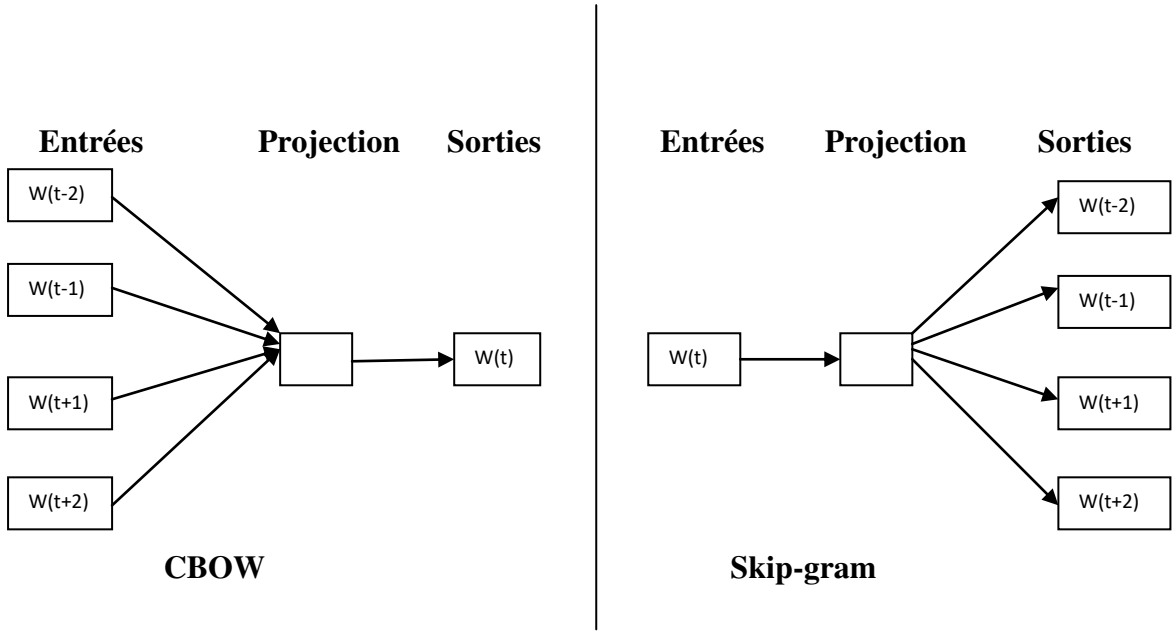


Figure II.10 : comparaison entre le modèle CBOW et le modèle Skip-gram.

II.3.3.5 le modèle Glove (Global Vectors for Word Representation)

GloVe est un modèle de Word Embedding qui succède le modèle Word2Vec. Ce dernier est un algorithme d'apprentissage non supervisé permettant d'obtenir des Word Embedding, qui porte sur les statistiques globales de co-occurrence des mots dans un corpus, et les représentations résultantes présentent des sous-structures linéaires intéressantes de l'espace vectoriel des Words Embedding**[31]**.

Ce modèle apporte trois modifications au modèle Skip-gram du Word2vec qui sont :

- Glove utilise la variable de distribution non probabiliste. (est un modèle basé sur la co-occurrence contrairement au word2vec qui est un modèle prédictif utilise les probabilités)
- Il ajoute deux paramètres de modèle scalaire pour chaque mot i : b_i pour les mots cibles centraux et c_i pour les mots de contexte.
- Il remplace le poids de chaque perte par la fonction h (x_{ij}).

Par conséquent, l'objectif de GloVe est de minimiser la fonction de perte, pour cela il propose un objectif pondéré des moindres carrés **J** qui vise directement à minimiser la différence entre le produit des vecteurs de deux mots et le logarithme de leur nombre de co-occurrences :

$$\boldsymbol{J = \sum_{i,j=1}^{V} h(X_{ij})(w_i^T v_i + b_i + c_j - logX_{ij})^2} \qquad \textbf{II.7}$$

Où w_i et b_i sont le mot vector et bias respectivement du mot i, x_{ij}est le nombre de fois que le mot i se produit dans le contexte du mot j et h est une fonction de pondération qui attribue un poids relativement inférieur aux co-occurrences rares et fréquentes.

Le modèle glove n'a pas une grande différence avec le modèle word2vec, les deux modèles apprennent des représentations géométriques (vecteurs) des mots à partir de leur co-occurrence **[33].**

II.4 Le Word Embedding en recherche d'information

Apprendre aux machines à répondre automatiquement aux questions posées en langage naturel sur n'importe quel sujet ou dans n'importe quel domaine a toujours été un objectif de longue date dans l'intelligence artificielle. Pour arriver à leur fin, les chercheurs se sont basés

sur les réseaux de neurones. D'ailleurs de nombreux travaux ont montré que les approches d'apprentissage par les réseaux de neurones sont très efficaces dans plusieurs tâches de la RI **[34][35]** et question-réponse **[36]**).

Ces travaux se décomposent en deux grandes catégories :

La première utilise des représentations distribuées de mots (Word Embedding) obtenues par les modèles neuronaux pour les intégrer dans les fonctions d'appariement requête-document **[37].**

La deuxième catégorie de travaux consiste en des modèles d'appariement qui apprennent la pertinence des paires document-requête à partir des vecteurs sémantiques latents**[35][38].**Par exemple, le Deep Semantic Structured Model(DSSM)**[35]** qui est l'un des modèles les plus efficaces dans la tâche de recherche sur le web. Ce modèle applique un réseau de neurone profond sur la représentation d'un document et d'une requête obtenues par une méthode de hachage de mots qui permet d'apprendre leurs représentations latentes à partir de leur valeur de pertinence. Une extension du modèle DSSM, proposée dans **[39]**, elle s'appuie sur un réseau de convolution, appelée Convolutional Latent Semantic Model (CLSM) **[40]**. Notre travail s'insère dans l'utilisation des Word Embedding au niveau de l'expansion de requêtes. Par conséquent, nous décrivons dans ce qui suit quelques travaux réalisées dans ce contexte.

II.4.1 Expansion de requête en utilisant le Word Embedding [41]

Comme nous l'avons déjà cité auparavant, l'expansion de la requête consiste à ajouter de nouveau termes à la requête initiale ou alors revoir le poids de ses termes dans le but de cibler la recherche vers les documents pertinents.

Parmi les techniques utilisées dans l'expansion des requêtes le Word Embedding. Dans ce qui suit nous présentons quelques travaux basés sur cette technique :

-Approche de Roy et al. (2016)

Ils proposent dans leur approche trois versions pour utiliser les Words Embedding dont l'expansion de requêtes basée sur K-NearestNeighbor (KNN).

La première version calcule les K voisins les plus proches pour chaque terme de requête dans l'espace vectoriel basée sur la représentation des Words Embedding. Puis sélectionne l'ensemble des termes d'expansion en calculant la similarité moyenne entre chaque mot candidat et les termes de la requête.

La deuxième approche réduit l'espace de vocabulaire considéré par KNN en considérant uniquement les termes des K premiers documents retournés en réponse à la requête initiale de l'utilisateur.

La troisième approche, est une stratégie couteuse en terme de calcul, est appliqué pour réduire le nombre de voisins les plus proches, en appliquant un processus itératif en supposant que les voisins les plus proches sont semblables les uns aux autres.

Ces trois versions sont plus détaillées dans le troisième chapitre, pour laquelle nous proposons une extension.

-Approche de Rekabsaz et al. (2016) [41]

Cette approche recommande de choisir des termes similaires en fonction d'un seuil de similarité plutôt que de KNN Ils choisissent le seuil en le définissant pour n'importe quel terme. Ils utilisent le modèle word2vec skip-gram pour produire les Words Embedding qui sont utilisés pour calculer la similarité du cosinus. Les expériences menées sur TREC 6-8 et HARD 2005 intègrent cette méthode dans un modèle de langue de traduction pour la recherche ad hoc, ce dernier donne de meilleurs résultats en fonction de la précision moyenne (MAP)

-Approche de Zheng et Callan. (2015) [41]

Ils utilisent les Words Embedding obtenus par le modèle word2vec CBOW pour construire des vecteurs de « caractéristiques » pour chaque mot de requête. Ce vecteur est obtenus par la soustraction entre la moyenne des vecteurs des Words Embedding des termes de requête et la moyenne des vecteurs des Words Embedding des termes donnés. Cette technique consiste à prédire un poids cible pour chaque terme de la requête étant donné son vecteur caractéristique en utilisant la régression. Le poids cible de chaque terme est tiré des jugements de pertinence. En supposant que des jugements de pertinence sont disponibles pour la même collection.

Des expériences effectuées sur quatre collections de test TREC : Robust04, WT10t, GOV2 et ClueWeb-Cat-B, en utilisant trois modèles de recherche : BM25, le modèle de langue unigrame et le modèle de dépendance séquentielle. Les résultats montrent que les Words Embedding peuvent améliorer la précision de recherche ad hoc sans nécessiter une modélisation de réseau neuronal de bout en bout. En ce qui concerne la dimensionnalité des

Word Embeddings, Zheng et Callan (2015) trouvent que 100 dimensions fonctionnent mieux pour estimer les poids des termes, mieux que 300 voir 500 dimension.

- Approche de Zuccon et al. (2015) [41]

Ils proposent un modèle de traduction de langage neuronal (NLTM) qui intègre les Words Embedding. Ce modèle consiste à estimer la probabilité de traduction entre les termes comme la similarité (cosinus) des deux termes divisés par la somme de la similarité de cosinus entre le terme de traduction et tous les termes du vocabulaire. Les expériences d'évaluation de l'approche NLTM sur les collections TREC AP87-88, WSJ87-92, DOTGOV et MedTrack ont fournit des améliorations modérées par rapport aux modèles basés sur l'information mutuelle. L'analyse des différents paramètres des modèles des Words Embedding montrent que la dimensionnalité de l'espace vectoriel, la taille de la fenêtre de contexte et le type du modèle (CBOW ou Skip-gram) n'ont pas d'impact sur le modèle de langage de traduction neuronal NLTM.

II.5.Conclusion

Ce deuxième chapitre nous a permet de présenter les concepts de base de l'expansion de requêtes et le word embedding. Dans un premier lieu, nous avons présenté la classification les approches de l'expansion de requêtes et expliquer ses étapes. Dans un second lieu nous avons présenté les Word Embedding, le formalisme sur lequel ils se basent (les réseaux de neurones) et les différentes implémentations de Word Embedding. Enfin, nous avons présenté quelques travaux en Recherche d'Information qui utilisent les Word Embedding.

CHAPITRE III

APPROCHE PROPOSEE, IMPLEMENTATION ET RESULTATS

III.1 Introduction

L'expansion de requête est parmi les méthodes utilisée pour améliorer les résultats de la recherche d'information. Notre travail consiste à implémenter et tester une extension d'une approche d'expansion de requêtes proposée par « Roy et al » [46].

Dans la première section nous allons présenter l'approche de Roy et al [46]. Puis dans la seconde section nous expliquerons notre contribution en décrivons l'extension proposée, son principe et ses étapes.

Par la suite nous allons introduire l'environnement de développement de notre approche en précisant les outils et le langage utilisés pour sa mise en œuvre. Enfin nous présentons les résultats obtenus sur la collection de test TREC AP88.

III.2 L'approche de Roy et al. [46]

L'approche proposée par Roy et al. [46] pour l'extension de la requête est basée sur le Word Embedding, ce dernier est un mappage qui associe chaque mot ou phrase apparaissant dans une collection de documents à un vecteur dans R^n, où n est nettement inférieur à la taille du vocabulaire de la collection de documents. Autrement dit, si a et b sont deux mots, *A* et *B* leur Embedding alors on peut s'attendre à ce que la distance entre *A* et *B* reflète la relation sémantique entre a et b. Cette approche utilise les relations sémantiques et contextuelles entre les mots, où les termes liés à la requête sont obtenus avec la technique des K plus proches voisins KNN (K-Nearest Neighbours).

III.2.1 Objectif de l'approche

L'objectif de cette approche est d'améliorer la pertinence de la recherche d'information avec la modification de la requête de l'utilisateur par l'ajout des mots voisins de ceux employés initialement.

Pour la sélection des termes d'extension, Roy et al ont utilisé l'algorithme KNN sur les vecteurs des termes candidats à l'expansion. Trois versions de l'approche sont proposées. Nous les décrivons dans la section suivante.

III.2.2 Les versions de l'approche

Roy et al ont proposé trois versions de leur approche basées sur KNN qui sont : Pré-récupération, Post-récupération, Pré-récupération incrémentielle.

a. Pré-Récupération (Pre-retrieval)

La première version c'est la méthode la plus simple, elle calcule les K plus proches voisins (KNN) pour chaque terme de la requête dans l'espace vectoriel basé sur la représentation des Words Embedding elle n'utilise pas les documents les plus pertinents qui sont retournés au en réponse à la requête initiale de l'utilisateur. Elle utilise tous le vocabulaire de la collection comme source des termes d'expansion. Nous présentons dans ce qui suit la formalisation de cette version.

Soit la requête **Q** composée des mots $\{q_1, q_2, \ldots, q_m\}$. Les auteurs ont défini l'ensemble **C** des termes d'extension à ajouter à **Q** comme suit :

$$C = \cup_{q \in Q} NN(q) \qquad \text{III.1}$$

Où **NN** est l'ensemble des **K** termes les plus proches dans l'espace vectoriel et **Q** est la requête initiale.

Pour chaque terme d'extension **t** dans **C** ils ont calculé la similarité moyenne $\mathbf{Sim_{moy}(t, q)}$ entre **t** et tous les termes de la requête **Q** comme le montre l'équation suivante :

$$\mathbf{Sim_{moy}(t, q)} = \frac{1}{|Q|} \sum_{q_i \in Q} \mathbf{Sim_{cos}}(\vec{t}, \vec{q}_i) \qquad \text{III.2}$$

b. Post-Récupération (Post-retrieval)

Roy et al ont proposé une deuxième version qui est une variante de la première qui utilise l'ensemble des documents pseudo-pertinents (Pseudo Relevent-Documents noté **PRD**), qui sont les tops documents retournés en réponse à la requête initiale de l'utilisateur. Autrement dit, au lieu de chercher les termes voisins les plus proches dans toute la collection, ils ont réduit l'espace de vocabulaire considéré par KNN en considérant uniquement les termes utilisés dans la **PRD**.

Le reste de la procédure est le même que celui de la section précédente (Pré-récupération).

c. Pre-récupération incrémentielle (Pre-retrieval incremental)

Cette version est une extension de la première version. Dans celle-ci, au lieu de calculer les voisins les plus proches (KNN) en une étape, ils utilisent un processus itératif. Sachant que les termes les plus similaires sont les termes les plus probables pour devenir les termes de l'expansion.

Roy et al ordonne **NN(q)** dans l'ordre décroissant de similarité comme suit $t_1, t_2, \ldots t_N$; et ils ont éliminé les **k** derniers termes les moins similaires à **q**, pour obtenir **:** $t_1, t_2, \ldots t_{N-k}$;puis ils ont pris t_1 *et* $t_2, \ldots t_{N-k}$ dans l'ordre de similarité avec t_1 , ensuite ils ont éliminé les k termes les moins similaires à t_1 pour avoir $t'_2, \ldots t'_{N-2K}$ puis ils ont pris t'_2 et ils ont réitéré le processus pendant **I** itérations.

Comme le montre le tableau suivant :

	Comparer à q	Comparer à t_1	Comparer à t_2	Comparer à t_{I-1}
Nombre d'itérations	1ére itération	2ème itération	3ème itération	Ième itération
L'ensemble **C** des termes candidats	t_1 t_2 . . . t_N	t_2 t_3 . . . t_{N-k}	t_2' t_3' . . . t_{N-2k}'	T_{I-1}' T_I' . . . $t_{N-(I-1)k}$'

Tableau III.1: Tableau explicatif de l'approche Pre-retrieval incremental

- **L'ensemble des termes de la requête étendue**

Roy et al. se sont basés sur la technique des k plus proches voisins pour généraliser le processus du choix des termes d'expansion. Pour ce faire ils ont considéré la requête **Q** composée de **m** termes **{q_1,…,q_m}** puis ils ont construit l'ensemble $\mathbf{Q_C}$ des mots de requête bi-gramme comme suit :

$$\mathbf{Q_C = \{(q_1, q_2)\ (q_2, q_3) \ldots (q_{m-1}, q_m)\}}$$

Par la suite ils ont défini le vecteur Word Embedding pour le bigramme (q_i,q_{i+1}) comme une somme des deux vecteurs des termes correspondants :$\vec{q}_i + \vec{q}_{i+1}$. Finalement l'ensemble Q' des termes est donné comme suit :

$$Q' = Q \cup Q_C \quad \textbf{III.3}$$

La dernière étape de l'expansion de la requête est la reformulation de la requête. La méthode « Jelinek Mercer » est utilisée pour obtenir le modèle de la requête étendue, comme le montre la formule suivante :

$$P(w|Q_{exp}) = \alpha P(w|Q) + (1-\alpha)\frac{Sim(w,Q)}{\sum_{w \in Q_{exp}} Sim(w,Q)} \quad \textbf{III.4}$$

Où Q_{exp} est l'ensemble des tops **K** termes de l'ensemble **C,** w l'ensemble des K plus proches voisins termes et α est un paramètre de lissage.

Remarque :

Nous pouvons remplacer l'ensemble Q par Q' dans la formule **III.4** comme suit :

$$P(w|Q_{exp}) = \alpha P(w|Q') + (1-\alpha)\frac{Sim(w,Q')}{\sum_{w \in Q_{exp}} Sim(w,Q')} \quad \textbf{III.5}$$

III.3 Présentation de l'approche proposée :

Dans la section suivante nous allons présenter le principe de notre extension ainsi que sa formalisation.

III.3.1 Principe de l'approche proposée et sa formalisation

L'approche proposée par Roy et al. est d'intégrer les Word Embedding au niveau de l'extension de la requête. Le choix des termes d'expansion est réalisé par l'algorithme KNN. Ils ont proposé de représenter la requête initiale sous forme de deux représentations : uni-gramme et bi-gramme **(Formule III.3)**

Cependant, lors de la construction des vecteurs Word Embedding correspondants aux bigrammes, ils ont utilisé une simple sommation des vecteurs. C'est-à-dire l'apport des vecteurs Word Embedding est identique.

Nous proposons de revoir cette composition (construction) de vecteur de Word Embedding des bi-grammes en se basant sur l'hypothèse suivante : « les termes qui composent un bi-gramme ne sont pas de la même importance » . Par conséquent, leurs vecteurs correspondants

le sont aussi. L'exemple de bi-gramme suivant illustre cette hypothèse : « ordinateur personnel ».

Nous considérons que le terme « ordinateur » est plus important que le terme « personnel ».

Pour mettre en œuvre cette hypothèse, nous nous sommes basées sur les fréquences des termes dans la collection à savoir que si un terme apparait dans moins de document alors il est plus important. Par conséquent, lors de la combinaison des termes pour former les bi-grammes, on affecte plus de poids pour le terme le plus important dans cette composition.

Formellement, nous lions l'importance d'un terme à son idf (Inverse Document Frequency) donné par la formule suivante :

$$\boldsymbol{idf}_q = \frac{\boldsymbol{N}}{\boldsymbol{N_q}} \qquad \textbf{III.6}$$

Où $\boldsymbol{N}$ est le nombre de document dans la collection et $\boldsymbol{N_q}$ est le nombre de documents qui contiennent le terme **q.**

Ainsi, lors de la composition d'un bi-gramme **(q_i,q_{i+1})** nous procédons comme suit :

$$(\vec{q}_i, \vec{q}_{i+1}) = \boldsymbol{\gamma}\, \vec{q}_i + (\mathbf{1} - \boldsymbol{\gamma})\, \vec{q}_{i+1} \qquad \textbf{III.7}$$

Où $\boldsymbol{\gamma}$ est un paramètre qui reflète l'importance donnée au vecteur Embedding du terme q_i.

Nous avons estimé ce paramètre comme suit :

$$\boldsymbol{\gamma} = \frac{\boldsymbol{idf}_{q_i}}{\boldsymbol{idf}_{q_i} + \boldsymbol{idf}_{q_{i+1}}} \qquad \textbf{III.8}$$

Cette extension nous a permis de former un nouvel ensemble de bi-grammes $\boldsymbol{Q'_c}$ donc l'équation **III.3** devient :

$$\boldsymbol{Q''} = \mathbf{Q} \cup \boldsymbol{Q'_c} \qquad \textbf{III.9}$$

Et la formule **III.5** devient :

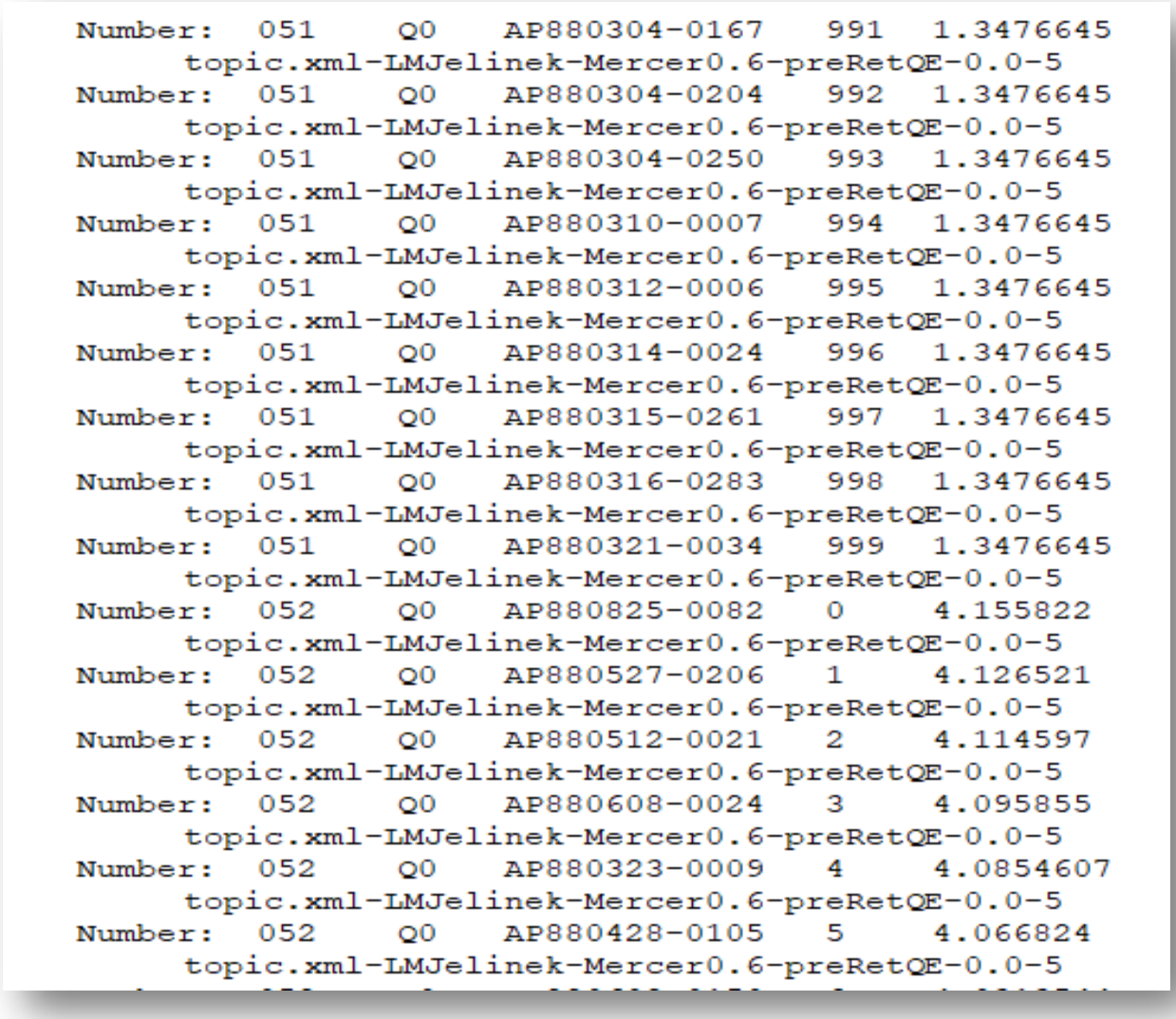
$$\boldsymbol{P}(\boldsymbol{w}|\boldsymbol{Q_{exp}}) = \boldsymbol{\alpha P}(\boldsymbol{w}|\boldsymbol{Q''}) + (\mathbf{1} - \boldsymbol{\alpha}) \frac{\boldsymbol{Sim}(\boldsymbol{w},\boldsymbol{Q''})}{\sum_{\boldsymbol{w} \in \boldsymbol{Q_{exp}}} \boldsymbol{Sim}(\boldsymbol{w},\boldsymbol{Q''})} \qquad \textbf{III.10}$$

III.3.2 La mise en œuvre de l'approche :

Nous présentons dans cette section les étapes de notre approche :

Une fois que la première recherche est effectuée avec le modèle de recherche « Jelinek Mercer », nous obtenons les tops documents (1000 premier documents) les plus pertinents pour chaque requête.

La figure suivante représente les résultats de la première recherche :



```
Number:  051    Q0    AP880304-0167   991   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880304-0204   992   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880304-0250   993   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880310-0007   994   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880312-0006   995   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880314-0024   996   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880315-0261   997   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880316-0283   998   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  051    Q0    AP880321-0034   999   1.3476645
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880825-0082   0     4.155822
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880527-0206   1     4.126521
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880512-0021   2     4.114597
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880608-0024   3     4.095855
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880323-0009   4     4.0854607
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
Number:  052    Q0    AP880428-0105   5     4.066824
      topic.xml-LMJelinek-Mercer0.6-preRetQE-0.0-5
```

Figure III.1 : Extrait d'un fichier résultats de la première recherche

Afin de mettre en œuvre notre extension nous avons suivi les étapes suivantes :

Etape01: « Extraction des word embedding »

Il existe plusieurs manières pour construire les Words Embedding, le modèle le plus connu est le word2vec qui a deux architecteurs Skip-gram et Cbow.

Cette étape permet d'extraire les Words Embedding qui représentent les vecteurs correspondants aux termes de notre collection AP88 ainsi que les termes de la requête. Ces Words Embedding sont stockés dans un fichier « Vectors.txt» qui est produit par l'algorithme d'apprentissage word2vec avec la version skip-gram en choisissant une taille de la fenêtre (Window size w=5) et une dimensionnalité des Words Embedding=200.

La figure suivante représente un extrait du fichier qui contient les Words Embedding du terme « national » :

```
national -1.407990 0.044497
0.528895 -0.179856 -0.175953 -1.743344 0.215967
0.715150 -1.065893 0.306093 -0.821624 -1.099479
0.630307 -0.936186 -0.920134 0.553093
0.532945 -0.109820 -0.082260 1.832795 -1.742606 0.889877
0.868576 -0.814404 -1.307484 -0.783597 1.665361 -0.906272
0.593886 -2.070964 0.032895 -0.182350 0.578443
0.001876 -0.326549 -0.959719 0.247176 0.395367 -1.294769
0.609206 -0.713926 -1.130420 -0.994259 -0.980100 0.659498
0.687110 -0.582110 0.895784 -1.915517 -0.165286 -0.584139
0.447221 -2.553148 0.741842 1.560176 0.248879
1.813685 -0.198859 -0.461532 1.841638
0.585640 -0.069136 -0.097661 -1.724856 -0.769121
0.657682 -0.061449 0.563144 2.390492 1.784991 0.330118 1.689531
0.930827 -0.772416 -0.507115 0.936008
0.434636 -0.335503 -0.486097 1.213945 1.167096 -0.898260
0.994521 -1.194756 0.062344 -0.583201 1.365290 -1.027198
0.402944 0.210794 0.386395 -1.259801 1.312434 0.961233 0.816744
0.008314 -0.342085 -0.175007 -0.322514 -0.670064
1.170803 -0.898314 0.633752 0.770234
0.190259 -0.524410 -1.518986 -0.137456 -0.590109 -0.581320 -0.68
3078 -0.334037 -0.555721 0.016646 0.111295 -0.990800
0.000470 -1.052460 -0.336174 -0.372346 -1.356404 -0.876472 -0.32
6313 -1.211311 -1.013645 -0.399889 -0.153072
0.864667 -0.807367 -0.034146 0.920450 -1.128648 0.190316
0.608200 0.353001 0.971748 1.247370 -0.028162 0.063326 -1.124037
0.896252 -1.268221 0.239400 -1.369036 0.768557
0.293530 -0.551657 0.645728 0.586776 -0.567379
1.032793 -0.275213 1.633355 0.674651 1.460056
```

Figure III.2 : Extrait d'un fichier qui contient les Words Embedding du terme « national »

Etape02 : « La lemmatisation des Words Embedding »

Après avoir récupérer ces vecteurs, nous devons faire la lemmatisation pour ces derniers.

La lemmatisation (Stemming en anglais) a pour objectif de construire un lemme d'un mot. Un lemme est tout ce qui reste du mot après avoir éliminé les affixes (préfixes et suffixes). Elle permet de substituer chaque mot par sa racine (lemme), qui est sous la forme infinitive si le mot est un verbe, soit sous la forme de singulier masculin si le mot est un nom, adjectif ou article **[47] [48]**.

Par exemple, le terme « national» illustré dans la **Figure III.2** devient : « nation»

Etape03 : «Extension de l'approche de Roy »

Pour mettre en œuvre notre approche nous avons utilisé l'implémentation fournie par Roy et al. laquelle nous avons étendue pour la prise en compte de l'extension proposée.

La section suivante présente les outils, APsI et environnement qui nous ont servi à l'implémentation de notre approche.

III.4 L'environnement de développement

Dans ce qui suit nous allons présenter les différents outils utilisés pour mettre en œuvre notre approche à savoir la bibliothèque Lucene, le langage de programmation java, TREC_EVAL et l'environnement Netbeans.

III.4.1 Lucene[49]

Lucene fonctionne sous différentes plateformes par exemple : Windows et Linux dans notre cas on a utilisé la version sous Windows.

Lucene est une bibliothèque gratuite, très flexible, efficace et effectif, on peut le télécharger sur : https://archive.apache.org/dist/lucene/java/. Dans notre approche on a utilisé la version lucene 3.6.0.

Lucene est une librairie open-source écrite en java, on l'intègre dans des projets Netbeans pour :

- ✓ Indexer de large collections de documents : extraction des mots clés des documents appartenant a une collection et les stocker dans un index.
- ✓ Rechercher des documents pertinents : pour répondre aux requêtes formulées par l'utilisateur, cette recherche peut être effectuée sur les collections de tests standards TREC.

III.4.1.1 L'architecture de Lucene

L'architecture de Lucene distingue les deux processus : l'indexation et la recherche

La figure ci-dessous montre l'architecture générale de Lucene :

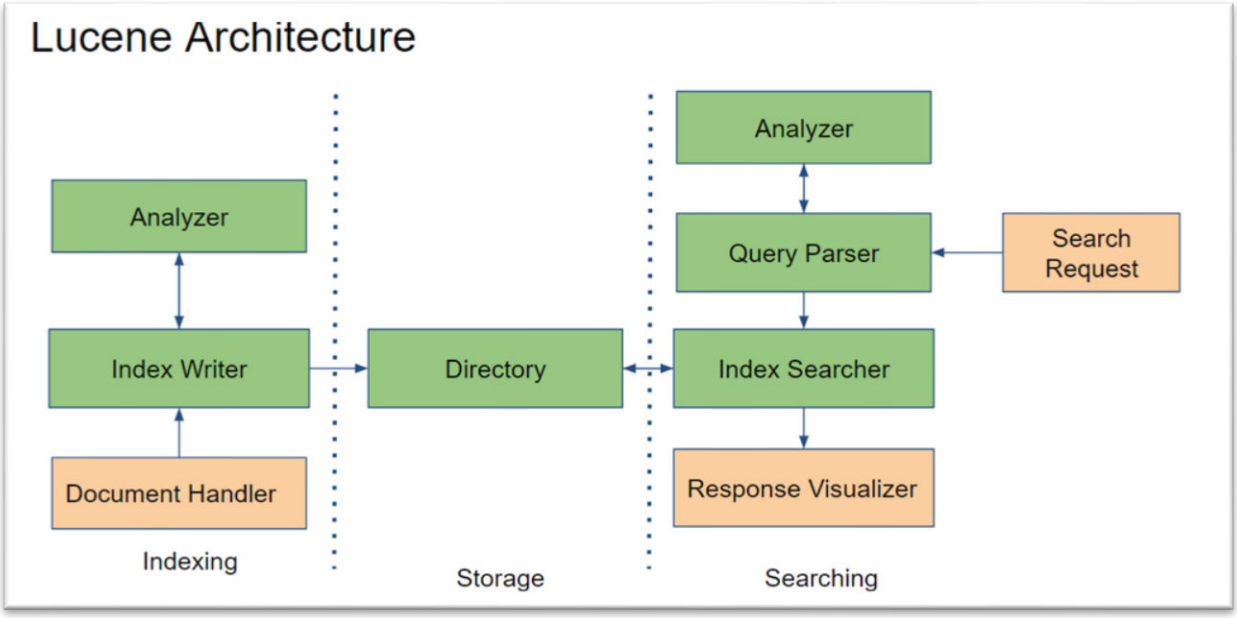


Figure III.3 : L'architecture de Lucene

Pour réaliser les deux processus « l'indexation et la recherche » on a importé un projet java développé par Roy sous Netbeans : « Luc4TREC » qui sera détaillé plus bas.

a. Le processus d'indexation

Nous avons indexé notre collection TREC « AP88 » pour l'extraction des mots clés de chaque document qui seront stockés dans un répertoire « index », pour se faire on utilisé la classe « commandLineIndexing.java », comme le montre la figure suivante :

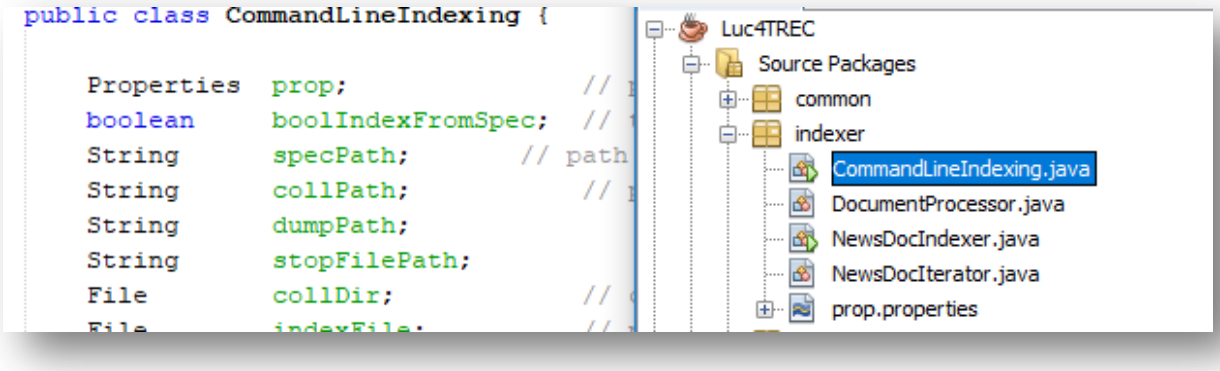


Figure III.4:La classe d'indexation

Cette dernière fait appel aux classes prédéfinies de lucene :

- IndexWriter : cette classe crée un nouvel index et ajoute des documents à un index existant
- Directoy : la classe Directory représente l'emplacement de l'indexe de lucene. Il existe plusieurs implémentations de cette classe : Directory, FSDirectory et RAMDirectory, dans notre cas nous utilisons FSDirectory.
- Analyser : avant que le texte soit dans l'index, il passe par l'Analyseur celui-ci est un filtre qui extrait les mots important pour l'index et supprime le reste.
- Document : la classe document représente un ensemble de champs.les champs d'un document représentent un document ou une métadonnée associées a celui-ci.
- Field : chaque document est un index contenant un ou plusieurs champs, intégré dans une classe intitulé Field, cela permet l'interrogation et/ou la récupération depuis l'index durant la recherche.

b. Le processus de recherche

Pour effectué la recherche sur la collection indexée, nous avons utilisé la classe « DocSearcher.java », comme le montre la figure suivante :

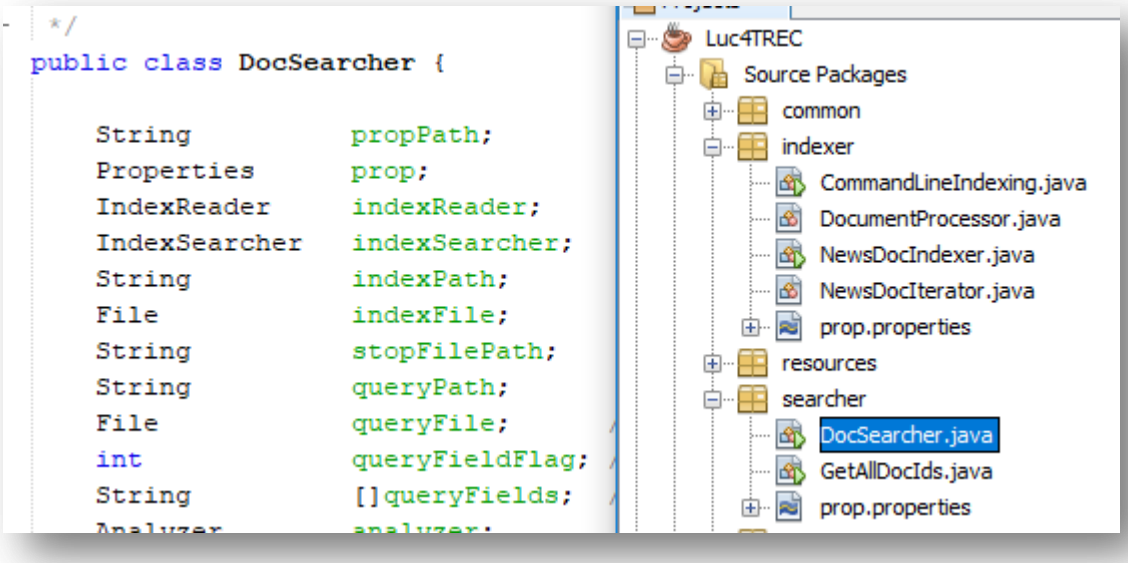


Figure III.5: La classe recherche

Cette dernière fait appel aux classes prédéfinies de lucene :

- IndexSearcher : c'est la classe qui donne accès aux index en recherche.
- Analyser : Tout comme pour l'indexation les analyzer font partie du processus de recherche fin de normaliser les critères de recherche.
- Query : c'est une classe abstraite qui représente la requête de l'utilisateur et utilisé par un IndexSearcher.
- QueryParser : un parseur de requête.
- Hits : Une collection d'éléments résultats de la recherche.
- Hit : Un élément de la collection des résultats.
- Document : Un document retrouvé et tel qu'il était lors de son ajout dans l'index (constitué des mêmes champs)

La figure suivante montre les résultats de la recherche :

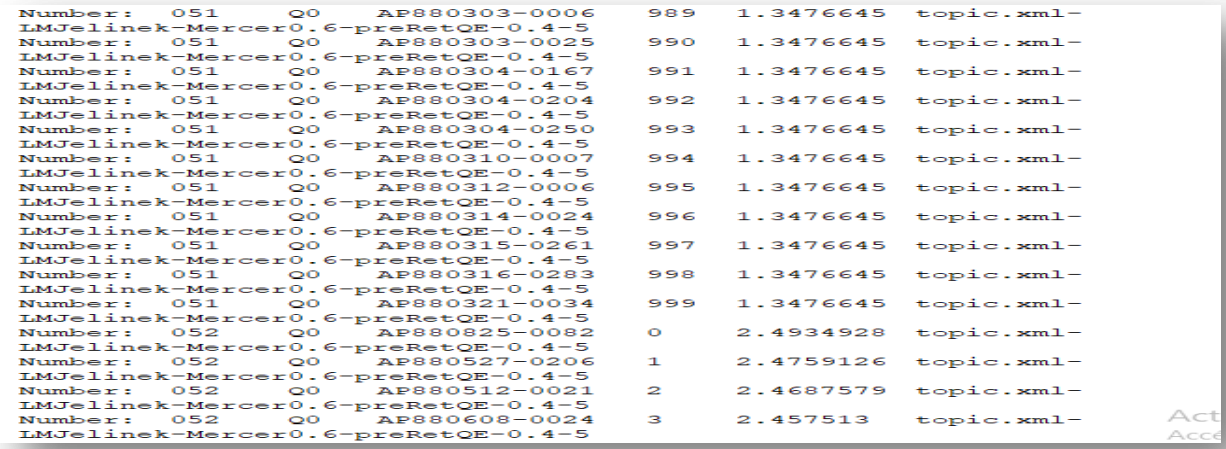


Figure III.6 : Les 1000 documents retournés.

III.4.2 Les API développées par Roy

Pour l'implémentation de notre extension nous avons utilisé les deux packages développés par Roy :

- **Luc4TREC :** ce projet est constitué de six packages dont essentiellement « common », « indexer » et « searcher ». Nous avons utilisé la classe « CommandeLineIndexing.java » du package « indexer » pour indexer notre collection TREC88, et la classe « DocSearcher.java » du package « searcher » pour effectuer la première recherche.
- **QEUsingW2V :** ce projet est composé de 3 pacckages : « common » dont nous avons utilisé la classe « CollectionStatistics.java » pour le calcul des probabilités , le package

« WordVectors » dont nous avons utilisé la classe « WordVec.java » pour le recalcul des probabilités et le dernier package «QEUsingW2V » qui comporte la classe « PotRetrievalQE.java » pour retourner les 1000 documents après notre extension.

III.4.3 TREC_EVAL [50]

L'évaluation des résultats de la recherche est réalisé avec TREC_EVAL sous Linux.

TREC_EVAL est un outil utilisé pour l'évaluation des résultats des documents triés par ordre de pertinence.

L'évaluation est basée sur deux fichiers sont :

- **Qrels**(Query relevances): contient les jugements de pertinences de l'utilisateur pour chaque requête.
- **Résultat**: contient le classement des documents renvoyés par le système RI.

III.4.4 Le langage de programmation java [51]

Java est un langage de programmation moderne né en 1995 développé par Sun Microsystems, aujourd'hui racheté par Oracle. C'est un langage orienté objet et compilé en bytecode qui est un langage intermédiaire indépendant de La machine, ce dernier est interprété par une machine virtuelle il est également typé (toute variable doit être déclaré avec un type qui est fournis soit par le langage ou par la définition des classes).

III.4.4.1 Caractéristiques [53]

- ✓ **Java est un langage de programmation orienté objet**

Objet : ensemble de données et de méthodes agissant exclusivement sur les données de l'objet.

Classe : description d'un ensemble d'objets ayant une structure de données commune et disposant des mêmes méthodes. Un objet est une instance de sa classe.

Héritage : permet de définir une nouvelle classe à partir d'une classe existante, à laquelle on ajoute de nouvelles données et de nouvelles méthodes.

- ✓ **Java est portable**

Une fois le programme java est créé, ce dernier est portable dont le code peut être exploité dans différents environnements (Windows, Mac, Linux, etc.).

- ✓ **Java est interprété**

Un code source doit être traduit dans le langage machine avant d'être exécuté.

Compilateur java traduit le code source java en bytecode (code portable) par la suite un interpréteur java spécifique a une machine donnée (JVM : Java Virtuel Machine) traduit et exécute le bytecode.

III.4.5 Netbeans [54]

L'EDI Netbeans est un environnement de développement pour java, placé en open source par Sun en juin 2000. En plus de java, il peut également supporter : python, C, C++, XML, et HTML, se concentrant principalement sur simplifier le développement d'applications java. L'EDI est lui-même écrit en java, ce qui permet de le faire tourner sur n'importe quel système d'exploitation.

NetBeans est disponible sous Windows, Linux, Mac OS X ou sous une autre version indépendante des systèmes d'exploitation (requérant une machine virtuelle java).Un environnement java développement Kit JDK est requis pour les développements en java.

La figure suivante illustre l'environnement de développement Netbeans :

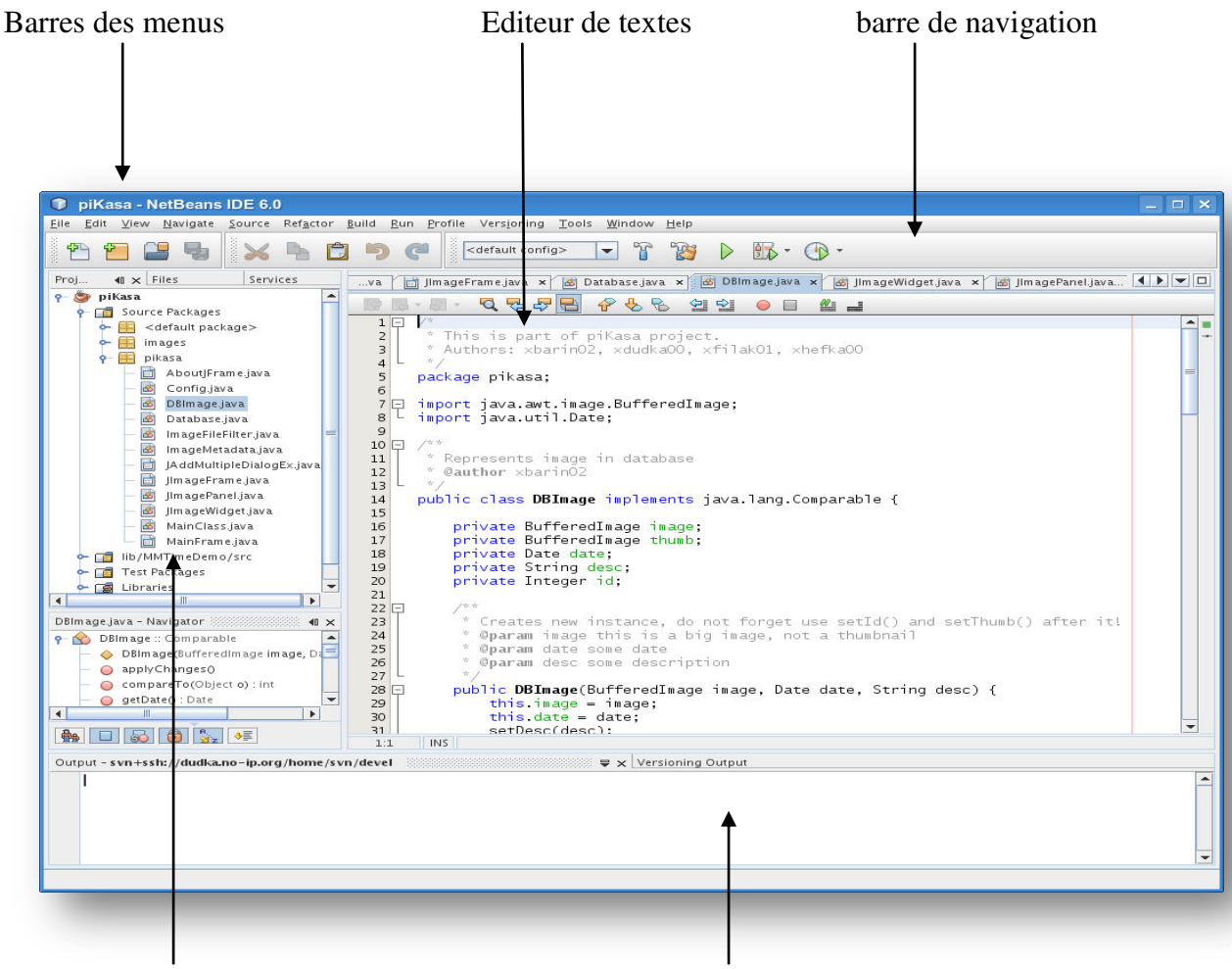


Figure III.7 : Environnement de développement de Netbeans

III.5 Evaluation et résultats

Dans cette section nous présentons dans un premier temps la collection et les requêtes utilisées ainsi que la mesure d'évalution adoptée. Ensuite, nous présentons les résultats obtenus par notre approche et celle de Roy et All [46]. Pour avoir une idée plus précise nous avons analysé aussi les résultats requête-par-requête.

III.5.1 La collection de test et les requêtes utilisées

Différentes collections de tests sont utilisées en recherche d'information. La collection que nous avons utilisée dans notre étude est : La collection **TREC AP88** (**A**ssociated **P**ress Newswire, 19**88**). Elle contient 79 919 documents.

Pour la recherche nous avons utilisée 50 requêtes issues des topics numérotées « 51-100 » de la collection TREC.

Afin d'évaleur les résultats, nous avons utilisé la mesure MAP, qui est la mesure la plus utilisée en Recherche d'information.

III.5.2 Présentation des résultats obtenus

Notre modèle à quatre paramètres à fixer. Le premier est le paramètre du modèle de recherche utilisé (Jelinek Mercer). Ce modèle à un seul paramètre λ qui permet de pondérer l'apport du modèle de la collection dans l'estimation du modèle du document. Nous avons fixé la valeur de ce paramètre à 0.6. Le second paramètre est α (voir formule III.5). C'est un parmètre qui permet de pondérer l'apport des termes d'expansion dans l'estimation du modèle de la requête finale. Nous avons varié sa valeur de 0 à 1 avec un pas de 0.1 sur l'approche de Roy et AL [46] et nous avons constaté que la valeur qui donne de meilleurs résultats est $\alpha = 0.2$.

Le troisième paramètre est le nombre de documents feed-back, que nous avons fixé à 5. Le dernier paramètre est le nombre de termes d'expansion que nous avons fixé à 10.

L'extension que nous avons proposée à l'approche de Roy et all, n'as pas de paramètre à estimer.

III.5.2.1 Résultats obtenus avant et après notre extension

Dans cette section nous présentons les résultats globaux et détaillés de notre approche et celle de Roy et all.

Le tableau ci-dessous montre les résultats en terme de précision (**MAP**) obtenus par l'approche de Roy et al et notre extension :

Résultat global	
Approche de Roy et ALL	**Notre l'extension**
0.2694	0.2688

III.2 Tableau : Résultats de la MAP globale

A partir de la table III.2 nous constatons que notre extension n'améliore pas l'approche de Roy et al. Cependant, la différence entre les deux modèles n'est pas énorme.

Pour avoir une idée plus précise de l'impact de notre extension nous avons analysé la précision requête par requête.

La figure suivante illustre la comparaison des résultats requête-par-requête du modèle de Roy et al et notre extension.

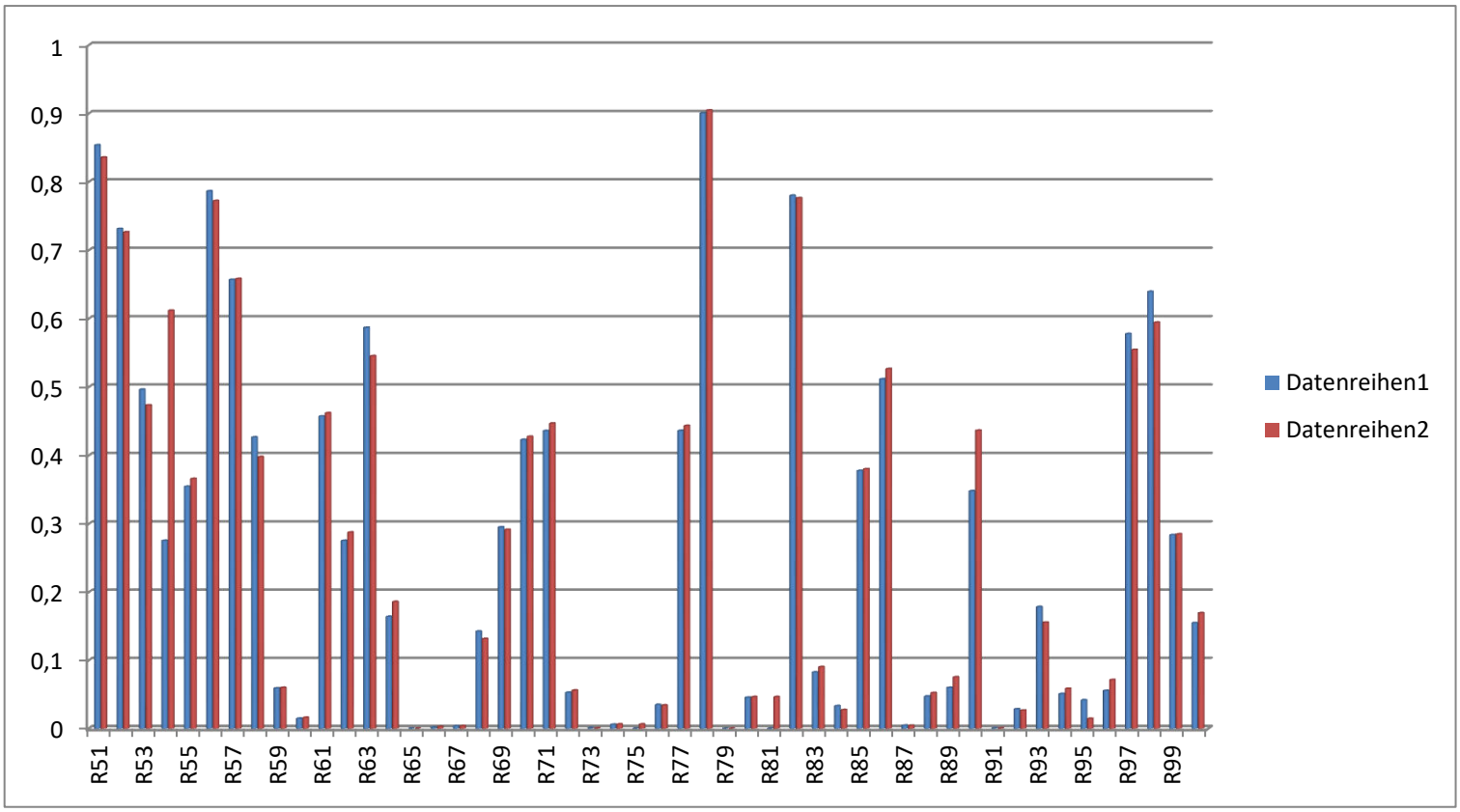


Figure III. 8 : Graphe illustrant les résultats de la MAP requête par requête

Dans le tableau suivant, nous avons selectioné que les 25 requêtes améliorées :

Requête	MAP avant extension	MAP après extension
R54	0.5700	0.6119
R55	0.3545	0.3660
R57	0.6571	0.6583
R59	0.0592	0.0599
R60	0.0147	0.0161
R61	0.4572	0.4621
R62	0.2754	0.2876
R64	0.1641	0.1859
R66	0.0017	0.0027
R67	0.0035	0.0037
R70	0.4231	0.4277
R72	0.0528	0.0560
R74	0.0058	0.0062
R77	0.4363	0.4435
R78	0.9006	0.9048
R80	0.0454	0.0463
R83	0.0825	0.0904
R85	0.3777	0.3802
R86	0.5117	0.5267
R88	0.0472	0.0523
R89	0.0599	0.0754
R94	0.0509	0.0585
R96	0.0555	0.0713
R99	0.2838	0.2850
R100	0.1549	0.1694

III.3 Tableau : Résultats de la MAP des requêtes améliorées

La figure III.8 montre que notre extension améliore 50% de requêtes. C'est à dire 25 requêtes sur les 50. Par exemple avec la requête numéro "64" (Hostage-Taking), notre extension obtient une précision de l'ordre de 0.1859 alors que l'approche de Roy et al obtient une

précision de 0.1641. Cette amélioration est liée au fait de notre pondération des vecteurs lors de la construction d'un bi-gramme; ici le bi-gramme est " Hostage-Taking". Nous pouvons remarquer que le terme « Hostage » est plus important que « Taking » dans le bi-gramme.

L'enseignement à retenir de ces résultats est d'essayer d'appliquer notre extension de manière sélective sur les bi-grammes, car ce n'est pas tous les bi-grammes qui ont un sens.

III.6 Conclusion

Dans ce chapitre nous avons présenté le principe de l'approche de de Roy et al [46]. Puis dans la seconde section nous avons expliqué notre contribution en décrivons l'extension proposée, sa formalisation et ses étapes.

Par la suite nous avons introduit l'environnement de développement de notre approche en précisant les outils et le langage utilisés pour sa mise en œuvre. Pour enfin terminer avec la présentation des résultats obtenus avant et après notre extension.

CONCLUSION GENERALE

Conclusion générale

Notre travail présenté dans le cadre de ce mémoire s'insère dans le domaine de la recherche d'information, plus particulièrement dans le cadre de la reformulation de la requête. Il porte sur l'extension d'une approche d'expansion de requête basée sur le Word Embedding présentée dans [46].

Pour mener à terme notre travail, nous avons donné un aperçu général sur la recherche d'information ainsi que le système de recherche d'information.

Nous avons ensuite présenté la technique du Word Embedding et ses quelques modèles qui se basent sur les réseaux de neurones, aussi nous avons cité quelques travaux qui utilisent le Word Embedding en recherche d'information.

Pour mettre en œuvre notre approche «Extension d'une approche d'expansion de requête en utilisant les Word Embedding » nous avons utilisé la bibliothèque Lucene, le langage de programmation Java et l'environnement NetBeans.

L'extension proposée n'a pas apporté d'amélioration globale par rapport au modèle de Roy et al [46]. Cependant, on a constaté des améliorations sur un certains nombre de requêtes. Ce qui est un bon indice.

Parmi les pistes à suivre pour améliorer les résultats de notre extension est d'identifier les requêtes qui nécEssitent l'introduction du faCteur de pondération proposé.

BIBLIOGRAPHIE

Bibliographie

[1] :Amirouche. « La recherche d'information ».cours Master2 ingénierie des systèmes d'informations. Université Mouloud Mammeri Tizi-Ouzou.2018/2019.

[2] : Hammache A. Recherche d'Information. « Un modèle de langue combinant mots simples et mots composés ».Thèse de doctorat à l'Université Mouloud Mammeri Tizi-Ouzou.2013.

[3] : Laure S. « Définition et évaluation de modèle de recherche d'information collaborative basé sur les compétences de domaine et les rôles des utilisateurs ». Recherche d'information [cs.IR]. Thèse doctorat à l'Université de Toulouse. 2014.

[4] : Mohand B. « Introduction à la recherche d'information ». Cours à l'Université Paul Sabatier.

[5] : Abbassi M. « Un modèle de reformulation des requêtes pour la recherche d'information sur le Web ». Thèse doctorat à l'Université de Ouargla.2013.

[7] : H. Aliane, Z.Alimazighi, R.O.Bouchaga, T.Djelliout**. «** Un Système de reformulation de requêtes pour la recherche d'information ». Thèse de master à l'université Houari Boumediene.2004.

[9] : Mahmoudi S M.« Indexation automatique et la recherche d'information dans les documents ».Cours à l'université de Téhéran. Année2006.

[11] : L Hlaoua. « Reformulation de requêtes par réinjection de pertinence dans les documents semi-structurés ».Thèse de master à L'université Paul Sabatier Toulouse.14 Décembre 2007.

[12] : Philipe M, Jean P C. « Modèle de langue par type de doxel pour l'indexation de documents structurés ».Thèse de master à l'Université de Grenoble. mars 2012.

[13] : Bouramoul A. « Recherche d'informations contextuelle et sémantique sur le web ». Thèse de doctorat à l'Université MENTOURI de Constantine.25/09/2011.

[14] : William L. « Modèle de langue basé sur l'analyse distributionnelle »..24 Août 2012.

[15] : B Audeh. « Reformulation Sémantique des requêtes par la recherche d'information ad hoc sur le web ». Thèse de doctorat à l'Ecole nationale supérieur des Mines de Saint-Etienne.6 Mars 2015.

Bibliographie

[16]: Hiteshwar K A, Akshay De. «Query expansion techniques for Information Retrieval: A survey ».Article(Information Processing & Management). 01 Août 2015.

[17]: C Caprinto, Giouanni Romand. « A survey of Automatic query expansion in information retrieval »Article(Automatic Query Expansion in: January). Janvier 2010.

[18]: Janod K, Morch M, Dufour R, Lianrés G. «Réseaux de neurones pour la représentation de contextes continus des mots ». Thèse à l'Université d'Avignon, LIA(France).

[20] : LEBRET R «Word Embedding for Natural LanguageProcessing ».Thèse de doctorat en informatique, ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE.2016

[21]: T.Mikolov, I.Sutskever, K.Chen, G.Courado, J.Dean. « Distributed Representation of words and phrases and their compositionality».Article d'une conference sur le Machine learning2013.

[22]: Tixier A. « Introduction to word embedding».Cours. Novembre 2015.

[23]: TOUZET C. «Les réseaux de neurones artificielles ». Cours. juillet 1992.

[24] : BENNANI Y. « Apprentissage par réseaux de neurones artificielle ».Cours à l'Université Paris13. 06 Juin 2014.

[25] : PETITJEAN G. « Introduction aux réseaux de neurones ».cours .

[26] : Msoaf M, Belmajdoub F. «L'application des réseaux de neurones de type dfedforward ». Xéme conférence internationale : conception et production Intégrées. 22 Janvier 2016.

[27] : Ruder S. « Word embedding ».Conférence on words embedding. 11 avril 2016.

[28]: Holzinges A.«Introduction to word embedding: word-vectors (word2vec/Glove)».Cours. 2016.

[29]: Pierrejean B. «Comprendre et utiliser les word embedding ».cours à l'Université Toulouse Jean Jaurès-CLLE-ERSS.

[30] : Pierre L, Bothna B. « Word2vec &Semvue Détection et visualisation de termes pertinents de maintenance ».cours.12 Décembre 2017.

[31] :J Pennington J, Socher R, Maning C. « GLOVE: Global vectors for word representation». Conférence on World Wide Web, pages 406–414. ACM. à l'Université Stanford.

[32]: Collobert R, Weston J. «A unified Architecture for Natural language processing».Conférence en Machine learning .2008.

[33]: Aston Zhang, Zachary C.Lipton, Alexander J.Smola. «Dive into deep Learning ». Livre.17 avril 2019.

[34]: Bengio et al.2007, Ranzato et al.2007,vincent et al.2008. « deep learning methods significantly ».Article de Journal of Machine Learning Research 11 (2010) 625-660.

[35]: Huang, P.-S .Gao,J.Deng, L.Acero, A.and Heck. «Learning deep structured semantic models for web search clickthrough data». In Proceedings of the 22nd ACMinternational conference on information & knowledge management, pages 2333-2338.2013.

[36]: Bordes, A.Chopra, S. and Weston, J.(2014).«Question answering ».In proceedings of the 2014 conference on Empirical Methods in Natural language processing (EMNLP), page 615-620.Association for Computational linguistics.2014.

[37]: Mitra, B.Nalisnick, E.Craswell, N.andCaruna, R.(2016). A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137.Extends WWW 20160 poster: Improving document ranking with dual word Embedding.2016.

[38]: Hu, B.Lu, Z.Li, H. and Chen, Q.(2014). « Convolutional neural network architectures for matching natural language sentences». In Advances in Neural Information Processing Systems, pages 2042.2050.Article.2014.

[39]: Sheen, YoHo, X.Gao, J.Deng,L.andMensil, G.(2014a). «A latent semantic model with convolutional pooling structure for information retrieval».In proceedings of the 23rd ACM International Conference on conference on Information & knowledge management, pages 101-110.ACM.2014.

[40]: Guo, J.Fan, Y.Ai, Q.and Croft, W.B.(2016). « A Deep Relevance Matching Model for ad hoc Retrieval». In the 25th ACM International conference on information and knowledge management, Indianapolis, United States.2014.

[41]: Kezan D, Ye Z, MdMustafizur R, Pinar K, Alex B, Brandon D, Heng-Lu C, Henna Kim, McNamara Q, Angert A, Banner E, VivekKhetan, McDonnell T, An Thanh Nguyen, Dan Xu, Byron C. Wallace, Maarten de Rijke, Matthew Lease. «Neuronal information retrieval: at the end of the early years ».10 November 2017.

[42]: Mathieu L. «Modèles neuronaux pour le traitement des langues».Cours. 3 février 2016.

[43] : Bouraoui JL, Guimier de Neef E, Gaillard B, Boualem M, Collin O. « Expansion sémantique de requêtes ».Cours. Mercredi 31 mars 2010.

[44]: 1. Arguello, J., Elsas, J.L., Callan, J., Carbonell, J.G. « Document representation and query expansion models for blog recommendation». ICWSM 2008(0), 1 (2008).

[45] : Masri AL, Berrut M, Chevallet C, J.P.: Wikipedia-based semantic query enrichment. In: Proceedings of the sixth international workshop on exploiting semantic annotations in information retrieval, pp. 5{8. ACM (2013).

[46] : Roy D, Debiyoti P, Mandar M. « Using word embedding for Automatic QueryExpansion ».Article.24 juin 2016.

[47] : Ounnaci I. « Recherche d'information dans les documents pédagogiques structurés adapté aux besoins spécifiques des apprenants ».Ingénierie des systèmes informatique. Mémoire à l'Univérsité Mouloud Mammeri TiziOuzou.

[48] : Andreea D. « Datamining-Indexation des documents ». Cours à l'Université Aix-Marseille.2012.

[49] : Ludovic J. « Introduction à Lucene et à Solr ». Cours.

[50] :Bham P, Hujun X, Noha E. « Evaluation of IR system based on TREC_EVAL ». Cours.26/10/2010.

[51]: Gauthier P, Laurent V. «Initiation à la programmation orienté objet avec le langage java».Cours.2013.

[52] : Kabil B. « Un Nouvel Algorithme de Stemmatisation pour l'Indexation Automatique de Documents non-structurés ». octobre 2013.

[53] : Juliette D. « La programmation en Java ».Cours à U.F.R D'informatique.2015.

[54]: Bernard D. «Netbeans/PHP».Cours.25 Mars 2010.

[55] : Bouraoui J, Guimier de Neef E, Gaillard B, Boualem M, Collin O. « Expansion sémantique de requêtes ». Cours à l'Université Catholique de LouvainMercredi 31 mars 2010.

WEBOGRAPHIE

Webographie

[6] : http://www.iro.umontreal.ca/~nie/IFT6255/Introduction.pdf

[8] : https://fr.wikipedia.org. Mot vide.

[10] :http://stephane.ayache.perso.luminy.univ-amu.fr/zoom/cours/Cours/RIVid/RI1.pdf
« Indexation et recherche d'information en vidéo ».

[19] : Https://fr.Wikipedia.org.word embedding.