République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

FACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes
Master Académique

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Conduite de projet informatique**

Thème

Extension d'un modèle basé sur le modèle de translation en utilisant le Word Embedding.

Mémoire soutenu publiquement le 09/07/2018 devant le jury composé de :

Président : Mr. Fillali Idir
Examinateur : Mr. Sadou Samir
Examinateur : Mr. Radja Hakim
Encadreur : Mr. Hammache Arezki

Présenté par

Mouzarine sabrina
Bouchemma sadia

2017/2018

Remerciement

Nous remercions le bon DIEU de nous avoir donné santé, courage, force et patience pour réaliser ce modeste travail.

Notre profonde gratitude et sincères remerciements vont à notre promoteur Mr A. HAMMACHE pour nous avoir confié ce travail, pour son suivi, sa disponibilité, son orientation et ses remarques pertinentes et précieuses.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous font en acceptant de juger ce mémoire de fin d'études.

Enfin, nous remercions toutes les personnes ayant contribué de près ou de loin au bon accomplissement de notre travail.

Dédicaces

Je dédie ce modeste travail à la mémoire de mes grands-parents, que dieu bénisse leurs âmes.

A ceux qui ont éclairé ma vie, ma très chère mère Ferroudja et mon cher père Ammar pour leur aide et leur soutien tout au long de mes études, et qui ont fait de moi ce que je suis aujourd'hui et j'espère qu'un jour je serai capable de leur donner au moins le minimum car quoiqu'on face on arrivera jamais à leurs rendre tout.

A mes chers frères Said et Youva.

A ma chère amie Sadia.

A tous mes oncles et mes tantes Rabah et sa femme Farida, Arezki et sa femme Djamila, Mohammed et sa femme Djahida, Fatma.

A mes cousins: Samir, Mourad, Madjid, Rayane, Gaya, Aris,Mayas.

Particulièrement Mouloud.

A mes cousines : Nassima, Farah, Ilham, Liticia et Alicia.

A tous mes ami(e)s et toute la promotion Informatique LMD 2017/2018.

Sabrina

Dédicaces

C'est avec un immense plaisir que j'écris ces lignes, et que je tiens à exprimer ma profonde gratitude à mes très chers parents et ma grande mère qui n'ont cessé de me soutenir, et de croire en moi pendant mes études, je les remercie du fond du cœur.

A mes soeurs : Razika, Malika, Nora, Kahina et à mon frère : Djamel et à sa femme Lamia, qui sont ce que j'ai de plus chers et qui ont toujours été là pour moi.

A ma collègue Sabrina.

A tous mes amis, et tous ceux qui me connaissent de prés ou de loin.

A toute la promotion Informatique LMD 2017/2018.

Sadia

Sommaire

Chapitre I : la recherche d'informations

Chapitre II: Le Word Embedding

Chapitre III: Approche et Expérimentation

Liste des figures :

Liste des tableaux :

Résumé

Notre travail se base sur l'extension d'un modèle basé sur le word embedding en introduisant la technique de clusterring.

Pour se faire, nous avons organisé notre mémoire en trois (03) chapitres :

Le premiers chapitre aborde les concepts de base de la recherche d'information tout en présentons les techniques d'indexation, les différents modèles de RI, la pondération ainsi que l'évaluation des performances d'un SRI.

Le deuxième chapitre présente un aperçu sur les réseaux de neurones, la technique du word embbeding, basée sur ces derniers, ses types et son utilisation en RI.

Le dernier chapitre est consacré à la présentation et l'implémentation de notre approche et les résultats de son évaluation.

Mots clefs : word embedding, clustering, modéle de langue, approche de zuccon.

Introduction générale

La recherche d'information est un ensemble de méthodes, et techniques pour l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information pertinente pour un utilisateur.

Aujourd'hui, avec l'explosion du volume d'informations disponible sous des formats hétérogènes produites par des sources d'informations distribuées ainsi qu'une multiplication des utilisateurs, la recherche d'information est devenue une tâche fastidieuse. Plusieurs outils d'accès à l'information (moteur de recherches, système de recherche d'informations(SRI)) ont été développés pour aider l'utilisateur dans cette tâche .Ces outils sont révélés efficaces dans la recherche d'information, mais ils sont encore principalement basés sur la correspondance exacte entre les termes de la requête et ceux des documents. Comme les requêtes sont généralement laconiques, et que les documents pertinents peuvent utiliser un vocabulaire différent, ces outils peuvent etre améliorés dans ce sens.

Idéalement, l'identification des documents pertinents devrait être basée sur la correspondance sémantique plutôt que sur la correspondance exacte des mots clés. Une technique dont les chercheurs ont utilisé pour remédier à ce problème est le word embedding. Mais l'utilisation de cette nouvelle technique ne réduit pas l'effort de la localisation des informations.

Pour remédier à ce problème de nombreuses approches ont été proposées, la plupart utilisant les relations existant entre les documents retournés pour orienter l'utilisateur dans sa recherche. Dans ce contexte, les techniques de clustering ont été largement employées afin de faciliter l'accès à l'information en regroupant les résultats aux thématiques similaires.

En combinant la technique de word embedding avec le clustering, on obtient de meilleurs résultats en réponse à la requête d'un utilisateur, l'approche que nous proposons et testons dans ce mémoire se base sur cette combinaison.

Dans notre projet, nous nous intéresserons à la technique du word embedding, précisément l'extension du modèle de recherche d'information en se basant sur le word embedding, et en intégrant la notion du clustering.

Pour ce faire nous allons organiser notre mémoire en trois (03) chapitres :

Le premiers chapitre aborde les concepts de base de la recherche d'information tout en présentons les techniques d'indexation, les différents modèles de RI, la pondération ainsi que l'évaluation des performances d'un SRI.

Le deuxième chapitre présente un aperçu sur les réseaux de neurones, la technique du word embbeding, basée sur ces derniers, ses types et son utilisation en RI.

Le dernier chapitre est consacré à la présentation et l'implémentation de notre approche et les résultats de son évaluation.

I.1 Introduction

Vu les grandes masses d'information qui se trouve aujourd'hui, l'utilisateur a de plus en plus des difficultés pour accéder à son besoin informationnel.

La recherche d'information (RI) traite de la représentation, du stockage, de l'organisation, et de l'accès à l'information. Le but d'un système de recherche d'information (SRI) est de retrouver, parmi une collection de documents préalablement stockées, les documents qui répondent au besoin de l'utilisateur exprimé sous forme de requête.

Dans ce chapitre nous présentons les concepts de la RI, en particulier en décrivant les notions du document, de requête, et de pertinence ainsi que les systèmes de recherche d'informations (SRI), les modèles de la RI et enfin l'évaluation des SRI.

I.2 Définition de la recherche d'information

Plusieurs définitions ont été attribuées à la recherche d'information, nous citons dans ce contexte les deux définitions suivantes :

Définition1 : La recherche d'information (RI) est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche dans une masse documentaire existante, les documents contenant l'information qui répond au besoin informationnel exprimé par l'utilisateur [1].

Définition 2 : La recherche d'information est une discipline de recherche qui intègre des modèles, et des techniques dont le but est de faciliter l'accès à l'information pertinente pour un utilisateur ayant un besoin en information [2].

I.3 Concept de base de la recherche d'information(RI)

La recherche d'information (RI) est un ensemble de méthodes, et de procédures ayant pour objectifs d'extraire à partir d'une collection de documents les informations qui répondent aux besoins informationnels exprimés par l'utilisateur sous forme de requête.

A partir de cette définition plusieurs concepts clés peuvent être définis, nous allons les clarifier dans ce qui suit :

I.3.1 Document et collection de documents

-Document : un support physique de l'information, qui peut être du texte, une page web, une image, une séquence vidéo, etc [3].

-Collection de documents : est l'ensemble des documents manipulés par un SRI (ou base documentaire ou encore corpus) [3].

I.3.2 Requête

Une requête constitue l'expression, dans un langage de requête, du besoin en information de l'utilisateur, une requête est un ensemble de mots clés, et elle peut être exprimée en langage naturel, booléen ou graphique [4].

I.3.3 La pertinence

La pertinence est un élément crucial et fondamental dans le domaine RI, elle peut être définie comme une mesure d'informativité du document à la requête, ou encore la correspondance entre un document et une requête [5].

Il existe deux types de pertinence :

-La pertinence système : c'est l'évaluation par le système de recherche d'information de l'adéquation entre les documents et une requête [6].

-La pertinence utilisateur: c'est l'évaluation par l'utilisateur de la pertinence des documents retrouvés par le système de recherche d'information (SRI) [6].

I.4 Définition du système de recherche d'information(SRI)

Un Système de Recherche d'Informations (SRI) est un ensemble de programmes informatiques qui permet de retourner à partir d'une collection de documents, ceux dont le contenu correspond le mieux à un besoin en informations d'un utilisateur, exprimé à l'aide d'une requête [7].

I.5 Le processus de recherche d'information

La figure suivante représente les différentes étapes de la RI dans un système de recherche d'informations (SRI) :

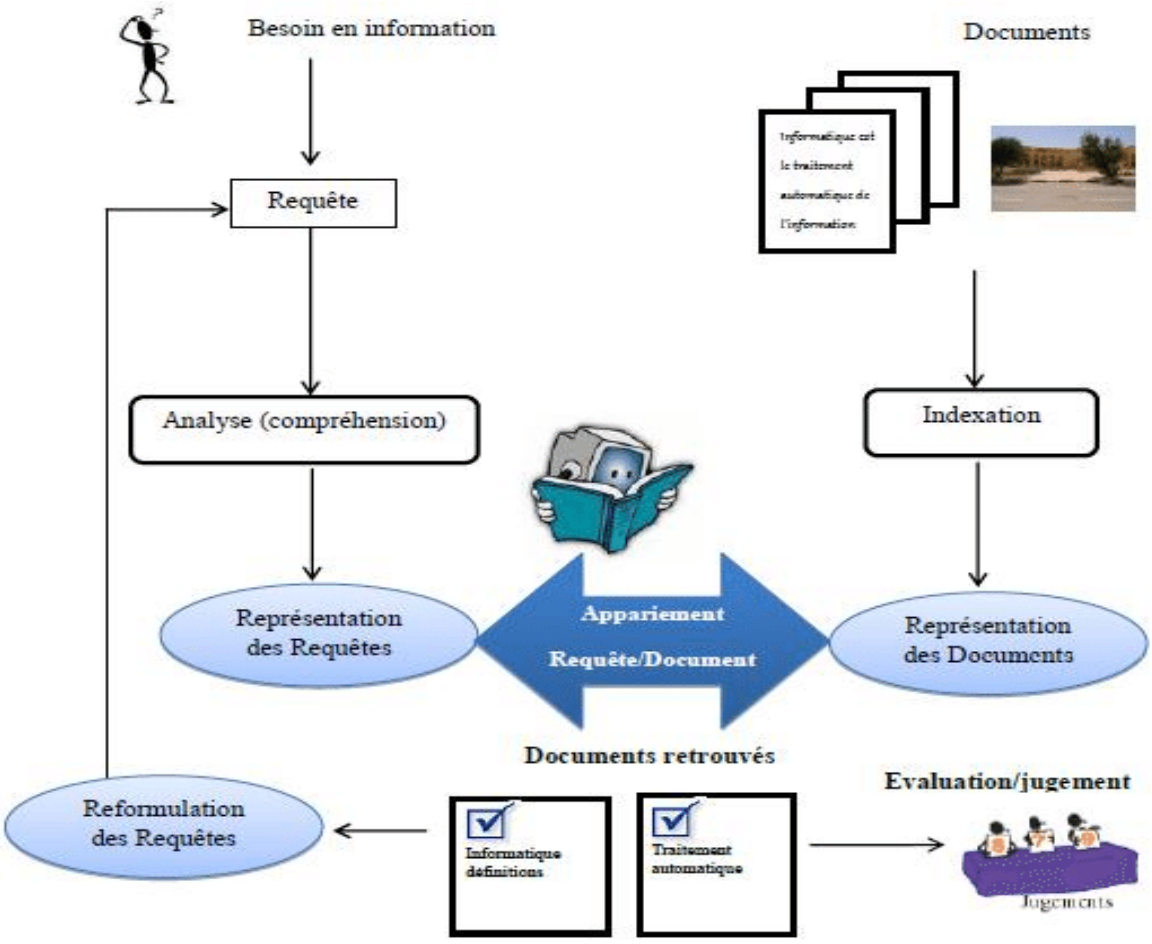


Figure I.1 -Processus de la recherche d'information-

A travers ce processus on a identifié les processus suivant : l'indexation, l'appariement, et la reformulation de la requête, que nous allons détailler ci-dessous :

I.5.1 L'Indexation

L'indexation est un processus qui transforme les documents en substituts appelées aussi descripteurs capables de représenter leurs contenus sémantiques, elle a pour objectif de créer une représentation interne (l'index) des documents en vue de faciliter la recherche. On trouve trois(3) types d'indexation: manuelle, semi-automatique et automatique.

-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 une 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 et le prix qu'elle exige (en temps et en nombre de personnes). De plus cette indexation est subjective, elle est liés aux facteurs humains, différents spécialistes peuvent indexer un document avec des termes différents.
Pratiquement inapplicable aux corpus de textes volumineux [4].

-Indexation semi-automatique: L'indexation est réalisée par un programme informatique et un indexeur humain. Le choix final des termes d'indexation à partir du vocabulaire fourni, est laissé à l'indexeur humain (généralement spécialiste du domaine) [4].

-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, elle est particulièrement adaptée aux corpus volumineux, elle passe par un ensemble d'étapes pour créer l'index d'une façon automatique [4].

Le schéma suivant détaille le processus d'indexation automatique :

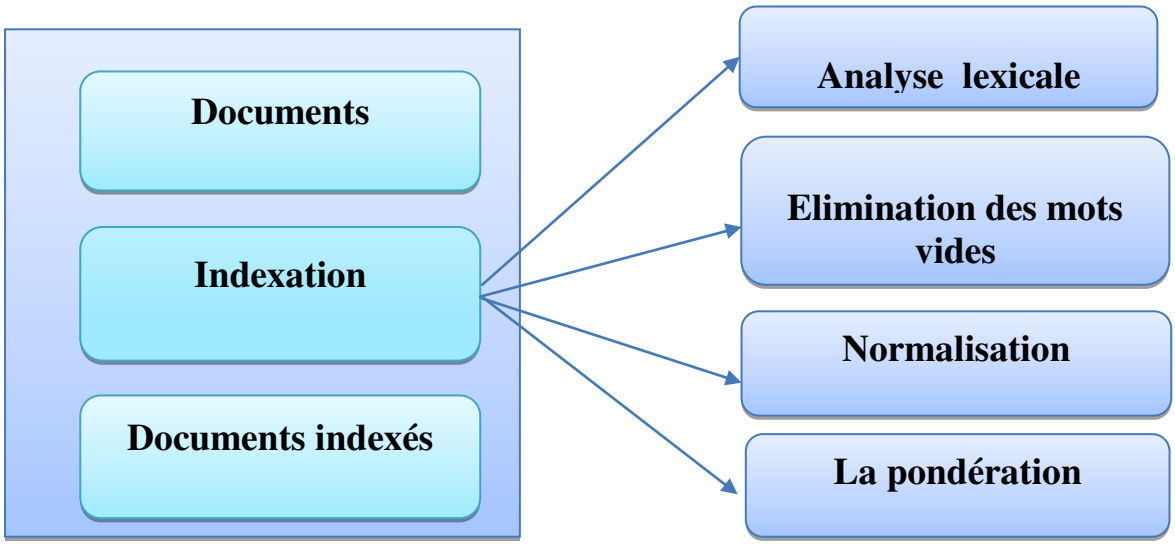


Figure I.2 -Le processus de l'indexation automatique-

Les étapes de l'indexation automatique définies dans la figure I.1 sont détaillées ci-dessous :

- **L'analyse lexicale (tokenization en anglais) :**

Elle permet de convertir un texte de document en liste de token, ce dernier est un groupe de caractères constituant un mot significatif dans un document.

L'analyse lexicale inclut les étapes suivantes :

-La conversion de la casse (majuscules en minuscules).

-L'élimination des accents.

-La tokénisation : Elle consiste à découper le document en un ensemble d'unité lexicale(mot ou token). Chaque unité lexicale ou un radical est une séquence de caractères entourées par des séparateurs d'unités. Elle permet alors de reconnaitre les espaces, les chiffres, les ponctuations, etc[8].

- **L'élimination des mots vides :**

Cette étape consiste à identifier les mots vides, ou mots qui n'apportent pas de sens au texte (prépositions, déterminants, adverbes, conjonctions, etc.), et les éliminer, on distingue deux techniques pour l'élimination des mots vides :

-L'utilisation d'une liste des mots vides (aussi appelée anti-dictionnaire ou stoplist).

-L'utilisation des mesures statistiques : éliminer des mots dépassant un certain nombre d'occurrences dans la collection ou les mots rares de la collection [8].

- **La normalisation des termes :**

L'objectif est de ramener les mots de la même famille à leur forme normale, la normalisation s'applique dans les documents, et dans la requête, plusieurs traitements de normalisation existent dont principalement :

-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, suffixes). Elle permet de substituer chaque mot par sa racine (lemme), qui est soit 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 adverbe.

-La racinisation (ou troncature): son principe consiste à construire une racine commune à un ensemble de mots de la même famille en tronquant les mots en questions à partir d'un certain rang avec la suppression des préfixes et des suffixes [8].

(Ex : cheval, chevaux, chevalier, chevalerie, chevaucher→"cheva"(mais pas "cavalier")).

- **La pondération :** Est une fonction fondamentale en RI. Tous les modèles de recherche se basent sur la pondération des termes.

L'idée 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 un document, plus son poids dans ce document doit être élevé).

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

Notant:

$tf_{t,d}$(*term frequency*): 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 estfréquent dans un document, plus il est important dans la description de ce document.

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 [8].

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

$$W_{t,d}=tf_{t,d} * idf_t \qquad \textbf{I.1}$$

I.5.2 L'appariement document-requête

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

Ce score de similarité entre le document et la requête est donné par une fonction nommée *Retrieval Status Value* notée RSV (d,q), où **d** est un document et **q** est une requête.
Traditionnellement le système de recherche retourne à l'utilisateur une liste de documents classés par RSV.

Cette fonction est fondamentale pour la RI car c'est elle qui détermine comment comparer la requête aux documents indexés. Le processus d'indexation et la fonction d'appariement constituent les deux éléments essentiels du modèle de recherche [9].
Il existe deux(2) types d'appariement :
-Appariement exact: Le résultat est une liste de documents respectant exactement la requête spécifiée avec des critères précis. Les documents retournés ne sont pas triés.
-Appariement approché: Le résultat est une liste de documents censés être pertinents pour la requête. Les documents retournés sont triés selon un ordre de mesure. Cet ordre reflète le degré de pertinence document/requête.

I.5.3 La reformulation de la requête

La qualité d'un SRI dépend de sa capacité à retrouver des documents pertinents pour l'utilisateur, elle est donc liée à l'indexation (au choix des termes par le système), et au choix des termes que l'utilisateur a fait pour formuler sa requête.
C'est pour résoudre les problèmes liés à un mauvais choix de termes, et pour pallier les lacunes de l'indexation qu'ont été introduits les mécanismes de reformulation de requête.
La reformulation consiste à réajuster les poids des termes de la requête ou à rajouter des termes reliés à ceux de la requête initiale, cette reformulation peut être réalisé par l'utilisateur, dans ce cas elle est dite manuel, ou par le système (dite automatique), comme elle peut être réalisé conjointement par l'utilisateur et le système, dans ce cas elle est dite semi-automatique [9].
Nous présentons dans ce qui suit les principales techniques de la reformulation :

I.5.3.1 La réinjection de pertinence(ou relevance feedback)

L'information sur la pertinence des documents retournés en réponse à la requête initiale est utilisée pour l'améliorer en se basant sur les deux types de réinjection de pertinence existants :

- **Réinjection de pertinence explicite :**

L'idée est de faire participer l'utilisateur dans le processus de recherche de sorte à améliorer l'ensemble final des résultats. Le procédé de base est le suivant :

-L'utilisateur formule sa requête.

-Le système lui renvoie un premier ensemble de résultats de recherche.

-L'utilisateur marque quelques documents retournés comme pertinents ou non pertinents (en pratique, seuls les 10 (ou 20) premiers documents classés sont examinés).

-L'idée principale consiste à sélectionner les termes importants des documents considérés comme pertinents, et améliorer l'importance de ces termes dans la nouvelle formulation de requête.

-Une nouvelle requête sera reformulée par le système à partir de la requête initiale en utilisant l'information sur la pertinence (nouveaux termes sélectionnés).

-La nouvelle requête est obtenue à partir de la requête initiale en appliquant un algorithme spécifique par exemple l'algorithme de Rocchio qui a été proposé pour la réinjection de pertinence dans le modèle vectoriel [4].

La formule de Rocchio pour la réinjection de pertinence est donnée comme suit :

$$Q_{m=}\ \alpha Q_0 + \beta \frac{1}{|R|} \sum_{d_p \in R} d_p - \gamma \frac{1}{|NR|} \sum_{dnp \in NR} d_{np} \qquad \textbf{I.2}$$

Où

$\boldsymbol{Q_m}$: Le vecteur de la nouvelle requête reformulée.

$\boldsymbol{Q_0}$: Le vecteur de la requête initiale.

$\boldsymbol{d_p}$(respectivement$\boldsymbol{d_{np}}$): Le vecteur associé à un document pertinent retrouvé (respectivement non pertinent).

R : C'est l'ensemble des documents pertinents de la collection.

NR : C'est l'ensemble des documents non pertinents de la collection.

α, β, γ : C'est des constantes telles que α + β + γ =1.

- **Réinjection de pertinence implicite (pseudo-réinjection de pertinence) :**

L'idée est d'utiliser les résultats de la recherche en vue d'améliorer les résultats du SRI sans l'intervention de l'utilisateur.

On suppose uniquement que les tops **m** documents retournés sont considérés comme pertinents, et on les utilise pour reformuler la requête initiale [4].

La variante de la formule de Rocchio pour la réinjection de pertinence explicite est exprimée par la formule suivante :

$$Q_{m=}\alpha Q_0 + \beta \frac{1}{|R|}\sum_{d_p \in R} d_p \qquad \textbf{I.3}$$

Où

$\boldsymbol{Q_m}$: Le vecteur de la nouvelle requête reformulée.

$\boldsymbol{Q_0}$: Le vecteur de la requête initiale.

$\boldsymbol{d_p}$: Le vecteur associé à un document pertinent retrouvé (respectivement non pertinent).

R : C'est l'ensemble des documents pertinents de la collection.

α et **β** sont des constantes.

I.6 Les modèles de recherche d'informations

Un modèle de RI a pour rôle de fournir une formalisation du processus de recherche, il permet de créer une représentation interne pour un document ou pour une requête basée sur des termes pondérés issus de l'indexation, ainsi il permet de définir une méthode de comparaison entre une représentation du document, et celle de la requête afin de détecter leurs degrés de correspondance(ou de similarité).

Les modèles de recherche sont classés en trois classes principales : le modèle booléen, les modèles vectoriels et les modèles probabilistes.

I.6.1 Le modèle booléen

Le modèle booléen est basé sur la théorie des ensembles. Dans ce modèle, les documents et les requêtes sont représentés par des ensembles de mots clés.

Chaque document est représenté par une conjonction logique des termes non pondérés qui constitue l'index du document. Un exemple de représentation d'un document est comme suit : d = t1∧t2∧t3...∧ tn.

Une requête est une expression booléenne dont les termes sont reliés par des opérateurs logiques (OR, AND, NOT) permettant d'effectuer des opérations d'union, d'intersection, et de différence entre les ensembles de résultats associés à chaque terme.

Un exemple de représentation d'une requête est comme suit : q = (t1∧t2) ∨ (t3∧t4).

La fonction de correspondance est basée sur l'hypothèse de présence/absence des termes de la requête dans le document, et vérifie si l'index de chaque document dj implique l'expression logique de la requête q [7].

Le résultat de cette fonction est donc binaire est décrit comme suit :

$$RSV(d,q)=\begin{cases}1 & Si\ d\ appartient\ a\ l'ensemble\ décrit\ par\ q \\ 0 & Sinon\end{cases} \qquad \textbf{I.4}$$

I.6.2 Le modèle vectoriel

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

Si on a un espace **T** de termes d'indexation de dimension **n**, T=$\{t_1, t_2, \dots, t_n\}$. Un document $\boldsymbol{d_i}$ est représenté par le vecteur$d_i(w_{t1}, w_{t2}, \dots, w_{tn})$. Une requête **q** par le vecteur $q(w_{q1}, w_{q2}, \dots, w_{qn})$.

Où $\boldsymbol{w_{qj}}$ est le poids de terme $\boldsymbol{t_j}$ dans la requête **q**, et $\boldsymbol{w_{ij}}$ son poids dans le document $\boldsymbol{d_i}$, ce poids peut être soit une forme de $tf_{t,d}$ * idf_tsoit un poids attribué manuellement par l'utilisateur [3].

Le modèle vectoriel offre des moyens pour la prise en compte du poids du terme dans le document. Plusieurs schémas de pondération ont été proposés qui sont la pondération locale et la pondération globale.

La pondération locale permet de mesurer l'importance du terme dans le document, elle correspond à une fonction de fréquence d'occurrence du terme dans le document, exprimée ainsi :

$$tf_{t,d} = 1 + \log(f(t_i, d_j)) \qquad \textbf{I.5}$$

Où $f(t_i, d_j)$ est la fréquence du terme $\boldsymbol{t_i}$ dans le document $\boldsymbol{d_j}$.

La pondération globale **idf** dépend de la fréquence document du terme est exprimée comme suit

$$\text{idf} = log(\frac{N}{n_i}) \qquad \textbf{I.6}$$

Où

$\boldsymbol{n_i}$ est la fréquence en document du terme considéré. et **N** est le nombre total des documents de la collection.

La pertinence du document $\boldsymbol{d_i}$ pour la requête **Q** est mesurée comme le degré de corrélation des vecteurs correspondants.

Cette corrélation peut être exprimée par l'une des mesures présentées dans le tableau suivant :

Mesures	**Formules**
Le produit scalaire	Score(q,d_i)=$\sum_{j=1}^{\|T\|} w_{qj} * w_{ij}$
La mesure de cosinus	Score(q,d_i)=$\frac{\sum_{j=1}^{\|T\|} w_{qj} * w_{ij}}{\sqrt{\sum_{j=1}^{\|T\|} {w_{qj}}^2 * \sum_{j=1}^{\|T\|} {w_{ij}}^2}}$
La mesure de Dice	Score(q,d_i)=$\frac{2 * \sum_{j=1}^{\|T\|} w_{qj} * w_{ij}}{\sum_{j=1}^{\|T\|} {w_{qj}}^2 + \sum_{j=1}^{\|T\|} {w_{ij}}^2}$
La mesure de Jaccard	Score(q,d_i)=$\frac{\sum_{j=1}^{\|T\|} w_{qj} * w_{ij}}{\sum_{j=1}^{\|T\|} {w_{qj}}^2 + \sum_{j=1}^{\|T\|} w_{ij}^2 - \sum_{j=1}^{\|T\|} w_{qj} * w_{ij}}$

Tableau I.1- Les mesures de similarité utilisée dans le modèle vectoriel-

I.6.3 Les modèles probabilistes

Le principe du modèle probabiliste 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.

I.6.3.1 Modèle probabiliste de base

Ce modèle est fondé sur le calcul de la probabilité de pertinence d'un document pour une requête.

Le principe de base consiste à retrouver des documents qui ont en même temps une forte probabilité d'être pertinents, et une faible probabilité d'être non pertinents.

Soit **Q** une requête utilisateur et **d** un document, le modèle probabiliste tente d'estimer la probabilité que le document **d** appartient à la classe des documents pertinents (ou non pertinents) [7].

Un document est sélectionné si la probabilité qu'il soit pertinent pour **Q** notée **P(R|d)** est supérieure à la probabilité qu'il soit non pertinent pour **Q**, notée **P(NR|d)**.

RSV (d,Q) est donné par :

$$\text{RSV (d,Q)} = \frac{\text{P(R|d)}}{\text{P(NR|d)}} \qquad \textbf{I.7}$$

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

$$\text{RSV (d,Q)} = \frac{P(d|R)}{P(d|NR)} \qquad \textbf{I.8}$$

P(d|R) (respectivement P(d|NR)) est la probabilité que le document appartient à l'ensemble **R** des documents pertinents (respectivement à l'ensemble **NR** des documents non pertinents).

Il existe d'autres méthodes pour estimer ces probabilités comme dans le modèle BIR(*Binary Independance Retrieval*).

Les probabilités P (d|R) et P(d|NR) sont données par:

$P(d|R)=p(t_1 = x_1, t_2 = x_2, t_3 = x_3, \dots |R)=\prod_i p\ (t_i = x_i|R)=\prod_i p\ (t_i = 1\ |R)^{x_i} *$

$p(t_i = 0|R)^{1-x_i}$ **I.9**

$P(d|NR)=p(t_1 = x_1, t_2 = x_2, t_3 = x_3, \dots |NR)=\prod_i p\ (t_i = x_i|NR)=\prod_i p\ (t_i = 1\ |NR)^{x_i} *$

$p(t_i = 0|NR)^{1-x_i}$ **I.10**

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

I.6.3.2 Le modèle de langue

Le modèle de langue est un modèle probabiliste, inspiré des techniques statistiques de modélisation de langue en informatique linguistique [3].

L'utilisation des modèles de langue en RI remonte à 1998. Le principe de ce modèle consiste à construire un modèle de langue $\boldsymbol{M_d}$ pour chaque document **d**, puis de calculer la probabilité qu'une requête **q** puisse être générée par le modèle de langue du document, soit $P(q|M_d)$ [5].

Cette probabilité est alors exprimée comme suit :

$$RSV(d,q)=P(q=(t_1,t_2, \dots, t_n)|M_d)=\prod_i P(t_i\ |M_d) \qquad \textbf{I.11}$$

$P(t_i|M_d)$ est donnée par :

$$P(t_i|M_d)=\frac{tf(t_i,d)}{\sum tf(t,d)} \qquad \textbf{I.12}$$

Où **tf(t|d)** est la fréquence d'occurrence du n-gramme (du mot) $\boldsymbol{t_i}$ dans le document **d** et $\boldsymbol{\sum tf(t,d)}$ correspond à la taille du document .

Pour remédier au problème posé par des mots de la requête absents dans le document, qui ont pour effet d'avoir la probabilité P(q|M_d) nulle, des techniques de lissage (smoothing) sont utilisées.

Le lissage consiste à assigner des probabilités non nulles aux termes, qui n'apparaissent pas dans les documents [10].

Il existe différentes techniques de lissage qu'on va présenter ci-dessous :

-Lissage de Laplace :

Cette méthode consiste à ajouter la fréquence un(1) à tous les termes, appelé aussi ajouter-un.

La probabilité de cette technique est exprimée ainsi :

$$P_{add-one}\ (t|M_d)=\frac{tf(t,d)+1}{\sum tf(t,d)+1} \qquad \textbf{I.13}$$

-Lissage de Good-Turing :

Cette méthode permet l'ajustement de la fréquence **tf** d'un terme en une fréquence **tf*** dite corrigée exprimée ainsi :

$$\text{Tf*= (tf+1)*}\frac{n_{t+1}}{n_t} \qquad \textbf{I.14}$$

Où $\boldsymbol{n_t}$ est le nombre de termes de fréquence **t**.

Ainsi pour tout terme l'estimation de sa probabilité devient alors :

$$P_{GT}\ (t|M_d)=\frac{tf^*(t,d)+1}{\sum tf^*(t,d)+1} \qquad \textbf{I.15}$$

-Lissage par interpolation (Jelinek-Mercer) :

Ce type de lissage consiste à combiner le modèle de langue considéré avec un ou plusieurs modèles de références, dans le cas de collection de documents on pourrait par exemple d'estimer le modèle de documents en le combinant avec le modèle de la collection.

Le modèle de documents est exprimé ainsi :

$$P_{JM}(t_i|d) = (1\text{-}\beta)P_{ML}(t_i|d) + \beta P_{ML}(t_i|C) \qquad \textbf{I.16}$$

Les modèles $P_{ML}(t_i|d)$et $P_{JM}(t_i|C)$ sont estimés selon le maximum de vraisemblance.

-Lissage par interpolation (Dirichlet) :

Ce lissage se base sur le lissage précédent en exploitant la valeur de **β** en fonction de la taille de l'échantillon. Dans ce cas la formule s'écrit comme suit :

$$P_{Dir}(t_i|d) = \frac{|d|}{|d|+\mu} P_{ML}(t_i|d) + \frac{\mu}{|d|+\mu} P_{ML}(t_i|C)$$

$$= \frac{|d| * P_{ML}(t_i|d) + \mu * P_{ML}(t_i|C)}{|d|+\mu}$$

$$= \frac{tf(t_i|d) + \mu * P_{ML}(t_i|C)}{|d|+\mu} \qquad \textbf{I.17}$$

Avec $P_{ML}(t_i|C) = \frac{tf(t_i|C)}{|C|}$

Où |d| est la taille du document.

$tf(t_i|d)$est la fréquence du terme $\mathbf{t_i}$ dans le document **d**, $\boldsymbol{\mu}$ un paramètre appelé pseudo fréquence.

I.7 Evaluation des SRI

L'évaluation constitue une étape importante dans la mise en œuvre d'un SRI, elle permet de mesurer les caractéristiques du système en termes de qualité de service, et de facilité d'utilisation, et d'évaluer la performance du système.

Le but d'un SRI est de retrouver l'information recherchée par l'utilisateur (ou information pertinente) et de la lui retourner dans un délai acceptable, en la lui présentant sous une forme aisément exploitable. Ceci implique la capacité du système à retrouver les documents pertinents d'une part, et la manière de présenter les résultats d'autre part.

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

Le principal objectif d'un système de recherche d'information est de restituer à l'utilisateur tous les documents pertinents et de rejeter tous les documents non pertinents.

Cet objectif est évalué à l'aide de différentes mesures d'évaluation. On présente ci-dessous les plus utilisées :

I.7.1.1 La précision et le rappel

Le rappel et la précision sont deux mesures de base pour évaluer les performances des systèmes de recherche d'information. Le principe de ces deux mesures est basé sur la connaissance à priori des documents pertinents de la collection d'une part, et d'autre part la partition de l'ensemble des documents restitués par le SRI en deux catégories : documents pertinents et documents non pertinents.

La figure suivante illustre la partition de la collection de tests pour une requête :

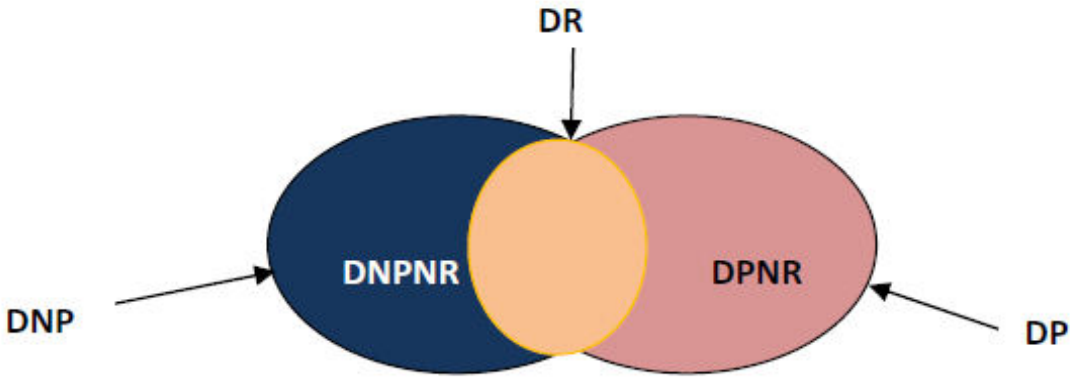


Figure I.3-Répartition des documents d'une collection suite à une requête-

DP : Documents pertinents

DNP : Documents non pertinents

DR : Documents retrouvés

DPNR : Documents pertinents non retrouvés

DNPNR : Documents non pertinents non retrouvés

-Le rappel : C'est la capacité du SRI à retrouver les documents pertinents de la collection, il mesure la proportion de documents pertinents restitués parmi tous les documents pertinents disponibles. Si le rappel vaut 1 c'est que les documents pertinents disponibles ont tous été restitués par le système, inversement si le rappel vaut 0 c'est

qu'aucun document pertinent n'a été restitué. Cette mesure permet aussi de déterminer le silence, c'est-à-dire la proportion de documents pertinents non trouvés.

Le rappel est donné par la proportion suivante :

$$\text{Rappel}=\frac{Le\ nombre\ de\ documents\ perinents\ retrouvés}{Le\ nombre\ de\ documents\ pertinents}=\frac{|DPR|}{|DP|} \quad \textbf{I.18}$$

-La précision : Elle calcule la capacité du SRI à retrouver uniquement les documents pertinents. La précision permet de mesurer la fraction des documents pertinents parmi ceux qui ont été retrouvés par le système.

Une précision égale à 1 signifie que le système n'a retrouvé que des documents pertinents.

La précision est donnée par la proportion suivante :

$$\text{Précision}=\frac{Le\ nombre\ de\ documents\ perinents\ retrouvés}{Le\ nombre\ de\ documents\ retrouvés}=\frac{|DPR|}{|DR|} \quad \textbf{I.19}$$

Les deux métriques ne sont pas indépendantes, quand l'une augmente, l'autre diminue.

Il est facile d'avoir 100% de rappel, il suffit de donner tous les documents disponibles comme réponse à chaque requête. Cependant, la précision dans ce cas serait très basse. De même, il est possible d'augmenter la précision en donnant très peu de documents en réponse, mais le rappel en souffrira. Il faut donc utiliser les deux métriques conjointement.

-Courbe Rappel/Précision : Les différentes valeurs de précision aux différents points de rappel servent à tracer la courbe Rappel/précision qui a en général l'aspect suivant comme le représente la figure ci-dessous.

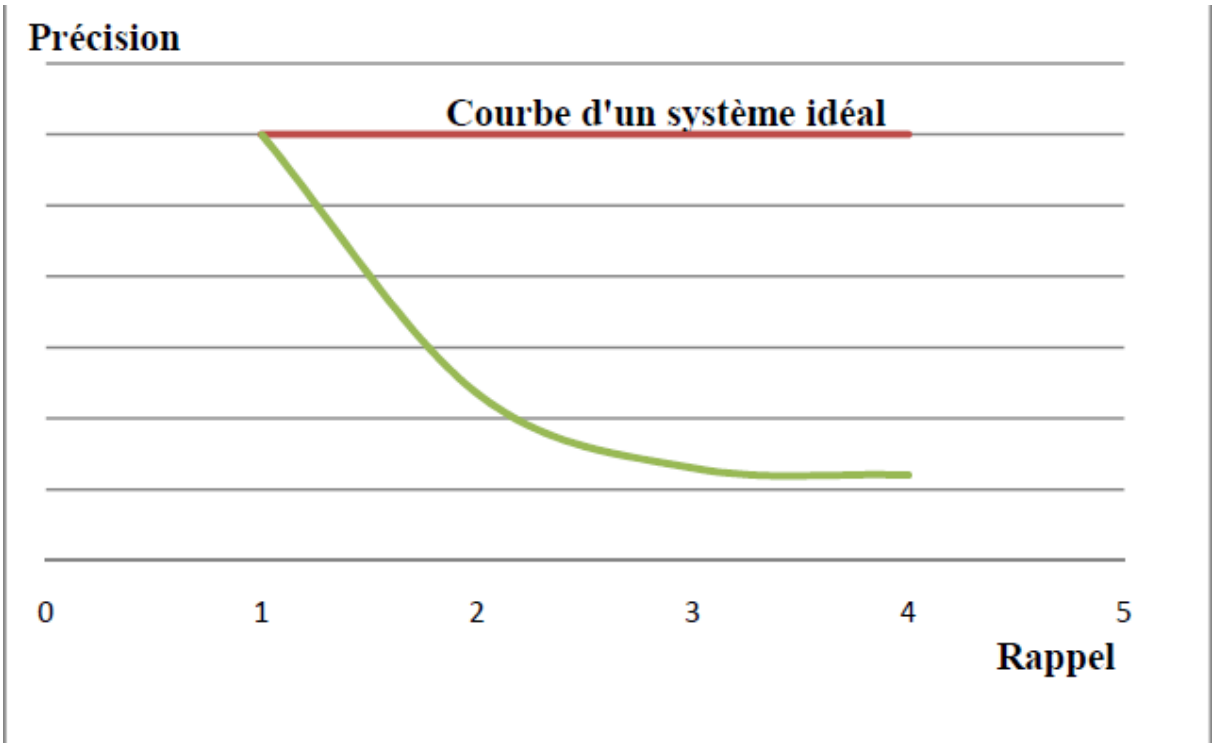


Figure I.4 –courbe rappel précision-

I.7.1.2 Bruit et silence:

Ce sont d'autres mesures complémentaires au Rappel et précision.

$$\text{Bruit : B}=\frac{Le\ nombre\ de\ documents\ non\ perinents\ retrouvés}{Le\ nombre\ de\ documents\ retrouvés}=\frac{|DNPR|}{|\text{DR}|} \quad \textbf{I.20}$$

$$\text{Silence: S}=\frac{Le\ nombre\ de\ documents\ non\ perinents\ retrouvés}{Le\ nombre\ de\ documents\ pertinents}=\frac{|DNPR|}{|DP|} \quad \textbf{I.21}$$

I.7.1.3 Précision moyenne (AVG-P), la MAP et la R-précision :

-La R-précision: La R-précision est la précision à n quand n est égal au nombre total de documents pertinents.

Cette mesure est plus réaliste pour l'étude de l'ordonnancement en tête de liste, mais pour l'obtenir, il est nécessaire de connaître au préalable le nombre de documents pertinents disponibles dans la collection pour une requête donnée.

La R-précision pour une requête **q** est la précision calculée au rang **R** (*Ranking*).

$$\text{R\_prec= P@R} = \frac{|DPR|}{|R|} \qquad \textbf{I.22}$$

Où **R** est le nombre de documents pertinents pour la requête **q**.

- Précision moyenne (AVG-P) : c'est une valeur moyenne des valeurs de précision à chaque observation d'un document pertinent pour une requête donnée (AP_q), pour chaque document pertinent retrouvé on calcule sa précision (Pr(d_i)) comme suit :

$$\Pr(d_i) = \begin{cases} \frac{r_{ni}}{n_i} sid_i estretrouvé \\ 0 \qquad\qquad Sinon \end{cases} \qquad \textbf{I.23}$$

Le calcul de AP_q se fait comme suit :

$$AP_q = \frac{1}{N}\sum_{i=1}^{N} \Pr(d_i) \qquad \textbf{I.24}$$

$\boldsymbol{n_i}$:est le rang du document d_i qui a été retrouvé et qui est pertinent pour la requête.

$\boldsymbol{r_{ni}}$: est le nombre de documents pertinents retrouvé au rang n_i.

N : est le nombre total de documents pertinents pour la requête.

- La MAP : La précision moyenne est une mesure de performance globale. Elle est une moyenne de précision sur un ensemble de points de rappel, calculée pour un ensemble **M** de requêtes comme suit :

$$MAP = \frac{1}{M}\sum_{j=1}^{M} AP_{qj} \qquad \textbf{I.25}$$

I.7.2 Les collections de test

Les collections de test sont utilisées afin d'évaluer la performance du SRI, une collection de test est composée d'un corpus de documents, une liste de requêtes prédéfinies et des jugements de pertinences qui sont établis par des experts pour définir les documents pertinents.

Plusieurs collections de test ont été construites, parmi elles ADI (82 requêtes, 25 documents), CACM(3204 requêtes ,64 documents), CISI(1460 requêtes,112 documents), MED(1033 requêtes, 30 documents), TIME(425 requêtes,23) [11].

Les compagnes d'évaluation:

Les compagnes d'évaluation en RI travaillent pour évaluer le degré de pertinence des réponses du SRI à une requête en utilisant des collections de test. Chaque compagnie est

constituée d'un certain nombre de tâches fournissant des résultats, et un protocole d'évaluation pour chaque tâche. Les compagnies de TREC (*Text Retrieval Conference*) sont devenues la référence en ce qui concerne l'évaluation des SRI mais on peut également citer les compagnies CLEF (*Cross-Language Evaluation Forum*), NTCIR (*NII Tesbeds and Community for Information access Research*), Amarllis et INEX [1].

Conclusion:

Nous avons présenté dans ce premier chapitre les concepts fondamentaux de la recherche d'information. Le rôle d'un système de recherche d'information est de retrouver les informations pertinentes à partir d'une grande masse de données en passant par plusieurs étapes telle que l'indexation, la recherche et la reformulation pour arriver à répondre au besoin de l'utilisateur. Dans le chapitre suivant nous allons présenter une des techniques les plus récentes utilisées dans la recherche d'information qui est le "Word embedding".

II.1 Introduction

Nous allons présenter dans ce chapitre le Word Embedding qui est une nouvelle technique d'apprentissage basée sur les réseaux de neurones. Principalement nous présentons l'intégration de cette technique dans les modèles de recherche d'information, et ce qu'apporte comme améliorations de la performance en recherche d'information.

Nous organisons alors notre chapitre comme suit : dans la première section nous présentons un aperçu sur les réseaux de neurones, dans la deuxième section nous présentons le fonctionnement du word embedding.

II.2 Le modèle de sac de mots

Le modèle de sac de mots est un algorithme simple utilisé dans le traitement du langage naturel.

Ce type de représentation ne permet pas de capturer le sens véhiculé par les mots et les relations potentielles entre ces mots, et l'ordre dans lequel les termes sont présentés. Afin de capturer des représentations fines, une des approches qui pourraient être explorées et qui connaît un engouement considérable est l'exploitation d'apprentissage profond basée sur les réseaux de neurones.

II.3 Les réseaux de neurones formels

Un réseau de neurones est composé de neurones, reliés entre eux par des synapses. On commence par une couche d'entrée dans laquelle chaque neurone correspond à un vecteur. Puis une ou plusieurs couches de neurones cachées et enfin une couche de neurone de sortie [18].

II.3.1 Fonctionnement d'un neurone formel

Un neurone formel (artificiel) est une unité de traitement qui reçoit des données en entrée, sous la forme d'un vecteur, et produit une sortie réelle. Cette sortie est une fonction des entrées et des poids des connexions [19].

La **figure** suivante représente un exemple d'un neurone formel avec trois (3) entrées :

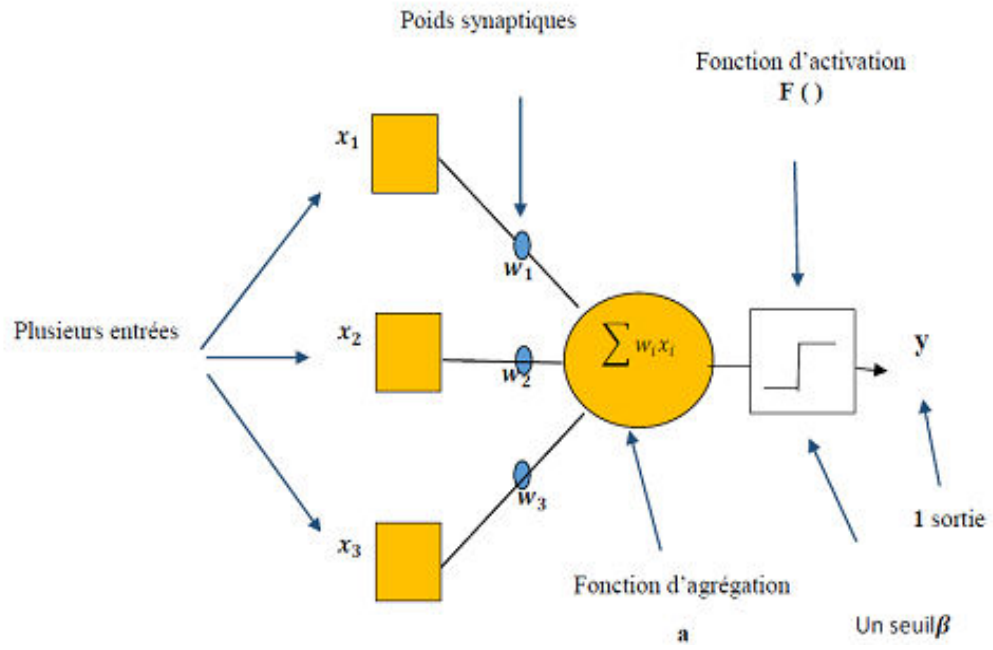


Figure II.1 -représente un exemple d'un neurone formel avec trois (3) entrées-

Les composants d'un neurone formel peuvent donc être définis de cette façon :

$x_1, x_2, x_3, \ldots, x_n$**:** constituent les valeurs d'entrées (valeurs scalaires).

$w_1, w_2, w_3, \ldots, w_n$: constituent la pondération de chaque flèches ou synapses.

a : constitue une valeur scalaire d'activation se calcule comme suit :

$$a=\sum(w_i, x_i) \qquad \text{II.1}$$

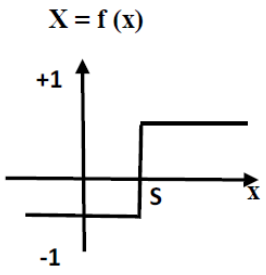
β: constitue le seuil (ou biais).

y : est la sortie.

F() : Fonction d'activation

C'est une fonction qui permet de définir l'état interne du neurone en fonction de son entrée totale, appelée aussi fonction de transfert.

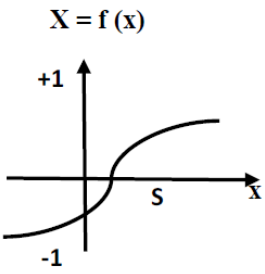
Les fonctions les plus souvent utilisées sont représentées ci-dessous :



a: fonction à seuil

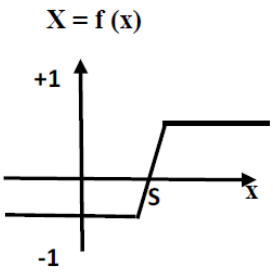
(**S** , la valeur du seuil)

$$f(x)=\begin{cases}\mathbf{0}\ \boldsymbol{if}\ \mathbf{0} > x \\ \mathbf{1}\ \boldsymbol{if}\ \boldsymbol{x}\ \geq \mathbf{0}\end{cases}$$



b: fonction sigmoïde

$$f(x) = \frac{1}{1+e^{-\beta x}}$$



C : fonction linéaire par morceaux.

$$f(x) = \begin{cases}\mathbf{0}\ \boldsymbol{if}\ \boldsymbol{x} \leq\ \boldsymbol{x_{min}} \\ \boldsymbol{mx} + \boldsymbol{b}\ \boldsymbol{if}\ \boldsymbol{x_{max}} > x >\ \boldsymbol{x_{min}} \\ \mathbf{1}\ \boldsymbol{if}\ \boldsymbol{x} \geq \boldsymbol{x_{max}}\end{cases}$$

II.3.2 Les types des réseaux de neurones

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle, les types de réseaux de neurones sont présentés ci-dessous :

II.3.2.1 Perceptron simple

Le perceptron est un type de réseau de neurone considéré comme le type de réseaux neuronal le plus simple, il se compose de deux couches de neurones : La première couche est composée de cellules d'entrée, la deuxième couche fournit la réponse, comme il dispose d'un unique neurone de sortie.

Il est utilisé pour la classification des données en deux catégories. L'impossibilité de traiter les problèmes non linéaires avec les réseaux de type Perceptron a été résolue par un réseau multicouche [22].

II.3.2.2 Réseau de neurone multicouche (ou singulier)

C'est un réseau entièrement connecté, il était le premier et le plus simple type de réseau de neurones artificiels conçu, les neurones sont arrangés par couche.

Il n'y a pas de connexion entre les neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches précédentes**.**

Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurone d'entrée, neurone de sortie [20].

Le schéma suivant présente l'architecture d'un réseau de neurone multicouche :

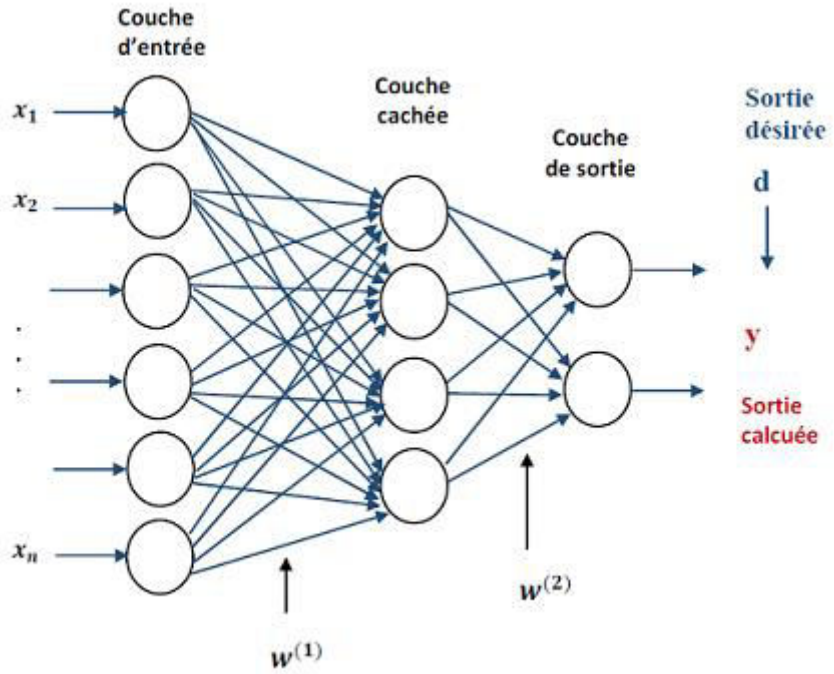


Figure II.2-Réseau de neurone multicouche-

II.3.2.3 Réseau à connexions récurrentes

C'est un réseau de neurones artificiels présentant des connexions récurrentes. Les neurones sont reliés par des arcs qui possèdent des poids. La sortie d'un neurone est une combinaison non linéaire de ses entrées [23].

Le schéma suivant présente l'architecture d'un réseau de neurone à connexion récurrente :

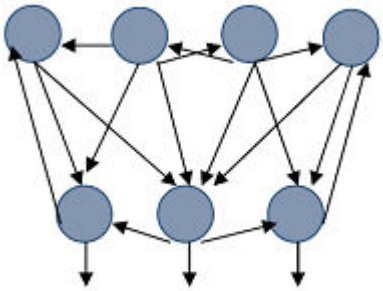


Figure II.3 –Réseau de neurones à connexion récurrentes-

II.4 Le word embedding

Les words embeddings permettent de représenter sous forme d'un vecteur chaque mot (ou terme) d'un corpus, en utilisant les termes qui apparaissent autour du mot, aussi appelés contexte. Cette représentation permet d'identifier les termes qui sont utilisés dans le même contexte [13].

Cette technique est couramment utilisée dans le domaine de traitement automatique du langage naturel [31].

II.5 Les Modèles du word embedding

II.5.1 Le modèle de langue neuronal

Ce modèle se base sur les réseaux de neurones, il consiste généralement à calculer la probabilité d'un mot $w_t$étant donné de ses n-1 mots précédents [19] selon l'équation suivante :

$$P(w_1, \dots, w_{i-1+n}) = \prod_i P(w_i | w_{i-1}, .., w_{i-1+n}) \quad \textbf{II.2}$$

Avec $P(w_t | w_{t-1}, .., w_{t-1+n}) = \frac{count(w_{t-1+n}, \dots, w_{t-1}, w_t)}{count(w_{t-1+n}, \dots, w_{t-1})}$ **II.3**

Ce modèle doit effectuer deux tâches : Premièrement, la projection des indices des mots du contexte dans un espace continu afin d'obtenir leurs représentation vectorielles. Ces représentations sont ensuite concaténées pour construire l'entrée de la couche cachée. Deuxièmement, la probabilité de chaque mot dans une liste de mots est calculée en sortie du réseau.

Le principe de ce modèle est exprimé dans le schéma suivant :

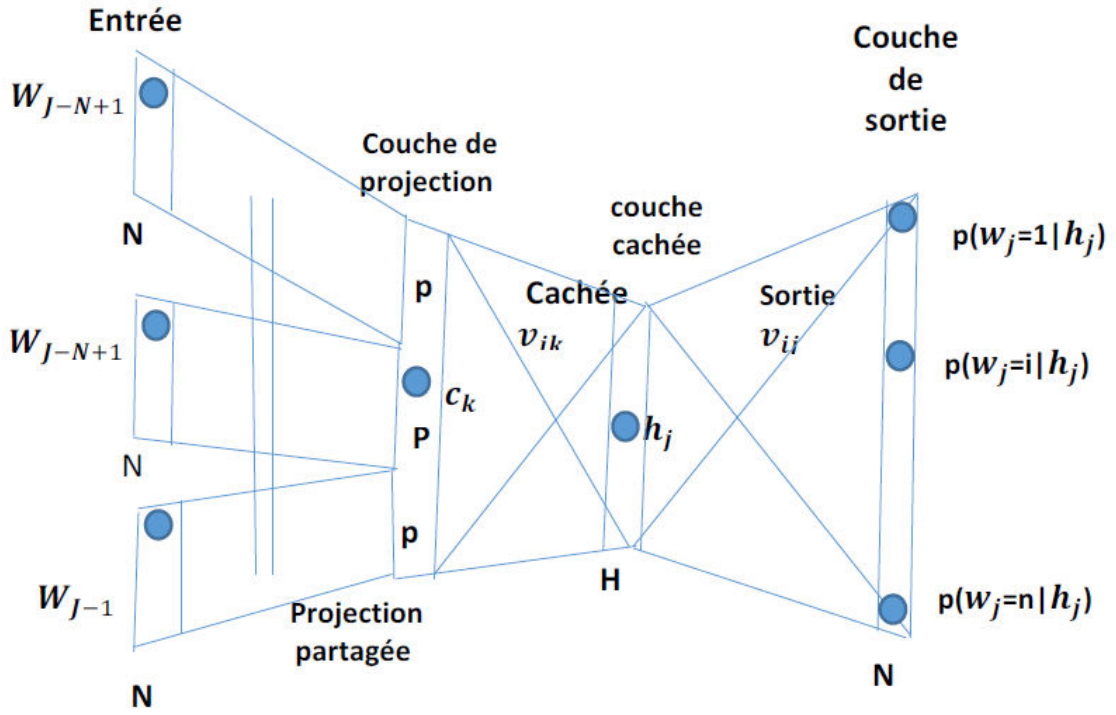


Figure II.4 -Le modèle de langue neuronal-

On calcule la probabilité de chaque mot **w** à la couche de sortie du réseau neuronal en utilisant la fonction softmax suivante :

$$P(w_t|w_{t-1},..,w_{t-1+n}) = \frac{\exp(h^T v' w_t)}{\sum_{w_{i\in v}} \exp(h^T v' w_t)} \qquad \textbf{II.4}$$

h est le vecteur de sortie de l'avant dernière couche du réseau (la couche cachée dans le réseau multicouches), tandis que **v'w** est l'embedding de sortie du mot **w**, c'est-à-dire sa représentation dans la matrice pondérale de la couche softmax. Notez que

même si **v'w** représente le mot **w**, il est appris séparément du word embedding d'entrée **vw**, car les multiplications dans lesquelles les deux vecteurs sont impliqués diffèrent (**vw** est multiplié par un vecteur d'index, **v'w** avec **h**).

II.5.2 Le modèle Collobert et Weston

Collobert et Weston (C&W) présentent une approche pour la modélisation neuronale du langage. Pour éviter le calcul de la couche de sortie sur tout le vocabulaire, ils ont proposé d'utiliser une autre fonction de coût pour l'apprentissage. En effet, au lieu d'utiliser l'entropie croisée, qui maximise la probabilité du prochain mot sachant les mots précédents, Collobert et Weston entraînent le modèle de langage à discriminer entre deux classes : si le mot du milieu en entrée du réseau est lié à son contexte ou non. Autrement dit, le réseau de neurones est entrainé pour discriminer les scores entre un mot correct et un autre "corrompu". La "corruption" consiste à remplacer aléatoirement le mot central par un autre mot du vocabulaire V [19]. L'entrainement du réseau de neurones consiste à minimiser la fonction de coût, sur tous les termes G dans le corpus, cette fonction est exprimée ainsi :

$$E(g)=\sum_{g\in G}\sum_{g'\in V}\max(0.1-S(g)+S(g')) \qquad \textbf{II.5}$$

Où **S()** est le score d'existence du terme, **g** est le terme correct choisi à partir d'un ensemble de tous les terme possibles dans le corpus **G**, **g'** est le terme "corrompu" choisi pour chaque terme **g**.

Ce modèle produit ainsi les embeddings comme résultat qui possèdent déjà une relation avec plusieurs word embeddings qui sont connues.

II.5.3 Le modèle Glove (Global Vectors for Word Representation)

GloVe est un modèle qui permet de construire des words embeddings. Il sert à effectuer des calculs de cooccurrence de mots à partir d'un corpus, et les représentations résultantes présentent des sous-structures linéaires intéressantes de l'espace vectoriel (word embeddings).

La formule suivante montre le principe de ce modèle :

$$J=\sum_{i,j=1}^{V} f(X_{ij})(w_i^T w_j^{\sim} + b_i + b_j - \log(X_{ij})^2 \qquad \textbf{II.6}$$

Où $w_i$et b_i sont les vecteurs du mot et biais respectivement du mot **i**.

$\boldsymbol{w_j^{\sim}}$ et $\boldsymbol{b_j}$sont les vecteurs de mot de contexte et biais respectivement du mot **j**,

$\boldsymbol{X_{ij}}$est le nombre de fois que le mot **i** se produit dans le contexte du mot **j,**

f est une fonction de pondération qui attribue un poids relativement inférieur aux cooccurrences rares et fréquentes.

GloVe prend une matrice plutôt que l'ensemble du corpus comme entrée [27].

II.5.4 Le modèle Word2vec

Ce modèle permet de construire des représentations continues des mots d'un texte ou d'une séquence, en se basant sur la notion de contexte auquel appartient le mot. Il permet de construire des représentations vectorielles (word embedding) permettant de capturer la sémantique exprimée par chacune des occurrences. Par la suite, ces représentations sont utilisées pour comparer les termes entre eux et exprimer sous forme de distances potentielles entre les termes.

Le modèle word2vec est un réseau de neurone composé de deux architectures skip-gram et cbow qui seront détaillé ci-dessous.

II.5.4.1 Le modèle CBOW

La première architecture permet de prédire un mot Wi en fonction de son contexte.

La construction des words embeddings se fait en calculant la somme des termes du contexte, puis en appliquant sur les vecteurs résultants un classifieur log-linéaire pour prédire le mot cible[25].

Etant donné un mot cible w_t et une séquence de mots du vocabulaire **W={**$w_{t-2}; w_{t-1}; w_{t+1}; w_{t+2}$**}** (où w_{t-k}précède et w_{t+k}suit $w_t$par **k** positions), l'objectif du modèle CBOW est de maximiser la probabilité de prédire correctement le mot

cible w_t. Le modèle CBOW génère un vecteurv_t, correspondant au mot cible w_t, comme moyenne des vecteursdes mots dans la séquence **W**, c'est-à-dire

$$v_t = \frac{1}{|w|}\sum_{i=1}^{w} v_i \qquad \textbf{II.7}$$

La fonction objective de CBOW à son tour n'est que légèrement différente de celle du modèle de langue :

$$S = \frac{1}{I}\sum_{i=1}^{I} \boldsymbol{log} P(w_1, \dots, w_{t-1+n}) \qquad \textbf{II.8}$$

Où **I** correspond au nombre de mots dans le corpus, le modèle reçoit une fenêtre de n mots autour du mot cible.

Le schéma ci-dessous illustre l'architecture du modèle CBOW.

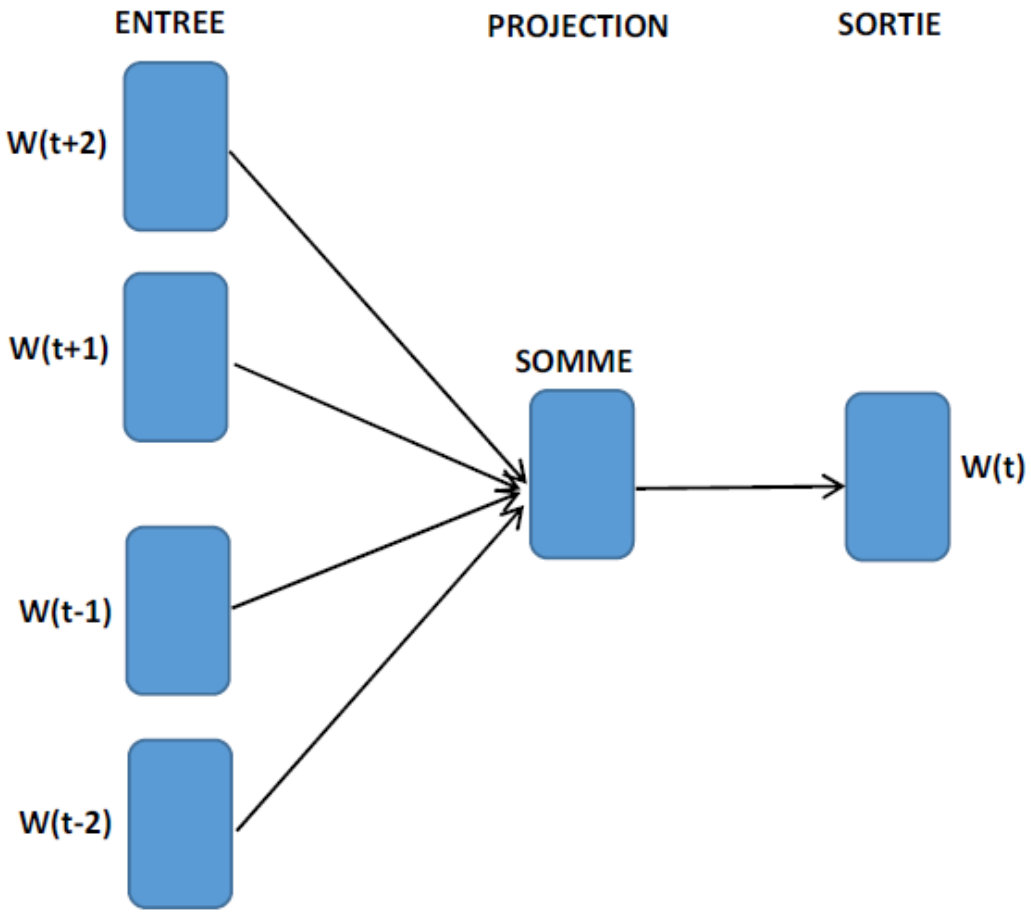


Figure II.5-Architecture du modèle CBOW-

Au lieu de prendre les **n** mots précédents comme dans le modèle de langue, ce modèle reçoit une fenêtre de **n** mots autour du mot cible w_t, il est très utile pour identifier les mots qui manquent dans une phrase.

II.5.4.2 Le modèle skip-gram

L'architecture skip-gram est l'inverse de CBOW. Le mot cible w_i est maintenant utilisé comme entrée, et les mots de contexte sont utilisés en sortie.

Elle consiste à prédire, pour un mot donné, le contexte dont il est issu [19].

Etant donné un corpus **W={**$w_{t-2}; w_{t-1}; w_{t+1}; w_{t+2}$**}**, le modèle Skip-gram sert à prédire les représentations des contextes **c** de ces mots de **W**.

Ce modèle construit des représentations vectorielles des mots de contextes.

L'objectif du modèle skip-gram est de maximiser la probabilité logarithmique moyenne suivante :

$$S=\frac{1}{|w|}\sum_{i=1}^{|w|}\sum_{-|w|\leq j\leq |w|, j\neq 0}\log(w_{i+j}|w_i) \qquad \textbf{II.9}$$

La taille de la fenêtre de contexte **|w|** détermine quels mots entourant le mot cible w_i sont considérés pour le calcul de la probabilité logarithmique (où la fenêtre est centrée autour du mot cible).

La probabilité d'un mot de sortie est calculée selon la fonction softmax:

$$P(w_0|w_I)=\frac{\exp({v_{w0}}^T v w_I)}{\sum_{w=1}^{|V|}\exp({v_w}^T v' w_I)} \qquad \textbf{II.10}$$

Où $\boldsymbol{v_{wI}}$et $\boldsymbol{v_{w0}}$ sont les représentations vectorielles des vecteurs d'entrée et de sortie, respectivement, et $\sum_{w=1}^{|V|}\exp({v_w}^T v' w_I)$ est le facteur de normalisation, dont le rôle est de normaliser les résultats internes du produit dans tous les mots de vocabulaire (|V| est la taille du vocabulaire). L'échantillonnage négatif est utilisé pour réduire la complexité du calcul.

Le schéma ci-dessous illustre l'architecture du modèle skip-gram :

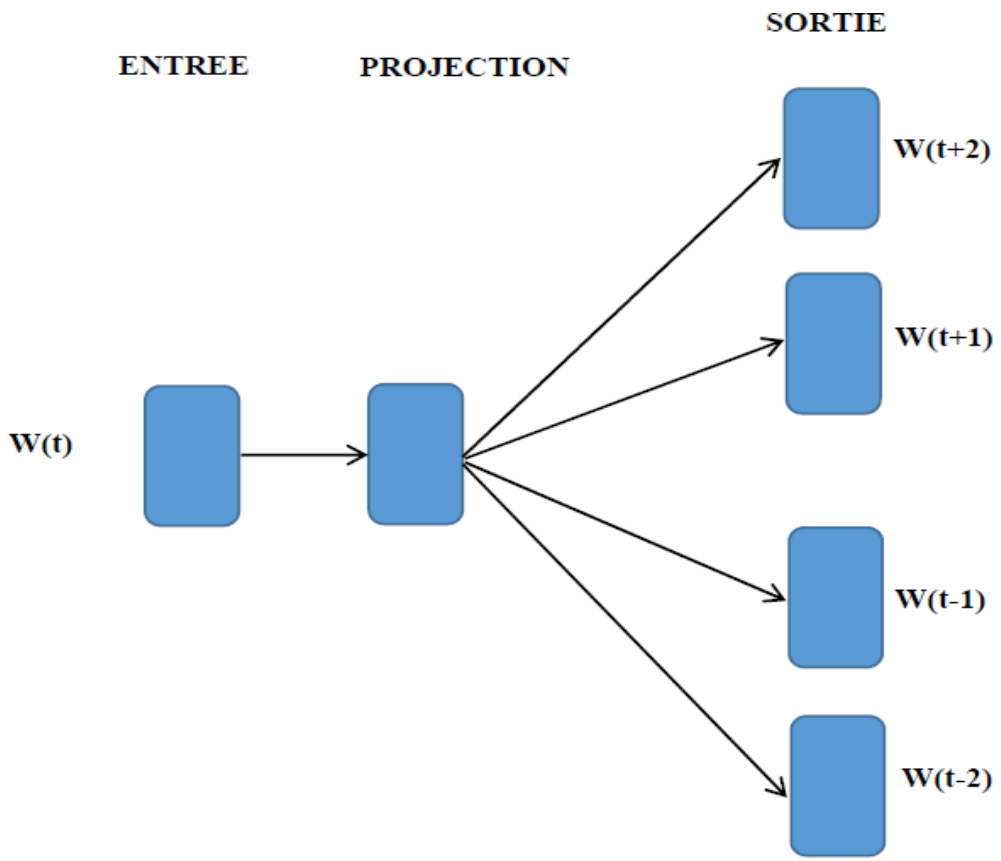


Figure II.6-Architecture du modèle skip-gram-

Comparativement au CBOW, les réseaux Skip-gram apprennent de meilleures représentations sémantiques et sont plus adaptés aux mots peu fréquents [25].

II.6 L'utilisation du Word Embbeding En RI

Pour répondre à la requête utilisateur, la majorité des modèles de RI représentent les documents et les requêtes sous forme de « sacs de mots » (bag of words) pondérés ou un sac de concepts issus d'une ontologie linguistique ou construits automatiquement par des techniques de type LSI (Latent Semantic Indexing) ou LDA (Latent Dirichlet Allocation).

La construction d'une représentation complexe pour un document, et la prise en compte de la structure permettent de résoudre le problème du sac de mots qui

n'informe pas sur l'emplacement, et la distribution des mots dans le texte, pour cela une nouvelle technique est apparue qui est le word embedding.

Différentes études ont été effectuées dans la recherche d'information en intégrant le word embedding. Nous allons citer ci-dessous quelque unes d'entre elles :

- [31]ont proposé une nouvelle méthode de recherche d'information . À savoir, le modèle de langage généralisé(GLM) qui est un modèle de recherche de base qui utilise le word embeddingpour une meilleure performance en recherche d'information.

-[32] ont proposé une méthode d'expansion de requête basée sur le voisin le plus proche (Nearest Neighbour). Ils ont décrit deux méthodes qui utilisent les words embedding de termes de requête.

La première méthode est une méthode d'expansion de requêtes qui utilise les termes sémantiquement similaires pour étendre une requête.

La deuxième approche est une simple variation de la première qui utilise des words embedding en conjonction avec un ensemble de documents pseudo pertinents.

- [33] attirent l'attention sur la question du changement de sujet lorsqu'on utilise directement le modèle word embedding basés sur des fenêtres (par exemple, word2vec et GloVe). Leurs études montrent un fort potentiel d'amélioration des performances dans la recherche de documents lors du filtrage des termes qui causent le changement de sujet.

-[34] utilisent les words embeddings obtenus par le modèle word2vec (cbow) pour construire des vecteurs de « caractéristiques » pour chaque mot de requête. Ce vecteur est obtenu par la soustraction entre la moyenne des vecteurs des words embeddings des termes de requête et les vecteurs des word embeddings des termes donnés. Cette technique consiste à prédire un poids cible pour chaque terme de requête étant donné son vecteur caractéristique en utilisant la technique de régression. Le poids cible de chaque terme est tiré des jugements de pertinence. Ils supposent que des jugements de pertinence sont disponibles pour la même collection.

-[17] proposent le modèle de traduction de langage neuronale (NLTM) qui intègre les word embeddings. Ce modèle consiste à estimer la probabilité de traduction entre les termes comme le cosinus entre les vecteurs word embedding des termes. Le modèle

de traduction utilisé avant l'arrivé des words embeddings est basé sur l'information Mutuelle (IM) pour estimer la probabilité de traduction. Les expériences d'évaluation de l'approche NLTM sur la recherche ad hoc sont effectuées sur les collections TREC AP87-88, WSJ87-92, DOTGOV ont montré des améliorations modérées par rapport aux modèle de traduction basé sur l'IM. L'analyse des différents paramètres des modèles des words embeddings 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. Par contre le choix du corpus pour l'apprentissage des words embeddings montre que l'efficacité du modèle NLTM semble généralement plus élevée lorsque les words embeddings sont estimés à l'aide de la même collection dans laquelle la recherche doit être effectuée.

II.7 Conclusion

L'utilisation des words embeddings ont réalisé un progrès dans la recherche d'information. La construction des words embeddings est basée sur les modèles neuronaux. Nous avons présenté dans ce chapitre ces modèles tels que C&W, Word2vec et Glove ainsi les différents types de réseaux de neurone qui ont contribué à l'architecture de ces modèles. Nous avons vu aussi quelques travaux qui ont exploité le word embedding en recherche d'information (RI).

Le troisième chapitre est consacré à la présentation et l'implémentation de notre approche, ainsi que les résultats que nous avons obtenus.

III.1 Introduction

Le word embedding est une nouvelle technique qui se base sur la représentation vectorielle des mots de la requête et ceux des documents, cette technique a été implémentée dans plusieurs travaux pour améliorer les résultats de la recherche d'information.

Notre travail se focalise sur l'extension d'un modèle basé sur le modèle de translation en utilisant le word embedding en lui intégrant une autre technique qui est le clustering.

Dans un premier temps, nous présentons l'approche de Zuccon et all [17], puis nous présentons notre extension de cette approche qui se base sur le word embedding et le clustering. Enfin nous présentons l'implémentation de notre approche ainsi que les résultats obtenus comparativement à l'approche de Zuccon et all [17].

III.2 Modèle de Zuccon et all. [17]

Avant de présenter l'approche de Zuccon et all [17] nous présentons d'abord quelques rappels sur le modèle de langue et le modèle de traduction en RI.

III.2.1 Le modèle de langue

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

Cette probabilité est alors exprimée comme suit :

$$RSV(d,q)=P(q=(t_1,t_2,\dots,t_n)|M_d)=\prod_i P(t_i\,|M_d) \qquad \textbf{III.1}$$

$P(t_i|M_d)$ est donnée par :

$$P(t_i|M_d)=\frac{tf(t_i,d)}{\sum tf(t,d)} \qquad \textbf{III.2}$$

Où **tf(t|d)** est la fréquence d'occurrence du n-gramme (du mot) $\boldsymbol{t_i}$ dans le document **d** et $\sum \boldsymbol{tf(t,d)}$ correspond à la taille du document.

III.2.2 Modèle de traduction [17]

Le modèle de traduction est utilisé pour déterminer la sémantique entre les termes de la requête et ceux des documents, il se base sur la correspondance sémantique plutôt que sur la correspondance exacte entre les termes.

La formule du modèle de traduction est donnée ci-dessous :

$$p(w|d)=\sum p(w|u) * p(u|d) \qquad \textbf{III.3}$$

Où$P(u|w) = \frac{\cos(u,w)}{\sum_{u' \in V} \cos(u',w)}$

III.2.3 L'approche de Zuccon et all [17]

Cette approche a été proposée par Zuccon et all [17]. Où L'appariement sémantique est modélisé comme une traduction entre les termes de la requête et ceux des documents. Un élément essentiel de ces modèles est l'estimation de la probabilité de traduction entre les termes. L'estimation de cette probabilité est basée sur la similarité entre les termes des documents et ceux de la requête.

Dans le cadre de la modélisation du langage, le modèle de Zuccon se base sur le modèle de traduction et le word embedding, en effet les documents sont classés en fonction de la vraisemblance.

$$log_p(q|d) = \sum_{i:c(qi:d)>0} log \frac{p_s(qi|d)}{\propto_d p(qi|c)} + nlog \propto d + \sum_{i=1}^{n} log_p(qi|C) \qquad \textbf{III.4}$$

On faisant une estimation avec le lissage Dirichlet sur $p_s(qi|d)$ on obtient :

$$p(w|d)=\frac{p_{ml}(w|d)+|d|+\mu P(w|C)}{|d|+\mu} \qquad \textbf{III.5}$$

Avec $p(w|d)=\sum p(w|u) * p(u|d)$ est la probabilité de translation.

Où $P(u|w) = \frac{\cos(u,w)}{\sum_{u' \in V} \cos(u',w)}$ le word embedding entre les vecteurs des termes.

III.3 Extension de notre approche

En se basant sur l'approche précédente proposée par Zuccon en 2015. Nous avons proposé une amélioration de cette approche en introduisant la notion du clustering.

Le clustering est une technique qui permet la classification des documents selon une thématique, l'utilisation du clustering dans la recherche d'information sert à mieux localiser les documents pertinents selon la sémantique des termes de la requête. D'autre part le word embedding prend en charge la représentation des termes selon leurs sémantiques.

Dans notre approche, on a appliqué le word embedding sur les clusters pour plus d'efficacité par apport à la recherche qui se base sur la correspondance sémantique entre les termes de la requête, et les documents offert par le word embedding, et la performance par apport à la recherche sur les clusters plutôt que sur la collection. Nous avons utilisé le modèle Dirichlet étendu dans la section précédente dans la formule **III.4** comme technique de lissage on aura la formule suivante :

$$P(t|d) = \propto p(t|d) + \beta p(t|clus) + \gamma p(t|C) \qquad \textbf{III.6}$$

On combinant les équations **III.3** et **III.4 et III.6** et on obtient la formule ci-dessous qu'on a utilisée dans notre approche:

$$\text{Score}(d_i, Q0) = \log\left[\frac{1}{(|d|+\mu+|clus|)}\left(((|d| + |clus|) * (\beta_1 \sum_{u\in d} p(w|u) * p(u|d)) + (\beta_2 \sum_{w\in clus} p(w|u) * p(u|clus))) + \mu * p(w|C))\right] - \log\left(\frac{\mu}{(|d|+\mu+|clus|)} * p(w|C)\right) + \log\left(\frac{\mu}{(|d|+\mu+|clus|)}\right) \qquad \textbf{III.7}$$

Avec :

|d|:La taille du document.

|clus|:La taille du cluster.

P(w|u) : le word embbeding du terme w, calculé avec la formule :

$$P(w|u) = \frac{\cos(w,u)}{\sum_{u' \in V} \cos(u',w)}$$

P(u|d) : La probabilité d'apparition du terme u dans le document d, calculée ainsi :

$$P(u|d) = \frac{tf(u|d)}{|d|}$$

tf(u|d) est le nombre d'apparition du terme u dans d.

P(u|clus) : la probabilité d'apparition du terme u dans le cluster, calculée ainsi :

$$P(u|clus) = \frac{tf(u|clus)}{|clus|}$$

tf(u|clus) est le nombre d'apparition du terme u dans le cluster.

P(w|C) : la probabilité d'apparition du terme w dans la collection C, calculée ainsi :

$$P(w|C) = \frac{tf(w|C)}{|C|}$$

tf(w|C) est le nombre d'apparition du terme w dans la collection C.

$\boldsymbol{\beta_1, \beta_2, \mu}$: des facteurs.

III.4 Implémentation de notre l'approche

Dans cette section nous allons présenter la mise en œuvre de notre approche, et nous présentons les étapes de son fonctionnement.

III.4.1 La mise en œuvre :

Dans cette partie, nous allons présenter le fonctionnement global de notre approche qui est illustré dans le schéma suivant :

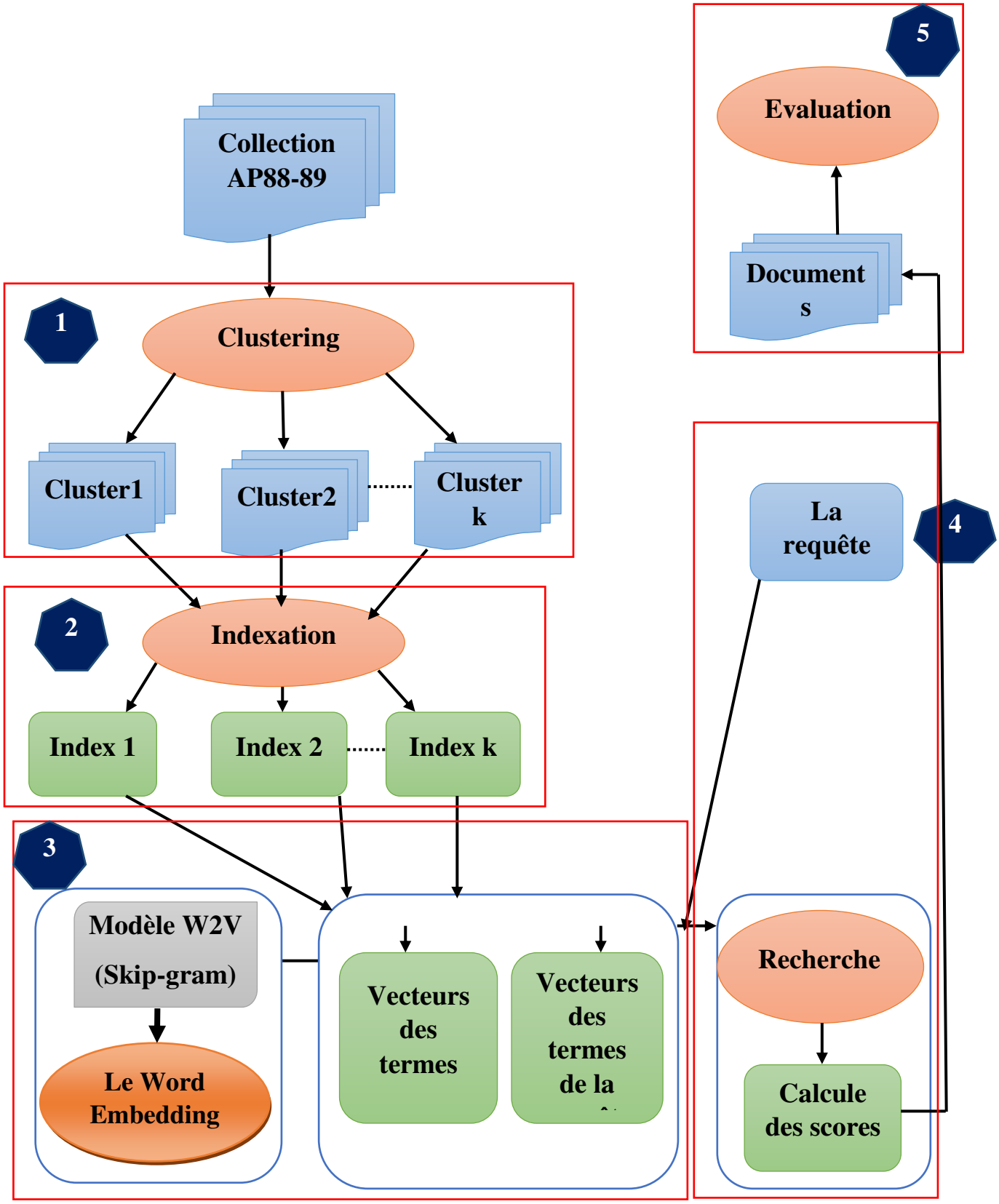


Figure III.1-Fonctionnement de notre approche-

III.4.2 Les étapes de notre approche

Pour implémenter notre approche, nous avons procédés en suivant les étapes mentionnées dans la **figure III.1** précédente :

-Etape1 : Appliquer le clustering sur la collection AP88-89.

Le clustering est une technique qui permet de diviser la collection en un nombre de clusters spécifié par l'utilisateur. La recherche à base de clusters exige que les documents soient d'abord organisés en groupe.

Pour les documents d'un cluster, il faut établir une mesure de similarité par paires de documents (ou à distance), puis choisir un algorithme de classification des documents en fonction de leur similitude (la distance), parmi les mesures de similarité les plus utilisé on a la mesure cosinus, la mesure de Jaccard.

Il existe plusieurs algorithmes de classification, des études ont montrées que le meilleur algorithme à utiliser est K-means, c'est pour cela que nous l'avons utilisé dans notre approche. K-means procède en suivant trois passes pour le partitionnement. Le nombre K est une entrée à l'algorithme qui spécifie le nombre désiré de clusters.

Dans le premier passage, l'algorithme prend les premiers K documents chacun comme le centre de gravité d'un cluster unique.

Avec $\mathbf{d_i}$ les documents du cluster, et **n** le nombre de documents du cluster.

Chacun des documents restants est ensuite comparé à ces centres de gravité en fonction de la distance euclidienne entre eux, et affecté au cluster avec la plus proche distance.

Dans les passages suivants, les centres de gravité du cluster sont recalculés en se basant sur les clusters formés dans le passage précédent, et la composition du groupe de documents sont réévalués en fonction de ces nouveaux centres de gravité. Ainsi de suite jusqu'à l'obtention des clusters finaux.

-Etape2: Indexer tous les clusters.

Appliquer le processus d'indexation sur chaque cluster en suivant les étapes suivante :

-Analyse lexicale : convertir un texte de document en liste de token.

-Elimination des mots vides: identifier les mots vides, ou mots qui n'apportent pas de sens au texte (prépositions, déterminants, adverbes, conjonctions, etc.), et les éliminer.

-Normalisation : construire une racine commune à un ensemble de mots de la même famille.

-La pondération : 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.

-Etape3: Appliquer le word embedding sur les clusters.

Nous rappelons qu'il existe plusieurs manières pour construire les words embeddings, le modèle le plus connu est word2vec qui est constitué de deux architectures skip-gram et cbow.

La figure suivante représente un exemple d'un fichier contenant les words embeddings obtenus à partir de l'algorithme de l'architecture skip-gram :

```
million -0.598626 0.168560 0.100276 0.097245 -0.152787 0.181717 0.370927 0.051000 0.307305 0.089384 0.303052 0.034036 -0.
when -0.486064 -0.315827 0.378562 -0.226375 -0.199393 -0.164619 -0.162890 0.081020 -0.202473 -0.040595 -0.348307 0.052471
also -0.395618 -0.219534 -0.194099 -0.192652 -0.129365 0.062787 -0.072128 0.141475 -0.380290 0.171661 0.000245 0.066383 -
other -0.252458 0.009199 -0.170528 -0.074996 -0.015604 -0.083963 -0.104796 0.164736 -0.310748 0.066786 -0.092161 0.141546
government -0.170247 0.208490 -0.172631 -0.049062 -0.415661 0.182263 -0.236325 0.197461 -0.304121 -0.176338 0.042329 -0.1
no -0.065733 0.076820 -0.060722 -0.150825 -0.150548 -0.085774 -0.456832 0.235236 -0.304697 0.271279 -0.227852 0.046814 -0.
there -0.245133 0.004285 0.046899 -0.080726 -0.334048 0.101451 -0.254830 0.092315 -0.407013 -0.107048 -0.346844 0.066634 -
than -0.337815 0.163228 0.048884 -0.223775 -0.169114 0.441644 0.199840 0.094097 0.077837 -0.265983 0.114695 0.322736 -0.5
president -0.415721 -0.231940 -0.227120 0.048899 -0.415781 -0.003490 -0.025815 0.060169 -0.066231 -0.068288 0.131063 -0.1
she -0.002385 -0.014100 0.263448 0.015986 -0.193196 -0.341938 -0.353444 -0.232914 -0.043796 -0.011940 -0.400492 -0.117182
all -0.311677 0.096146 0.192131 -0.140614 -0.104757 0.129783 -0.223018 0.236932 -0.086537 0.072250 -0.259994 0.344800 -0.5
last -0.508820 -0.110750 -0.024805 -0.051022 -0.245636 -0.042631 -0.089053 0.145496 0.122002 0.160329 -0.265576 0.264416 -
out -0.196364 0.105552 0.027216 -0.137675 -0.202448 0.109282 -0.140526 -0.207267 0.321602 -0.245785 -0.279958 0.116526 -0.
years -0.672428 -0.148352 0.172778 -0.096573 -0.139128 0.423026 -0.276296 -0.251843 0.029688 0.172046 -0.086135 0.405066 -
state -0.151402 0.103554 -0.246124 -0.088363 -0.701540 0.110767 -0.275657 -0.026822 -0.213432 0.000748 0.080760 0.017368 -
some -0.036428 0.113979 -0.069539 -0.150096 -0.042315 -0.081210 -0.223316 0.231157 -0.110476 0.060543 -0.309328 0.068636 -
note -0.914992 -0.598222 -0.000998 0.362763 0.018473 0.443680 0.211316 0.080483 -0.391725 0.523700 0.688830 0.165248 -0.2
```

Figure III.2 -exemple du fichier skip-gram contenant les words embeddings-

L'ensemble de ces probabilités seront stocké dans un fichier qui a l'extension .**ser,** il sera chargé lors du processus de la recherche.

-Etape4 :

Après avoir chargé le fichier **.ser** du word embedding, et récupéré les vecteurs de tous les termes, nous allons effectuer la recherche comme suit :

-L'architecture du modèle de langue neuronal word2vec prend en entrés le vecteur de chaque terme de la requête, fait une projection de ce vecteur sur l'espace des 200 dimensions vectoriel de chaque cluster, calcule la distance de similarité entre ce vecteur et les n-1 et n+1 vecteurs des termes de chaque clusters sachant que n=5 correspond à la taille de la fenêtre du modèle skip-gram, et sélectionne les termes des vecteurs les plus proche par apport au vecteur de la requête.

-Identifier le document du cluster auquel appartient chaque terme, et calculer le score de similarité pour chaque terme retrouvé dans chaque cluster.

La formule qui permet de calculer le score entre les words embeddings des documents de chaque cluster, et les requêtes est donnée comme suit :

$$\text{Score}(d_i, Q0) = \log[\frac{1}{(|d|+\mu+|clus|)}\big(((|d| + |clus|) * (\beta_1 \sum_{u \in d} p(w|u) * p(u|d)) + (\beta_2 \sum_{w \in clus} p(w|u) * p(u|clus))) + \mu * p(w|C))] - \log\left(\frac{\mu}{(|d|+\mu+|clus|)} * p(w|C)\right) + \log(\frac{\mu}{(|d|+\mu+|clus|)}) \qquad \textbf{III.9}$$

-Etape5 : Evaluation des résultats

A la fin du processus de la recherche un fichier **.ser** est créé, ce fichier contient tous les documents retournés pour chaque requête (23 requêtes) ainsi le scores de chaque documents, et le modèle utilisé(DirichletLM).Voici un exemple de ce fichier :

```
051 Q0 AP880731-0085 0 21.977088529593118 DirichletLM
051 Q0 AP880318-0287 1 16.36548635975742 DirichletLM
051 Q0 AP880316-0292 2 16.218939798498035 DirichletLM
051 Q0 AP880406-0267 3 15.718714585451073 DirichletLM
051 Q0 AP880325-0293 4 15.478037477918097 DirichletLM
051 Q0 AP881108-0085 5 15.32555877475062 DirichletLM
051 Q0 AP880706-0311 6 15.078196833862815 DirichletLM
051 Q0 AP881108-0253 7 14.737376027674674 DirichletLM
051 Q0 AP880627-0045 8 14.544700709028884 DirichletLM
051 Q0 AP880517-0253 9 13.457697954490522 DirichletLM
051 Q0 AP880412-0268 10 13.366843456247615 DirichletLM
051 Q0 AP881019-0037 11 12.90793099269255 DirichletLM
051 Q0 AP881118-0209 12 12.582485435939393 DirichletLM
051 Q0 AP881119-0047 13 12.492891639795612 DirichletLM
051 Q0 AP881119-0051 14 11.844483363750868 DirichletLM
051 Q0 AP880618-0073 15 11.746578235641651 DirichletLM
051 Q0 AP881014-0042 16 11.498429100290304 DirichletLM
051 Q0 AP880407-0258 17 11.465004899468212 DirichletLM
```

Figure III.3–exemple du fichier a évalué–

Après l'évaluation de ce fichier un autre fichier serra créer avec l'extension .**eval** qui contient la valeur de la **MAP** et la précision moyen, voici un exemple de ce fichier :

```
Number of queries  = 28
Retrieved          = 28000
Relevant           = 7015
Relevant retrieved = 868

Average Precision: 0.0552
R Precision      : 0.0988

Precision at   1 : 0.2500
Precision at   2 : 0.2321
Precision at   3 : 0.2500
Precision at   4 : 0.2500
Precision at   5 : 0.2500
Precision at   10 : 0.2429
Precision at   15 : 0.2405
Precision at   20 : 0.2304
Precision at   30 : 0.2167
Precision at   50 : 0.1929
Precision at   100 : 0.1550
Precision at   200 : 0.1084
```

Figure III.4–exemple du fichier résultat de l'évaluation –

III.5 Expérimentation

III.5.1 Environnement de développement

Nous allons présenter dans cette section les différents outils utilisés dans notre environnement à savoir la plateforme TERRIER, le langage java ainsi que l'environnement Netbeans.

II.5.1.1 L'outil de recherche TERRIER

Terrier est un outil de recherche open source, très flexible, efficace et effectif. Il implémente plusieurs fonctionnalités de recherche, et fourni une plateforme idéal pour le développement rapide, et l'évaluation pour les applications de recherche à grande échelle.

C'est une plateforme, complète et transparente pour la recherche, et l'expérimentation dans la recherche de texte, cette recherche peut être effectuée facilement sur les collections de tests standard TREC et CLEF.

Terrier peut indexer de large collections de documents jusqu'à 50 millions de documents, il est écrit en Java, fonctionne sous différentes plateformes: Windows, Mac OS X, Linux, Unix [28].

-Architecture de terrier: L'architecture de Terrier distingue les deux phases classiques: l'indexation et la recherche.

La figure suivante illustre l'architecture de terrier :

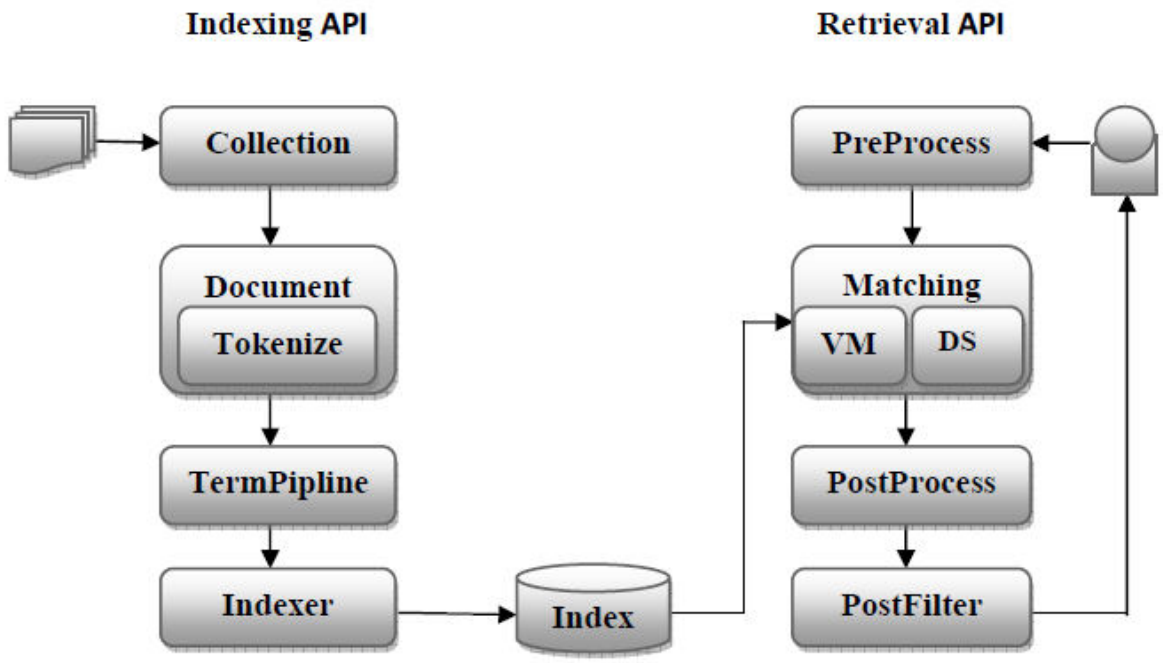


Figure III.5–Architecture de terrier-

a. L'indexation : l'indexation dans Terrier passe par les étapes suivantes :

1. Collection : lire les documents.
2. Document : extraire le texte brut.
3. Tokenizer : fragmenter (tokenizer) le texte extrait.
4. TermPipline : (chaine de traitement) traiter les tokens.
5. Indexer : indexer les tokens traités.

b. La recherche : ses étapes sont résumées ci-dessous :

1. Manager: Analyse la requête utilisateur.
2. Matching: Recherche les documents.
3. VMset DSMs: Score les documents.
4. PostProcess: Post-Traitement, met à jour les résultats globaux.
5. PostFilter: met à jour chaque document individuel retrouvé.

- Utilisation de Terrier dans l'indexation, la recherche et l'évaluation :

a. L'indexation : voici les différentes étapes de l'indexation sur la ligne de commande :

–Aller dans le dossier terrier :

cd terrier-3.6

– On spécifie le chemin de la collection:

./bin/trec_setup.sh «chemin vers la collection»

Cela donne comme résultat un fichier **collection.spec** qui est créé au niveau du dossier **etc,** et il contient les chemins vers les fichiers de la collection à indexer.

Pour indexer une collection on fait appel au fichier script **trec_terrier.sh** et en ajoutant le **–i** comme suit:

./bin/trec-terrier.sh –i

b. La recherche : Pour faire la recherche sur une collection indexée, on suit les étapes suivantes:

-Effectuer certaines configurations. La plupart des fonctionnalités de terrier sont contrôlées par des propriétés. On peut configurer ceci dans le fichier **Terrier.Propreties.Sample** qui se trouve dans le dossier **etc** de terrier, ou bien spécifier chaque propriété dans la ligne de commande. Dans ce qui suit, nous allons utiliser la ligne de commande pour spécifier les propriétés appropriées.

Pour faire la recherche et évaluer les résultats d'un lot de requêtes, on a besoin de savoir:

– Le chemin d'accès aux requêtes (aussi connu sous le nom de topic files), spécifié en utilisant le fichier **trec.topics** qui se trouve dans le dossier **etc**, au début ce fichier est vide, on a qu'à écrire le chemin des topics dedans.

–Le modèle de poids (exp: Dirichlet) à utiliser, spécifié en utilisant **trec.model**, si on veut choisir un modèle on a qu'à enlever le « # » qui se trouve au début de chaque modèle.

–Le fichier des jugements de pertinence correspondant (qrels) aux requêtes, spécifié par **trec.qrels**, de la même manière on va ajouter les qrels en écrivant le chemin vers ces qrels.

Pour commencer la recherche, l'option **–r** demande à Terrier d'exécuter une recherche sur un lot, c'est-à-dire rechercher les documents estimés être les plus pertinents pour chaque requête dans le fichier topics file. On effectue la recherche en exécutant la commande suivante :

./bin/trec_terrier –r

Si tout va bien, le résultat de la recherche est sauvegardé dans le fichier **.res** qui se trouve dans le dossier **var/results** appelé Dirichlet.res (pour le modèle Dirichlet), appeler chaque **.res** un exécutable.

c. L'évaluation : L'évaluation des résultats de recherche obtenus s'effectue en utilisant

l'option**–e** de trec_terrier:

./bin/trec_terrier –e

Terrier va aller dans le dossier **var/results**, évalue chaque fichier **.res**et enregistre les résultats dans un fichier **.eval** nommé comme le fichier **.res** correspondant.

II.5.1.2 Le langage java

Java est un langage de programmation, développé par **Sun Microsystems**(aujourd'hui racheté par **Oracle**).

Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux , etc [29].

–**Java est simple :** Java est un langage simple à prendre en main, basé sur le langage C/C++, mais laisse de côté les sources de problèmes (pointeurs, structures, gestion de la mémoire, héritage multiple, macros etc.).

– **Java est orienté objet :** L'utilisation des classes, héritage simple.

–**Java est distribué :** Java propose une API réseau standard. Cette dernière permet de manipuler, par exemple, les protocoles HTTP & FTP avec aisance.

Des API pour la communication entre des objets distribués (Remote Method Invocation).

-Java est interprété : Un code source doit être traduit dans le langage machine avant d'être exécuté.

-Compilateur: traduction du code source dans le langage binaire de la machine sur laquelle il sera exécuté.

-Interpréteur: idem qu'un compilateur, sauf qu'il procède par étapes successives de compilation et exécution. Chaque instruction est compilée puis exécutée, puis le tour à l'instruction qui suit, etc.

Compilateur Java traduit le code source Java en bytecode (code portable). Par la suite un interpréteur Java spécifique à une machine donnée (Java Virtual Machine : JVM Machine Virtuelle), traduit et exécute le bytecode.

-Java est indépendant de l'architecture : Le bytecode généré n'est pas lié à un système d'exploitation en particulier. De ce fait, il peut être interprété très facilement sur n'importe quel environnement disposant d'une JVM.

–**Java est portable :** Java est portable d'un système à un autre : int 32 bits alors qu'en C/C++ 16 ou 32 bits.

–**Java est robuste :** Pas de pointeurs.

Gestion de mémoire indépendante.
Mécanisme d'exceptions pour la gestion des erreurs.
Compilateur très contraignant.
Pas d'héritage multiple ni surcharge des opérateurs.
–**Java est sécurisée :** quatre niveaux de sécurité :
Langage et son compilateur contraignant.
Vérifier : vérifier le bytecode.
Class Loader : le chargeur de classe.
Security Manager : protection des fichiers et accès au réseau.

III.5.1.3 Netbeans

L'EDI NetBeans est un environnement de développement intégré, placé en open source par Sun en juin 2000, se concentrant principalement sur simplifier le développement d'application Java. Il fournit du support pour tous les types d'applications Java, depuis le client riche jusqu'aux applications d'entreprises multicouches, en passant par les applications pour les mobiles supportant Java.

NetBeans est disponible sous Linux, Solaris, Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java [30].

III.5.1.4 Les données utilisées

L'outil de recherche TERRIER peut travailler avec des collections TREC ou CLEF. La collection que nous avons utilisée pour nos expérimentations est : **TREC AP88-89** (***A**ssociated **P**ressnewswire*, 19**88**).

Collection	**documents**	requêtes utilisées
AP88-89	79,919	51–100 (23 requêtes)

Tableau III-1 : -les données de la collection AP88-89-

Pour la recherche nous avons utilisé 23 requêtes issues des topics numérotées « 51-100» de la collection TREC.

III.5.2 Présentation des résultats obtenus

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base : recherche simple (modèle de langue de base utilisant la méthode de lissage de Dirichlet), recherche basé sur l'approche de Zuccon pour les 23 requête utilisées.

Ces résultats seront comparés par la suite à ceux obtenus après l'application de notre approche.

III.5.2.1 Résultats obtenus avec la recherche simple

Le modèle de recherche utiliser dans cette approche est le modèle DirichletLM.

Le tableau suivant illustre les différents résultats d'évaluation obtenus de la MAP, P@5, P@10 et P@20 avec la variation de la valeur de μ du modèle de DirichletLM:

	MAP	**P@5**	**P@10**	**P@20**
μ=500	**0.1592**	**0.2184**	**0.2061**	**0.1847**
μ=1000	0.1533	0.2204	0.1884	0.1781
μ=1500	0.1498	0.2163	0.1929	0.1770
μ=2000	0.1464	0.2082	0.1878	0.1704

Tableau III.2 - Résultats obtenus avec la recherche simple (DirichletLM)-

D'après les résultats obtenus, on remarque que la meilleure valeur de la **MAP=0.1592** pour une valeur de **μ=500**.

Le tableau suivant illustre l'évaluation des requêtes (23 requêtes) :

Numéro de la requête	MAP
51	0.2855
57	0.1081
60	0.0023
69	0.1113
81	0.0001
86	0.0667
90	0.0750
95	0.0018
99	0.0631
101	0.0330
104	0.0001
110	0.0759
116	0.0017
117	0.1233
121	0.0025
123	0.0030
124	0.0795
132	0.0380
133	0.0010
134	0.4167
137	0.0370
146	0.1633
148	0.0250

Tableau III.3 -Résultats de l'évaluation des requêtes obtenus avec la recherche simple (DirichletLM)-

III.5.2.2 Résultats obtenus avec l'approche de Zuccon et all[17]

Nous allons appliquer le word embedding sur l'ensemble de la collection avec le choix de la meilleure valeur de **μ** obtenue lors de la recherche simple qui est égale à 500, et avec le même modèle DirichletLM.

Les résultats d'évaluation avec l'approche de Zuccon sont résumés dans le tableau suivant :

	MAP	**P@5**	**P@10**	**P@20**
μ=500	**0.2252**	0.2522	0.2391	0.2283
% d'amélioration par rapport au modèle DirichletLM	(**0.2252-**0.1592)/ 0.1592=0.41*100=41%	58%	50%	43%

Tableau III.4 -Résultats obtenus avec l'approche de Zuccon-

Les résultats obtenus dans ce tableau montrent qu'il y'a une amélioration de la MAP due à l'utilisation de la technique de word embedding.

Le tableau suivant montre l'évaluation des requêtes :

Numéro de la requête	MAP
51	**0.6694**
57	**0.6481**
60	**0.0150**
69	**0.1662**
81	**0.0354**
86	**0.1786**
90	**0.1554**
95	**0.0018**
99	**0.3867**
101	**0.0558**
104	**0.0059**
110	**0.3011**
116	**0.5000**
117	**0.1326**
121	**0.0580**
123	**0.0030**
124	**0.1732**
132	**0.0381**
133	**0.0100**
134	**0.4626**
137	**0.0477**
146	**0.2584**
148	**0.0268**

Tableau III.5 –Résultats de l'évaluation des requêtes obtenus avec l'approche de Zuccon-

En faisant une comparaison entre le tableau **III.3** et le tableau **III.5**, on remarque qu'il y a une amélioration de la **MAP** pour toutes les requêtes.

III.5.2.3 Résultats obtenus avec notre approche

Notre approche se focalise sur l'utilisation des deux techniques le word embedding et le clustering. Pour chaque requête, le cluster retourné représente le cluster qui contient tous les documents pertinents.

On varie le nombre K de clusters, et on observe la valeur de la MAP.

Le tableau suivant montre le résultat d'évaluation pour k=10 clusters :

	MAP	P@5	P@10	P@20
μ=500	**0.2806**	0.3001	0.2912	0.2907
% d'amélioration par rapport à l'approche de Zuccon	**(0.2806-**0.2252)/ 0.2252=0.24*100=24%	33%	29,3%	29,08%

Tableau III.6 -Résultats obtenus pour k=10 avec notre approche –

On remarque qu'il y a une amélioration de la MAP par apport à l'approche de Zuccon due à l'utilisation du clustering.

Les résultats d'évaluation des requêtes avec notre approche pour un nombre de clusters=10 sont résumés dans le tableau suivant :

Numéro de la requête	cluster	MAP
51	Cluster1	0.0909
57	Cluster10	0.0222
60	**Cluster6**	**0.0362**
69	Cluster2	0.0055
81	Cluster9	0.0182
86	**Cluster4**	**0.1822**
90	**Cluster9**	**0.1924**
95	**Cluster7**	**0.0024**
99	Cluster10	0.0278
101	Cluster1	0.0184
104	**Cluster2**	**0.0161**
110	**Cluster1**	**0.7522**
116	Cluster3	0.0182
117	**Cluster5**	**0.1495**
121	Cluster7	0.0013
123	**Cluster7**	**0.0066**
124	Cluster7	0.0875
132	**Cluster7**	**0.0391**
133	**Cluster7**	**0.0129**
134	Cluster3	0.0328
137	**Cluster1**	**0.0505**
146	Cluster1	0.1361
148	**Cluster8**	**0.0299**

Tableau III.7 –Résultats de l'évaluation des requêtes obtenus pour k=10 avec notre approche-

En comparant le tableau **III.5** (approche de Zuccon) et le tableau **III.7**, on remarque qu'il y a une amélioration de la **MAP** pour les requêtes **60, 86, 90, 95, 104, 110, 117, 123, 132, 133, 137, 148 (12 requêtes)**.

Le tableau suivant montre le résultat d'évaluation pour k=4 clusters :

	MAP	**P@5**	**P@10**	**P@20**
μ=500	**0.2332**	0.2301	0.2290	0.2269
% d'amélioration par rapport à l'approche de Zuccon	**(0.2332-**0.2252)/ 0.2252=0.03*100=3%	2%	1%	0,5%

Tableau III.8 -Résultats obtenus pour k=4 avec notre approche –

On remarque qu'il y a une petite amélioration de la MAP par apport à l'approche de Zuccon due à l'utilisation du clustering.

Le tableau suivant montre le résultat d'évaluation des requêtes pour un nombre de cluster k=4 :

Numéro de la requête	cluster	MAP
51	Cluster1	0.0329
57	Cluster2	0.0230
60	**Cluster4**	**0.0255**
69	Cluster2	0.0274
81	Cluster3	0.0098
86	Cluster2	0.0500
90	**Cluster3**	**0.1718**
95	**Cluster2**	**0.0036**
99	**Cluster2**	**0.4008**
101	**Cluster2**	**0.3923**
104	**Cluster2**	**0.0124**
110	**Cluster2**	**0.7912**
116	Cluster2	0.1180
117	Cluster1	0.0228
121	Cluster2	0.0307
123	Cluster1	0.0020
124	Cluster2	0.1067
132	**Cluster2**	**0.6867**
133	Cluster2	0.0013
134	Cluster2	0.0135
137	Cluster1	0.0022
146	Cluster2	0.0001
148	Cluster2	0.0039

Tableau III.9 –Résultats de l'évaluation des requêtes obtenus pour k=4 avec notre approche-

En comparant le tableau **III.5** (approche de Zuccon) et le tableau **III.9**, on remarque qu'il y a une amélioration de la **MAP** pour les requêtes **60, 90, 95, 99, 101, 104, 110, 132 (08 requêtes).**

Les résultats obtenus pour l'évaluation de notre approche montrent que :

-Pour **k=4**,le nombre de requêtes améliorées est 08 requêtes.

-Pour **k=10**, le nombre de requêtes améliorées est 12 requêtes.

Remarque : lorsqu'on augmente le nombre de clusters K, il y a une augmentation du nombre de requêtes améliorées.

On remarque qu'il y a une amélioration de la MAP par apport à l'approche de Zuccon due à la combinaison du word embedding et le clustering.

Le clustering a une influence sur la pertinence des résultats retournés par rapport à l'efficacité de la recherche puisque les documents les plus pertinents seront regroupés ensemble.

III.6 Conclusion

Dans ce chapitre, nous avons présenté le principe de notre approche et son fonctionnement, nous avons aussi présenté l'environnement d'implémentation à savoir la plateforme TERRIER, l'environnement Netbeans.

Enfin, nous avons présenté et discuté les résultats de notre approche vis-à-vis le modèle de recherche Dirichlet avec les résultats obtenus avec notre approche.

Une amélioration est constatée avec notre approche qui est basé sur la technique du word embedding et le clustering sur les requêtes.

Conclusion générale

Le travail présenté dans ce mémoire s'inscrit dans le cadre de la recherche d'information, il consiste à l'extension du modèle de langue basée sur le word embedding en utilisant le clustering.

Pour mener à terme notre travail, nous avons donné un aperçu général sur la recherche d'information ainsi le rôle des systèmes de recherche d'information, nous avons présenté par la suite la technique du word embedding et ses quelques modèles de base ainsi que son utilisation en recherche d'information. Nous avons implémenté notre approche sous la plateforme de recherche d'information Terrier basé sur le langage de programmation 'Java'. Pour la mise en œuvre nous avons utilisé l'environnement 'Netbeans'.

L'utilisation du word embedding avec le clustering dans notre approche a montré des améliorations par rapport aux résultats de la recherche simple et le modèle proposé par Zuccon et al [17].

Ce travail nous a permis d'approfondir nos connaissances sur la recherche d'information, de mettre l'accent sur la manière dont les systèmes de recherche d'information fonctionnent sur la plate-forme Terrier.

Références bibliographiques

[1] : Fatiha Boubkeur-Amirouche. Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets. Thèse de doctorat en informatique. L'Université Toulouse III - Paul Sabatier. 2008.

[2] : Jérôme Bondu. "Panorama d'outils de recherche d'informations gratuits et en ligne". Inter-Ligere Sarl. http://www.inter-ligere.com/article-30587376.html.2009.

[3] : Cours de RI. Master2 conduite de projet informatique. Université Mouloud Mammeri tizi-ouzou.2017.2018.

[4] : Cours de RI. Master2 ingénierie des systèmes d'informations. Université mouloud Mammeri tizi-ouzou.2017.2018.

[5] : M Hammache Arezki. Recherche d'information : un modèle de langue combinant mots simples et mots composés. Université Mouloud Mammeri de Tizi Ouzou.

[6] : Brigitte Simonnot. La pertinence en science de l'information : des modèles, une théorie. https://archivesic.ccsd.cnrs.fr/sic_00496291.2010.

[7] : Abbassi Meftah. Un modèle de reformulation des requêtes pour la recherche d'information sur le Web.

[8] : Mr.Ounnaci Iddir. Recherche D'information Dans Les Documents Pédagogiques Structurés Adaptée Aux Besoins Spécifiques Des Apprenants. Ingénierie Des Systèmes Informatiques. Université Mouloud Mammeri De Tizi-Ouzou.

[9] : Yaël Champclaux : Un modèle de recherche d'information basé sur les graphes et les similarités structurelles pour l'amélioration du processus de recherche d'information. https://tel.archives-ouvertes.fr/tel-00446372.2010.

[10] : Zhai, C., Lafferty, J. A study of smoothing methods for language models applied to add hoc information retrieval. In W.B Croft, D.J. Harper, D.H. Kraft, & J. Zobel (Eds), Processing of the 24th annual information ACM-SIGHIR Conference on Researche and Developement in information Retrieval, pp. 334-342, 2001.

[11] : Mr.Abdelkrim Bouramoul. Recherche d'informationcon textuelle et semantique sur le web. Informatique. Université MENTOURI de Constantine, 2011.

[12] : https://fr.wikipedia.org/wiki/Word_Embedding.

[13] : T. Mikolov, K. Chen, G. Corrado, & J. Dean. Efficient estimation of word representations in vector space. In Workshop at ICLR, 2013.

[14] : Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. The Journal of Machine Learning Research, 3, 1137–1155.

[15]: http://ruder.io/word-embeddings-1/.

[16]: Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research, 11(Feb):625–660.

[17] : Integrating and Evaluating Neural Word Embeddings in Information Retrieval Guido Zuccon1 Bevan Koopman; Peter Bruza Leif Azzopardi. Queensland University of Technology, Brisbane, Australia. Australian e-Health Research Centre, CSIRO, Brisbane, Australia. University of Glasgow, Glasgow, United Kingdom g.zuccon@qut.edu.au,bevan.koopman@csiro.au.p.bruza@qut.edu.au.leif.azzopardi@glasgow.ac.uk.

[18] : Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507.

[19] : Sahar Ghannay : Etude sur les représentations continues de mots appliquées à la détection automatique des erreurs de reconnaissance de la parole. https://tel.archives-ouvertes.fr/tel-01661491.2018.

[20] : Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Sackinger, E., and Shah, R. ¨(1993). Signature verification using a siamese time delay neural network. International Journal ofPattern Recognition and Artificial Intelligence, 7(04):669–688.

[21] : Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing. Proceedings of the 25th International Conference on Machine Learning – ICML '08, 20(1), 160–167.

[22] : https://fr.wikipedia.org/wiki/Réseau_de_neurones_artificiels.

[23] : https://fr.wikipedia.org/wiki/Réseau_de_neurones_récurrents.

[24]: https://fr.wikipedia.org/wiki/Réseau_de_neurones_boltzmann.

[25] : Mohamed Morchid : Réseaux de Neurones pour la Représentation de Contextes Continus des Mots. Université d'Avignon, LIA (France).

[26] : Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conférence on Empirical Methods in Natural Language Processing, 1532–1543. http://doi.org/10.3115/v1/D14-1162.

[28] : www.terrier.org.

[29] : www.oracle.com/java/.

[30] : https://netbeans.org/.

[31] : Ganguly et al. (2015). A Word Embedding Based Generalized Language Model for Information Retrieval.

[32] : Roy et al. (2016). Using Word Embeddings for Automatic Query Expansion

[33] : Rekabsaz et al. (2016). Uncertainty in Neural Network Word Embedding Exploration of Threshold for Similarity .

[34] : Zhang et Callan (2015). Neural Information Retrieval: A Literature Review.