REPUBLIQUE ALGERRIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSTE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Mémoire de Fin d'Etudes de Master Académique

Domaine : Mathématique et Informatique
Filière : Informatique
Spécialité : Système Informatiques

Présenté par : *Melle CHIKH Farida*
Melle HADADI Nawal.

Thème :

Extension d'un modèle d'expansion de requêtes pour la prise en compte de la représentation de type word embedding.

Mémoire soutenu publiquement le30 /09/2017 Devant le jury composé de :

Présidente : BOUGCHICHE.L
Encadreur : Mr A.Hammache.
Examinatrice : YESLI Yasmine
Examinateur :

Promotion 2016/2017

Remerciements

Avant tout je remercie « Dieu »pour m'avoir donnée la force, le courage et la volonté afin de mener à bien et à terme ce modeste travail.

Nous adressons aussi nos vifs remerciements à :

- *Notre promoteur Mr HAMMACHE.A pour la qualité de son encadrement, et son suivi durant toute la durée de la réalisation du projet. Nous tenons également à lui exprimer notre reconnaissance pour le temps qu'il nous a consacré, ainsi que pour ses encouragements.*
- *Aux membres du jury pour avoir consenti à examiner notre travail. Qu'ils trouvent ici toute notre reconnaissance.*
- *A nos parents, et nos chères familles pour leur soutien, leurs encouragements.*

Dédicaces

A mes très chers parents que j aime très fort, pour leurs sacrifices et leurs Encouragements Que dieu les protègent.

A ma chère grand mère que dieu la protège et la garde pour nous.

A mes anges gardiens :
Mes frères MAHDI, BILAL que dieu les gardent pour moi

. *A mes chères sœurs SIHAM, HANNANE, KAMILIA*
Qui ont été toujours à mes côtés.

A mes tantes et mes oncles

A mes cousins et mes cousines

Mes amies qui m'ont toujours aidé, orienté, conseillé et aimé

LYDIA, FAZIA, SAMIA, SIHAM, FARIDA, LYNDA

A mon très cher binôme FARIDA que j'aime beaucoup

Et toute sa famille.

A tous mes collègues de la promotion 2016/2017.

Toute personne qui m'a aidé de près
Ou de loin, je leurs souhaite la
Réussite et le bonheur.

Nawal

Dédicace :

Nous voici au bout de notre cursus .A la fin d'un long parcours d'études préliminaires à une spécialisation peut-être, au terme duquel j'élabore, avec mon binôme NAWAL le présent mémoire de fin d'étude que je dédie cordialement à :

- Mes adorables parents MOUHAMED ET MALIKA pour leur amour leur aide matérielle et surtout leur soutien moral ainsi que pour leurs précieux conseils ; encore mille merci à vous deux.
- Mes très chères sœurs : SADIA, TINHINANE, LOUIZA ET SON MARI RACHID.
- Et les petits enfants : RAZANE, ABDOU et MOUH.
- A Toute la famille CHIKH.
- Mon binôme et chère amie NAWAL, pour les merveilleux moments passés ensemble, et à sa famille.
- Mes amis : LIDIA, FAZIA, SIHAM, SAMIA, WAHIBA, HORIA Et tous les autres, la liste est longue, je m'en excuse.
- Et à tous ceux qui prêtent attention à ma thèse.

Dédicace FARIDA

Sommaire

Chapitre II : Word embedding

Chapitre III : Présentation de l'approche proposée et son évaluation.

Liste des figures

Liste des tableaux

Introduction générale

La recherche d'information est un domaine qui s'intéresse à la structure, à l'analyse, à l'organisation, au stockage, à la recherche, à la découverte de l'information, le défi est de pouvoir, parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur.

L'opérationnalisation de la RI est réalisée par des outils informatiques appelés systèmes de recherche d'Information (SRI), Le but principal d'un système de recherche d'information (SRI) est de retrouver les documents pertinents en réponse à une requête utilisateur.

Dans les Systèmes de Recherche d'Information, souvent la requête initiale est insuffisante pour permettre de sélectionner des documents pertinents qui répondent aux besoins des utilisateurs. Pour cela l'expansion de requêtes est une méthode conçue pour améliorer le processus de la recherche d'information.

Notre travail se situe dans le domaine de la recherche d'information(RI), particulièrement dans l'expansion de la requête qui consiste à ajouter de nouveaux termes à la requête initiale et pour cela nous utilisons la technique word embeddings.

Le word embeddings est une technique d'apprentissage qui vise à cartographier des mots à partir d'un vocabulaire en vecteurs de nombres réels dans un espace à faible dimension, cette technique permet du capturer le sens des mots. Les word embedding ont montrés aussi des propriétés d'analogie entre les termes, ils sont obtenus par des modèles tel que word2vect qui se base sur les réseaux de neurones il prend en entrée un grand corpus de texte et produit un espace vectoriel, chaque mot unique dans le corpus étant affecté d'un vecteur correspondant dans l'éspace,les mots qui partagent des contextes communs dans le corpus se trouvent à proximité l'un de l'autre dans l'espace

Notre travail consiste à intégrer les représentations distribuées de mots(les word embedding) obtenues par un modèle neuronale (Word2vect) dans les fonctions de modèle d'expansion de requêtes.

Le présent mémoire, comporte outre l'introduction, la conclusion, et la bibliographie,
Les trois chapitres suivants :

- **Chapitre I : «la recherche d'information »** décrit les généralités sur le domaine de la recherche d'information, notamment les concepts de bases de la recherche d'information, les modèles de recherche d'information, et l'évaluation des systèmes de recherche d'information.
- **Chapitre II : « Word embedding »** Ce chapitre traite l'exploitation de la technique word embedding en recherche d'information. Nous présentons notamment les fondements de cette technique à savoir les réseaux de neurones, ainsi que quelque modèle de word embeddings .
- **Chapitre III : « Présentation de l'approche proposée et son évaluation »**ce chapitre porte sur la présentation de l'approche proposée et l'expérimentation de l'approche en précisant les outils et le langage utilisés pour sa mise en œuvre. Ainsi que les résultats obtenus.

Ce travail se termine par une conclusion générale.

I.1 Introduction

Au début des années soixante, quelques années après l'invention de l'ordinateur, la RI est apparue comme une réponse au besoin de gérer l'explosion de la quantité d'informations. C'est la science de la recherche de l'information dans des documents qu'ils soient dans une base de données, dans une base documentaire ou sur le Web. Très tôt, le monde de la recherche s'est intéressé aux SRI pour mettre en œuvre des techniques et des mécanismes dans le but de pouvoir , parmi le volume important de documents disponibles, trouver ceux qui correspondent au mieux à l'attente de l'utilisateur.

L'objectif de ce chapitre est de présenter le domaine de la recherche d'information. Dans la première partie, nous présentons les concepts de base de la recherche d'information (éléments et processus).Dans la seconde partie, nous parlons des modèles de recherche d'information et dans la dernière partie de ce chapitre nous discutons de l'évaluation des systèmes de recherche d'information.

I.2 Définition de la Recherche d'Information (RI)

Plusieurs définitions de la recherche d'information ont vu le jour, nous citons les trois définitions suivantes :

- ✓ La recherche d'information est une branche de l'informatique qui s'intéresse à l'acquisition, l'organisation, le stockage, la recherche et la sélection d'information **[1].**
- ✓ La recherche d'information est une activité dont la finalité est de localiser et de délivrer des granules documentaires à un utilisateur en fonction de son besoin en informations **[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 **[3].**

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

Pour répondre aux besoins en informations de l'utilisateur, un SRI met en œuvre un certain nombre de processus pour réaliser la mise en correspondance des informations contenus dans un fonds documentaires d'une part, et les besoins en information des utilisateurs d'autre part **[0l].**

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

L'objectif d'un SRI est de relier le besoin en information d'un utilisateur, modélisé par une requête, avec un ensemble de documents en estimant leur pertinence par rapport à la requête. Dans ce qui suit, nous introduisons, dans un premier temps, les éléments de la RI. Dans un second temps, nous détaillons les processus de la RI.

I.4.1 Les éléments de de la RI

I.4.1.1 Requête

La requête constitue l'expression du besoin en information de l'utilisateur. Elle représente l'interface entre le SRI et l'utilisateur. Divers types de langages d'interrogation sont proposés dans la littérature. Une requête est un ensemble de mots clés, mais elle peut être exprimée en langage naturel, booléen ou graphique. Trois types de requêtes ont été identifiés **[4].**

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

I.4.1.2 Document

Un document est généralement défini comme support physique d'une information structurée, sous forme de mots, de sons, d'image, une séquence de vidéo, etc. plus précisément on peut le définir comme un ensemble de données informatives présentes sur un support, d'une façon permanente et lisible par l'homme ou par une machine .En informatique, le mot document ou document électronique est généralement synonyme de fichier **[5].**

I.4.1.3 Pertinence

La pertinence est une notion fondamentale et cruciale dans le domaine de la RI. Cependant, la définition de cette notion complexe n'est pas simple car elle fait intervenir plusieurs notions. Basiquement, elle peut être définie comme la correspondance entre un document et une requête ou encore comme une mesure d'information de document à la requête **[6].** Essentiellement deux types de pertinences ont été définis :

- ✓ **La pertinence système**

Est déterministe, objective et elle est définie à travers les modèles de la recherche d'information. Elle est souvent traduite par un score évaluant l'adéquation du contenu des documents vis-à-vis de celui de la requête **[7].**

- ✓ **La pertinence utilisateurs**

Est liée à la perception de l'utilisateur sur l'information renvoyée par le système. Elle est subjective, deux utilisateurs peuvent juger différemment un même document renvoyé pour une même requête. Elle peut évoluer dans le temps d'une recherche. Une information jugée non pertinente à l'instant $\boldsymbol{t}$ pour une requête peut être jugée pertinente à $\boldsymbol{t + 1}$ car la connaissance de l'utilisateur sur le sujet a évolué **[7].**

I.4.2 Les processus de la RI

Les processus de RI qui permet, à partir d'une requête, d'ordonnancer les documents est appelé **'' processus en U".** Il est décomposé en trois principales étapes : l'indexation, l'appariement document-requête et la reformulation de la requête. La figure suivante représente l'architecture générale d'un SRI

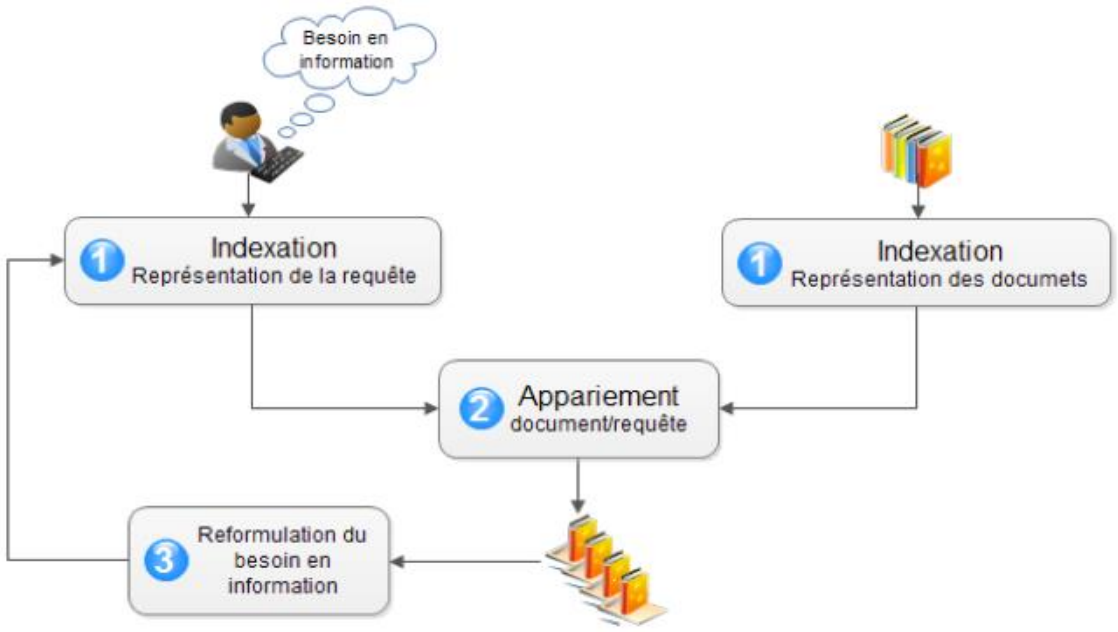


Figure I.1 : Processus en U de recherche d'information [8].

Ce processus est composé de trois fonctions principales

1. L'indexation consiste à extraire et à représenter le contenu des documents de manière interne sous forme d'index. Cette structure d'index permet de retrouver rapidement les documents contenant les mots clés de la requête.

2. Appariement document/requête, une fois les documents sont représentés sous forme interne d'index. Suite à une requête utilisateur, le système calcule la pertinence de chaque document vis-à-vis de la requête utilisateur selon une mesure de correspondance du modèle de RI, et retourne la liste des résultats à l'utilisateur.

3. La reformulation du besoin en information est l'étape qui permet de redéfinir le besoin de l'utilisateur au fur et à mesure de la session de recherche.

I.4.2.1 L'indexation

L'indexation a pour rôle d'extraire à partir d'un document ou d'une requête, une représentation paramétrée qui couvre au mieux à son contenu sémantique. Le résultat de l'indexation constitue le descripteur du document ou de la requête qui est une liste de termes significatifs pour l'unité textuelle correspondante, auxquels sont associés généralement des poids pour différencier leur degré de représentativité. L'ensemble des termes reconnus par le SRI est rangé dans une structure appelé dictionnaire constituant de langage d'indexation **[4].**

Bien que l'indexation se base sur des techniques relativement établies, il peut y avoir plusieurs indexations différentes d'un même texte, aussi valables les unes que les autres, en fonction de l'usage qui doit en être fait et du public auquel elles s'adressent.

L'indexation se décompose en trois phases schématisées dans la figure I.2.

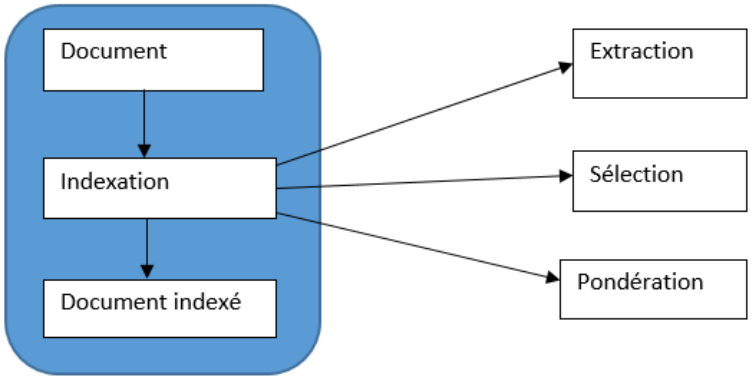


Figure I.2 : Indexation d'un document [09].

On distingue trois types d'indexation :

- **L'indexation manuelle :**

Lors de l'indexation manuelle, un expert dans le domaine choisit les termes qu'il juge pertinents dans la description du contenu sémantique du document. Ce type d'indexation permet d'avoir un vocabulaire d'index contrôlé ce qui permet d'accroître la consistance et la qualité de la représentation obtenue **[10].**

Néanmoins cette indexation présente plusieurs inconvénients :

- ✓ Approche subjective, puisque le choix des termes de l'indexation dépend de l'indexeur et de ces connaissances du domaine.
- ✓ Pratiquement inapplicable aux corpus de textes volumineux.
- ✓ Très couteuse en termes de temps.

- **L'indexation semi- automatique :**

La tâche d'indexation est réalisée ici conjointement par un programme informatique et un spécialiste du domaine **[11].** Une indexation automatique est d'abord lancée, qui extrait un ensemble de termes descripteurs du document.

Le choix final des termes d'indexation à partir de vocabulaire fourni, est laissé à l'indexeur humain (généralement spécialiste du domaine).

- **L'indexation automatique :**

Ce type d'indexation ne fait pas intervenir d'expert. L'indexation automatique repose sur des algorithmes associant automatiquement des descripteurs à des parties de document. Dans le cas des documents textuels, chaque mot est potentiellement un index du document qui le contient. Chaque terme selon un processus défini : extraction, suppression des mots vides, normalisation **[12].** Cette indexation :

- ✓ particulièrement adaptée aux corpus volumineux.
- ✓ adoptée en RI.

a. L'analyse lexicale

L'analyse lexicale permet de convertir le texte d'un document en un ensemble de termes. Un terme est une unité lexicale ou un radical. Elle permet de reconnaitre les espaces de séparation des mots, les chiffres, les ponctuations, etc. **[4].**Ces unités seront les candidats à l'indexation et à la recherche dans une requête. L'analyse lexicale inclut les étapes suivantes :

- ✓ La conversion de la casse (Les majuscules en minuscules).
- ✓ L'élimination des accents.
- ✓ La tokénisation (Une séquence d'unités lexicales élémentaires).

b. L'élimination des mots vides

Un des problèmes majeurs de l'indexation est d'arriver à extraire les termes significatifs tout en évitant les mots vides (Pronoms personnels, préposition, etc.).Les mots vides peuvent aussi être des mots athématiques(Les mots qui peuvent se retrouver dans n'importe quel document parce qu'ils exposent le sujet mais ne le traitent pas, comme par exemple contenir, appartenir, etc.).On distingue deux techniques pour éliminer les mots vides :

- ✓ l'utilisation d'une liste de mots vides (Aussi appelée anti-dictionnaire ou stop- List).
- ✓ l'élimination des mots dépassant un certain nombre d'occurrence dans la collection.

c. La lemmatisation

La lemmatisation consiste à représenter les différentes variantes d'un terme par un format unique appelé lemme ou racine, ce qui a pour effet de réduire la taille de l'index. Plusieurs stratégies de normalisation sont utilisés : la table de correspondance, l'élimination des affixes (L'algorithme de porter) **[13]**, la troncature, l'utilisation de N-grammes. L'inconvénient majeur de cette opération est qu'elle supprime dans certains cas la sémantique des termes originaux.

Par exemple la lemmatisation du verbe « positionner » (forme canonique) peut se faire par rapport à ce genre d'occurrences :
Positionnant, positionnait, positionnées, positions, position, positionna, positionnâtes, positionniez.

d. le choix de descripteurs

Elle consiste à déterminer le type d'unités élémentaires pour représenter les documents. On parle aussi de descripteur. L'objectif est d'avoir une représentation des documents permettant

une moindre perte d'information sémantique possible .On distingue plusieurs types de descripteurs, nous citons :

- ✓ **Les mots simples :** les mots simples du texte de document en éliminant les mots vides.
- ✓ **Les lemmes** ou les racines des mots extraits.
- ✓ **Les N-grammes :** qui sont une représentation originale d'un texte en séquence de N caractères consécutifs. On trouve des utilisations de bi-grammes et trigrammes dans la recherche d'information.
- ✓ **Les mots composés :** groupes de mots ou expressions (phrase en anglais) sont souvent plus riches sémantiquement que les mots qui les composent pris séparément.
- ✓ **Les concepts :** qui sont des expressions pris généralement d'une structure conceptuelle, tels que les thésaurus ou les ontologies.

e. La création de l'index

Un index est une liste de termes représentatifs du contenu de document .Mathématiquement l'index représente une relation binaire :

Index: $\boldsymbol{d_i \rightarrow \{t_{ij}\}}$

Ou $\boldsymbol{d_i}$ document de la collection et $\boldsymbol{t_{ij}}$ le $\boldsymbol{j^{ieme}}$terme de document $\boldsymbol{d_i}$

Des structures de stockage particulière sont nécessaires pour mémoriser les informations sélectionnées lors de processus d'indexation afin d'accélérer la réponse à une requête .Les moyens de stockages les plus utilisés sont :

- ✓ **Le fichier direct (maitre)**

C'est le fichier de base dans lequel sont stockées les données. L'opération peut durer quelques secondes sur un fichier maitre de quelques centaines d'enregistrements, cependant, elle peut se révéler très lente si la base atteint des milliers de documents **[14].**

- ✓ **Le fichier inverse**

Il est créé autour du fichier maitre. Ce fichier comme son nom l'indique est le résultat de l'inversion du fichier maitre. Plus exactement, au lieu de donner pour chaque document les mots et les fréquences qui le constituent, on donne pour chaque mot les documents qui le contiennent et sa fréquence dans chacun des documents **[14].**

La structure d'un fichier inverse associe à un document est composée de deux éléments :

- **Le vocabulaire (dictionnaire) :** ensemble des termes d'index de la collection auquel on peut éventuellement rajouter l'information sur le nombre de documents de la collection ou le terme apparait ou sa fréquence d'occurrence totale.
- **Les occurrences (postings) :** ensemble de toutes les listes contenants chaque terme du vocabulaire .ces listes peuvent éventuellement inclure la fréquence d'occurrence du terme dans le document qui le contient.

I.4.2.2 L'appariement document-requête

Il s'agit de l'expression du besoin d'information (requête), la recherche dans le corpus, et la présentation des résultats. Cette phase nécessite un modèle de représentation du besoin de l'utilisateur, appelé modèle de requêtes, ainsi qu'une fonction de correspondance qui doit évaluer la pertinence des documents par rapport à la requête. La réponse du système est un ensemble de références à des documents qui obtiennent une valeur de correspondance élevée. Cet ensemble est généralement présenté sous la forme d'une liste ordonnée suivant la valeur de correspondance.

- **Fonction de correspondance**

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

$\boldsymbol{d}$: Un document de la collection.

$\boldsymbol{q}$: La requête.

Il existe deux méthodes d'appariement :

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

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

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

Le résultat est une liste de documents censés être pertinents pour la requête. Les documents retournés sont triés selon leur score de pertinence vis-à-vis de la requête.

I.4.2.4 La reformulation de la requête

La reformulation du besoin en information est l'étape qui permet de redéfinir le besoin de l'utilisateur au fur et à mesure de la session de recherche. Cette étape peut être effectuée :

- ✓ **Manuellement**, dans le cas où l'utilisateur soumet lui-même une nouvelle requête.
- ✓ De façon **automatique**, lorsque le système de RI s'appuie sur les termes importants dans les documents les plus pertinents ou visités par l'utilisateur qui sont réutilisés.

Ce processus a pour but d'améliorer la performance du système en offrant un mécanisme de raffinement de la requête utilisateur. Les techniques utilisées durant ce processus se classent en deux catégories **[15]** : les méthodes locales et les méthodes globales.

- ➢ **Les méthodes locales :**

Elles s'appuient sur la technique dite de réinjection de pertinence (relevance feedback). En se basant sur la requête initiale, une liste de documents sera présentée à l'utilisateur, qui va ensuite choisir les documents qu'il juge pertinents. La requête initiale sera ensuite enrichie avec les termes des documents ainsi choisis.

- ➢ **Les méthodes globales**

La requête est réajustée en se basant sur des ressources externes telles que les bases lexicales, les thésaurus et les ontologies. Ainsi, des termes, non nécessairement présents dans la requête initiale, sont suggérés à l'utilisateur pour raffiner sa requête.

La figure 1.4, présente les principales techniques d'amélioration des SRI par reformulation de la requête initiale en y ajoutant de nouveaux termes. La reformulation peut se faire par expansion automatique de la requête, par combinaison de différentes présentations de la requête ou par réinjection de pertinence. Nous présentons dans ce qui suit ces trois principales techniques.

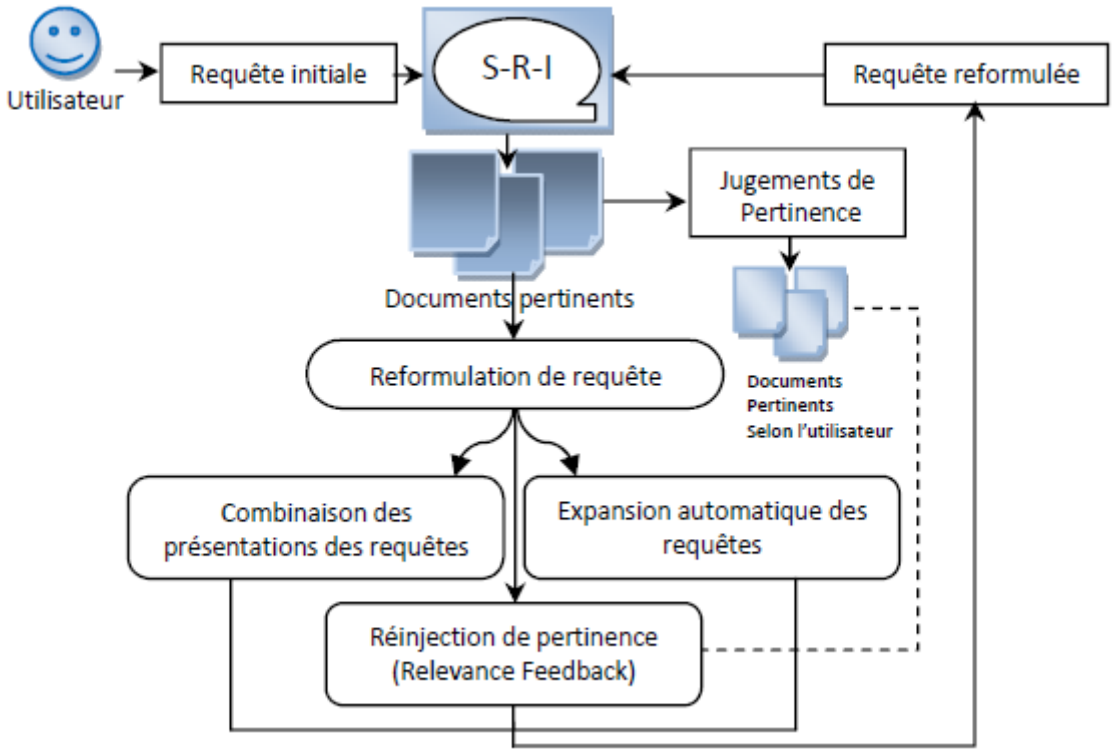


Figure 1.3 Techniques d'améliorations des SRI par reformulation de requêtes [04].

I.5 Les modèles de la RI

Le modèle de recherche représente le noyau d'un SRI. Il comprend la fonction de décision fondamentale qui permet d'associer à une requête, l'ensemble des documents pertinents à restituer. Il est utilisé pour la recherche d'informations proprement dite et est étroitement lié au modèle de représentation des documents et des requêtes **[2].**

Un modèle de RI a pour rôle de fournir une formalisation du processus de RI et un cadre théorique pour la modélisation de la mesure de pertinence.

Plusieurs modèles de recherche d'information en été proposés. Ils se déclinent en trois grandes catégories qui sont les modèles booléens, les modèles vectoriels et les modèles probabilistes.

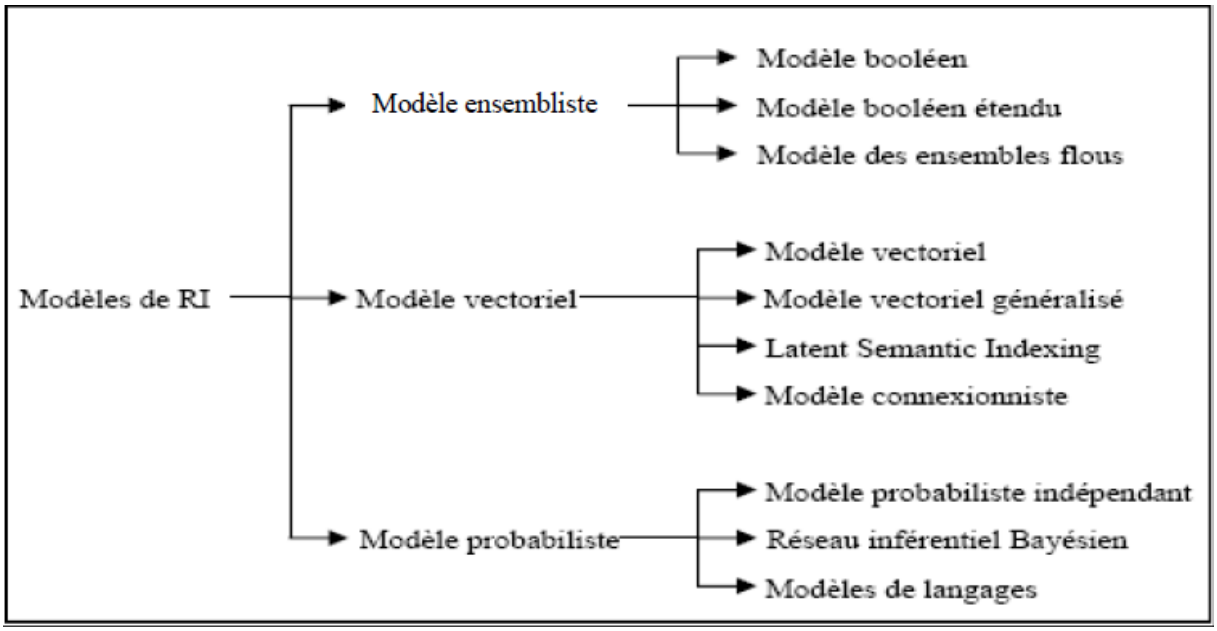


Figure I.4 : Taxonomie des modèles en RI [15].

I.5.1 Les modèles ensemblistes

Ce sont les modèles basé sur la théorie des ensembles et l'algèbre de Bool, dans le représentant le plus connu est le modèle booléen.

I.5.1.1 Les modèle booléens

Dans ce modèle, le document est représenté par un ensemble de termes. La requête est représentée par un ensemble de mots clés relies par des opérateurs booléens ($\boldsymbol{AND, OR\ et\ NOT}$) **[8].** L'appariement requête-document est strict et se base sur des opérations ensemblistes selon les règles suivantes :

$$\boldsymbol{RSV\ (d, t_i) = 1\ si\ t_i \in d\ 0\ si\ non} \quad \textbf{(I.1)}$$

$$\boldsymbol{RSV\ (d, t_i\ AND\ t_j) = 1\ si\ (t_i \in d) \wedge (t_j \in d), 0\ si\ non} \quad \textbf{(I.2)}$$

$$\boldsymbol{RSV\ (d, t_i\ OR\ t_j) = 1\ si\ (t_i \in d) \vee (t_j \in d), 0\ si\ non} \quad \textbf{(I.3)}$$

$$\boldsymbol{RSV\ (d, NOT\ t_i) = 1\ si\ t_i \notin d, 0\ si\ non} \quad \textbf{(I.4)}$$

Avec :

$\boldsymbol{q}$ **:** Requête.

$\boldsymbol{d}$ **:** Document.

$\boldsymbol{t_i}$**:** Terme.

$\boldsymbol{RSV}\ (d, q)$ (**R**etrieval **S**tatut **V**alue) : fonction de correspondance entre un document d et différentes formes de requêtes.

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

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

Et pour résoudre certaines problèmes de ce modèle, des extensions ont été proposées parmi elle on trouve : le modèle booléen basé sur la théorie des ensembles flous **[16]** ,le modèle booléen étendu **[17].**

I.5.2 Les modèles algébriques

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

I.5.2.1 Modèle vectoriel

Dans ce modèle, un document est représenté sous forme d'un vecteur dans l'espace vectoriel composé de tous les termes d'indexation. Les coordonnées d'un vecteur document représentent les poids des termes correspondants **[8].** Formellement, un document $\boldsymbol{d_i}$ est représenté par un vecteur de dimension $\boldsymbol{n}$

$$\boldsymbol{d_i = (w_{i1}, w_{i2}, \dots, w_{in})\ \ pour\ i = 1, 2, \dots, m.} \qquad \textbf{(I.5)}$$

Avec :

$\boldsymbol{w_{ij}}$ est le poids du terme $\boldsymbol{t_j}$dans le document $\boldsymbol{d_i}$,

$\boldsymbol{m}$ est le nombre de documents dans la collection,

$\boldsymbol{n}$ est le nombre de termes d'indexation.

Une requête $\boldsymbol{q}$ est aussi représentée par un vecteur de mots-clés défini dans le même espace vectoriel que le document.

$$\boldsymbol{q = (w_{q1}, w_{q2}, \dots, w_{qn})} \qquad \textbf{(I.6)}$$

Avec :

w_{qj} est le poids de terme t_j dans la requête q. Ce poids peut être soit une forme de $tf * idf$ (Pondération), soit un poids attribué manuellement par l'utilisateur.

Les facteurs tf et idf permettent de combiner les pondérations locale et globale d'un terme :

- ✓ tf (**T**erm **F**requency) : cette mesure est proportionnelle à la fréquence de terme dans le document (pondération locale) .Elle peut être utilisée telle quelle ou selon plusieurs déclinaisons(log(tf),présence/absence ,etc.).
- ✓ idf (**I**nverse of **D**ocumment **F**requency) : ce facteur mesure l'importance d'un terme dans toute la collection (pondération globale). Un terme qui apparait souvent dans la base documentaire ne doit pas avoir le même impact qu'un terme moins fréquent. Il est généralement exprimé comme suite : $log(N/df)$ où df est le nombre de documents contenant le terme et N est le nombre total de documents de la base documentaire.

- ✓ La mesure $tf * idf$ est une bonne approximation de l'importance d'un terme dans un document, particulièrement dans des corpus de documents de tailles homogènes.

La pertinence du document 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 suivantes :

- **Le produit scalaire :**

$$RSV(d,q) = \sum_{j=1}^{n} w_{qj} * w_{ij} \qquad \textbf{(I.7)}$$

- **La mesure du cosinus :**

$$RSV(d,q) = \frac{\sum_{j=1}^{n} w_{qj} * w_{ij}}{(\sum_{j=1}^{n} w^2_{qj})^{1/2} * (\sum_{j=1}^{n} w^2_{ij})^{1/2}} \qquad \textbf{(I.8)}$$

- **La mesure de Dice :**

$$RSV(d,q) = \frac{2 * \sum_{j=1}^{n} w_{ij} * w_{qj}}{\sum_{j=1}^{n} w^2_{qj}) + \sum_{j=1}^{n} w^2_{ij}} \qquad \textbf{(I.9)}$$

- **La mesure de Jacard :**

$$RSV(d,q) = \frac{\sum_{j=1}^{n} w_{ij} * w_{qj}}{\sum_{j=1}^{n} w_{ij} * w_{qj} + \sum_{j=1}^{n} w_{ij} * w_{qj}} \quad \textbf{(I.10)}$$

L'un des avantages du modèle vectoriel réside dans sa simplicité conceptuelle et de mise en œuvre. En outre, il permet de trier les résultats d'une recherche à travers une mesure de similarité document/requête, en plaçant en tête les documents jugés les plus similaires à la requête. Cependant, ce modèle ne permet pas de modéliser les associations entre les termes d'indexation. Chacun des termes est considéré comme indépendant des autres.

I.5.3 Les modèles probabilistes

Ces modèles reposent sur la théorie des probabilités, dans ces modèles, la pertinence d'un document vis-à-vis d'une requête est vue comme une probabilité de pertinence document/requête.

I.5.3.1 Le modèle probabiliste de base

Le principe de base de ce modèle consiste à présenter les résultats d'un SRI dans un ordre basé sur la probabilité de pertinence d'un document vis-à-vis d'une requête. Etant donnée une requête utilisateur notée $\boldsymbol{q}$ et un document $\boldsymbol{d}$, le modèle probabiliste tente d'estimer la probabilité que le document $\boldsymbol{d}$ appartienne à la classe des documents pertinents (non pertinents) **[9].**Un document est alors sélectionné si la probabilité qu'il soit pertinent à $\boldsymbol{q}$, notée $\boldsymbol{P(R|d)}$**,**est supérieur à la probabilité qu'il soit non pertinent à $\boldsymbol{q}$,noté $\boldsymbol{P(NR|d)}$Le score d'appariement entre le document et la requête $\boldsymbol{q}$ noté $\boldsymbol{RSV(d,q)}$,est donnée par :

$$RSV(d,q) = \frac{P(R|d)}{P(NR|d)} \quad \textbf{(I.11)}$$

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

$$RSV(d,q) = \frac{P(d|R)}{P(d|NR)} \quad \textbf{(I.12)}$$

Différentes méthodes sont utilisées pour estimer ces différentes probabilités. La plus connue est celle du modèle BIR(Binary Independance Retrieval). On considère dans ce modèle que la variable document $\boldsymbol{d(t_1 = x_1, t_2 = x_2, .., t_n = x_n)}$ est représenté par un ensemble

d'évènements qui dénotent la présence ($x_i = 1$) ou l'absence ($x_i = 0$) d'un terme dans un document. En supposant que ces évènements soit indépendants, d'où l'appellation BIR, les probabilités de pertinence (respectivement de non pertinence) d'un document, notées $P(d|R)$(Respectivement $P(d|NR)$), sont données par :

$$P(d|R) = P(t_1 = x_1, t_2 = x_2, \dots, tn = t_n = x_n|R) = \prod_{i=1}^{n} P(t_i = x_i|R) \quad \textbf{(I.13)}$$

$$P(d|NR) = P(t_1 = x_1, t_2 = x_2, \dots, tn = t_n = x_n|R) = \prod_{i=1}^{n} P(t_i = x_i|NR) \quad \textbf{(I.14)}$$

La distribution des termes suit une loi de Bernoulli ; $P(d|R)$ peut s'écrire :

$$P(d|R) = \prod_{i=1}^{n} P(t_i = 1|R)^{x_i} * P(t_i = 0|R)^{1-x_i} \quad \textbf{(I.15)}$$

$$P(d|NR) = \prod_{i=1}^{n} P(t_i = 1|NR)^{x_i} * P(t_i = 0|NR)^{1-x_i} \quad \textbf{(I.16)}$$

En posant :

$P(t_i = 1|R) = p_i$ $\qquad$ $P(t_i = 0|R) = 1 - p_i$

$P(t_i = 1|NR) = q_i$ $\qquad$ $P(t_i = 0|NR) = 1 - q_i$

$RSV\ (d, q)$ Peux s'écrire, après transformation comme suit :

$$RSV(d, q) = \frac{p_i^{x_i} * (1-p_i)^{(1-x_i)}}{q^{x_j} (1-q_i)^{(1-x_j)}} \quad \textbf{(I.17)}$$

En se ramenant à la fonction log et après un petit développement, la fonction *RSV* s'écrit alors

$$RSV(d, q) = \sum_{i;x_i=1} log \frac{p_i(1-q_i)}{q_i(1-p_i)} \quad \textbf{(I.18)}$$

Pour estimer les probabilités p_i et q_i on suppose l'existence d'une collection de test, et si en outre en propose connus l'ensemble R des documents pertinents et l'ensemble NR des documents non pertinents alors en peux aisément estimer les probabilités p_i et q_i en utilisant les propositions définie en tableau suivant comme suite :

	Contenant ti	Non contenant ti	Totale
#doc pertinents	r_i	$n - r_i$	N
#doc non pertinents	$R_i - r_i$	$N - n - R_i + r_i$	$N - n$
Totale (collection)	R_i	$N - R_i$	N

Tableau I.1 Distribution de probabilités de pertinence des termes d'un corpus d'apprentissage.

En a donc :

$\boldsymbol{p_i = \frac{r_i}{n}}$ **et** $\boldsymbol{q_i = \frac{Ri - r_i}{N - n}}$

Ainsi la $\boldsymbol{RSV\ (d/q)}$ se réduit à :

$$\boldsymbol{RSV(d,q) = \sum_{i;xi=1} log \frac{r_i(N - R_i - n + r_i)}{(1 - r_i)(R_i - r_i)}} \quad \textbf{(I.19)}$$

Les reproches faits à ce modèle sont deux types :

- ✓ L'indépendance des termes n'est pas toujours vérifiée (en particulier pour les termes concurrents)
- ✓ L'impossibilité d'estimer ses paramètres si des collections d'entrainement ne sont pas disponibles.

I.5.3.2 Le modèle de langue

Les modèles de langue émergent comme une nouvelle tendance en recherche d'informations. Le nom «modèle de langue » est hérité du domaine de la reconnaissance de la parole, ou on tente de créer un modèle statistique pour modéliser une langue et ainsi de déterminer la probabilité d'apparition d'un mot en fonction du modèle et des signaux acoustique **[10].**

Contrairement aux modèles classique de RI, les modèles de langue ne modélisent pas directement la notion de pertinence d'un document $\boldsymbol{d}$ face à une requête $\boldsymbol{q}$. La pertinence d'un document vis-à-vis d'une requête est vu comme la probabilité que la requête soit généré par le modèle de langue du document.

Un modèle de langue se construit en utilisant une fonction de probabilité $\boldsymbol{P}$ qui assigne une probabilité $\boldsymbol{P(s)}$ à un mot ou à une séquence de mots $\boldsymbol{s}$ dans un corpus. Cette fonction permet

alors d'estimer la probabilité d'une séquence quelconque de mots dans la langue modélisée, ou de façon plus générale, d'estimer la probabilité de générer cette séquence de mots à partir du modèle de langue.

Formellement, soit $\boldsymbol{M_d}$ le modèle de langue du document $\boldsymbol{d}$; la pertinence de vis-à-vis d'une requête $\boldsymbol{q}$ revient à estimer$\boldsymbol{P(q|M_d)}$, c'est-à-dire, la probabilité que la requête $\boldsymbol{q}$ soit générer par $\boldsymbol{M_d}$. Etant donne une requête $\boldsymbol{q}$, cette pertinence est mesurée par :

$$\boldsymbol{RSV(d,q) = P(q = (t_1, t_2, \dots., t_n)|M_d) = \prod_i^n p(t_i|d)} \quad \textbf{(I.20)}$$

$\boldsymbol{p(t_i|d)}$peut être estimer en se basant sur l'estimation maximale de vraisemblance du terme ti. Elle est données par :

$$\boldsymbol{P(t_i) = \frac{|t_i|}{\sum t_i} = \frac{Nbr\ d'occurence du\ terme\ t\ dans\ C}{Nbr\ totale\ des\ terme\ dans\ C}} \quad \textbf{(I.21)}$$

Donc en aura :

$$\boldsymbol{P(t_i|d) = \frac{tf(t_i|d)}{\sum_t tf(t|d)}} \quad \textbf{(I.22)}$$

I.6 Evaluation des systèmes de recherche d'information

L'évaluation d'un système de RI, permet de vérifier l'efficacité des modèles mis en œuvre pour l'identification des documents pertinents **[6].**

Plusieurs critères peuvent entrer en jeu dans le processus d'évaluation d'un SRI, tels que :

- Le temps de réponse
- La présentation claire des résultats
- L'effort fourni par l'utilisateur pour récupérer l'information pertinente
- La pertinence des documents retournés.

Dans cette section, nous présentons le cadre d'évaluation d'un système de RI ainsi que les mesures d'évaluation sous-jacentes.

I.6.1 Les collections de test

Une collection (ou corpus) de test constitue le moyen d'évaluation des SRI. Elle est généralement composée d'un ensemble de document, d'un ensemble de requêtes et des jugements de pertinence, associés à ces requêtes. L'évaluation d'un SRI, consiste à comparer

les résultats retournés par ce dernier par rapport aux jugements de pertinence. Des mesures d'évaluations, décrites dans la section suivante, sont utilisées pour effectuer cette comparaison .Les collections de test sont les résultats de projets d'évaluation qui se sont multipliés depuis les années 1970, on peut citer la collection CACM, la collection CISI, la campagne CLEF et la campagne TREC.

La compagne TREC consiste à ce jour la compagne de référence dans le cadre de l'évaluation des systèmes de recherche d'information et cela depuis son lancement 1992. L'objectif de cette compagne est de proposer une plate-forme qui réunit des collections de test, des tâches spécifiques et des protocoles d'évaluation pour chaque tâche afin de mesurer les différentes stratégies de recherche**[4].**

I.6.3 Les mesures d'évaluation

La démarche de validation en RI se base sur l'évaluation expérimentale des performances du modèle ou du système proposé. L'évaluation des performances d'un modèle de RI, permet de paramétrer le modèle, d'estimer l'impact de chacune de ses caractéristiques et de fournir des éléments de comparaison entre modèles.

Cette évaluation peut porter sur plusieurs critères : le temps de réponse, la pertinence, la qualité et la présentation des résultats, etc. Le critère le plus important est celui qui mesure la capacité du système à satisfaire le besoin en information de l'utilisateur, c'est à dire la pertinence. Deux facteurs permettent d'évaluer ce critère. Le premier est le **taux de rappel** et le deuxième est **le taux de précision**.

- **Rappel :**

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

$$\text{Rappel} = \frac{\text{nombre de documents pertinents trouvés}}{\text{nombre de documents pertinents}}$$

- **Précision :**

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

$$\text{Précision} = \frac{\text{nombre de documents pertinents trouvés}}{\text{nombre de documents trouvés}}$$

Le rappel et la précision permettent aussi de définir le silence documentaire et le bruit documentaire qui représentent respectivement le nombre des documents pertinents non retrouvés et le nombre de documents non pertinents retrouvés. La figure 1.2 schématise ces deux variables dans l'ensemble des documents.

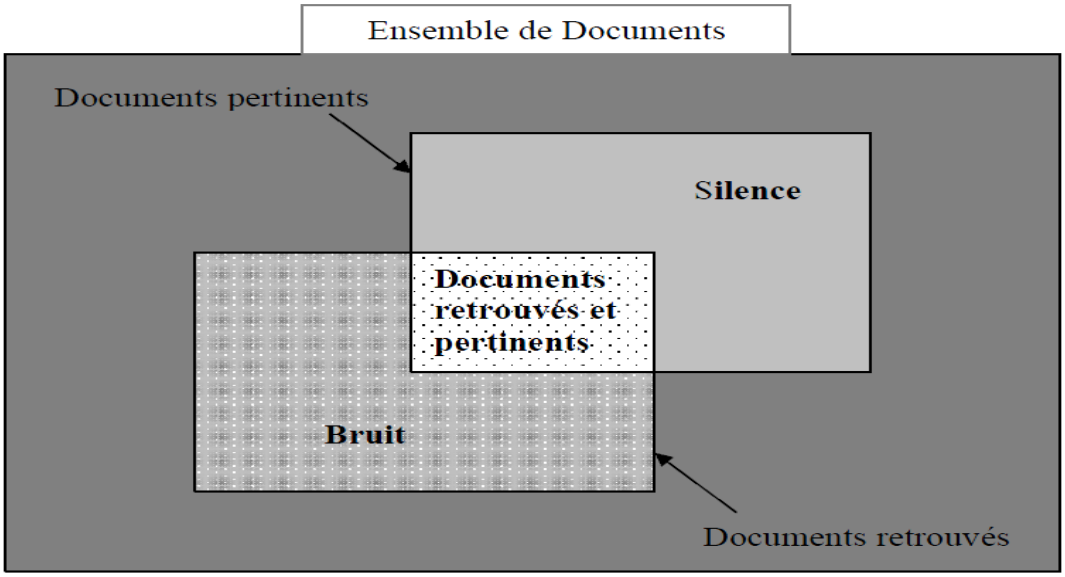


Figure I.5 : Bruit et Silence [18]

I.6.3.5 Les mesures alternatives

Plusieurs mesures alternatives ont été proposées, parmi elles, nous trouvons :

- **R-précision** : cette précision mesure la proportion des documents pertinents retrouves après que R documents ont été retrouves, ou R est le nombre de documents pertinents pour la requête considérée **[11].**

- **Précision moyenne (Mean Average Precision-MAP)**

C'est la moyenne des précisions moyennes (Mean Average precision) obtenues sur l'ensemble des requêtes à chaque fois qu'un document pertinent est retrouvé :

$$MAP = \frac{\sum_{q \in Q} APq}{|Q|}$$

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

I.7 Conclusion

Dans ce premier chapitre, nous avons présenté, les notions de base de la RI .En premier lieu nous avons introduit les éléments principales de la RI : requête, document et pertinence. Nous avons aussi décrit les différentes étapes d'un processus de RI, à savoir, l'indexation, l'appariement et la reformulation de la requête. Et enfin nous avons traité les modèles de la RI et l'évaluation des systèmes de RI. Ce chapitre nous a aidé à comprendre les fondements de la recherche d'information pour avancer dans notre travail.

II.1 Introduction

Le word embedding, est une technique pour capturer le «sens» d'un mot via un vecteur à faible dimension. Ils sont utitilisés dans de nombreuses tâches de la recherche d'informations (RI) et dans le traitement de la langue naturelle (PNL), telles que questions réponses ou la traduction d'une langue à une autre.

Dans ce chapitre nous présentons le word embeddibng et les modèles de recherche basé sur le word embedding qui sont réalisés à base de réseaux de neurones, puis le fonctionnement de ces modèles. Et enfin nous présentons quelques travaux utilisons le word embedding en Ri.

II.2 Historique

L'apprentissage non supervisé de word embedding a exceptionnellement connu de succès dans nombreuses taches de PNL (Processing Natural Language) qui consiste à remplacer les anciennes méthodes utilisé dans la représentation distribué.

Depuis les années 1990, les modèles spatiaux vectoriels ont été utilisés dans la distribution sémantique. Au cours de cette période, de nombreux modèles pour estimer les représentations continues de mots ont été développés, y compris l'analyse sémantique latente (LSA) et l'allocation latente de Dirichlet (LDA).

Bengio et al. Ont utilisé le terme embeddings en 2003 et ils l'ont utilisé dans un modèle de langue neuronal. En premier lieu Collobert et Weston en 2008**[19]**, ont montré l'utilité des word embedding. Ils ont conçu une architecture unifiée pour le traitement du langage naturel qui établit non seulement des word embeddings comme une technique utile pour les tâches en aval, mais introduit également une architecture de réseau neuronal qui constitue la base de nombreuses approches actuelles.

Cependant, la notion de word embedding a étais connu et propagé grâce à l'étude effectuer par Mikolov et al en 2013. **[20]** qui a créé word2vec, un outil qui permet une formation et une utilisation parfaites des embeddings.

En 2014, Pennington et al. Ont lancé GloVe, qui est un autre modèle de word embedding.

II.3 Définition de word embedding

Le ***word embedding*** est une méthode d'apprentissage automatique qui vise à représenter les mots d'un vocabulaire dans des vecteurs de réels dans un espace à faible dimension. En s'appuyant sur un grand corpus de textes, de telles représentations vectorielles peuvent être calculées pour capturer à la fois des informations syntaxiques et sémantiques sur les mots. Ces

word embeddings, lorsqu'ils sont ensuite utilisés comme données d'entrée, se sont révélés être un grand atout pour une grande variété de tâches en traitement automatique du langage naturel (TALN) **[21].**

Avec le word embedding le mot est représenté sous forme d'un vecteur de nombres réels. Ce vecteur code la position du mot dans un espace multidimensionnel dans lequel les mots proches syntaxiquement ou sémantiquement sont également proches dans cet espace. Par exemple les vecteurs qui représentent les villes du monde sont très proches dans ce nouvel espace. Ces représentations se nomment « word embedding ». La figure II.1 illustre dans une image 2D cette représentation. On voit que tous les mots associés aux nombres, qu'ils soient écrits en chiffres ou en lettres sont proches. Que les noms de métiers sont regroupés également. Alors que d'un point de vue symbolique, tous ces mots sont totalement différents entre eux.

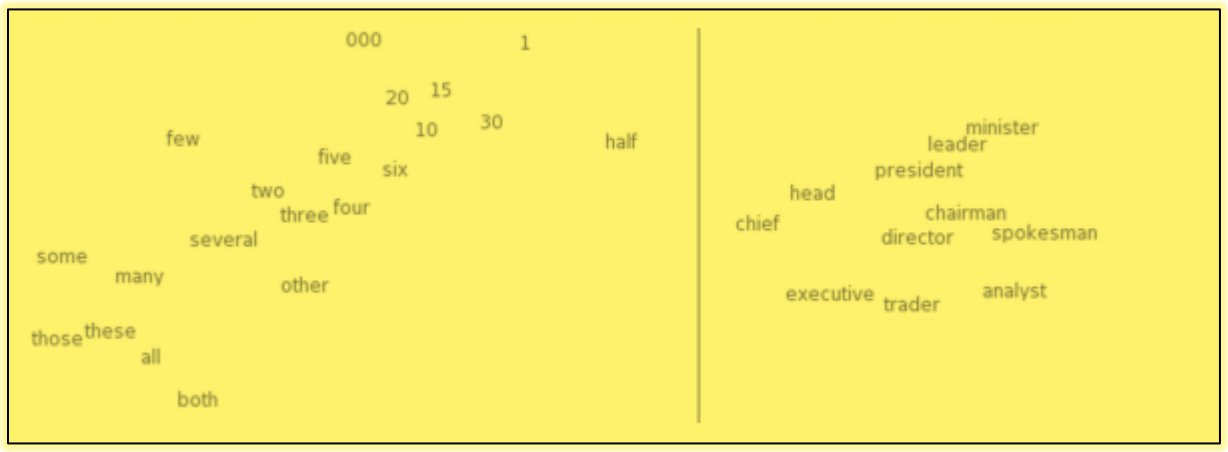


Figure II.1 Les visualisations des embeddings de mots. A gauche : numéro région ; à droite : région emploi [22].

Ces représentations montrent d'autres propriétés intéressantes : les analogies entre les mots sont représenté par la différence entre leurs vecteurs. Par exemple il semble qu'il y ait une différence constante male-femelle :

$$\boldsymbol{W(woman) - W(man) \simeq W(aunt - W(uncle)}$$

$$\boldsymbol{W(woman) - W(man) \simeq W(queen) - W(king)}$$

Ces nouvelles représentations sont tout à fait pertinentes et ont permis d'améliorer les performances de nombreuses tâches de traitement des langues **[23]**.

II.4 Les réseaux de neurones artificiels

II.4.1 Définition

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau **[24].**

D'une manière générale le réseau de neurone comporte (**Figure II.2**)

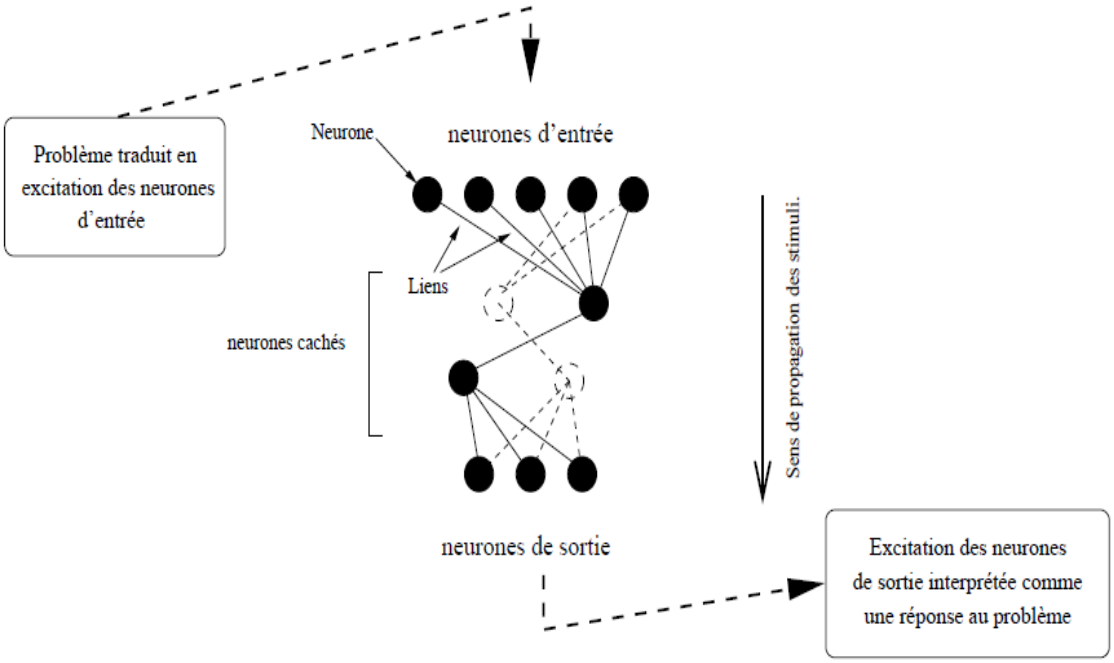


Figure II.2 : Schéma générale d'un réseau de neurones [25].

1. Des neurones d'entrée, auxquels on attribue une extraction en fonction des données que le réseau doit traiter
2. D'autres neurones au travers desquels l'excitation des neurones d'entrée se propage et est modifiée ;
3. Des neurones de sortie dont l'état d'excitation fournit une réponse au problème posé en entrée.

II.4.2 Apprentissage d'un réseau de neurones

On appelle « apprentissage » des réseaux de neurones la procédure qui consiste à estimer les paramètres des neurones du réseau, afin que celui-ci remplisse au mieux la tâche qui lui affectée **[26].**

Au niveau des algorithmes d'apprentissage, il y a deux grandes classes selon que l'apprentissage est dit « supervisé » ou « non supervisé »

- ✓ **Apprentissage supervisé**

L'apprentissage supervisé est une méthode qui utilise directement les connaissances d'un expert et essaye de reproduire ces connaissances **[27].**

- ✓ **Apprentissage non supervisé**

L'apprentissage non supervisé est une méthode qui essaye de dériver des généralisations à partir des données, de segmenter l'espace de données **[28].**

I.4.3 Fonctionnement d'un neurone

Un neurone est défini par trois caractéristiques : son état, ses connexions avec d'autres neurones et sa fonction de transfert ou d'activation comme illustre la figure suivantes :

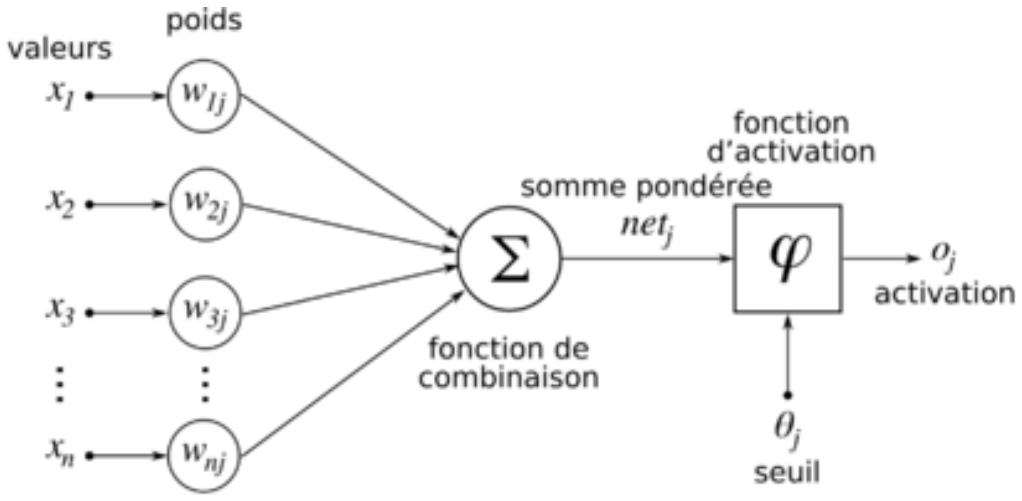


Figure II.3 Structure d'un neurone artificiel [29].

- ✓ Chaque neurone artificiel est un processeur élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones amont.

- ✓ A chacune de ces entrées est associé un poids $\boldsymbol{w}$(weight) représentatif de la force de la connexion.
- ✓ Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones avals.

Comportement de neurone artificiel

Il passe par deux phases :

1. Un neurone reçoit $\boldsymbol{n}$ entrées (des neurones de la couche précédente), $\boldsymbol{x1}, \ldots, \boldsymbol{xn}$ et produit une sortie. La première étape est habituellement le calcul de la somme pondérée des entrées selon l'expression suivante :

$$\boldsymbol{S}(\sum_{i=1}^{n} \boldsymbol{W_i X_i} + \boldsymbol{W_0}) \qquad \textbf{(II.1)}$$

Où les ($\boldsymbol{wi}$) sont des nombres réels, appelés poids, qui pondèrent les entrées

2. A partir de la valeur de la somme, une fonction de transfert appelée fonction d'activation calcule la valeur de l'état du neurone. C'est cette valeur qui sera transmise aux neurones avals.

I.4.4 Fonction d'activation

La **fonction d'activation** (ou **fonction de seuillage**, ou encore **fonction de transfert**) sert à introduire une non-linéarité dans le fonctionnement du neurone **[30].**

Les fonctions de seuillage présentent généralement trois intervalles :

1. en dessous du seuil, le neurone est non-actif (souvent dans ce cas, sa sortie vaut 0 ou -1) ;
2. aux alentours du seuil, une phase de transition ;
3. au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1).

La figure suivante montre des exemples de fonctions d'activation

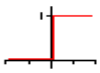
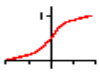
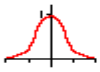
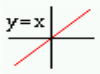
Pas unitaire		$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$
Sigmoïde		$f(x) = \frac{1}{1+e^{-\beta x}}$
Linéaire Seuillée		$f(x) = \begin{cases} 0 & \text{if } x \leq x_{min} \\ mx+b & \text{if } x_{max} > x > x_{min} \\ 1 & \text{if } x \geq x_{max} \end{cases}$
Gaussienne		$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$
Identité		$f(x) = x$

Figure II.4 Exemple de fonctions d'activation [31].

II.4.5 Architecture des réseaux de neurones

Un réseau de neurones est un ensemble de processeurs élémentaires, les neurones qui sont largement connectés les uns aux autres et qui sont capables d'échanger des informations au moyen des connexions qui les relient **[32].**

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Parmi ses modèles en trouve.

- **Réseau multicouche** : les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche et les connexions ne se font qu'avec les neurones des couches avales. Comme illustre la figure suivante :

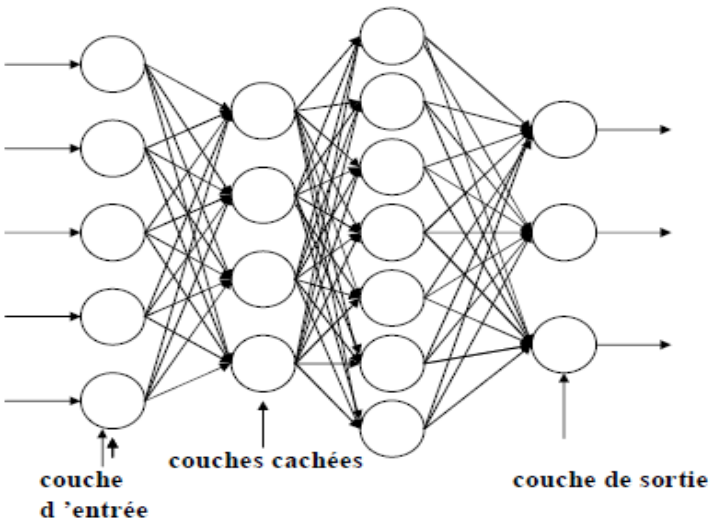


Figure II.5 : réseau multicouche avec deux couches cachées [33]

- **Réseau à connexions récurrentes** : les connexions récurrentes ramènent l'information en arrière par rapport au sens de propagation défini dans un réseau multicouche.

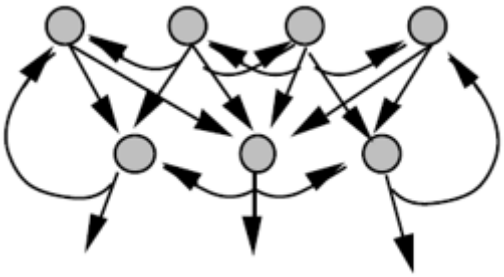


Figure II.6 : Réseau à connexions récurrentes [34]

- **Réseau à connexion complète** : c'est la structure d'interconnexion la plus générale. Chaque neurone est connecté à tous les neurones du réseau

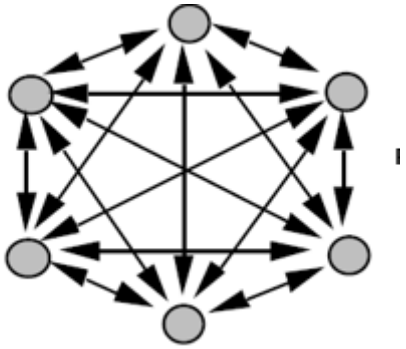


Figure II.7 : Réseau à connexion complète [34].

II.5 Modèles de langue neuronaux pour la RI

Les représentations distribuées de mots et de documents sont connues depuis longtemps en **RI [35]** ces type de représentation de mots sont très étroitement liés avec les modèles de langue. La qualité des modèles de langue est mesurée en fonction de leur capacité à apprendre une distribution de probabilité sur les mots dans un vocabulaire $\boldsymbol{V}$. En fait, de nombreux modèles d'intégration de mots à la fine pointe de la technologie tentent de prédire le prochain mot dans une séquence, on prenant considération certaine mesure. En outre, les modèles d'intégration de mots sont souvent évalués à l'aide de la perplexité,

Avant d'entrer dans les détails des modèles d'intégration de mots, nous allons brièvement parler de certains fondamentaux de modélisation de langue**.** Nous supposons les normes de notation suivantes :

Nous supposons un corpus d'apprentissage contenant une séquence de mots d'apprentissage $\boldsymbol{T}$, $\boldsymbol{w1}, \boldsymbol{w2}, \boldsymbol{w3} \ldots, \boldsymbol{wT}$ des mots qui appartient à un vocabulaire $\boldsymbol{V}$ dont la dimension est $|\boldsymbol{V}|$. Les modèles considèrent généralement un contexte de mots du **n** mots. Nous associons chaque mot avec une intégration d'entrée $\boldsymbol{Vw}$ avec des dimensions $\boldsymbol{d}$ et une intégration de sortie $\boldsymbol{V'w}$ .Nous optimisons finalement une fonction objective $\boldsymbol{J_\theta}$ par rapport à nos paramètres de modèle $\boldsymbol{\theta}$ et notre modèle produit un score $\boldsymbol{f_\theta(x)}$ pour chaque entrée $\boldsymbol{x}$.

Les modèles de langue tentent généralement de calculer la probabilité d'un mot $\boldsymbol{w_t}$ avec les mots précédents $\boldsymbol{n-1}$, c'est-à-dire $\boldsymbol{p(w_t|w_{t-1}, \cdots w_{t-n+1})}$. En appliquant la règle de la chaîne de Markov, nous pouvons rapprocher le produit d'une phrase ou d'un document complet en fonction des probabilités de chaque mot donné avec ses $\boldsymbol{n}$ mots précédents :

$$\boldsymbol{p(w_1, \cdots, w_t) = \prod_i p(w_i|w_{i-1}, \cdots w_{i-n+1})} \qquad \textbf{(II.2)}$$

Dans les modèles de langue basés sur le n-gramme, on peut calculer la probabilité d'un mot en fonction des fréquences de ses n-grammes constitutifs :

$$\boldsymbol{p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{count(w_{t-n+1}, \cdots, w_{t-1}, w_t)}{count(wt-n+1, \cdots, wt-1).}} \qquad \textbf{(II.3)}$$

Le $\boldsymbol{n = 2}$ donne des probabilités de bigram, tandis que $\boldsymbol{n = 5}$ avec le lissage de Kneser-Ney conduit à des modèles lissés de 5 grammes qui ont été considérés comme une base de référence forte pour la modélisation de langue.

Dans les réseaux de neurones, nous atteignons le même objectif en utilisant la fonction de softmax bien connue :

$$\boldsymbol{p(w_t|w_{t-1}, \cdots, w_{t-n+1}) = \frac{exp(h^\top v'_{wt})}{\sum wi \in V exp(h^\top v' wi))}} \qquad \textbf{(II.4)}$$

Le produit intérieur $\boldsymbol{h^\top v'_{wt}}$ est le vecteur de sortie de l'avant-dernière couche de réseau (la couche cachée dans le réseau multicouche), tandis que $\boldsymbol{v'_w}$est l'intégration de sortie du mot $\boldsymbol{w}$, c'est-à-dire sa représentation dans la matrice de poids de la couche softmax. Notez que, bien que $\boldsymbol{v'_w}$représente le mot $\boldsymbol{w}$, il est appris séparément de l''intégration de mot d'entrée $\boldsymbol{v_w}$, car

les multiplications dans lesquelles les deux vecteurs sont impliqués diffèrent ($\boldsymbol{v_w}$ est multiplié par un vecteur d'index, $\boldsymbol{v'_w}$ avec $\boldsymbol{h}$).

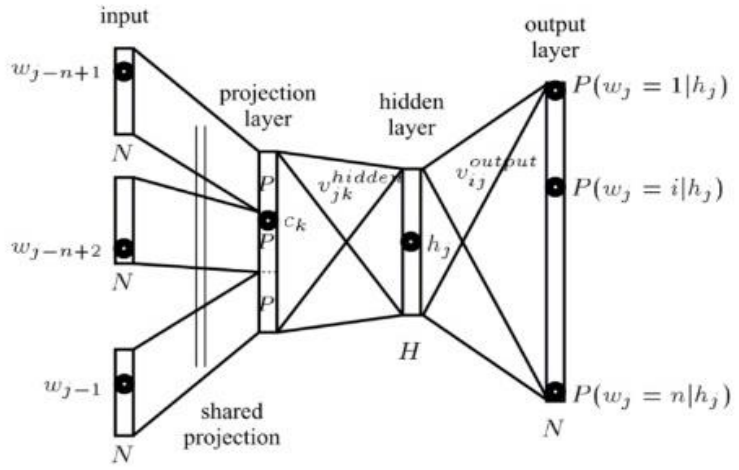


Figure II.8 Un modèle de langue neuronale [36]

Notez que nous devons calculer la probabilité de chaque mot $\boldsymbol{w}$ à la couche de sortie du réseau neuronal. Pour ce faire efficacement, nous effectuons une multiplication matricielle entre $\boldsymbol{h}$ et une matrice de poids représenté par $\boldsymbol{v'_w}$ de tous les mots $\boldsymbol{w}$ dans $\boldsymbol{V}$. Nous obtenons ensuite le vecteur résultant, qui est souvent appelé logit.

En utilisant cette fonction softmax, le modèle tente de maximiser la probabilité de prédire le mot correct à chaque étape $\boldsymbol{t}$. L'ensemble du modèle tente donc de maximiser la probabilité logarithmique moyenne de l'ensemble du corpus :

$$\boldsymbol{J\theta = \frac{1}{T} log\ p(w1, \cdots, wT)} \qquad \textbf{(II.5)}$$

Par analogie, grâce à l'application de la règle de la chaîne, il est généralement formé pour maximiser la moyenne des probabilités de logarithme de tous les mots dans le corpus donné leurs $\boldsymbol{n}$ mots précédents :

$$\mathbf{J\theta = \frac{1}{T}\sum_{t=1}^{T} log\ p(w_t|w_{t-1}, \cdots, w_{t-n+1})} \qquad \textbf{(II.6)}$$

II.6 Exemple des modèles de langue neuronal

II.6.1 Le modèle classique de langue neuronal

Le modèle classique de langue neuronale proposé par Bengio et al. en 2003 se compose d'une couche caché qui prédit le mot suivant dans une séquence comme dans la figure **II.9** :

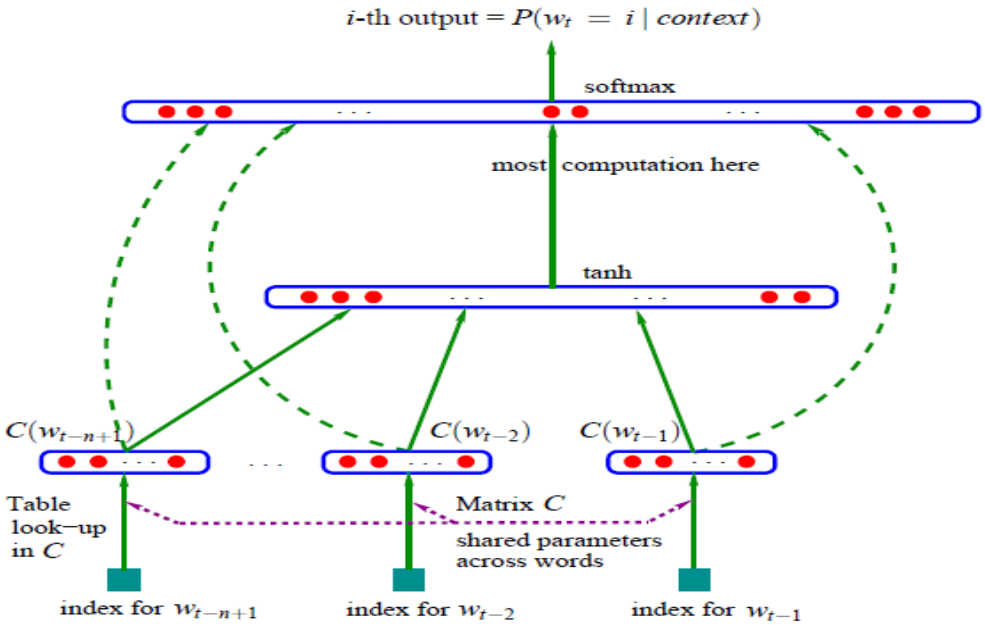


Figure II.9 Modèle de la langue neuronal classique [37]

Leur modèle maximise ce que nous avons décrit ci-dessus comme l'objectif du modèle de langue neuronal prototypique :

$$J\theta = \frac{1}{T}\sum_{t=1}^{T} log\, f(wt_t, w_{t-1}, \cdots, w_{t-n+1}) \qquad \textbf{(II.7)}$$

$f(w_t, w_{t-1}, \cdots, w_{t-n+1})$ Est la sortie du modèle, c'est-à-dire la probabilité $p(w_t \mid w_{t-1}, \cdots, w_{t-n+1})$ calculé par le softmax, où n est le nombre de mots précédents introduits dans le modèle.

Bengio et al. sont l'un des premiers à introduire ce que nous appelons maintenant en tant que word embedding, un vecteur de fonctionnalité de mots à valeur réelle dans R. Leur architecture constitue une grande partie de prototype sur lequel les approches actuelles se sont progressivement améliorées. Cependant, les blocs de construction généraux de leur modèle se retrouvent dans tous les modèles actuels de langue neuronal et les word embedding. Ceux qui sont :

- ✓ **Couche d'incorporation :** une couche qui génère des word embeddings en multipliant un vecteur d'index par une matrice des word embeddings.
- ✓ **Couche (s) intermédiaire (s) :** une ou plusieurs couches qui produisent une représentation intermédiaire de l'entrée, par exemple une couche entièrement connectée qui applique une fonction non linéarité lors de la concaténation des word embeddings de n mots précédents.
- ✓ **Couche softmax :** la couche finale qui produit une distribution de probabilité sur les mots dans V.

II.6.2 Le modèle C et W

Collobert et Weston **[38]** (ainsi C & W) montrent en 2008 que les word embeddings sur un ensemble de données suffisamment important ont une signification syntaxique et sémantique et améliorent les performances sur les tâches en aval .

Leur solution pour éviter de calculer le softmax coûteux est d'utiliser une fonction objective différente : au lieu du critère de Bengio et al., qui maximise la probabilité que le prochain mot ait donné les mots précédents, Collobert et Weston forment un réseau qui produit un score plus élevé f_θ pour une séquence de mots correcte (une séquence de mots probable dans le modèle de Bengio) que pour une séquence incorrecte. À cet effet, ils utilisent un critère de classement par paire, qui ressemble à ceci :

$$J\theta = \sum_{x \in X} \sum_{w \in V} max\{0, 1 - f\theta(x) + f\theta(x^{(w)})\} \qquad \textbf{(II.8)}$$

Ils échantillonnent correctement les fenêtres x contenant n mots de l'ensemble de toutes les fenêtres X possibles dans leur corpus. Pour chaque fenêtre x, ils produisent alors une version incorrecte x (w) corrompue en remplaçant le mot central de x par un autre mot w de V. Leur objectif maximise maintenant la distance entre les résultats obtenus par le modèle pour la fenêtre correcte et la fenêtre incorrecte avec une marge de 1. Leur architecture de modèle, représentée sur la figure 3 sans l'objectif de classement, est analogue au modèle de Bengio et al.

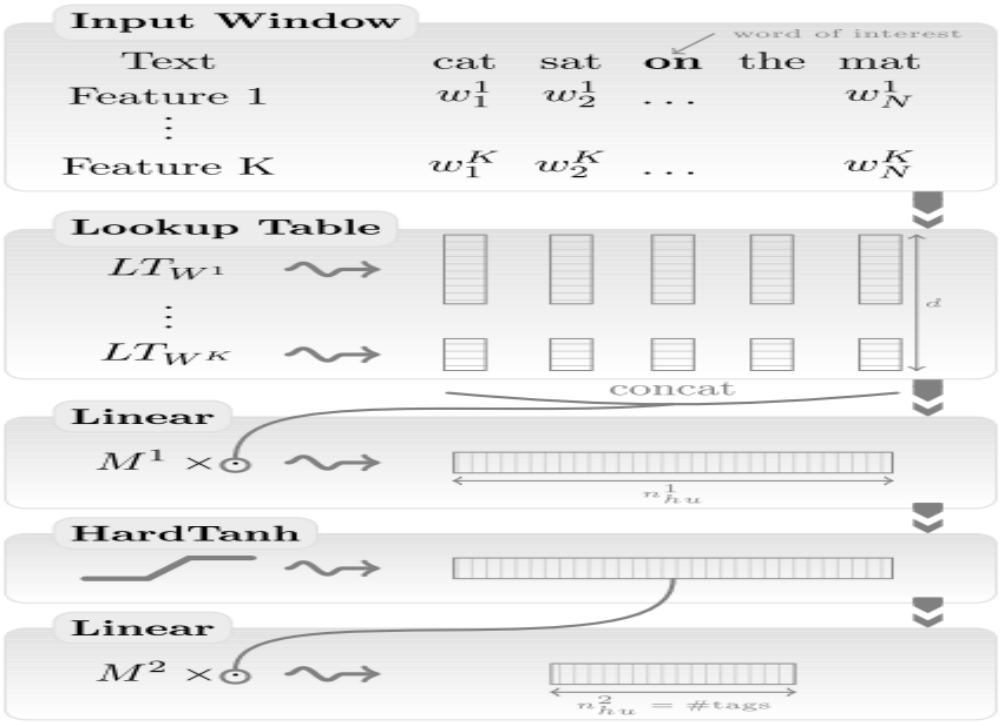


Figure II.10 Le modèle C et W sans objectif de classement [39]

Le modèle de langue résultant produit des embeddings qui possèdent déjà des relations connues , par exemple, les pays sont regroupés ensemble et des mots syntaxiquement similaires occupent des emplacements similaires dans l'espace vectoriel. Alors que leur objectif de classement élimine la complexité du softmax, ils conservent la couche cachée intermédiaire de Bengio et al. (La couche HardTanh de la figure II.10), ce qui constitue une autre source de calcul coûteux. En raison de cela, leur modèle complet fonctionne pendant sept semaines au total avec | V | = 130000 | V | = 130000.

II.6.3 Le modèle Word2vect

Word2vec est un groupe de modèles associés qui sont utilisés pour produire des word embeddings . Ces modèles sont des réseaux de neurones peu profonds qui sont formés à reconstruire des contextes linguistiques de mots. Word2vec prend comme entrée un grand corpus de texte et produit un espace vectoriel, généralement de plusieurs centaines de dimensions , chaque mot unique dans le corpus étant affecté à un vecteur correspondant dans l'espace. Les vecteurs de mots sont positionnés dans l'espace vectoriel de telle sorte que les mots qui partagent des contextes communs dans le corpus sont situés à proximité l'un de l'autre dans l'espace **[40].**

Word2vec est la méthode et le logiciel le plus populaire pour apprendre le word embedding aujourd'hui, il a été proposé par Mikolov et al. en 2013. Ce dernier est un modèle qui utilise un réseau de neurone à trois couches avec une couche cachée, deux variantes ont été proposées : skip-gram qui prédit le contexte étant donné le mot actuel, tandis que le sac-de-mots continu prédit le mot actuel étant donné son contexte. Ces architectures offrent deux avantages principaux par rapport au modèle C et W et au modèle linguistique Bengio:

- ✓ Ils éliminent la couche cachée coûteuse.
- ✓ Ils permettent au modèle de langue de prendre en compte un contexte supplémentaire.

II.6.3.1 Approche par sac de mots continus (CBOW)

Est un modèle qui permet de construire des word embeddings , permet de calculé la probabilité d'un mot cible étant donné son contexte il est basé sur le modèle de langue neuronal.la figure suivante elustre l'architecture CBOW :

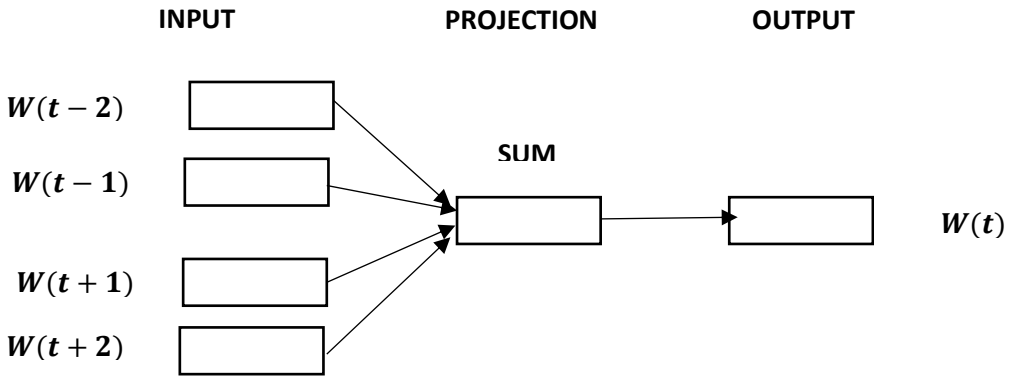


Figure II.11 : Sac de mots continu [41] .

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

$$J\theta = \frac{1}{T}\sum_{t=1}^{T} log\ p(w_t|w_{t-n},\cdots,w_{t-1},w_{t+1}\ldots w_{t+n}) \qquad \textbf{(II.9)}$$

Au lieu d'alimenter $\boldsymbol{n}$ mots précédents dans le modèle, le modèle reçoit une fenêtre de $\boldsymbol{n}$ mots autour du mot cible $\boldsymbol{w_t}$ à chaque pas de temps t.

Exemple détaillé sur le modèle CBOW :

CBOW est très utile pour identifier les mots qui manquent dans une phrase, voici un exemple détaillé qui explique l'architecture du modèle CBOW et son fonctionnement :

Supposant qu'on a le texte suivant : « Latent Semantique Indexing », on va sélectionner le mot «Indexing » qui est le mot cible qu'on veut chercher,

La taille de la couche d'entrée et la couche de sortie dépond de la taille du corpus, et la taille de la couche cachée est représentée par la dimension de système.la figure suivante monte un exemple qui utilise le modèle CBOW :

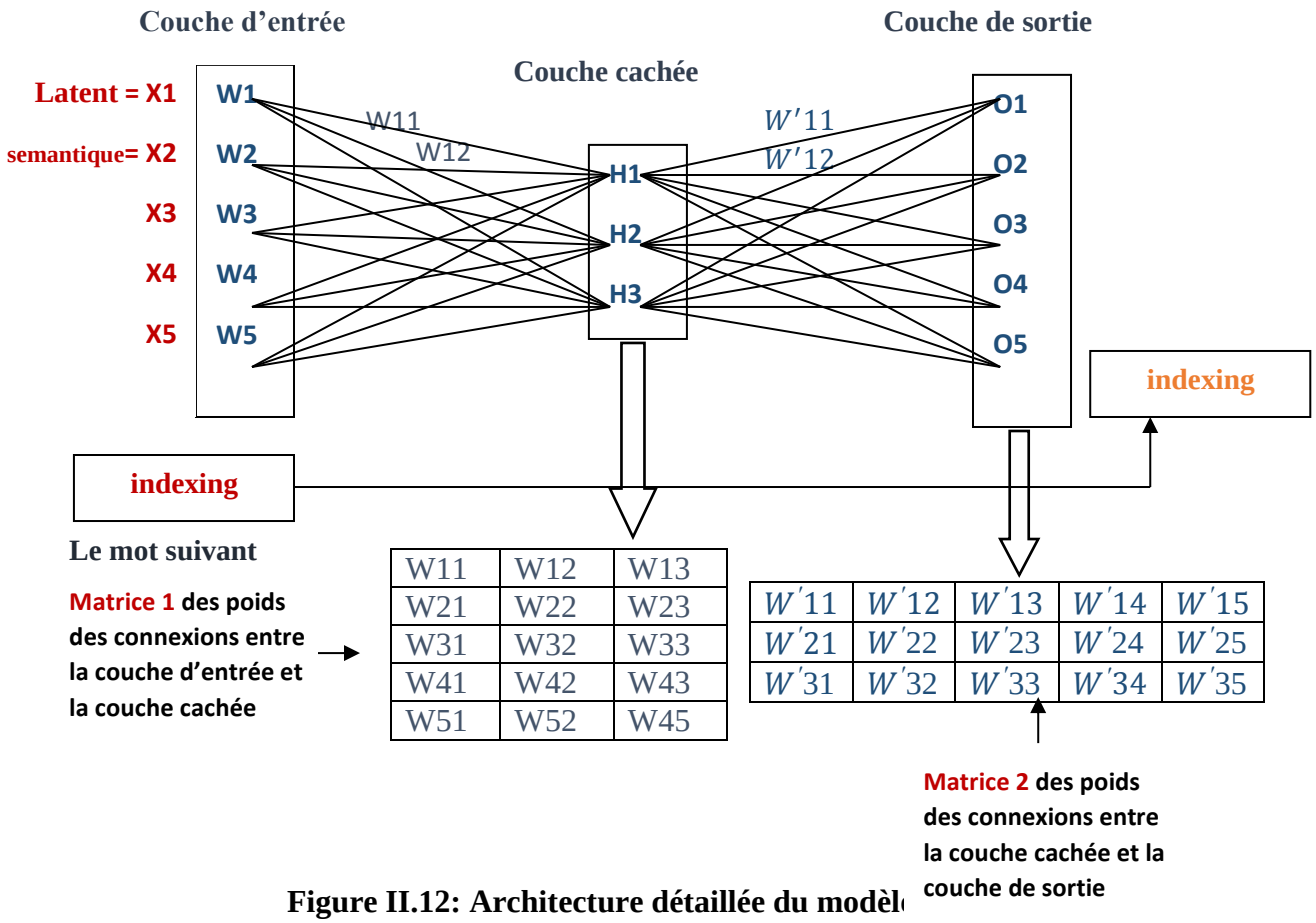


Figure II.12: Architecture détaillée du modèl

$\boldsymbol{x_1, x_2, \dots, x_3}$ sont des valeurs attribuées aux termes $\boldsymbol{w_1, w_2, \dots, w_3}$, et $\boldsymbol{x_1}$ et $\boldsymbol{x_2}$ sont égaux à 1, et les autres sont fixés à 0.

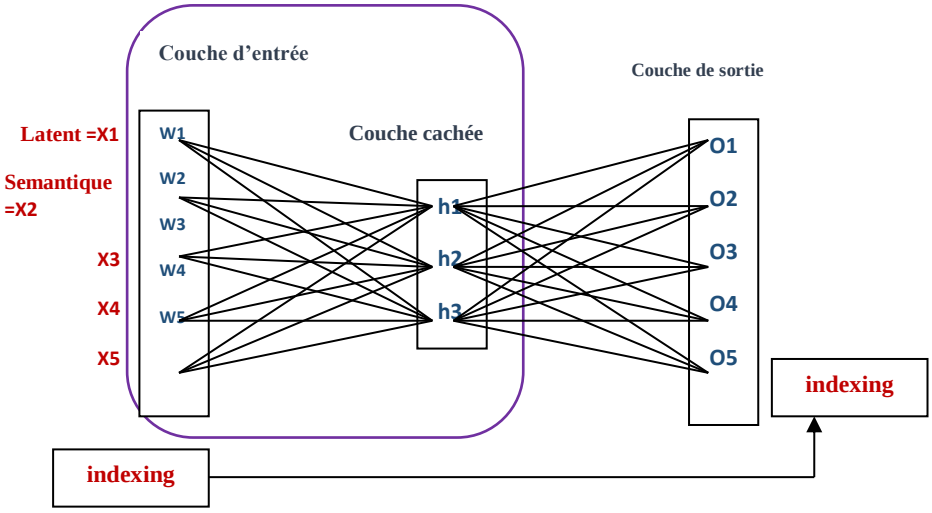
$\boldsymbol{W_{ij}}$: C'est le poids des connexions entre la couche d'entrée et la couche cachée.

W'_{ij}: C'est le poids des connexions entre la couche caché et la couche de sortie .

CBOW passe par deux phases :

a. **Forward propagation** : dans cette phase nous allons calculer les poids de la couche d'entrée vers la couche cachée, ensuite de la couche cachée vers la couche de sortie :

a.1 Forward propagation : de la couche d'entrée vers la couche cachée :



$h_1=(w_{11}x_1+w_{21}x_2 + w_{31}x_3 + w_{41}x_4 + w_{51}x_5)$

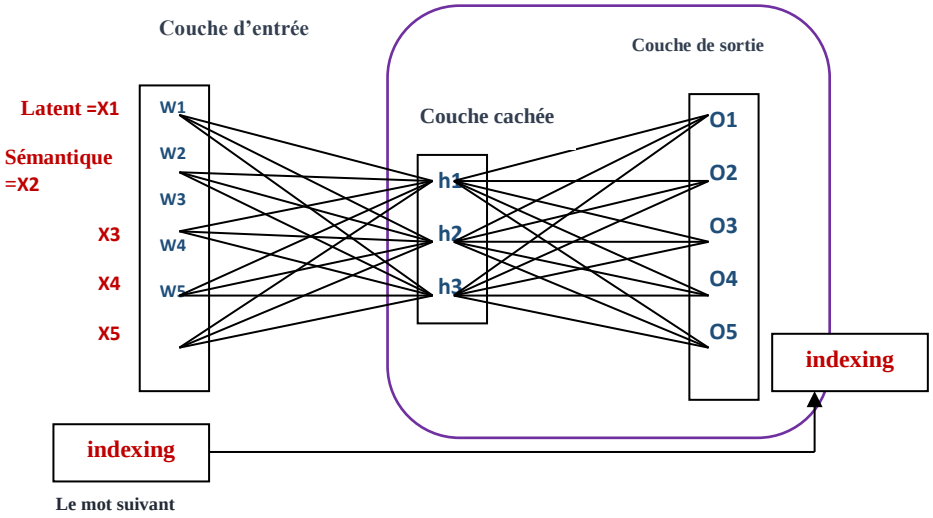
$h_2=(w_{12}x_1+w_{22}x_2 + w_{32}x_3 + w_{42}x_4 + w_{52}x_5)$ $\begin{vmatrix} h_1 \\ h_2 \\ h_2 \end{vmatrix} = w^T x =$

$$\begin{vmatrix} w_{11}+w_{21}+w_{31}+w_{41}+w_{51} \\ w_{12}+w_{22}+w_{32}+w_{42}+w_{52} \\ w_{13}+w_{23}+w_{33}+w_{43}+w_{53} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{vmatrix}$$

$h_3=(w_{13}\ x_1+w_{23}x_2 + w_{33}x_3 + w_{43}x_4 + w_{53}x_5)$

$w^T x$ est la matrice transposée de matrice

a.2 Forward propagation : de la couche cachée vers la couche de sortie :



Le calcule de poids pour la couche cachée se fait en deux parties :

Partie1 :

En appliquer la fonction de sommation : c'est la somme des poids de la couche cachée multiplié par les poids des connexions entre cette couche et la couche de sortie, soit Net (Oi) ce poids calculé.

Net (O1) $=u_1 = w'_{11}h_1 + w'_{21}h_2 + w'_{31}h_3$

Net (O2) $=u_2 = w'_{12}h_1 + w'_{22}h_2 + w'_{32}h_3$

Net (O4)= $u_3 = w'_{13}h_1 + w'_{23}h_2 + w'_{33}h_3$ = Net (O=)w^Th = $\begin{vmatrix} w'_{11} & w'_{21} & w'_{31} \\ w'_{12} & w'_{22} & w'_{32} \\ w'_{13} & w'_{23} & w'_{33} \\ w'_{14} & w'_{24} & w'_{34} \\ w'_{15} & w'_{25} & w'_{35} \end{vmatrix} * \begin{vmatrix} h_1 \\ h_2 \\ h_2 \end{vmatrix}$

Net (O3) $=u_4 = w'_{14}h_1 + w'_{24}h_2 + w'_{34}h_3$

Net (O5) $=u_5 = w'_{15}h_1 + w'_{25}h_2 + w'_{35}h_3$

w^Th est la matrice transposée de **matrice2.**

Partie2 :

Le calcul de la softmax de sortie

$$\textbf{Out}(O_1)= y_1=\frac{e^{Net(O_1)}}{e^{Net(O_1)}+e^{Net(O_2)}+e^{Net(O_3)}+e^{Net(O_4)}+e^{Net(O_5)}}=\frac{e^{u_1}}{e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5}}$$

$$\textbf{Out}(O_2)= y_2=\frac{e^{Net(O_2)}}{e^{Net(O_1)}+e^{Net(O_2)}+e^{Net(O_3)}+e^{Net(O_4)}+e^{Net(O_5)}}=\frac{e^{u_2}}{e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5}}$$

De la même on peut calculer y_3, y_4 *et* y_5

La fonction softmax pour un mot W_j (cible) donnant les mots de contexte W_I est alors calculée comme suit :

$$\textbf{P}(W_j|W_I) = y_j = \frac{e^{Net(O_j)}}{\sum_{j'=1}^{V} Net(O_{j'})} = \frac{e^{(u_j)}}{\sum_{j'=1}^{V} e^{(u_{j'})}}$$

Partie3 :

Calcul d'erreur : nous utilisant le calcul d'erreur pour corriger les poids des connexions entre les trois couches.

Soit w_O le mot cible (Indexing), w_I le contexte des mots donné (Latent sementique), et V la taille de contexte d'entrée. L'objectif est d'optimiser la probabilité conditionnelle du mot w_O sachant w_I, la fonction de perte est définie comme suit :

$$\textbf{Max P}(w_O|w_I)\textbf{= max (log}(y_{j^*})\textbf{)}$$

Par exemple :

$$\textbf{E}(O_4)\textbf{= log P}(w_4|w_I)\textbf{ = log }(\frac{e^{u_4}}{e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5}})= log_e(e^{u_4})- log_e(e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5})$$

$$= u_4-(e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5})$$

Comme on peut aussi minimiser l'erreur comme suit :

$$\textbf{E = - log P}(w_4|w_I)\textbf{,}\text{ alors}$$

$$\textbf{E}(O_4) = -(e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5}) - u_4$$

On peut généralisé ainsi :

$$\textbf{E = log }\sum_{j'=1}^{V} e^{u_{j'}} - u_{j^*}$$

Où, j^* est l'index du mot cible dans la couche cachée

Maintenant prenant la dérivation de E par rapport à $\boldsymbol{u_4}$ nous allons avoir :

$$\frac{dE(O_4)}{dE(u_4)} = \frac{u_4}{e^{u_1}+e^{u_2}+e^{u_3}+e^{u_4}+e^{u_5}} - \frac{dE(u_4)}{dE(u_4)} = \mathbf{Out}(\boldsymbol{O_4}) - \frac{dE(u_4)}{dE(u_4)} = \boldsymbol{y_4 - 1}$$

Nous remarquons que u_4 est la valeur du mot cible O_4 (Indexing)

Nous pouvons généraliser cette dérivation comme suit :

$$\frac{dE}{du_j} = \frac{d\,(log\sum_{j'=1}^{V} e^{u_{j'}} - u_{j^*})}{du_j} = \frac{e^{u_j}}{\sum_{j'=1}^{V} e^{u_{j'}}} - \frac{d(u_{j^*})}{du_j} = y_j - t_j := e_j$$

$$t_j = \frac{d(u_{j^*})}{du_j}$$

$t_j = 1$ si $(t_j = t_{j^*})$ si non $t_j = 0$

- **La rétro propagation (back-propagation) :**

  ✓ de la couche de sorite vers la couche cachée : Cette phase nous permet de mettre à jour les poids de tous les neurones de la couche cachée vers la couche de sortie ($W'_{11}, W'_{12}, W'_{13}, \ldots, W'_{35}$).

Etape1 :

prendre le gradient de 'E' par rapport à W'_{11}.

$$\frac{dE(O_1)}{dW'_{11}} = \frac{dE(O_1)}{d_{u_1}} * \frac{d_{u_1}}{dW'_{11}}$$

$$\frac{dE(O_1)}{d_{u_1}} = (y_1 - 0) = e_1$$

$$\frac{d_{u_1}}{dW'_{11}} = \frac{d(W'_{11}h_1 + W'_{21}h_2 + W'_{31}h_3)}{dW'_{11}} = h_1$$

$$\frac{dE(O_1)}{dW'_{11}} = \frac{dE(O_1)}{d_{u_1}} * \frac{d_{u_1}}{dW'_{11}} = e_1 * h_1$$

Etape 2 :

mettre à jour le poids W'_{11}.

$$W'^{new}_{11} = W'_{11} - \eta \frac{dE(O_1)}{dW'_{11}} = W'_{11} - \eta\,(e_1 - h_1)$$

η représente un rang d'apprentissage $\in$ [0,1]

de la meme façon nous pouvons calculer le poids de $W'_{12}, W'_{13}, W'_{21} \ldots, W'_{53}$.

✓ De la couche cachée vers le couche d'entrée : Cette phase nous permet de mettre à jour les poids de tous les neurones de la couche cachée vers la couche de sortie ($W_{11}, W_{12}, W_{13}, \ldots, W_{35}$).

Etape1 : prendre le gradient de 'E' par rapport à W_{11}.

$$\frac{dE}{dW_{11}} = \frac{dE}{dh_1} * \frac{dh_1}{dW_{11}}$$

$$\frac{dh_1}{dW_{11}} = (\frac{dE}{du_1} * \frac{du_1}{dh_1}) + (\frac{dE}{du_2} * \frac{du_2}{dh_1}) + (\frac{dE}{du_3} * \frac{du_3}{dh_1}) + (\frac{dE}{du_4} * \frac{du_4}{dh_1}) + (\frac{dE}{du_5} * \frac{du_5}{dh_1})$$

$$= (e_1 W'_{11}) + (e_2 W'_{12}) + (e_3 W'_{13}) + (e_4 W'_{14}) + (e_5 W'_{15})$$

$(\frac{dE}{dh_1} * \frac{du_1}{dh_1})$ est l'erreur propagée à partir du neurone de mot de sortie O_1 de la couche de sortie

$(\frac{dE}{dh_2} * \frac{du_2}{dh_1})$ est l'erreur propagée à partir du neurone de mot de sortie O_2 de la couche de sorti

$$\frac{dh_1}{dW_{11}} = \frac{d(W_{11}x_1 + W_{21}x_2 + W_{31}x_3 + W_{41}x_4 + W_{51}x_5)}{dW_{11}} = x_1$$

$$\frac{dE}{dW_{11}} = (\frac{dE}{dh_1} * \frac{dh_1}{dW_{11}}) = ((e_1 W'_{11}) + (e_2 W'_{12}) + (e_3 W'_{13}) + (e_4 W'_{14}) + (e_5 W'_{15})) * (x_1)$$

Etape 2 : mettre à jour le poids W_{11}

$$W_{11}{}^{new} = W_{11} - \eta \frac{dE}{dW_{11}} = W_{11} - \eta((e_1 W'_{11}) + (e_2 W'_{12}) + (e_3 W'_{13}) + (e_4 W'_{14}) + (e_5 W'_{15})) * (x_1)$$

De la même manière nous pouvons calculer$W_{12}, W_{13}, W_{21}\ldots, W_{53}$

II.6.3.2 Approche de type skip-gram

L'entraînement du modèle à pour objectif d'obtenir le vecteur du mot permettant d'exprimer le même sens que le mot en cours, ie : la représentation qui permet de déduire son contexte , comme on peut le voir sur la figure II.13 :

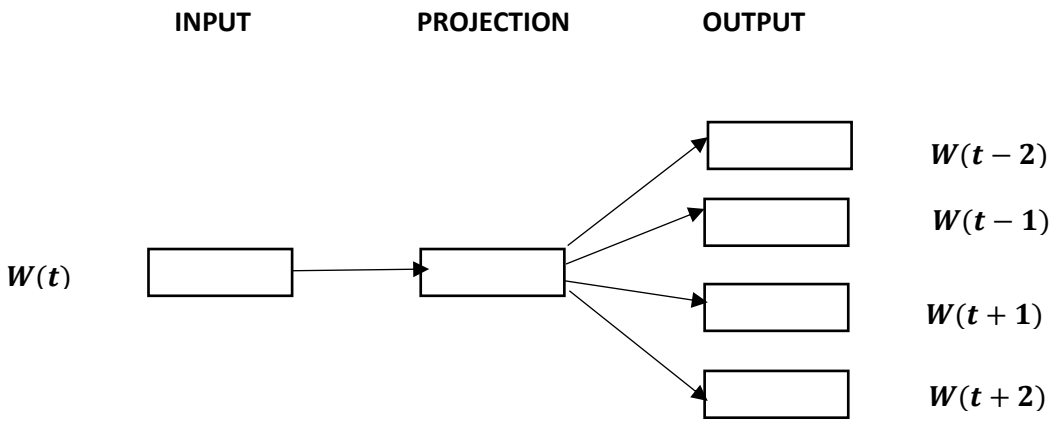


Figure II.13 : Skip-gram [42]

Formellement, le modèle a pour objectif, pour une séquence de mots d'entraînement $w1, w2,$ wT de maximiser la moyenne des logs attribués aux fenêtres de mots k :

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-k}^{k} logP(w_{t+j}|w_t)\right]_{(j\neq 0)} \qquad \textbf{(II.10)}$$

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

$$P(w_i|w_j) = \frac{exp\left(U_{w_i}^T V_{w_j}\right)}{\sum_{l=1}^{V} exp\left(U_l^T V_{w_j}\right)} \qquad \textbf{(II.11)}$$

Où V est la taille du vocabulaire.

II.6.4 Le modele Glove

Glove est un algorithme d'apprentissage non supervisé pour l'obtention de représentations vectorielles pour les mots. Glove s'effectue sur des statistiques cumulées globales de cooccurrence de mot-mots à partir d'un corpus, et les représentations résultantes présentent des sous-structures linéaires intéressantes du mot dans un espace vectoriel **[43].**

Dans cette méthode, ils font en premier une cooccurrence matrice qui base sur l'équation II.14.

$$\boldsymbol{X_{i,j} = \sum_{wi,wj} \frac{1}{dis(wi,wj)}} \qquad \textbf{(II.12)}$$

Nous montrons directement l'objectif de ce modèle qui est :

$$\boldsymbol{J = \sum_{i,j} f(X_{ij})(w_i^T w_j - logX_{ij})^2} \qquad \textbf{(II.13)}$$

Où $\boldsymbol{f(X_{ij})}$ est la fonction du poids pour chaque paire de mots, et il est défini comme :

$$\boldsymbol{f(X_{ij}) = \begin{cases} (X_{ij}/X_{max})^{\alpha} \; if \; X_{ij} < X_{max} \\ 1 \; autrement \end{cases}} \qquad \textbf{(II.14)}$$

L'algorithme d'apprentissage répète à travers chaque élément $\boldsymbol{X_{ij}}$ de cooccurrence $\boldsymbol{X}$ et met à jour.

II.7 Le word embedding en recherche d'information :

La recherche d'information se réfère à la recherche initiale effectuée par un utilisateur : une seule requête, sans autre interaction ou rétroaction, sur la base duquel un système RI s'efforce de renvoyer un classement précis des documents.

Récemment, de nombreux travaux ont montré que les approches d'apprentissage par les réseaux de neurones sont très efficaces dans plusieurs tâches de RI (Appariement de textes **[44][45](**Bengio et al., 2006 ; Huang et al., 2013), et tâches de question-réponse**[46]** (Bordes et al., 2014)). Une première catégorie de travaux utilise des représentations distribuées de mots (word embeddings) obtenues par les modèles neuronaux pour les intégrer dans les fonctions d'appariement requête-document **[47]**.

La deuxième catégorie de travaux consiste en des modèles d'appariement qui apprennent la pertinence des paires document-requête à partir des vecteurs sémantiques latents **[48][49].** Par exemple, le Deep Semantic Structured Model (DSSM) **[50]** est connu pour être un des modèles les plus efficaces dans la tâche de recherche sur le web. Ce modèle applique un réseau de neurones profond sur la représentation d'un document et d'une requête obtenues par une méthode de hâchage de mots qui permet d'apprendre leurs représentations latentes à partir de leur valeur de pertinence. Une extension du modèle DSSM, proposée dans (Shen et al., 2014) **[51]** s'appuie sur un réseau de convolution, appelée Convolutional Latent Semantic Model (CLSM) **[52].**

II.6 Expansion de requête en utilisant word embedding :

L'expansion de la requête consiste à ajouter de nouveaux termes à la requête initiale ou alors revoir le poids de ses termes dans le but de cibler la recherche vers les documents pertinents **[53]**.

Parmi les techniques utilisé dans l'expansion des requêtes en trouvent le word embedding. Dans ce qui suit nous présentons quelques travaux qui se focalisent sur cette technique :

- **Sordoni et al en 2014 :**

Ils ont étudié l'expansion des requêtes, évaluant la recherche ad hoc sur les collections TREC.

Suggestion des requêtes étudiées par les deux chercheurs Sordoni et al en 2015 [54] :

Al et sordoni ont proposé une approche supervisée de Minimisation Quantum Entropy(QEM). Elle consiste à trouver une représentation sémantique des concepts, tel que des mots ou des phrases, pour l'expansion de requêtes. Cette approche permet aux documents et aux requêtes de se trouver dans un espace plus vaste et de transporter plus d'information sémantique que les concepts. QEM s'efforce d'augmenter le pouvoir de représentation dans l'espace vectoriel ce qui permet d'obtenir une plus grande résolution .QEM réalise notamment une précision améliorée .la capacité de QEM à trouver des termes d'expansion utiles pour des requêtes plus longues en raison d'une résolution sémantique plus élevé. QEM est utilisé dans l'expansion de requêtes dans les journaux Wikipedia.

- **Ganguly et al. (2015) [55]:**

Propose un modèle de langue généralisé(GLM) pour l'intégration de word embedding avec une modélisation de langage de probabilité de requêtes. La similarité sémantique entre les termes de documents et de requêtes est mesurer la similarité cosinus entre les embeddings obtenu par l'approche CBOW de modèle Word2Vec.

- **Gupta et al. (2014) [56]** : a propose une approche d'encodeur automatique pour l'expansion de requêtes de script mixte.
- **ALMasri et al. (2016) [57] :**

Propose une méthode heuristique pour l'expansion de requêtes VEXP en fonction de la similarité des word embedding.

- **Zamani et Croft (2016)[58] :**

Ils ont proposé deux approches. La première EQE1 (Embedding-based Query Expansion) suppose les conditions d'indépendance entre les termes d'expansion, tandis que la deuxième EQE2 suppose que la similarité sémantique entre deux termes est indépendante de la requête.

- **Roy et al. (2016)[59]** :

Ils ont proposé trois approches qui utilisent le word embedding dans l'expansion de requêtes basé sur les k-voisins plus proches(KNN) .la première approche (i) compte les k voisins pour chaque terme de requête dans l'espace vectoriel de word embedding .la deuxième approche réduit l'espace de vocabulaire considéré par KNN en considérant uniquement les termes des tops documents retournés par le modèle de pertinence. La troisième approche, est une stratégie

couteuse en terme de calcul,est appliqué pour réduire le nombre de voisins les plus proches ,en supposant que les voisins les plus proches sont semblables les uns aux autres

- **Amer et al. (2016) [60] :**

Utilisent le word embedding pour l'expansion de requêtes personnalisées dans le domaine de la recherche des livres sociaux.

II.7 Conclusion

Dans ce deuxième chapitre, En premier lieu nous avons parlé sur le word embedding et les réseaux de neurones et ses architecture après nous avons introduit les principales modèles de word embedding et leur fonctionnement qui se bases sur les réseaux de neurones : modèle classique de langue, **C et W** modele, word2vect (CBOW, Skip-gram) et le modèle Glove.
Enfin nous avons traité l'utilité de word embedding dans la recherche d'information ah doc et l'expansion de requêtes .Ce chapitre nous a aidé à comprendre le but de word embedding et les difficultés qui les a remédié ce qui concerne la recherche sémantique et syntaxique et l'amélioration qui porte à la recherche d'information par rapport aux modèles de recherche précédents.

Le chapitre suivant est consacré à la présentation et l'expérimentation de notre approche.

III.1 Introduction

Ce dernier chapitre est consacré à la présentation de l'approche proposée, et les résultats obtenus. La première section, nous décrivons l'approche proposée. Dans la seconde section, nous abordons l'expérimentation de l'approche en précisant les outils et le langage utilisés pour sa mise en œuvre, ainsi nous présentons quelques résultats expérimentaux obtenus sur la collection de test TREC AP88.

III.2 Présentation de l'approche proposée

III.2.1 Principe de l'approche

Le besoin en information de l'utilisateur est exprimé par une requête qui est exécutée par le système de recherche d'information. Dans certains cas cette requête n'est pas complète et claire, alors elle retourne des résultats qui ne sont pas satisfaisants.
Pour remédier à ce problème, Nous utilisons l'expansion de la requête pour améliorer la requête initiale en ajoutant des termes générés, classés et choisis pour reformuler enfin cette requête.

Dans notre approche nous utilisons la technique word embedding pour la sélection des termes d'expansion .nous rappelons que le word embedding a pour objectif de représenter les termes sous forme de vecteur d'un espace d'une dimension réduit dans notre cas (200 dimensions).

III.2.2 Formalisation de l'approche

Nous présentons dans cette section, l'emplacement de notre approche dans le SRI

III.2.2.1 Emplacement de notre approche dans un système de recherche d'information

Nous présentons dans la figure suivante l'emplacement de notre approche dans l'architecture d'un SRI. Nous expliquons ensuite les étapes de notre approche.

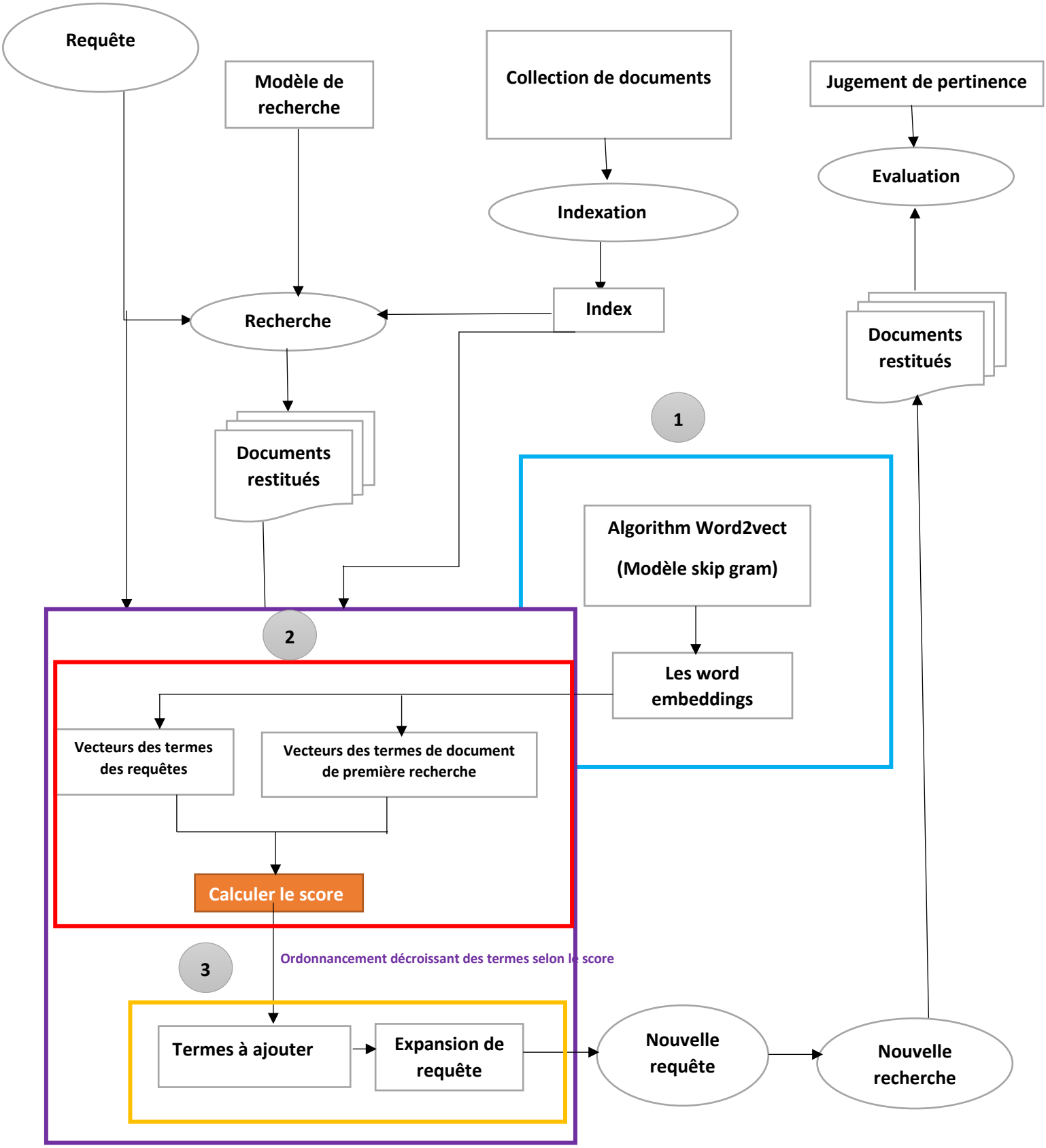


Figure III.1 : Emplacement de notre approche dans un SRI.

a. **Les étapes de l'approche :** une fois que la première recherche est effectuée avec le modèle de recherche BM25 nous obtenons les mille documents les plus pertinents, à ce niveau en intégrons les word embedding. Afin de réaliser notre approche, nous avons suivi les étapes suivantes :

- **Etape 01 : extraction des word embedding**

Cet étape permet d'extraire les word embedding qui représentent les vecteurs correspond aux termes de notre collection AP88 ainsi les termes des requêtes. Notre embedding sont stocké dans un fichier **« vectors_ap8889_skipgram_s200_w20_neg20_hs0_sam1e-4_iter5.txt »** ce fichier est produit par l'algorithme d'apprentissage **Word2vect** par son modèle **skip gram**.
La figure suivante représente un extrait de fichier qui contient les word embedding :

```
future -0.332784 -0.353664
0.102814 -0.495613 -0.259367 -0.098757
0.067439 -0.256459 -0.250686 0.421359 0.298603
0.027290 -0.021085 -0.188576 -0.202211 -0.350216 -0.192615
0.433190 -0.160695 0.230655 -0.260912 0.075215 0.178258
0.026363 -0.257999 -0.031147 0.193919 -0.341385 -0.120098
0.255371 0.184951 0.186962 -0.166631 0.089827
0.435518 -0.682795 0.135153 0.078873
0.120199 -0.128637 -0.009728 0.231643 -0.100674 -0.247482
0.092683 0.054301 -0.087291 0.200882 0.047700 -0.012300
0.172792 -0.258321 0.003506 0.222842 -0.170090 -0.104514
0.034759 0.025637
0.040477 -0.469322 -0.130720 -0.357463 -0.015615
0.014417 -0.093778 0.121895
0.017808 -0.321895 -0.047997 -0.374758 -0.338372 -0.040954
0.376622 -0.011388 0.060620 -0.026029 0.007188 0.186717
0.105085 0.025419 -0.083844 0.021663 -0.117888 -0.025971
0.007823 -0.005896 -0.069800 -0.024535 0.008428 -0.219325
0.362621 0.058019 0.178190 -0.098573 0.021364 -0.164936
0.435256 0.047656 -0.159797 -0.307847 -0.191355
0.189707 -0.190985
0.087738 -0.081636 -0.195177 -0.309346 -0.237668 -0.241349
0.428861 -0.053471 0.154840 0.356313 -0.399727 0.110976
0.201181 0.470589 0.114466 0.194234 0.205545 -0.232971
0.112617 -0.091266 -0.027655 -0.328227 -0.101458 0.136851
0.200264 0.188496 -0.068235 -0.220051 -0.337514
```

Figure III.2 extrait d'un fichier contient les word embedding.

Ce fichier contient tous les termes de collection et de requêtes (93951 termes) chaque terme (exemple « future » dans la **Figure III.1**) est représenté par un vecteur de 200 dimension et la taille de la fenêtre est 5.

- **Etape 02 :**

Cet étape consiste à classer les termes des n premiers documents en utilisant la similarité de leur vecteurs (word embedding) avec l'ensemble des vecteurs de termes de la requête cette similarité est basé sur la fonction cosinus.

Pour ce faire nous procédons comme suit :

$\boldsymbol{Q}$ **:** Requêtes.

$\boldsymbol{R}$ **:** Document restitué par la première recherche**.**

$\boldsymbol{t_i}$ **:** Terme document restitue.

$\boldsymbol{t_j}$ **:** Terme de la requête.

$\boldsymbol{v_{ti}}$: Vecteurs de termes de la requête.

$\boldsymbol{v_{tj}}$: Vecteurs de termes de documents.

En calcule la similarité entre deux vecteurs avec la formule suivante :

$$\cos(v_{ti}, v_{tj}) = \frac{v_{ti} * v_{tj}}{|v_{tj}| * |v_{tj}|} \qquad \textbf{(III.1)}$$

On calcule le score de chacun des termes des documents restitués par la fonction **III.1** suivante :

$$score(t_j) = \sum \cos(v_{ti}, v_{tj}) \qquad \textbf{(III.2)}$$

$\forall\, t_i \in Q$

$t_j \in R$

Après le calcul du score nous ordonnons les termes selon leur score.

- **Etape 03 :**

Cette étape consiste à l'expansion de la requête :

Après l'ordonnancement des termes, on choisit le nombre de termes a ajouté pour la requête originale et enfin on effectue une nouvelle recherche avec la nouvelle requête obtenue avec notre approche.

b. L'implémentation de l'approche :

Pour implémenter notre approche nous avons modifié la classe QueryExpansion de la plateforme terrier.

- La première étape de notre approche est implémentée par l'algorithme word2vec qui nous a permet de récupérer les words embeddings des documents et des requêtes

- La deuxième étape permet le calcul des scores de termes des documents restitués et l'ordonnancement décroissant des termes selon le score.

L'algorithme suivant formalise notre approche :

Debut //debut de l'algorithme

//Remplir la table

Table < clé,valeur> termdocument

//clé=identifiant de terme, valeur=la fréquence de terme dans les documents

Pour (i=0 jusqu'à taille de Top_Document)

//Récupérer l'identifiant du terme et sa fréquence dans le document

terms[][]

Pour (j=0 jusqu'à taille terms [0])

Si(terme existe dans plusieurs documents)

Faire la somme de ces frequences

Finsi

Mettre terms [0][j] ,terms [1][j] dans table //terms [0][1]comme premier argument

// et terms [j][1]comme deuxième argument

Fin pour

Fin pour

//le calcul des scores des termes d'expansion

Table< clé,valeur>resultatFinal //clé :terme , valeur : vecteur**(la variable terme contient les terme et vecteur contient les vecteur correspond)**

String []termerequete //tableau des termes des requetes

Table < clé,valeur> termdocument

Pour (i=0 jusqu'à taille de termdocument)

- Récupérer les termes des documents (terme)
- Récupérer les vecteurs des termes de documents de la table resultatFinal(**vectTerme)**

Fin pour

scoretermEm=0; //initialisation de score

Table < valeur1> scors //liste contient les scores des termes

Table < valeur2> termeajouter //liste contient les termes correspond au scores

Pour (i=0 jusqu'à taille de termerequete)

- Récupérer les termes des requêtes (termereque)
- Récupérer les vecteurs des termes des requêtes de la table resultatFinal (**vectTermereq)**

Si ((vectTermereq!=null)&& (vectTerme!=null))

-calcul la similarité cosinus entre le vecteur de terme de la requête et le vecteur de terme de documents.

Avec la formule :

$$score(t_j) = \sum \cos(v_{ti}, v_{tj}) \qquad \textbf{(III.3)}$$

Fin si

Si (scoretermEm!=0.0)

-Mettre les scores obtenus dans une liste (scors)

-Mettre les termes correspond au score dans une liste (termeajouter)

Fin si

Fin pour

Fin pour

Pour (i=0 jusqu'à taille de la liste score)

-Trier les scores dans la liste scors

- Trier les termes dans la liste termeajouter

Fin pour

– La troisième étape permet l'ajout des termes ordonnés à la requête originale

Pour (i=0 jusqu'à nombre de termes à ajouter)

-ajout des termes à la requête originale

Fin pour

III.3 L'environnement de développement

Dans ce qui suit nous allons présenter notre environnement technique et préciser les différents outils utilisés : la plateforme Terrier, le langage de programmation JAVA et Netbeans.

III.3.1 La plateforme Terrier

Terrier est l'abréviation de TERabyteRetrIEver qui est un moteur de recherche flexible, robuste, et efficace. Il est facilement déployable à grande échelle des collections de documents. Il a été créé par le département d'informatique de l'université Glasgow de Royaume-Uni en 2000. Dans le but de fournir une plateforme flexible pour le développement rapide des applications de recherche d'information. La recherche peut être effectuée facilement sur les collections de test standard TREC et CLEF [01].

Terrier est un produit open source écrit en Java, qu'il a été mis à disposition, du général public depuis novembre 2004 .Il a été conçu pour fonctionner sur la plupart des systèmes d'exploitation actuels tels que Windows ou Linux.

Avec sa conception modulaire, il facilite son extension en cas de besoin, et il propose aussi une grande palette de configuration et cela dans le but de satisfaire ses utilisateurs et leurs fournir une meilleure maniabilité. Ce système est sauvent utilisé dans l'innovation et la mise en œuvre de nouvelles méthodes pour la recherche d'information. Comme tous moteurs de recherche, Terrier permet :

- ✓ L'indexation classique : extraction des mots clés des documents appartenant à une collection et les stocker dans un index.
- ✓ Recherche des documents pertinents pour répondre aux requêtes formulées par l'utilisateur.
- ✓ Evaluation des résultats de la recherche.

III.3.1.1 Architecture de Terrier

La figure ci-dessous montre l'architecture générale de Terrier

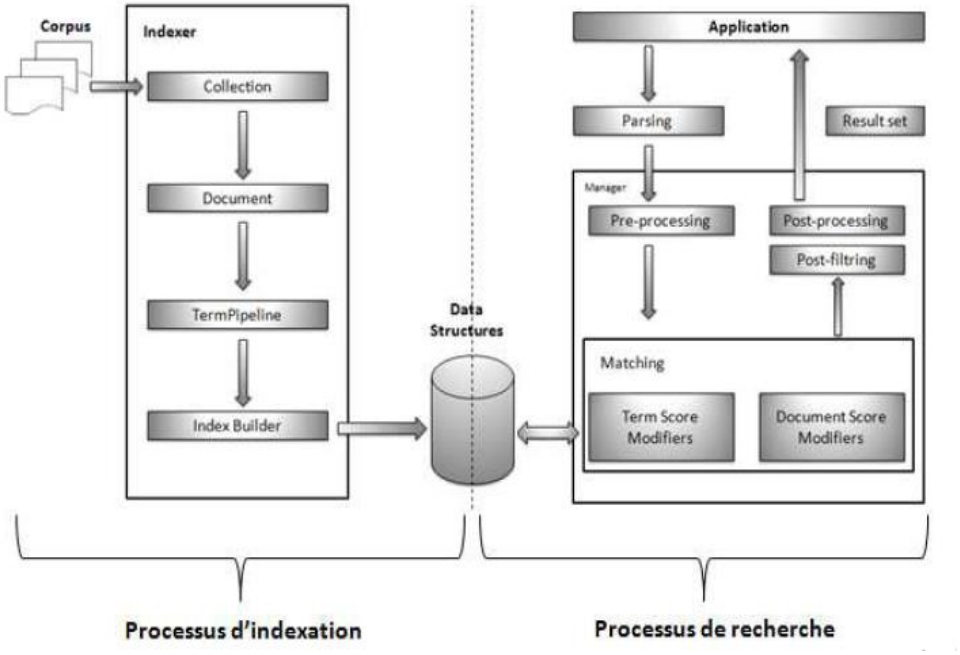


Figure III.3 : Vu d'ensemble de l'architecture de terrier [02].

III.3.1.1.2 Le processus d'indexation

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

Les quatre procédures sont :

1. Splitter la collection de documents : consiste à parcourir l'ensemble de corpus reçu en entrée par Terrier et envoyer chaque document à l'étape suivante.
2. Extraction des termes (Tokenize Document) : pour chaque document de la collection, un processus visant à extraire les termes du document nommé «tokenisation».Ce processus retourne comme résultat l'ensemble de termes de document qu'on peut adopter comme termes d'indexation.
3. Traitement des termes extraits avec Term- Pipeline : consiste à l'élimination des mots vides et la lemmatisation des termes.
4. La construction de l'index.

La figure ci-dessous donne une vue d'ensemble d'interaction des composants principaux impliqués dans le processus d'indexation.

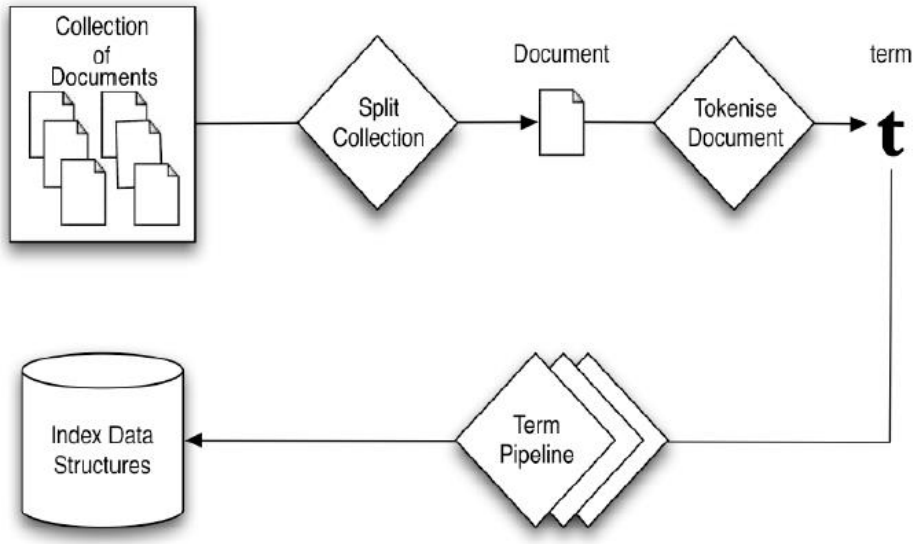


Figure III.4 : Le processus d'indexation dans Terrier [02].

Le processus d'indexation avec Terrier consiste à analyser tous les documents de la collection spécifiée afin de produire un ensemble de termes d'indexation «index » et les ranger dans la structure « index data structure ».

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

- **Org.terrier.indexing :** ce package contient les différentes classes permettant de réaliser un ensemble d'opérations sur la collection des documents, dans le but d'extraire les termes de tous les documents de la collection
- **Org.terrier.terms :** les classes de ce package permettent d'effectuer un ensemble de traitements sur les termes extraits. Parmi ces traitements, l'élimination des mots vides, lemmatisation des termes, …etc.
- **Org.terrier.structures :** les classes de ce package permettent la construction d'un ensemble de structures ou un ensemble de données stockées. Parmi ces structures, on a :
  - ✓ **Lexicon :** contient les informations sur chaque terme de la collection (Terme, Id terme, nombre de documents qui contiennent le terme, fréquence du terme dans la collection, Offset dans le fichier inverse).
  - ✓ **Direct index :** il enregistre pour un document les termes qui apparaissent dans ce dernier. Il est souvent utilisé pour la reformulation de la requête, la classification et la comparaison des documents.
  - ✓ **Inverted index :** contrairement à l'index direct, il enregistre pour un terme les documents dans lesquels il apparait, il contient aussi la position de chaque terme et sa fréquence dans ces documents.
  - ✓ **Document index :** contient des informations sur les différents documents de la collection.

III.3.1.1.2 Le processus de recherche

Un des principaux objectifs de Terrier est de faciliter la recherche d'information. Terrier implémente pour cette raison certain nombre de fonctionnalités de recherche qui offre un large choix pour le développement de nouvelles applications et pour les tests en recherche d'information. Durant le processus de recherche, chaque requête doit passer par les étapes suivantes :

1. Query : est une classe abstraite qui représente la requête. Terrier supporte trois modèles de requête :

- ***Single Term Query :*** désigne la requête qui contient un seul terme.
- ***Multi Term Query :*** désigne la requête qui contient plusieurs termes.
- ***Field Query :*** terme qualifié par un champ.

2. Parsing : est l'opération qui se charge de tokenizer la requête.

3. Pré-processing : est l'oppération qui applique le TermPipeline à la requête. Elimine les mots vides et les lemmatises.

4. Matching : est l'opération responsable de l'initialisation du Weighting Model et du calcul des scores entre la requête et les documents.

- ✓ **Weighting Models :** est une interface qui assigne un score pour chaque terme de la requête dans le document. Terrier offre plusieurs classes qui implémentent cette interface, parmi eux (TFIDF, BM25).
- ✓ **Document Score Modifiers :** permet de modifier le score des documents en fonction du langage de la requête.
- ✓ **Term Score Modifiers :** permet de modifier le score des documents en fonction de la position des termes.

5. Post-processing : est l'opération appropriée pour implémenter des fonctionnalités qui apportent un changement à la requête originale. Un exemple de Post-processing est l'expansion de requête, puis exécute une autre fois le Matching avec cette nouvelle requête. Un autre exemple de Post-processing est l'application de clustering.

6. Post-filtering : est l'étape finale du processus de recherche de Terrier où une série de filtres peut enlever les documents déjà recherchés, qui ne satisfont pas une condition donnée, par exemple, dans le contexte d'un moteur de recherche Web, un post filtre peut être utilisé pour réduire le nombre de documents recherchée depuis le même site Web, afin d'augmenter la diversité des résultats. A la suite des étapes précédentes, Terrier s'occupe de retourner un ensemble de documents triés selon l'ordre décroissant de leurs scores.

La figure ci-dessous donne une vue d'ensemble d'interaction des composants de Terrier dans la phase de recherche

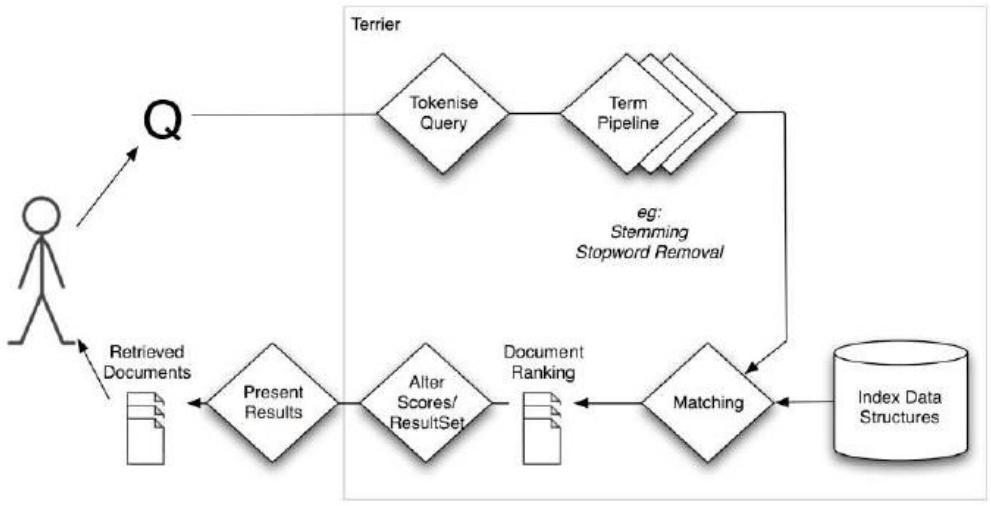


Figure III.5 : Le processus de recherche dans Terrier [02].

III.3.2 Le langage Java

Java est un langage de programmation moderne développé par Sun Microsystems (Aujourd'hui racheté par oracle). Une de ses plus grandes forces est son excellente portabilité : une fois le programme java est créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc [03].

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 :** Java est un langage purement orienté objet.
  - ✓ Tout est classe.
  - ✓ Héritage simple.
  - ✓ Une librairie plus de classes est fournie.

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

II.3.3 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'applications 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 [04].

L'EDI est lui-même écrit en Java, ce qui permet de le faire tourner sur n'importe quel système d'exploitation.

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

La figure suivante illustre l'environnement de développement Netbeans :

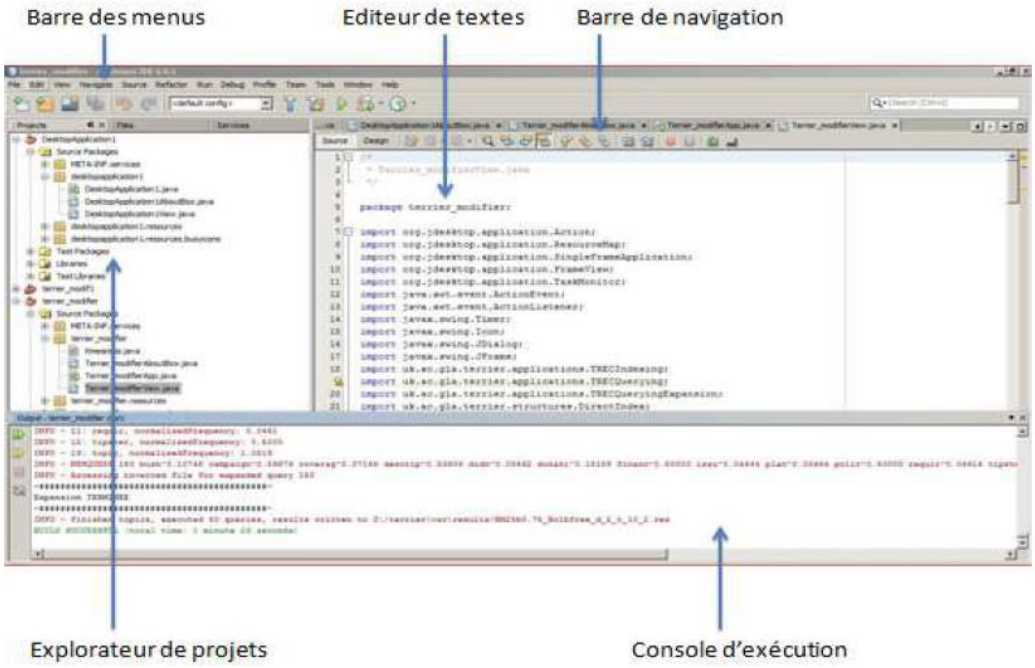


Figure III.6 : Environnement de développement de Netbeans.

III.4 Expérimentation

III.4.1 Collection de test utilisée

Différentes collections de tests sont utilisées en recherche d'information. La collection que nous avons utilisée pour nos expérimentations est : **TREC AP88** (**A**ssociated Press newswire, 19**88**). Pour la recherche nous avons utilisé 50 requêtes issues des topics numérotées « 51-100 » de la collection TREC.

III.4.2 Evaluation et Résultats

Pour évaluer les performances de notre approche, nous devons tout d'abord fixer les résultats de base : recherche simple.

Ces résultats seront comparés par la suite à ceux obtenus après la reformulation avec notre approche en appliquant les paramètres de recherche.

III.4 .2.1 Résultats de la recherche simple

Avant de procéder à notre approche, nous allons commencer par illustrer les résultats obtenus avec la recherche simple.

Le tableau ci-dessous montre la précision (MAP) obtenue avec le modèle de recherche BM25.

Recherche simple	
Modèle de recherche	BM25
MAP	0.1334

Tableau III .1 Résultat obtenu avec la recherche simple (BM25).

III.4 .2.2 Résultats obtenus avec notre approche

Dans cette étapes, nous allons présenter les résultats obtenus avec le modèle de recherche, après une recherche simple effectue avec le modèle de recherche BM25 et notre modèle d'expansion de requête, qui utilise les tops documents retournés par la recherche simple pour extraire les termes d'expansion et reformuler les requêtes. Pour cela nous avons fait varier le nombre de documents de 5 à 30 avec un pas de 5 et pour chaque étape nous avons fait varier le nombre de termes d'expansion de 10 à 30 avec un pas de 5

Nombre de documents	Nombre de terme a ajouté	MAP
5	10	0.1348
	15	0.1351
	20	0.1352
	25	0.1354
	30	**0.1356**
10	10	0.1345
	15	0.1347
	20	0.1348
	25	0.1354
	30	0.1349
15	10	0.1347
	15	0.1348
	20	0.1349
	25	0 ,1350
	30	0.1349
20	10	0.1315
	15	0.1311
	20	0.1304
	25	0.1354
	30	
30	10	0.1350
	15	
	20	
	25	
	30	0.1336

Tableau III .2 Résultat obtenu avec notre approche.

Les résultats obtenus montrent qu'une amélioration mineure a été observée avec les valeurs suivantes :

Nombre de documents=5

Nombre de termes=30

On analysons les résultats obtenue par l'évaluation par requête :

- Nombre de requêtes amélioré : 12
- Nombre de requêtes dégradées : 6
- Nombre de requêtes nulles : 32

La figure suivante illustre les résultats obtenue on évalue requête par requête

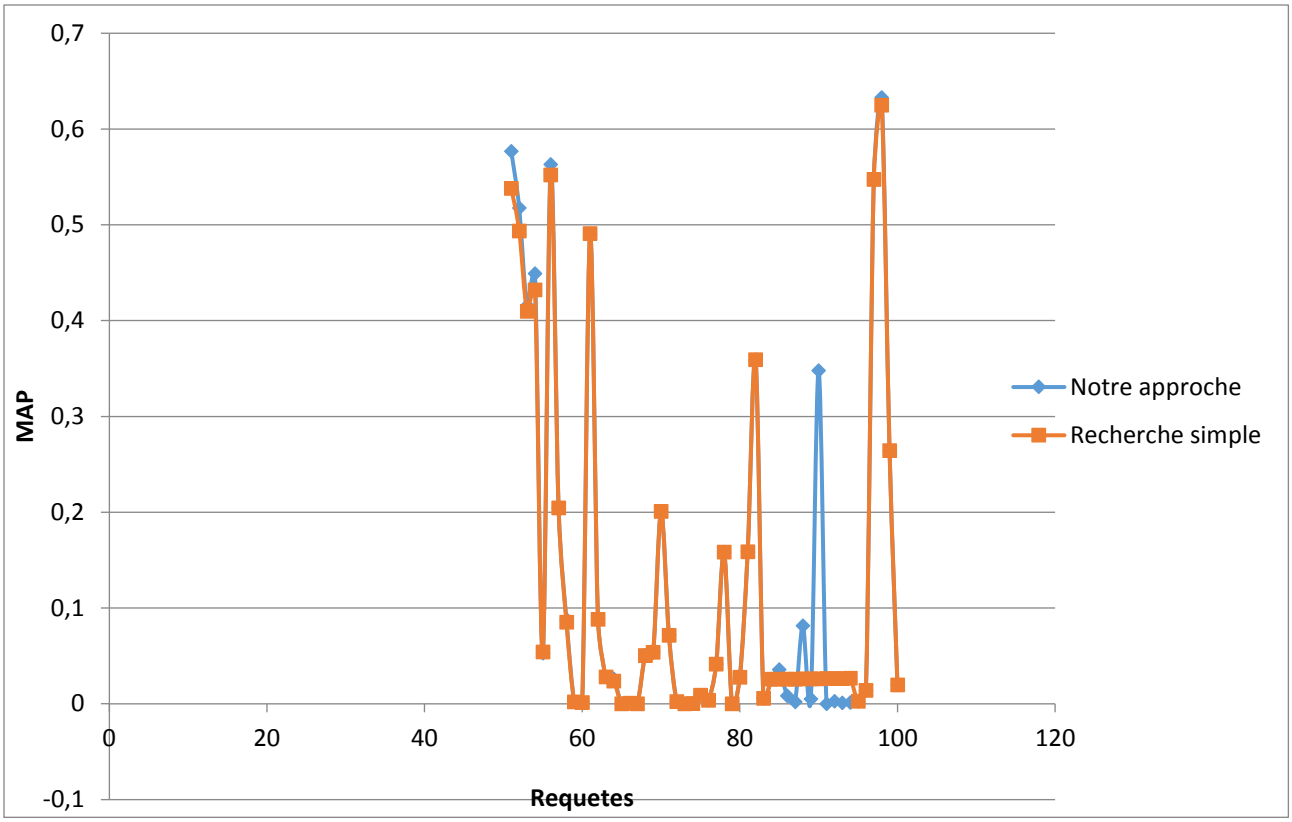


Figure III. : Résultats d'évaluation par requête.

III.5 Conclusion

Dans ce chapitre, nous avons présenté le principe de notre approche et son fonctionnement, nous avons aussi présenté l'environnement d'implémentation et les tests de notre approche.

Enfin, nous avons présenté et discuté les résultats de notre approche vis-à-vis du modèle de recherche BM25, une amélioration est résulté avec notre approche basée sur le word embedding.

Conclusion générale

Notre projet se situe dans le domaine de la recherche d'information, plus Particulièrement dans le cadre de la reformulation de la requête. Il porte sur l'Implémentation et l'évaluation d'une méthode de sélection de termes d'expansion basée sur la technique des word embeddings.

Pour mener à terme notre travail, nous avons donné un aperçu général sur la recherche d'information ainsi que les systèmes de recherche d'information. Nous avons ensuite étudié la technique word embedding et quelque modèles qui se basent sur les réseaux de neurones.

Nous avons aussi étudié l'utilisation de cette technique « word embedding »dans la recherche d'information.

Pour mettre en œuvre notre approche d'expansion de requêtes basées sur les word embedding, nous avons utilisé la plateforme terrier. Des améliorations ont été constatés par rapport à la recherche simple

Ce travail nous a permis d'aborder le domaine de la recherche d'information, d'enrichir nos connaissances et plus précisément :

- Approfondir nos connaissances sur la recherche d'information.
- Mettre l'accent sur la manière dont les systèmes de recherche d'information fonctionnent dans la plate-forme Terrier.
- Améliorer nos connaissances sur les réseaux de neurones.

Bibliographie

[1] Boughanem M, Jaques S : « Recherche d'information état des lieux et perspectives » Lavoisier, 2008.

[2] **:** Hernandez N. "Ontologie de domaine pour la modélisation du contexte en recherche d'information", thèse de doctorat en informatique, Université Paul Sabatier. (2006)

[3] : Daoud M. "Accès personnalisé à l'information : approche basée sur l'utilisation d'un profil utilisateur sémantique dérivé d'une ontologie de domaines à travers l'historique des sessions de recherche", thèse de doctorat en informatique, Université Paul Sabatier. (2009).

[4]MR.ABDELKRIM BOURAMOUL. « Recherche D'information Contextuelle et semantique sur le web », thèse de doctorat en informatique, Université MENTOURI de Constantine ,2011

[5] Jian-Yun Nie, cours hiver 2004; Module recherche d'information ; université Montreal. Départment D'informatique et de recherche opérationelle (I.R.O) Hiver 2004.

[6] Hammache Arezki : "le recherché d'information : modèle de langue combinant mots simple et mots composés". Thèse de doctorat en informatique, Université Mouloud Mammeri de Tizi Ouzou, 2013.

[7] C. CLEVERDON. Progress in documentation. Evaluation of information retrieval systems. Journal of Documentation, 1970.

[8] Bilel MOULAHI : « Définition et évaluation de modèles d'agrégation pour l'estimation de la pertinence multidimensionnelle en recherche d'information » UNIVERSITÉ DE TOULOUSE, 2015.

[9] C. Tambellini. "Un système de recherche d'information adapté aux données incertaines : adaptation du modèle de langue". Thèse de doctorat en informatique, Université de Nice-Sophia Antipolis-UFR sciences, 2007.

[10] Ren,F, Fan, L, Nie, J-Y. SAAK Approach: How to Acquire Knowledge in an Actual Application System. International Conference on Artificial Intelligence and Soft Computing, Honolulu, pp.136-140, 1999.

[11] C. Jacquemin, B. Daille, J. Royanté, and X. Polanco. In vitro evaluation of a program for machine-aided indexing.Inf. Process. Manage, 2002.

[12] Tamine L, Boughanem M, Daoud M "Evaluation of contextual information retrieval effectiveness: overview of issues and research". Journal of Knowledge and
Information Systems. Volume 24 Issue 1, pp. 1-34. Springer, Londres, United Kingdom. July 2010.

[13] Porter MF. An algorithm for suffix stripping. In: Readings in information retrieval. San Francisco, CA: Morgan Kaufmann Publishers Inc.. p. 313-316. 1997.

[14] Mohamed Ben Aouicha **:** « Une approche algébrique pour la recherche d'information structurée ». Thèse de doctorat en informatique, université Paul Sabatier.

[15] Boubekeur-Amirouche F. Contribution à la définition de modèles de recherche d'information flexibles basés sur les CP-Nets. Thèse de doctorat. Toulouse : Université Paul Sabatier ; 2008.

[16]Kraf,D.Hand Buell,D.A.Fuzzy sets and generalized Boolean retrieval systems.International on Man-Manchine Studies,19:pp.49-56-1983

[17]Salton,G. ,E.A Fox,H. Wu.Extended Boolean information retrieval system .CACM 26(11),app.1022-1039,1983.

[18]Soule-Dupuy C., Recherche et exploration d'information, Objectifs de la RI, Cours Master 2IH, 2006/2007.

[19] 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.

[20] : **Mikolov et al**. (**2013c**) introduced a new evalua- tion scheme based on word analogies that probes the finer structure of the word vector space.

[21] Rémi Philippe LEBRET "**Word Embeddings for Natural Language Processing",** thèse de doctorat en informatique, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2016

[22] Mikolov, Tomas; Sutskever, Ilya; Chen, Kai; Corrado, Greg; Dean, Jeffrey (2013). "Distributed Representations of Words and Phrases and their Compositionality".

[23] R. Lebret and R. Collobert, "Word embeddings through hellinger pca," in Proceedings of the 14th Conference of the European Chapter of the Association for Computational linguistics(EACL). Association for Computational Linguistics, 2014, pp. 482–490.

[24] J. Anderson, E. Rosenfeld, Neurocomputing : Foundations of research, MIT Press, Cambridge, Second printing, ISBN 0-262-01097-6, 1988.

[25] Philippe POINCOT « Classification et recherche d'information bibliographique par l'utilisation des cartes auto-organisatrices, applications en astronomie » thèse de doctorat en informatique, 15 décembre 1999

[26] livre Réseaux de neurones G.Dreyfus J-M. Martinez M.Samuelides M.B.Gardon F.Badran S.Thiria L.Hérault sous la derection de Gérard Dreyfus Deuxième tirage 2002

[27]Younès Bennani « Apprentissage Connexionniste »Edition Hermès Sience ,2006

[28]G.Drefus,JM Martinez ,M.Samuelides,MB Gordon ,F Badran,S.Thiria,L.Hérault « Réseaux de neurones »Editions Eyrolleses,2002.

[29] Diagramof an artificial neuron crée par Chrislb , le 14 Juilet. 2005

[30] M.T. Hagan, H.B. Demuth, M. Beale, «Neural Network Design», PWS Publishing Compagny, 1995.

[31] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[32] J.C. Principe, N.R. Euliano, W.C. Lefebvre, «Neural and Adaptive Systems : Fundamentals through Simulations», Wiley, 2000.

[33] Jerzy Korczak, LSIIT, ULP ; « Réseaux neuronaux Perceptron Multi-Couche » ,2005.

[34] Marc Parizeau, *Réseaux de Neurones* (Le perceptron multicouche et son algorithme de retropropagation des erreurs), Université Laval, Laval, 2004, 272 p.

[35] Indexing by Latent Semantic Analysis. **Scott Deerwester**. Graduate Library School. University of Chicago. Chicago, IL 60637. Susan T. Dumais. George W.

[36] Yoshua **Bengio**, Réjean Ducharme, Pascal Vincent, Christian Jauvin. learning a clustering of the words (Brown et **al**., 1992, Pereira et **al**., 1993, .

[37] S. **Bengio** , Y. **Bengio**, Taking on the curse of dimensionality in joint for the structured language model, Proceedings of the **2003** conference on Empirical Bryan Perozzi , Rami **Al-**Rfou ,

[38] 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.

[39] Ronan **Collobert**, Jason Weston, Léon Bottou, Michael Karlen, Koray A standard benchmark setup is described in detail by Toutanova et **al**. (2003).

[40] Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 1–12.

[41] T **Mikolov**, I Sutskever, K Chen, GS Corrado, J Dean. Advances in neural information **2013** .

[42] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean "istributed Representations of Words and Phrases and their Compositionality"

[43] Jeffrey Pennington, Richard Socher, Christopher D. Manning GloVe: "Global Vectors forWord Representation" Computer Science Department, Stanford University, Stanford, CA 94305.

[44] Bengio et al., 2007; Ranzato et al., 2007; Vincent et al., 2008; ... 2007; Salakhutdinov et al., 2007 deep learning methods significantly

[45] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic
models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 2333–2338. ACM.

[46] Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 615–620. Association for Computational Linguistics.

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

[48] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 2333–2338. ACM.

[49] Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In Advances in Neural Information Processing Systems, pages 2042–2050.

[50] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 2333–2338. ACM.

[51] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014a). A latent semantic model with convolutionalpooling structure for information retrieval. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 101–110. ACM.

[52] Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A Deep Relevance Matching Model for Ad-hoc Retrieval. In The 25th ACM International Conference on Information and Knowledge Management, Indianapolis, United States.

[53] Mustapha BAZIZ : « indexation conceptuelle guidée par ontologie pour la recherche d'information ». Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, Décembre 2005.

[54] Sordoni, A., Bengio, Y., and Nie, J.-Y. (2014). Learning Concept Embeddings for Query Expansion by Quantum Entropy Minimization. In AAAI, pages 1586–1592.

[55] Ganguly, D., Roy, D., Mitra, M., and Jones, G. J. (2015). Word embedding based generalized language model for information retrieval. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 795–798. ACM.

[56] Gupta, P., Bali, K., Banchs, R. E., Choudhury, M., and Rosso, P. (2014). Query Expansion for Mixed-Script Information Retrieval.

[57] ALMasri, M., Berrut, C., and Chevallet, J.-P. (2016). A Comparison of Deep Learning Based Query Expansion with Pseudo-Relevance Feedback and Mutual Information. In European Conference on Information Retrieval, pages 709–715. Springer.

[58] H **Zamani**, WB **Croft**. Proceedings of the **2016** ACM on International Conference on the Theory of ...,**2016**. 29, **2016**. Estimating Embedding Vectors for Queries.

[59] Roy, D., Paul, D., and Mitra, M. (2016). Using Word Embeddings for Automatic Query Expansion. In ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR). arXiv:1606.07608.

[60] Amer, N. O., Mulhem, P., and Gery, M. (2016). Toward Word Embedding for Personalized Information Retrieval. In ACM SIGIR Workshop on Neural Information Retrieval (Neu-IR). arXiv:1606.06991.

[61] G. Amati. Probabilistic Models for Information Retrieval based on Divergence from Randomness. PhD thesis, Department of Computing Science, University of Glasgow, 2003.

[62]Researching and Building IR applications using Terrier Craig Macdonald & Ben He Department of Computing Science University of Glasgow,2008

[63]Object-oriented Programming with Java Barry J. Holmes, Daniel T. Joyce

[64] http://netbeans.org/features/java/osgi.html