MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

Mémoire en vue de l'obtention du diplôme master

Spécialité : Ingénierie des Systèmes d'Information

Thème

Applicationdu Deep Learning dans la Recherche d'Information

Réalisé par :

- **DENDANI Lamia**
- **KERMAS Dyhia**

Encadré par :

M me S. FELLAG

Année d'étude : 2019/2020

Remerciements

Tout d'abord, nous tenons à remercier le bon Dieu tout puissant de nous avoir donné le courage, la force et la patience d'achever ce modeste travail.

Nous voulons particulièrement dans un premier temps, remercier notre directrice de mémoire Madame ***FELLAG Samia*** d'avoir accepté d'encadrer notre travail. Nous tenons à lui exprimer notre profonde reconnaissance pour sa patience, sa disponibilité, son implication et surtout ses judicieux conseils du début à la fin, qui ont contribué à alimenter nos réflexions et nos motivations.

Nous désirons aussi remercier Madame ***Amirouche***, chef du département Informatique, ainsi que les professeurs de la faculté d'informatique d'avoir transmis leur savoir tout au long de ces cinq années, qui a contribué à la réussite de nos études universitaires.

Nous adressons nos vifs remerciements aux membres du jury d'avoir bien voulu examiner et évaluer notre travail.

Enfin, nous remercions tous ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail.

Dédicaces

À la mémoire de ma grand-mère paternelle Fetima.

Je dédie ce modeste travail

À mes très chers parents. Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour leurs sacrifices, qu'ils ont consenti pour mon instruction et mon bien être. Puisse Dieu, le très haut, leurs accorder santé, bonheur et longue vie.

À ma petite sœur *Elena.*

À mes frères *Aghiles, Malik* et *Mohammed-Ouassim.*

À ma sœur *Radia* et son mari *Mohand Arezki.*

À ma petite nièce *Ouardia,* ma petite boule d'énergie positive !

À toute ma famille et particulièrement mes cousines *Dahbia, Ouiza, Camélia, Zahra* et *Yasmine* qui ont toujours été là pour moi. Leurs soutiens inconditionnels et leurs encouragements ont été d'une grande aide.

À mes chers oncles et tantes, leurs époux et épouses, pour leurs encouragements ; à qui je souhaite le bonheur, la santé et la prospérité, ainsi qu'à leurs enfants.

À mes beaux-parents *Tassadit* et *Rabah* pour leurs soutiens et leurs compréhensions.

À mon fiancé *Hocine* d'avoir été un homme aussi merveilleux et attentionné et qui était toujours à mes côtés, je le remercie de ne m'avoir jamais déçu.

À mes très chères amies *Lydia*, *Sabrina* et *Biba*, je les remercie pour leurs présences, leurs écoutes, leurs confiances en moi et leurs soutiens durant mon chemin des études supérieurs.

À ma très chère amie et binôme *Dyhia,* avec laquelle j'ai pris beaucoup de plaisir à travailler. Nous avons formé une belle équipe, je la remercie pour tout ce qu'elle m'a apporté au cours de ces sept années partagées.

Lamia

Dédicaces

Avec l'expression de ma reconnaissance et mon amour sincère, je dédie ce modeste travail :

À la femme, qui a souffert sans me laisser souffrir, qui n'a jamais dit non à mes exigences et qui n'a épargné aucun effort pour me rendre heureuse : ma chère maman *Fatima*.

À l'homme, qui a sacrifié sa vie pour m'offrir le confort nécessaire pour achever mes études et réussir dans ma vie : mon papa *Chabane*.

Aux plus belles deux sœurs du monde *Katia* et *Lilia* et à mon cher frère *Aghiles*, qui ont toujours été là pour moi et pour m'orienter dans mes choix.

À mes adorables deux petits cousins *Jugurta* et *Micipsa*, qui ont toujours apporté de la joie dans mon cœur par leurs présences, à qui je souhaite une vie pleine de bonheur et de réussite.

À mes grands-parents, mes oncles, leurs femmes et leurs enfants, et mes tantes qui se sont toujours inquiétés pour moi dans mes moments de stress et de difficulté. Que dieu leur donne une longue et joyeuse vie.

À mes meilleures amies *Lydia* et *Mélissa* qui ont toujours étaient à l'écoute, pour me conseiller, me soutenir et surtout me motiver.

À toutes les personnes qui m'ont apporté un savoir, une connaissance et une expérience dans ma vie.

Sans oublier ma plus belle rencontre depuis 7 ans, ma chère amie *Lamia* avant d'être mon binôme pour son soutien moral et ses judicieux conseils tout au long de ces années.

Dyhia

Résumé

Les développements récents des modèles de recherche d'information neuronaux ont montré des résultats satisfaisants sur les reconnaissances des formes et les traitements automatiques du langage naturel. La capacité de représenter avec précision du texte est essentielle à la compréhension du langage. Dans ce mémoire, nous décrivons une architecture neuronale pour la représentation des textes afin de calculer la similarité entre les paires document-requête. Nous nous sommes intéressées à intégrer les relations sémantiques entre les mots par une représentation vectorielle des mots de type *word2vec* pour résoudre le problème du bag of words (BOW), en utilisant les réseaux neuronaux profonds (DNN) afin de classer un ensemble de documents pour une requête donnée. Le réseau gère comme entrée un ensemble de vecteurs de termes (requête ou document) de haute dimension puis produit la représentation vectorielle correspondante à la séquence d'entrée. Cette représentation vise à capturer la sémantique relationnelle du document en projetant son contenu conceptuel dans la hiérarchie de la ressource externe.

Mots clés : recherche d’information, apprentissage profond, représentation vectorielle, représentation sémantique, appariement document/requête.

Tables des matières

Table des figures

Les services internet et web voient leur nombre d'accès (visites et visiteurs) continuer de croître, poussés par une évolution technologique qui ne faiblit pas. L'explosion quantitative des données numériques a obligé les chercheurs à trouver de nouvelles manières concernant le stockage, la recherche, l'analyse et la représentation des données comme le big data[1] et le data mining[2], pour pallier aux nombreux problèmes d'accès à l'information. Cette nouvelle explosion a propulsé la recherche d'information au premier plan.

L'enjeu d'un système de recherche d'information (SRI) est de satisfaire l'utilisateur, en lui renvoyant les informations pertinentes qu'il recherche. Les modèles de classement neuronal pour la recherche d'information (RI) utilisent des techniques d'intelligence artificielle (IA) telle que l'apprentissage profond (Deep Learning).

Les approches du « deep learning » sont largement exploitées dans le traitement d'images, notamment dans la reconnaissance d'objets [Socher et al., 2012b] et la classification d'images [Ciregan et al., 2012], également dans la reconnaissance de la parole [Graves et al., 2013] et le traitement du langage naturel (TAL) [Socher et al., 2012a]. Ces techniques ont été aussi utilisées en RI, dans le but d'améliorer les résultats de recherche. Elles sont exploitées aussi bien dans le processus de calcul de similarité (l'appariement) que dans la construction des représentations pour les documents et les requêtes. Ces travaux, dits Neural Language Modeling, partagent la caractéristique de représentation vectorielle des mots [Mikolov et al., 2013b], des séquences [Severyn et Moschitti, 2015], des paragraphes et du document tout entier [Le et Mikolov, 2014].

Ce travail se situe dans le contexte de la recherche d'information (RI) utilisant des techniques d'intelligence artificielle basées sur les réseaux de neurones profonds (Deep Learning). Il s'intéresse à une des tâches d'un SRI qui est la représentation des documents/requêtes.

L'objectif de ce travail est de proposer un nouveau modèle, utilisant les méthodes du deep learning, afin de construire un modèle de représentation basée sur la sémantique des textes, et permettant de pallier aux problèmes de l'inadéquation du vocabulaire relatifs aux représentations par sac de mots, ou bag of words (BoW), utilisées dans les modèles classiques de la RI. Ce problème apparaît lorsque les séquences de textes à apparier n'utilisent pas le même vocabulaire, même si leurs sujets sont liés. Les représentations en BoW ignorent donc plusieurs aspects, tels que la structure du texte et le contexte des mots. Ces caractéristiques sont très importantes et permettent de différencier deux textes utilisant les mêmes mots et dont les informations exprimées sont différentes.

[1] Big data : un concept permettant de stocker un ensemble très volumineux de données sur une base numérique.

[2] Data mining : un processus d'extraction de connaissances à partir de grandes bases de données, à l'aide des méthodes statistiques, mathématiques et de reconnaissance de formes.

Organisation du mémoire

Ce mémoire est constitué de 4 chapitres :

Dans le premier chapitre intitulé « La recherche d'information », nous allons rappeler les concepts de base de la RI, en commençant par définir les notions de : document, requête et pertinence ainsi que le processus d'indexation, d'appariement et de reformulation de la requête. Ensuite, nous décrivons les trois modèles de base de la recherche d'information à savoir : le modèle booléen, le modèle vectoriel et le modèle probabiliste. Enfin, l'évaluation des systèmes de recherche d'information basée sur les collections de tests et nous terminons ce chapitre avec une conclusion.

Dans le deuxième chapitre « Les réseaux de neurones », nous définissons qu'est-ce qu'un réseau de neurones artificiels et un réseau de neurones profond avec leurs architectures, en mettant le point sur l'apprentissage des deux réseaux de neurones, dont le premier est effectué à l'aide de trois approches distinctes ; et le deuxième avec une notion appelée rétro-propagation (correction des erreurs). Nous présentons également plusieurs typologies de réseaux de neurones profonds, adaptées soit au traitement de l'image, au son ou encore au texte.

Dans le troisième chapitre « l'application du deep learning à la recherche d'information », nous passons d'abord en revue l'état de l'art des travaux antérieurs qui exploitent les approches du deep learning dans la recherche d'information. Parmi ces travaux, ceux qui sont basés sur l'appariement de deux textes (document/requête) ; l'objectif principal de cette catégorie de travaux est de calculer le score de pertinence d'un document vis-à-vis d'une requête. Nous citons également les travaux qui visent la construction des représentations pour les documents et les requêtes en se basant sur la sémantique généralement. Nous clôturons ce chapitre par une synthèse des travaux présentés et une conclusion.

Dans la dernière partie de ce mémoire : chapitre 4 intitulé « contribution », nous exposons notre contribution, nous décrivons l'architecture de notre modèle basé sur l'état de l'art du chapitre 3, et l'algorithme qui explique son mode de fonctionnement. Nous finissons ce présent mémoire par une conclusion générale et des perspectives à entreprendre pour enrichir notre travail.

Chapitre I

Recherche d'information (RI)

Introduction

La recherche d'information (*RI*) fut fondée peu de temps après l'avènement des premiers ordinateurs, et constitue certainement la plus ancienne application de l'informatique à l'accès aux documents électroniques. Après la révolution d'internet et ses milliards de pages accessibles sur la toile, la RI est plus que jamais d'actualité.
La définition la plus admise pour la recherche d'information : consiste à trouver des ressources, habituellement des documents qui sont de nature non structurée (des textes) pour répondre à un besoin d'information parmi une large collection qui est généralement stockée dans des serveurs.
L'objectif principal de la RI est de répondre au besoin d'information des utilisateurs. Pour ce faire, un système de recherche d'information (*SRI*) utilise des modèles et des algorithmes permettant de sélectionner dans une collection les informations (items, documents., etc.) les plus pertinentes répondant à ce besoin en information.
Dans ce chapitre, nous présentons les concepts de base de la RI, nous commençons par définir les notions de : document, requête et de pertinence ainsi que les processus d'indexation, de recherche et de reformulation de la requête. Ensuite, nous décrivons les trois techniques (modèles) basiques de la recherche d'information. Enfin l'évaluation des systèmes de recherche d'information.

I.1 Concepts de base de la recherche d'information

Le processus de la recherche d'information est réalisé à partir d'un outil informatique appelé système de recherche d'information. Un système de recherche d'information (*SRI*) est un ensemble de techniques informatiques permettant de retourner des documents spécifiques dits pertinents à partir d'une collection de documents répondant le mieux à un besoin en information d'un utilisateur exprimé sous forme de requête.
Un SRI est l'interface entre l'utilisateur et la masse de documents [Allala et Mouas, 2017], intègre un ensemble de procédures et d'opérations pour la gestion, le stockage, la recherche et l'exploration des documents pertinents répondant aux besoins en information des utilisateurs.
Le fonctionnement global d'un SRI est représenté schématiquement par *le processus en U* (figure I.1).

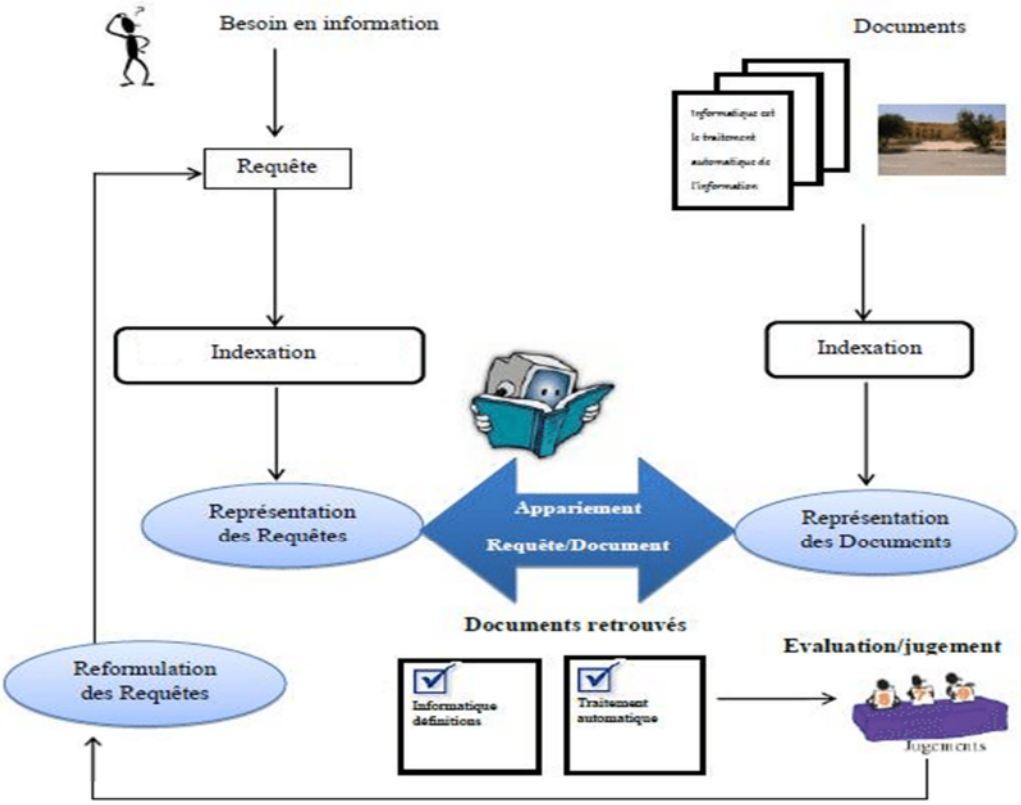


Figure I.1: Architecture générale d'un système de recherche d'information (le processus en U). [Adil et al., 2015]

La figure met particulièrement en évidence les 3 notions de base d'un SRI : document et requête qui sont des conteneurs d'information, et la pertinence ; ainsi que ses 3 fonctionnalités principales: l'indexation, l'appariement et la reformulation de requêtes.

I.1.1 Document

Un document représente toute unité retournée par un SRI à l'utilisateur comme réponse à son besoin en information. Un document peut être un texte, une page web, une image, une bande vidéo,…, etc.
Un ensemble de documents forment une collection autrement dite base documentaire ou corpus.

I.1.2 Requête

Une requête est l'expression mentale formulée par l'utilisateur afin d'exprimer son besoin en information. Plusieurs types d'interrogations proposées en RI pour exprimer une requête :

- Une liste de mots clés : cas des systèmes SMART [Salton, 1971] et OKAPI [Robertson et Walker, 1999].
- Langage naturel : cas des systèmes SMART [Salton, 1971] et SPIRIT [Fluhr et Debili, 1985], sous forme de texte libre (langage ordinaire).
- Langage booléen : cas du système DIALOG [Bourne et Anderson, 1979], un mot clé ou une liste de mots clés incluant des opérateurs logiques.
- Langage graphique : cas du système NEURODOC [Lelu et François, 1992], à partir d'une interface graphique.

I.1.3 Pertinence

La notion de pertinence est très complexe, elle constitue l'objectif principal de la RI. Grâce à cette notion le SRI juge si un document doit être retourné à l'utilisateur ou pas. Un document retourné est dit document pertinent, signifie qu'il répond au besoin en information de l'utilisateur. Elle peut être définie comme étant la correspondance entre un document et une requête, ou encore une mesure d'informativité du document à la requête.
Il existe deux niveaux de pertinence : pertinence utilisateur et pertinence système.

- **Pertinence utilisateur :** c'est le jugement apporté par l'utilisateur aux documents restitués par le SRI comme réponse à sa requête. La pertinence utilisateur est subjective [Hammache, 2013], car un document peut être jugé différemment pour la même requête par deux utilisateurs distincts chacun selon sa satisfaction.
- **Pertinence système :** inversement à la pertinence utilisateur, cette pertinence est objectif [Hammache, 2013]. C'est l'évaluation calculée par le SRI de l'adéquation entre des documents et une requête d'où chaque document pertinent retourné par le SRI possède un degré de pertinence vis-à-vis de la requête.

I.1.4 Indexation

L'indexation est un processus qui consiste à extraire un ensemble de mots clés (*sac de mots*) d'un document (ou une requête) qui couvre au mieux son contenu sémantique afin de le décrire le plus précisément possible et le retrouver au sein d'un corpus. Autrement dit, l'indexation est donc la réduction du volume des données d'un document par le biais d'une représentation de ce document par des mots clés [El Hachani, 1997].
Le but de l'indexation est de construire au mieux une représentation interne (*index*) des documents qui va faciliter et accélérer la recherche.
L'ensemble de ces mots clés est stocké dans un descripteur appelé fichier index. Ce dernier peut contenir des mots simples ou un groupe de mots qui peuvent être extraits de trois façons : manuelle, automatique ou semi-automatique.

I.1.4.1 Indexation manuelle

Chaque document du corpus est analysé par un expert en indexation (indexeur) ou un documentaliste. L'indexation manuelle permet la recherche par concepts (sujets, thèmes) ; meilleure que la recherche par mots simples. Néanmoins, ce mode d'indexation est très coûteux en temps surtout pour construire le vocabulaire et de classifier les documents par concepts lorsqu'il s'agit des collections de document très volumineuses (le web par exemple). Cette indexation basée sur un jugement humain est donc subjective, plusieurs indexeurs affectent des termes différents à un même document.

I.1.4.2 Indexation semi-automatique

Ce type d'indexation est assuré par une combinaison d'un expert du domaine et un programme informatique, le choix final des descripteurs reste au spécialiste du domaine ou documentaliste, qui intervient souvent pour établir des relations sémantiques entre mots-clés et choisir les termes significatifs.

I.1.4.3 Indexation automatique

C'est l'approche adoptée en RI à cause de la taille volumineuse du corpus et du fait qu'elle remédie au problème du choix des descripteurs : fournit toujours le même index pour le même

document, ce qui reflète la qualité du système. Chaque document est analysé à l'aide d'un processus entièrement automatisé qui comprend cinq phases comme suit : l'analyse lexicale, l'élimination des mots vides, la normalisation (lemmatisation ou racinisation), la pondération des termes et enfin la création de l'index. Dans ce qui suit nous allons définir chacune de ces étapes.

I.1.4.3.1 Analyse lexicale

C'est l'étape initiale indispensable pour définir l'ensemble des termes significatifs qui seront extraits automatiquement d'un document. Elle implique la conversion d'une chaîne de caractères (le texte du document) en une séquence d'unités lexicales élémentaires dites *tokens*. Un terme (*token* : mot simple ou groupe de mots) est donc une suite de caractères séparée par un blanc ou signe de ponctuation, caractère spécial ou un nombre. Cette étape inclut la conversion de la casse (majuscule en minuscule) et l'élimination des accents.
Elle dépend très fortement de la langue, ce qui mène tout d'abord à identifier la langue d'un document pour définir le moyen à adopter afin de séparer les mots.

I.1.4.3.2 Elimination des mots vides

Tous les mots d'un document sont-ils également importants ? Clairement non [Manning et al, 2008] ; elle consiste à retirer les mots qui donnent pas une description précise du concept du document, appelés mots vides (*stop words*), et ils sont conservés dans un anti-dictionnaire (*stoplist*). Ces mots sont tous les mots les plus courants et porteurs d'une information faible comme les articles "le, ce", les verbes fonctionnels "être, avoir", les conjonctions "et, ou" les adverbes "comme, bientôt", etc. La suppression de ces mots vides engendre la diminution du stockage et de la taille de l'index.

I.1.4.3.3 Normalisation

Beaucoup de mots qui restent de la précédente étape ont des formes légèrement différentes et un sens très similaire. L'objectif de la normalisation est d'obtenir une seule forme (lemme ou racine) pour représenter ces variances morphologiques.

- **Lemmatisation :** c'est une analyse linguistique, consiste à trouver la forme canonique d'un mot à condition qu'il ait du sens et existe dans le dictionnaire.
  Exemple : infinitif pour les verbes : recherchons →rechercher,
  masculin singulier pour les noms: professionnelles →professionnel, etc.
- **Racinisation :** (*stemming/ désuffixation*) c'est de ramener toutes les formes d'un même mot à une seule racine en tronquant les suffixes, terminaisons de la conjugaison, les marques de genre, du pluriel, etc. Cette approche est utilisée dans l'algorithme Porter stemming [Porter, 1980] qui est largement adapté à la langue anglaise.
  Exemple : dynamic, dynamics, dynamically seront représentés par le terme dynamic.

La normalisation peut s'appuyer sur plusieurs stratégies à savoir : table de correspondance, troncatures simples [Denjean, 1989], étiquetage, utilisation des N_grammes [Adamson et Boreham, 1974] et suppression des suffixes [Porter, 1980].

I.1.4.3.4 Pondération des termes

La pondération est une fonction fondamentale en RI [Fellag, 2018] ; mesure l'importance d'un terme dans un document ou une requête afin d'éliminer les termes trop fréquents ou trop rares,

Par exemple comme le suggère la loi de Zipf [Zipf, 1949] : 'la fréquence d'utilisation d'un mot dans un texte était inversement proportionnelle à son rang'. Cette importance dépend de deux fréquences statistiques : pondération globale et pondération locale.
L'approche la plus simple consiste à attribuer un poids à chaque terme qui est égal au nombre d'occurrences d'un terme *t* dans le document *d*.
Ce schéma de pondération est appelé *term frequency* et est noté $tf_{t,d}$ [Manning et al., 2008], il représente la pondération locale; définit le degré d'informativité (représentativité) de ce terme dans le document comme suit :

$$tf = n_{oc}$$

tf: plus un terme est fréquent dans un document plus il est important dans la description de ce document.

$$tf_{t,d} = nf_{t,d}$$

$\boldsymbol{nf_{t,d}}$ **:** nombre de fois où *t* apparaît dans *d*.
La pondération globale est le pouvoir de discrimination d'un terme dans le corpus, autrement dit, l'inverse du nombre d'occurrence d'un terme dans la collection de documents et est notée idf_t (*Inverse Document Frequency* d'un terme *t*).

$$idf_t = log\left(\frac{N}{df_t}\right)$$

t : un terme.
N : le nombre total de documents dans une collection.
$\boldsymbol{df_t}$ **:** le nombre de documents de la collection contenant un terme *t* (fréquence du document).

La combinaison des deux facteurs : fréquence de terme (*Term Frequency*) et fréquence inverse du document (*Inverse Document Frequency*) produit un poids pour chaque terme dans un document. Ce poids est calculé comme suit :

$$W_{t,d} = tf_{t,d} \times idf_t$$

$\boldsymbol{W_{t,d}}$ **:** poids d'un terme *t* dans un document *d*.
Le *TF* IDF* c'est une évaluation de la pertinence d'un document vis-à-vis d'un terme.

I.1.4.3.5 Création de l'index

L'index est l'ensemble des termes pondérés de chaque document ; ordonnés alphabétiquement où chaque terme est muni de sa fréquence (nombre d'occurrences) dans les documents où il apparait et d'un pointeur vers le début d'une liste inversée *'posting list*' [Zobel et Moffat, 2006][Mechach, 2016] contenant une série de couples :

- Les identifiants *d* des documents contenant *t*, représentés par des numéros de documents ordinaux ; et
- L'ensemble de fréquences $f_{d,t}$ associées des termes *t* dans le document *d*. [Boucham, 2009]

Les fichiers index rendent la recherche plus rapide grâce au fichier inverse (l'ensemble des listes inversées) qui peut aussi stocker les positions des termes.

I.1.5 Appariement document-requête

Les SRI intègrent un processus de recherche/décision qui permet de trier et sélectionner l'information jugée pertinente pour l'utilisateur. Ce processus intègre une fonction d'appariement basée sur une mesure de similarité sémantique entre les termes de la requête

(indexée) d'un utilisateur et les descripteurs des documents de la collection. Le résultat de cette comparaison détermine le degré de pertinence (ressemblance) d'un document du corpus par rapport à une requête.
Cette fonction est notée *RSV(Q, D)* (*Retrieval Status Value*), où *Q* est une requête et *D* est un document de la collection.
Il existe deux types d'appariement :

- **Appariement exact :** le résultat est une liste de documents respectant exactement la requête spécifiée avec des critères précis. Les documents retournés ne sont pas triés.
- **Appariement approché :** le résultat est une liste de documents pertinents pour la requête. Les documents retournés sont triés selon un ordre de degré de pertinence document /requête.

I.2 Les modèles de recherche d'information

Un modèle de recherche d'information représente le noyau d'un SRI. Le modèle remplit deux fonctions :

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

Dans ce qui suit nous présentons les modèles da la RI classique basés sur trois théories différentes :

- **Théorie des ensembles :** où un document est représenté sous un ensemble de termes, de même pour une requête, exemple du modèle booléen.
- **Théorie algébrique :** basée sur le calcul des distances algébriques pour mesurer la pertinence d'un document vis-à-vis d'une requête dans un espace vectoriel, exemple du modèle vectoriel.
- **Théorie probabiliste :** basée sur le calcul de la probabilité de pertinence d'un document pour une requête.

On distingue alors trois catégories de modèles : modèles booléens, modèles vectoriels et modèles probabilistes.

I.2.1 Le modèle booléen

Le modèle booléen [Salton, 1971] dit logique est le premier modèle de la RI. Il repose sur la théorie des ensembles et de l'algèbre de Boole.
Dans ce modèle, un document est représenté par l'ensemble des termes qui l'indexent, et une requête est une expression booléenne constituée d'un ou plusieurs termes reliés à l'aide des connecteurs logiques AND (∧), OR (∨) et NOT (¬).
Lors de l'appariement, le SRI retourne les documents qui correspondent exactement à la requête et la pondération des termes est définie selon la fonction de présence/absence du terme dans le document (1/0 respectivement).
Le score de similarité (*RSV : Retrieval Status Value*) entre un document *d* et une requête *q* est déterminé comme suit :
$RSV(d, t_i) = 1$ si t_i appartient *d* ; *0* sinon.
$RSV(d, q_i \ AND \ q_j) = 1$ si $RSV(d, q_i) = 1$ et $RSV(d, q_j) = 1$; *0* sinon.

RSV(d, q_i OR q_j) = 1 si *RSV(d, q_i) = 1* ou *RSV(d, q_j) = 1* ; *0* sinon.
RSV(d, NOT q_i) = 1 si *RSV(d, q_i) = 0* ; *0* sinon.
L'avantage de ce modèle repose sur sa simplicité de mise en œuvre et sa capacité de ne retourner que les documents qui répondent au besoin en information de l'utilisateur grâce à l'appariement exact. La principale faiblesse de ce modèle est son critère de pertinence binaire qui cause le non-ordonnancement des documents selon un degré de pertinence (aucun terme n'est plus important qu'un autre soit 0 soit 1). Parmi les inconvénients de ce modèle on trouve aussi la formulation booléenne des requêtes non accessible à un large public et le nombre de documents retournés pouvant être considérable.

I.2.2 Le modèle vectoriel

Le modèle le plus populaire en RI proposé par [Salton, 1971] et concrétisé dans le système SMART. Dans ce modèle l'ensemble des termes générés par l'indexation forme un espace vectoriel à *n* dimensions sous la forme $< t_1, t_2, \dots, t_n >$. Un document et une requête sont représentés à l'aide d'un *'vecteur de poids'* [Msaaf et Belmajdoub, 2015] dont les coordonnées dépendent de la fréquence d'un terme dans le document ou la requête. La représentation de ces derniers peut être comme suit :
Document : $d_j = (w_{1j}, w_{2j}, \dots, w_{nj})$
Requête : $q = (w_{1q}, w_{2q}, \dots, w_{nq})$
Où $\boldsymbol{w_{iq}}$ **:** est le poids de terme t_i dans la requête *q*,
et $\boldsymbol{w_{ij}}$ **:** son poids dans le document d_j.
La distance entre le vecteur document et le vecteur requête traduit la similarité entre un document et une requête qui permet de retourner les documents correspondants à une requête lors de l'appariement : plus la distance est petite plus le document est pertinent pour la requête. Pour calculer cette distance plusieurs mesures sont mises à dispositions dont les plus courantes :

- **Le produit scalaire :**

$$RSV(D_j, Q_k) = \sum_{i=1}^{N} d_{ij} q_{ik}$$

- **La mesure du cosinus :**

$$RSV(D_j, Q_k) = \frac{\sum_{i=1}^{N} d_{ij} q_{ik}}{\sqrt{\sum_{i=1}^{N} d_{ij}^2 \times \sum_{i=1}^{N} q_{ik}^2}}$$

- **La mesure de Dice :**

$$RSV(D_j, Q_k) = \frac{2 \sum_{i=1}^{N} d_{ij} q_{ik}}{\sum_{i=1}^{N} ({d_{ij}}^2 + {q_{ik}}^2)}$$

- **La mesure de Jaccard :**

$$RSV(D_j, Q_k) = \frac{\sum_{i=1}^{N} d_{ij} q_{ik}}{\sum_{i=1}^{N} {d_{ij}}^2 + \sum_{i=1}^{N} {q_{ik}}^2 - \sum_{i=1}^{N} d_{ij} q_{ik}}$$

Contrairement au modèle booléen, l'expression des requêtes est plus simple et sa performance est meilleure grâce à la pondération des termes permettant d'ordonner les documents par ordre de pertinence décroissante vis-à-vis de la requête, mais dans ce modèle les résultats

d'appariement sont partiels. L'inconvénient majeur du modèle vectoriel est qu'il est fondé sur l'hypothèse de l'indépendance entre les termes alors que la sémantique d'un document est dans les liaisons entre ses termes. Pour remédier à cette critique, un autre modèle à été conçu : le modèle vectoriel généralisé (*Generalized Vector Space Model*) [Wong et al, 1985] dans le but d'intégrer la dépendance entre les termes.

I.2.2.1 Le modèle LSI

Le modèle LSI (*Latent Semantic Indexing*) est une variante du modèle vectoriel ; il se base sur la décomposition des termes d'indexation. La technique LSI construit un espace sémantique de termes à partir d'un corpus de textes, cet espace est ensuite réduit avec la méthode de la décomposition en valeurs singulières (*Singular Value Decomposition* en anglais, *SVD*) [Haddi et Messabih, 2019] en regroupant les termes co-occurrents (similaires) dans les mêmes dimensions. Cette réduction est le noyau de ce modèle car elle extrait les relations sémantiques et permet d'obtenir en résultat une nouvelle matrice terme-document améliorée.
Pratiquement, la technique LSI construit une matrice terme-document *A(m×n)* qui contient les fréquences tf_{ij} (l'occurrence du terme *i* dans le document *j*) où *m* est le nombre de termes qui apparaissent dans toute la collection, et *n* le nombre de documents. Ensuite, la matrice terme-document *A* est décomposée à l'aide de la technique mathématique SVD pour obtenir à la fin un nouvel espace contenant des matrices terme-document et un vecteur de requête transformé dans cet espace.
La puissance de la méthode LSI réside dans cette représentation transformée dont les documents et les requêtes sémantiquement similaires seront plus proches qu'avec les mots-clés. La pertinence d'un document vis-à-vis d'une requête dans ce modèle est définie par des mesures de distance dans un espace vectoriel.

I.2.3 Le modèle probabiliste

I.2.3.1 Le modèle de base

Le premier modèle de base a été proposé par *Maron* et *Kuhns* au début des années 60 ; Il se base sur le principe de classement probabiliste (*Probability Ranking Principle*), ce modèle considère la RI comme un espace d'événements possibles où un évènement peut être le jugement de pertinence par un utilisateur sur un document par rapport à une requête, il consiste donc 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.
La pertinence entre un document et une requête est mesurée par le rapport entre la probabilité qu'un document *D* donné soit pertinent pour une requête *Q*, notée *P(R/D)*, et la probabilité qu'il soit non pertinent, notée *P(NR/D)*, où *R* est l'évènement de pertinence et *NR* de non pertinence. Ces probabilités sont estimées par les probabilités conditionnelles selon la présence ou absence d'un terme dans un document pertinent ou non pertinent.

I.2.3.2 Le modèle de langue

C'est un modèle basé sur des méthodes statistiques appliquées sur un corpus d'entrainement ; utilisé dans les applications du traitement automatique de la langue (la reconnaissance de la parole, traduction automatique, étiquetage des catégories syntaxiques et la recherche d'information).

Pratiquement, ce modèle utilise une fonction de probabilité $P(q|M_d)$ qui consiste à estimer la probabilité qu'une requête q puisse être générée à partir du modèle de langue M_d. Le modèle le plus utilisé est le modèle uni-gramme où les textes sont générés à partir de mots simples, c'est-à-dire *M* génère des séquences de 1 mot simple de manière indépendante. La probabilité est exprimée ainsi :

$$P(q|M_d) = \prod_{t \in Q} P(t|M_d)$$

La probabilité $P(t|M_d)$ peut être estimée par maximum de vraisemblance (*Maximum Likelihood*). Elle est donnée par :

$$P(t|M_d) = \frac{tf(t,d)}{|d|}$$

Où : $\boldsymbol{tf}_{(t,d)}$ **:** est la fréquence du terme *t* dans le document *d*.

Pour pallier au problème des fréquences nulles (zéro) posé par l'absence d'un ou plusieurs mot de la requête dans un document (qui ont pour effet l'obtention d'un résultat nul pour la probabilité $P(t|M_d)$), plusieurs moyens peuvent être utilisés dont la technique appelée lissage (*Smoothing*) : le lissage de *Laplace* (ajouter-un), le lissage de *Good Turing*, le lissage de *Backoff*, le lissage par interpolation, etc. Leur principe est d'assigner des probabilités différentes de zéro pour les termes absents.

I.3 Reformulation de requête

En général, la requête initiale est formulée à partir de quelques mots-clés issus des connaissances de l'utilisateur sur un domaine donné, et cette formule permet rarement d'aboutir à un résultat qui satisfait explicitement son besoin en information. La RI a alors envisagé comme une suite de formulations et de reformulations de requêtes jusqu'à la satisfaction de ce dernier. Elle consiste de manière générale à enrichir la requête de l'utilisateur en ajoutant des termes permettant de mieux exprimer son besoin [Elyaleb, 2009][Sriti, 2012], autrement dit, il s'agit de modifier la requête initiale en ajoutant des termes plus significatifs et appropriés, on parle alors d'expansion de la requête de l'utilisateur. On distingue deux approches de reformulation de la requête dont : réinjection de pertinence (ou *Relevance Feedback*) et expansion automatique de la requête (ou *Query Expansion*).

I.3.1 Réinjection de pertinence

La réinjection de pertinence (rétroaction ou propagation) est la technique la plus répandue en RI. Un processus de reformulation automatique consistant à utiliser l'information de pertinence en analysant les documents retournés en réponse à la requête initiale afin de l'améliorer, autrement dit, les documents restitués supposés pertinents pour la recherche de la requête initiale servent comme outil pour transformer cette dernière en une nouvelle requête dont le but est de répondre le plus optimale possible au besoin en information de l'utilisateur. Cette transformation est faite soit de manière implicite (par le SRI) ou de manière explicite (par l'utilisateur).

I.3.1.1 Réinjection explicite

Ce procédé commence par la formulation de la requête initiale par l'utilisateur, ensuite le système lui renvoie la liste des documents résultats, puis l'utilisateur juge la pertinence des documents retournés de la recherche sur sa requête. À son tour le système sélectionne les termes importants et une nouvelle requête est formulée à partir de la requête initiale en

utilisant les termes considérés importants en appliquant un algorithme spécifique de réinjection de pertinence.

I.3.1.2 Réinjection implicite

Appelée aussi pseudo-réinjection de pertinence (*Blind Relevance Feedback*), l'information sur la pertinence des résultats retournés est déduite automatiquement par le SRI, elle pallie la contrainte de sollicitation de l'utilisateur pour juger la pertinence utilisée dans la réinjection explicite. Au lieu de juger explicitement les documents, cette méthode consiste à supposer que les premiers documents renvoyés par le système sont pertinents, sans les juger, ensuite ces documents sont utilisés pour reformuler la requête.

I.3.2 Expansion de requête

Les termes utilisés dans les requêtes utilisateur peuvent avoir plusieurs sens, ce qui rend ces requêtes ambigües d'une part, d'une autre part, ces termes ne correspondent pas souvent aux termes utilisés par les auteurs des documents pour décrire le même concept. L'expansion de requête est l'une des techniques utilisées pour résoudre ce problème, elle consiste à élargir la requête initiale en ajoutant des termes corrélés (liés sémantiquement) avec les termes de celle-ci.

Le choix de ces termes corrélés est déterminé à partir d'une étude statistique de corrélation des termes appliquée soit sur l'ensemble de tous les documents du corpus dite une *analyse locale*, soit sur des ressources externes comme le thésaurus. Ce dernier contient un vocabulaire standardisé regroupant l'ensemble des termes et les relations sémantiques et de synonymie entre les mots et les expressions d'un domaine particulier, cette technique est appelée *analyse globale*.

I.4 Evaluation d'un SRI

L'évaluation d'un SRI peut être élaborée selon des critères subjectifs ou objectifs qui reviennent à définir deux approches respectivement paradigme usager et paradigme système.

- **Paradigme usager** [Hammache, 2013][Chaudiron et Ihadjadene, 2002] : cette approche fait appel au jugement de l'utilisateur sur le temps de réponse, la présentation des résultats, l'effort requis par l'utilisateur pour trouver les documents satisfaisants ,etc.
- **Paradigme système** [Hammache, 2013][Chaudiron et Ihadjadene, 2002] : vise l'efficacité du système notamment sa performance d'appariement entre la requête et la base documentaire qui se traduit en sa capacité à retourner des informations pertinentes à la requête de l'utilisateur. La qualité d'un système doit être mesurée en comparant les réponses du système avec les réponses idéales que l'utilisateur espère recevoir selon des métriques pour quantifier la pertinence des réponses, parmi ces métriques le rappel et la précision détaillés ci-dessous.

L'évaluation est une étape importante qui contribue à la mise en œuvre d'un modèle de RI : elle permet de paramétrer le modèle, estimer l'impact de chacune de ses caractéristiques et enfin de fournir des éléments de comparaison entre les modèles. Cette évaluation se base sur deux éléments : les collections de tests et des mesures d'évaluation.

I.4.1 Collection de tests

Une collection de tests est constituée d'un ensemble de documents (corpus documentaire) et un ensemble de requêtes. Ces documents sont indexés manuellement avec un vocabulaire contrôlé, et les requêtes sont évaluées par des experts afin de déterminer les réponses souhaitées qui seront classées comme documents pertinents. Les résultats d'une recherche automatique sont comparés aux réponses souhaitées dans le but de mesurer la performance en termes de précision et rappel (jugement de pertinence).

I.4.2 Les mesures d'évaluation

L'évaluation du système sur la qualité des documents retournés à une requête consiste à répartir la base documentaire en 3 classes schématisées dans la figure I.2 :

1. Les documents pertinents correspondants réellement à la requête.
2. Les documents non pertinents ne correspondants pas réellement à la requête.
3. Les documents retournés par le SRI.

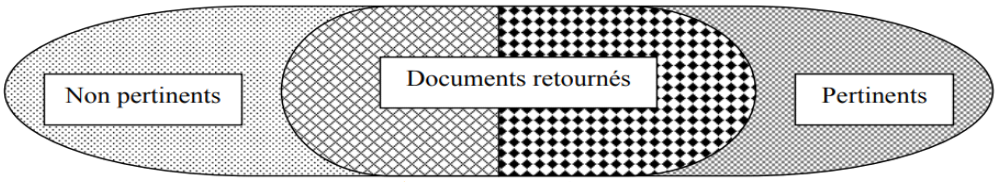


Figure I.3: Evaluation, classification des documents. [Boucham, 2009]

Cette notion de pertinence est définie à l'aide de deux métriques :

- **Précision :** détermine la capacité du système à ne choisir que les documents pertinents du corpus. Elle mesure la proportion de documents pertinents retrouvés *(PR)* parmi tous les documents retrouvés *(R)* par le système, avec la formule :

$$Précision = {}^{PR}/_{R}$$

- **Rappel :** détermine la capacité du système à retrouver tous les documents pertinents du corpus. Elle mesure la proportion de documents pertinents retrouvés parmi tous les documents pertinents *(P)* dans la base, avec la formule :

$$Rappel = {}^{PR}/_{P}$$

Ces deux métriques ne sont pas indépendantes, il y'a une forte relation entre elles : quand l'une augmente, l'autre diminue. Il ne signifie rien de parler de la qualité d'un système en utilisant seulement une des métriques.
Une précision égale à 1 signifie que le système n'a retrouvé que des documents pertinents.
Un rappel égal à 1 signifie que tous les documents pertinents ont été retrouvés.
Il existe deux notions fondamentales complémentaires à la précision *(P)* et au rappel *(R)* qui sont quantifiées pour mesurer cette qualité : respectivement le bruit et le silence.

- **Le bruit :** fait référence à l'ensemble des documents non pertinents restitués à l'utilisateur, exprimé par :

$$B = 1 - P$$

- **Le silence :** fait référence aux documents pertinents existants mais que le système n'a pas restitué, exprimé par :

$$S = 1 - R$$

Conclusion

Dans ce chapitre nous avons présenté les principales notions et concepts de la recherche d'information ainsi que les systèmes de recherche d'information (SRI) de manière générale. Initialement, nous avons introduit des notions de bases telles que le besoin en information d'un utilisateur (la requête), le document et la pertinence. Nous avons également décrit le processus de base d'un SRI, notamment le module d'indexation des documents et des requêtes ainsi que le processus d'appariement entre requêtes et documents (mesure de similarité/distance entre requêtes et documents) et la reformulation de requête. De plus, nous avons défini et expliqué les trois différents modèles basiques de la RI. Pour clôturer ce chapitre, nous avons évoqué la notion d'évaluation des systèmes de recherche d'information traitée à l'aide d'une collection de tests et des mesures d'évaluation.

À travers les différentes sections que nous avons étudiées, on peut constater que la recherche d'information définit des modèles et des systèmes afin de faciliter l'accès à un ensemble de documents se trouvant dans des bases documentaires. L'objectif est donc de permettre aux utilisateurs de retrouver les documents dont le contenu répond de façon optimale à leur besoin en information, il s'agit alors de retourner l'ensemble de documents pertinents. De là, on peut déduire que la notion de pertinence (jugement de pertinence) dépend de la satisfaction de l'utilisateur d'une part, et des différents sens portés par les termes de la requête d'une autre part.

Dans le chapitre suivant, nous allons présenter les différents concepts des réseaux de neurones et du deep learning, pour ensuite pouvoir établir un lien entre la recherche d'information et l'apprentissage profond (*Deep Learning*).

Chapitre II

Les Réseaux de neurones et le Deep Learning

Introduction

L'intelligence artificielle (*IA*) (*Artificial Intelligence* ou *AI*, en anglais nommé par John McCarthy) est apparu pour la première fois dans l'article de [Turing, 1950] qui s'est posé la question "les machines peuvent-elles penser ?".
Le but de l'IA est de concevoir des systèmes capables de reproduire le comportement de l'humain dans ses activités de raisonnement [Aïmeur, 2011], autrement dit, "Des systèmes qui pensent comme les humains" [Haugeland, 1985] et aller jusqu'à le battre dans des défis (Test de Turning) que d'après lui un ordinateur doit posséder la fonctionnalité d'apprentissage afin de pouvoir s'améliorer d'une façon autonome, et comme l'affirme aussi le patron de l'intelligence artificielle chez Facebook Yan le Cun : "Il n'y a pas d'intelligence sans apprentissage", d'où vient l'approche du machine learning (*ML*, *apprentissage automatique* en français). L'apprentissage automatique est la discipline qui vise à ce qu'une machine soit capable d'apprendre par elle-même sans être explicitement programmée. Un sous-ensemble de ML appelé *deep learning* (*DL*, *apprentissage profond* en français) vise à traiter et résoudre des tâches plus complexes comme la reconnaissance de formes, le traitement du signal, la mémorisation, la généralisation avec un apprentissage en profondeur et de précision davantage, en s'appuyant sur le mécanisme neuronal biologique traduit en réseaux de neurones artificiels (multicouches).
Dans ce chapitre nous allons mettre le point sur les concepts principaux des réseaux de neurones artificiels et multicouches.

II.1 Réseaux de neurones et réseaux de neurones profonds

II.1.1 Définition des réseaux de neurones

Un réseau neuronal artificiel (*RNA*) est un système de la technologie de l'information fondé sur des modèles mathématiques largement calqué du fonctionnement nerveux du cerveau humain.
Les réseaux de neurones artificiels[1] 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. [Touzet, 1992]
C'est un ensemble de neurones artificiels interconnectés qui constitue une architecture de calcul. Un RNA fonctionne dans un premier temps avec des exemples d'apprentissage en modifiant les paramètres du réseau jusqu'à obtention d'un résultat désiré et certain pour de nouvelles données.

II.1.2 Structure d'un réseau de neurones

Le neurone formel apparu en 1943 par Warren McCulloch et Walter Pitts est la représentation schématique du fonctionnement du cerveau humain. La figure II.1 montre les équivalences entre la structure du neurone biologique et neurone formel.

[1] Les réseaux de neurones artificiels = Les réseaux de neurones (Neural network) = les réseaux de neurones formels

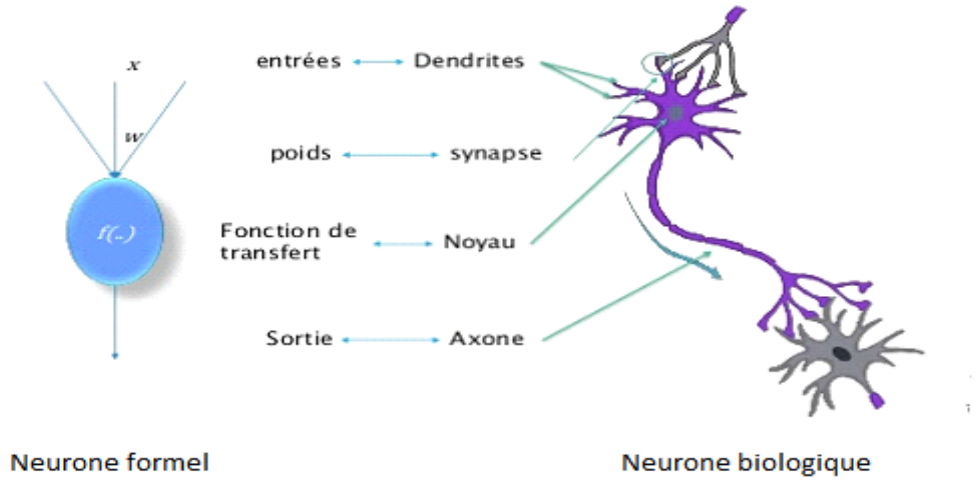


Figure II.2: Les composants d'un neurone formel et biologique. [Werfelli, 2015]

Comme illustré dans la figure II.1 le neurone formel reçoit un ensemble de signaux appelés valeurs d'entrées *x* en provenance des neurones voisins. Ces valeurs sont pondérées chacune avec un poids *w* (référé à weight en anglais) afin de reproduire l'efficacité synaptique reliant deux unités, autrement dit, la force de la relation entre les neurones ; Plus la valeur d'un poids *w* est importante, plus l'intensité du signal entrant est forte, et donc, plus l'entrée correspondante est influente. Une seule sortie est engendrée de ces valeurs d'entrées en passant par la fonction de transfert *f* et cette sortie sera transmise aux neurones adjacents.
Le comportement du corps cellulaire se traduit dans le neurone artificiel en deux phases à l'aide de deux fonctions mathématiques : la fonction de combinaison et la fonction d'activation montrées dans la figure II.2.

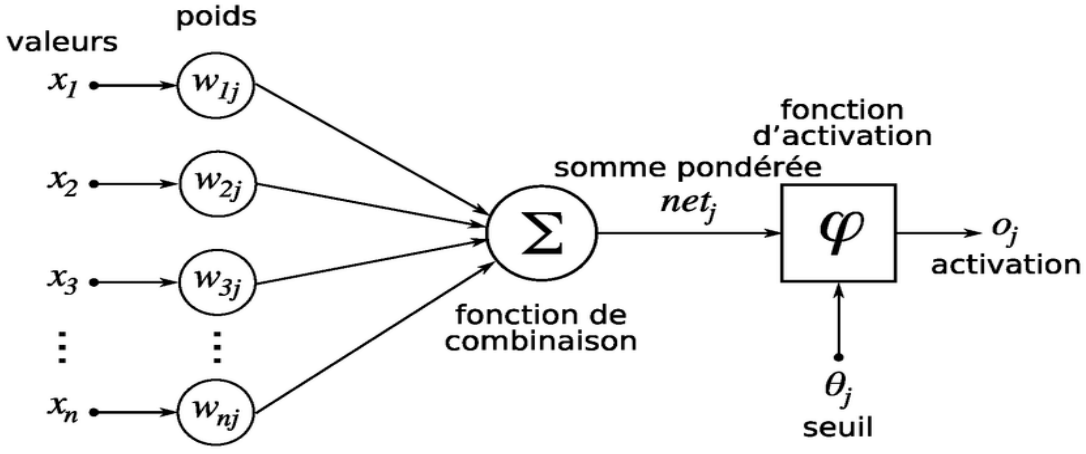


Figure II.3: Structure d'un neurone formel (Perceptron) [Rosenblatt,1957].

- **La fonction de combinaison :** la fonction de la 1ère phase qui correspond au calcul de la somme des poids des entrées qui sera utilisée pour constituer la fonction de la deuxième phase. Cette somme est calculée selon l'équation suivante :

$$Z = \sum_i w_i x_i$$

$\boldsymbol{w_i}$ **:** poids de l'entrée *i.*

$\boldsymbol{x_i}$ **:** signal de l'entrée *i.*

- **Fonction d'activation :** la fonction d'activation est la fonction de transfert qui relie la sommation pondérée au signal de sortie [Msaaf et Belmajdoub, 2015]. Elle effectue une transformation non-linéaire contrôlée par un seuil. Il existe plusieurs formes possibles de la fonction d'activation *F(Z)*, la figure II.3 illustre celles les plus utilisées. Les fonctions de seuillage présentent généralement trois intervalles :
  - En dessous du seuil, le neurone est non-actif (dans ce cas, sa sortie vaut *0* ou *-1*) ;
  - Aux alentours du seuil, une phase de transition ;
  - Au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut *1*).

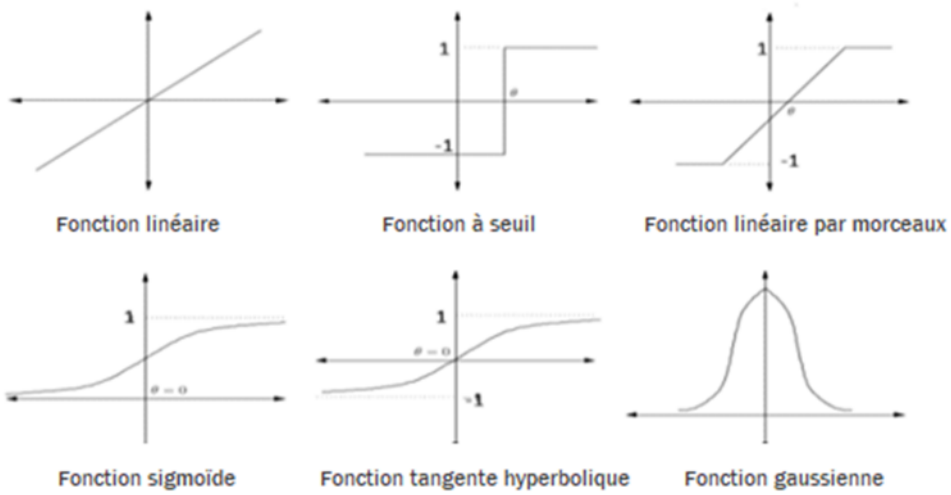


Figure II.4: Fonctions d'activation les plus utilisées. [Msaaf et Belmajdoub, 2015]

La structure d'un réseau de neurones est l'association d'au moins deux couches de neurones formels liées avec le poids : une couche d'entrée et une de sortie, chaque couche avec une tâche précise et des neurones spécialisés ; plus le nombre de couches augmentent plus la capacité de résoudre des problèmes complexes converge et permet de donner un résultat plus précis et fiable grâce à des paramètres d'entrées ajustables. Ces dernières sont les sorties reçues des neurones voisins.
Un RNA peut être monocouche (entrées et sortie) qui est la forme la plus simple du réseau de neurones appelé perceptron (figure II.2), ou doté de plusieurs couches intermédiaires cachées (*Hidden Layers* en anglais) nommé réseaux de neurones multicouches (figure II.4) ; ces deux concepts seront détaillés dans ce qui suit.

II.1.3 Notion de profondeur

Le deep (profond) learning (*DL*) tente de modéliser les problèmes à forte complexité avec un haut niveau d'abstraction par exemple les expressions révélées du visage, en utilisant l'architecture d'un réseau de neurones dit profond qui revient au nombre considérable de dizaines voire de centaines de couches de neurones appelées cachées (*Hidden Layers*) de plus que la couche d'entrée et la couche de sortie. Chaque couche comporte un ensemble de neurones non-connectés entre eux mais connectés avec les neurones de la couche suivante comme le montre la figure II.4.

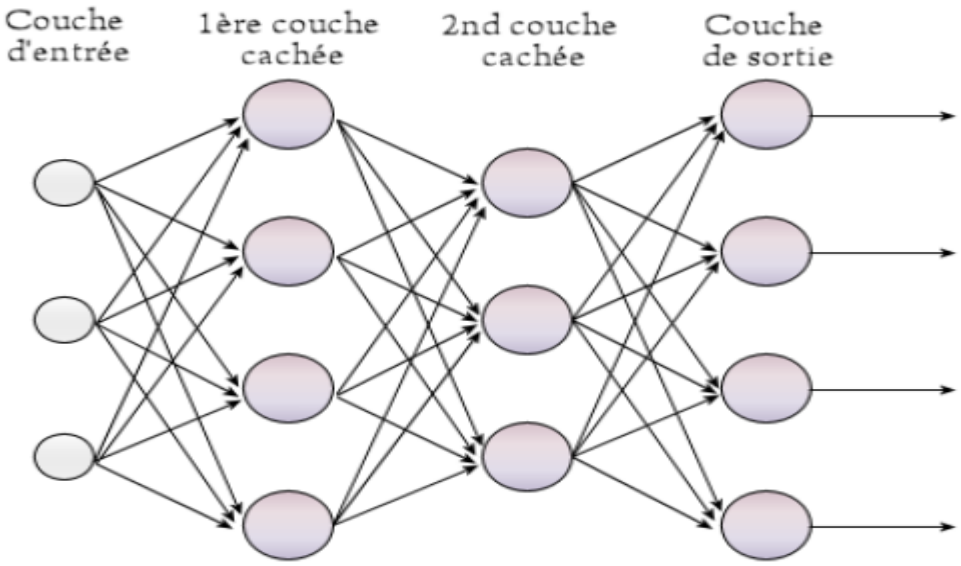


Figure II.5: Réseaux de neurones multicouches (profonds). [Pottiez et Duquesnoy, 2018]

II.2 L'apprentissage des réseaux de neurones

L'apprentissage est le processus le plus remarquable des réseaux de neurones ; permet à un réseau de neurones d'apprendre à partir des exemples d'entrainement fournis de l'extérieur en utilisant des algorithmes d'optimisation qui cherchent à minimiser l'écart entre les réponses d'un réseau de neurones et les réponses souhaitées. L'apprentissage implique la modification itérative des pondérations des connexions entre les neurones avant d'atteindre les résultats désirés. Le réseau donc apprend à chaque itération et garde ses résultats en mémoire pour s'en servir plus tard afin de généraliser pour de nouveaux exemples non vus dans la base d'apprentissage.

Les méthodes d'apprentissage sont distinguées en trois catégories distinctes :

- **Apprentissage supervisé :** (*Supervised Learning*) le réseau de neurones s'entraîne sur un ensemble de données étiquetées (entrées-sorties). Pour chaque entrée une prévision de poids est attribuée pour avoir une sortie, cette dernière est comparée à la sortie initiale à l'aide d'une fonction d'erreur. Un processus de modification de ces poids est itéré afin de minimiser le taux d'erreur jusqu'à produire un résultat suffisamment fiable.
- **Apprentissage non-supervisé :** (*Unsupervised Learning*) dans cette catégorie la base d'apprentissage n'est pas étiquetée (sans les réponses correctes). Le réseau se base sur l'analyse des données et apprend par lui-même les éléments qui peuvent être pertinents. Cet apprentissage s'effectue grâce à la minimisation d'une fonction, appelée fonction de coût [Stricker, 2000], qui indique dans quelle mesure ces prévisions sont proches de l'objectif à atteindre et quel ajustement doit être apporté aux poids.
- **Apprentissage par renforcement :** (*Reinforcement Learning*) appelé aussi apprentissage semi-supervisé. C'est apprendre à agir par essai et erreur. Dans ce paradigme, un agent peut percevoir son état et effectuer des actions. Après chaque action, une récompense numérique est donnée. Le but de l'agent est de maximiser la récompense totale qu'il reçoit au cours du temps [Coulom, 2002] ; autrement dit c'est une approche qui adapte ses paramètres en fonction des réactions reçues de

l'environnement, qui fournit ensuite un retour d'information sur les décisions prises. Par exemple, un système modélisant un joueur d'échecs qui utilise le résultat des étapes précédentes pour améliorer ses performances.

II.3 L'apprentissage des réseaux de neurones profonds

II.3.1 Définition

Le deep learning ou apprentissage profond est un type d'intelligence artificielle dérivé du machine learning (apprentissage automatique), conçu sur la base de réseaux de neurones profonds, visant à imiter la « profondeur » des couches d'un cerveau humain, où il réunit une classe d'algorithmes d'apprentissage correspondants à ces architectures profondes.

II.3.2 Notion de rétro-propagation du gradient

Dans le cas des réseaux de neurones, les poids synaptiques[2] qui contribuent à engendrer une erreur importante se verront modifiés de manière plus significative que les poids qui ont engendré une erreur marginale, qui sont donc corrigés via la rétro-propagation.
La technique de rétro-propagation du gradient est l'algorithme classique de correction des erreurs basé sur le calcul du gradient de l'erreur pour chaque neurone d'un réseau de neurones, de la dernière couche vers la première. L'algorithme est itératif et procède donc par améliorations successives. La manière de calculer cette erreur peut varier selon le type d'apprentissage à effectuer. En appliquant cette étape plusieurs fois, l'erreur tend à diminuer et le réseau offre une meilleure prédiction.
L'algorithme suit ces étapes :

- Soient un échantillon $\vec{x}$ que l'on présente à l'entrée du réseau de neurones et $\vec{t}$ la sortie recherchée pour cet échantillon.
- On propage le signal en avant dans les couches du réseau de neurones: $x_k^{(n-k)} \rightarrow x_j^{(n)}$, avec *n* le numéro de la couche.
  La propagation vers l'avant se calcule à l'aide de la fonction d'activation *g*, de la fonction d'agrégation *h* (souvent un produit scalaire entre les poids et les entrées du neurone) et des poids synaptiques $\vec{w}_{jk}$, entre le neurone et le neurone $x_k^{(n-1)}$. La notation est alors inversée $\vec{w}_{jk}$: indique bien un poids de *k* vers *j*.
  $$x_j^{(n)} = g^{(n)}\left(h_j^{(n)}\right) = g^{(n)}\left(\sum_k w_{jk}^{(n)}\, x_k^{(n-1)}\right)$$
- Lorsque la propagation vers l'avant est terminée, on obtient à la sortie le résultat $\vec{y}$.
- On calcule alors l'erreur entre la sortie donnée par le réseau $\vec{y}$ et le vecteur $\vec{t}$ désiré à la sortie pour cet échantillon. Pour chaque neurone *i* dans la couche de sortie, on calcule (*g'* étant la dérivée de *g*) :
  $$e_i^{sortie} = \acute{g}\left(h_i^{sortie}\right)(y_i - t_i)$$
- On propage l'erreur vers l'arrière $e_i^{(n)} \rightarrow e_j^{(n-1)}$, grâce à la formule suivante :

[2] La synapse désigne une zone de contact fonctionnelle qui s'établit entre deux neurones, ou entre un neurone et une autre cellule.

$$e_j^{(n-1)} = \acute{g}^{(n-1)}\left(h_j^{(n-1)}\right)\left(\sum_i w_{ij}^{(n)} e_i^{(n)}\right)$$

Note : $e_i^{(n)} = e_i^{(sortie)} = (y_i - t_i)\frac{\partial yi}{\partial h_i^{(n)}}$

- On met à jour les poids dans toutes les couches :

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \lambda e_i^l x_j^{(l-1)}$$

Où $\boldsymbol{\lambda}$ **:** représente le taux d'apprentissage (de faible magnitude et compris entre *0,0* et *1,0*).

II.3.3 Quelques réseaux profonds

II.3.3.1 Boltzmann machine (BM)

Le réseau de Boltzmann [Ackley et al., 1985] est un réseau d'unités neuronales symétriquement connectées qui prennent des décisions stochastiques[3] (aléatoires) sur l'activation ou la désactivation (l'activation est liée à une fonction stochastique).

$$P[s_i = 1] = \frac{1}{1 + e^{-\frac{1}{T^t}}}$$

Où : $\boldsymbol{P[s_i = 1]}$ **:** représente la probabilité que l'activation du neurone soit à 1.
$\boldsymbol{T}$ **:** température ($T > = 0$).

$$t_i = \sum_{j \in N} W_{ij}.s_j$$

Les machines Boltzmann ont un algorithme d'apprentissage simple qui leur permet de découvrir les caractéristiques intéressantes dans des ensembles de données composés de vecteurs binaires. L'algorithme d'apprentissage est très lent dans les réseaux avec de nombreuses couches de détecteurs de fonctionnalités, mais il a la particularité d'inclure une couche cachée. Malgré la présence de cette couche le réseau est tout de même capable d'apprendre un comportement désiré.
Les machines Boltzmann sont utilisées pour résoudre deux problèmes :

1. Pour un problème de recherche, les poids sur les connexions sont fixes et sont utilisés pour représenter la fonction de coût d'un problème d'optimisation.
2. Pour un problème d'apprentissage, la machine Boltzmann doit trouver des poids sur les connexions afin que les vecteurs de données soient de bonnes solutions au problème d'optimisation défini par ces poids. Pour résoudre un problème d'apprentissage, les machines Boltzmann effectuent de nombreuses petites mises à jour à leur poids.

[3] Décisions Stochastiques représentent une famille $\{X_t\}_{t \in T}$ de variables aléatoires indexées par le temps *t*.

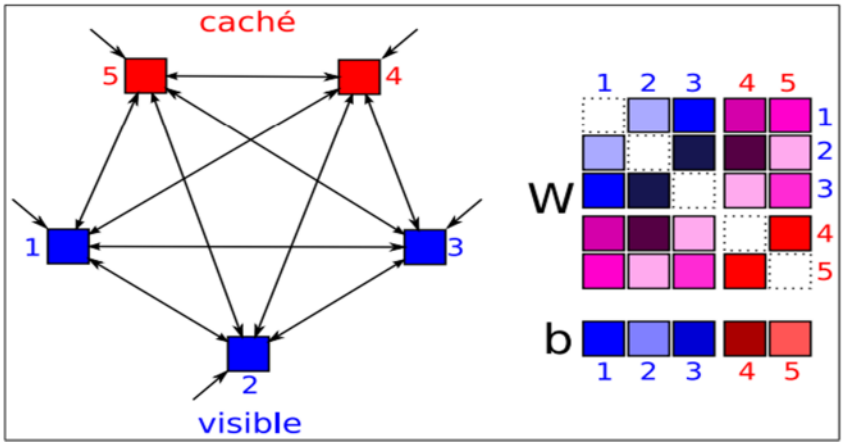


Figure II.6: Visualisation d'une machine de Boltzmann à cinq unités. [Breuleux, 2009]

Chaque unité reçoit également un biais qui lui est propre.
Les paramètres du modèle sont montrés à droite, par exemple, la flèche de l'unité 1 à l'unité 4 est représentée par le poids en (1, 4).
Plus un carré est clair, plus le poids est positif, plus il est foncé, plus le poids est négatif, tandis qu'un carré en pointillé à une valeur nulle. Les paramètres sont colorés en bleu, rouge ou violet dépendamment de s'ils sont entre unités visibles, cachées ou entre unités des deux types.
Notons que W est une matrice symétrique.

II.3.3.2 Restricted boltzmann machine (RBM)

Une machine Boltzmann restreinte (*RBM*) [Smolensky, 1986] est une machine Boltzmann où toutes les connexions sont entre les unités cachées et visibles.
⇒ Une RBM est un champ aléatoire de Markov[4] sur un graphe biparti[5].
⇒Il n'existe aucune connexion entre unités du même groupe (les unités sont séparées en unités visibles v_i et cachées h_j de sorte qu'il n'existe aucune connexion entre unités du même groupe). C'est donc dire que la matrice de poids W est zéro pour toute paire d'indices $\{i, j\}$ pour laquelle s_i et s_j sont tous les deux visibles ou cachés.
Étant donné l'indépendance entre les deux types d'unités, on peut simplifier un peu la représentation en les séparant explicitement et en réduisant la taille de matrice W de sorte qu'elle ne contienne que les poids non-nuls. On se retrouve alors avec des paramètres $\theta = (W, b, c)$ et l'énergie peut être exprimée de la manière suivante :

$$E(v, \quad) = \sum_i c_i v_i \sum_j b_j h_j \sum_{i,j} W_{i,j} v_i h_j$$

Il n'existe aucun poids liant les v ou les h entre eux et on a deux vecteurs de biais b et c qui agissent sur les unités cachées et visibles respectivement. La figure II.6 montre une machine de Boltzmann restreinte.

[4] Un champ aléatoire de Markov est un ensemble de variables aléatoires vérifiant une propriété de Markov relativement à un graphe non orienté. C'est un modèle graphique.
[5] Un graphe est dit biparti si on peut partager son ensemble de sommets en deux parties *A* et *B* tels qu'il n'y ait aucune arête entre éléments de *A* et aucune arête entre éléments de *B*. (Exemple : les arbres sont des graphes bipartis).

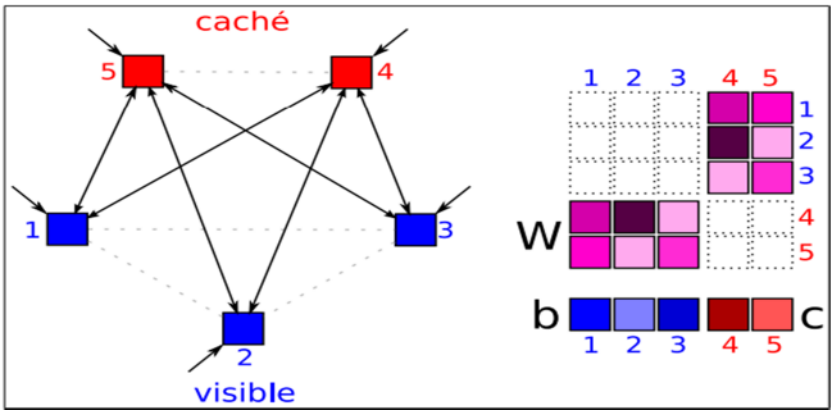


Figure II.7: Visualisation d'une machine de Boltzmann restreinte à cinq unités. [Breuleux, 2009]

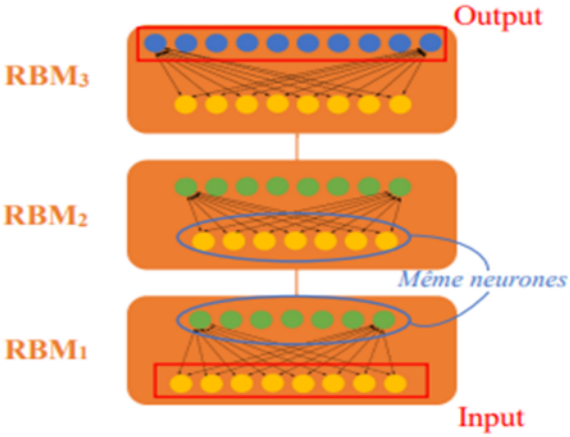
Par rapport à la figure II.6, notons que les poids entre unités visibles et les poids entre unités cachées ont disparus, les biais restent toutefois inchangés. Le léger changement de notation de la BM à la RBM est rendu explicitement ici : W est la matrice des connections entre unités visibles et cachées, b sont les biais des unités visibles seulement et c sont les biais des unités cachées seulement.

II.3.3.3 Deep neural network (DNN)

Les réseaux de neurones profonds (Figure II.4) appelés également le perceptron multicouche (*Multilayer Perceptron, MLP*). Un réseau de neurones est considéré profond s'il contient au moins une couche cachée ; il peut comporter des millions de neurones, organisés en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. Ils sont utilisés en deep learning pour concevoir des mécanismes d'apprentissage supervisé et non supervisé.

II.3.3.4 Deep belief network (DBN)

Comme son nom l'indique, il s'agit d'un réseau de croyance multicouche (profond), DBN se compose de deux types différents de réseaux de neurones : belief networks et machines boltzmann restreintes où la couche cachée de chaque sous-réseau sert de couche visible pour la suivante. Contrairement au perceptron et réseaux neuronaux de rétro-propagation, DBN est un algorithme d'apprentissage non supervisé.
Plusieurs couches de RBM peuvent être empilées les unes sur les autres pour former le réseau multicouche DBN [Varga et al., 2008][Benbakreti, 2019]. Chaque couche RBM utilise les neurones cachés de la couche RBM précédente comme entrée (voir figure II.7).



Dans la RBM1 la couche cachée est la couche avec les neurones verts.

Mais si on se place dans La RBM2 il s'agit de l'input (couche visible), etc.

Figure II.8: Décomposition d'un DBN en plusieurs RBM. [Rainero, 2017]

II.3.3.5 Deep auto-encoder

L'auto-encodeur est une technique de deep learning pour la réduction de la dimensionnalité et la représentation des données où les entrées et les sorties sont identiques. L'encodeur est un ensemble de couches de neurones (perceptron à une ou plusieurs couches cachées), qui traitent les données afin de construire de nouvelles représentations dites 'encodées'. Ensuite, les couches de neurones du décodeur, reçoivent à leurs tours ces représentations et les traitent afin d'essayer de reconstruire les données de départ. Les différences entre les données reconstruites et les données initiales permettent de mesurer l'erreur commise par l'auto-encodeur. L'entraînement consiste à modifier les paramètres de l'auto-encodeur afin de réduire l'erreur de reconstruction mesurée sur les différents exemples du jeu de données. Les entrées et les sorties sont identiques.
L'architecture d'un auto-encodeur est constituée de deux parties : l'encodeur et le décodeur. La figure suivante schématise un auto-encodeur simple, dont l'encodeur (encoder) traite des images (inputs), afin de les représenter comme des points dans un espace à deux dimensions (encoded representation), puis décode cette représentation (decoder), afin de retrouver les données de départ (output).

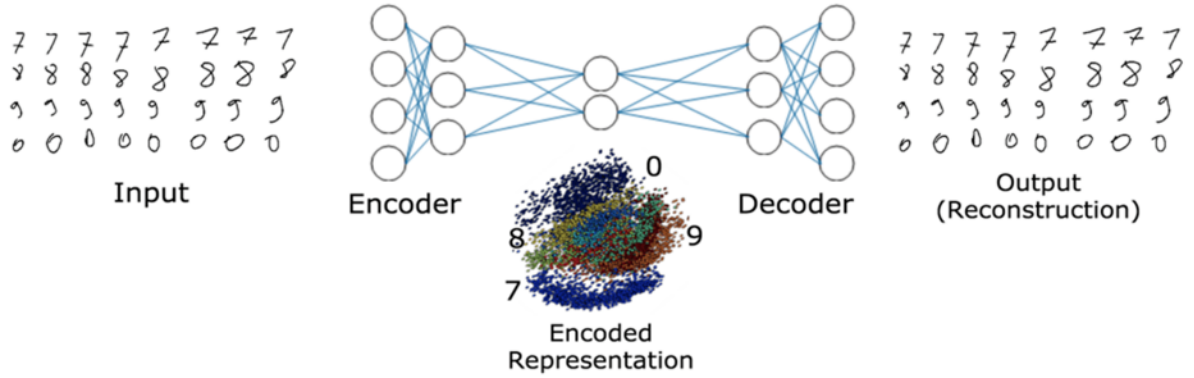


Figure II.9: Un auto-encodeur simple. [Bellin et al., 2016]

II.3.3.6 Deep stacking networks (DSN)

Les deep stacking networks ou réseaux d'empilage sont un type spécial du modèle profond équipé d'un apprentissage parallèle et évolutif. L'architecture du réseau est organisée en blocs (modules) de couches empilés les uns sur les autres, dont chaque bloc est vu comme un sous-réseau de neurones. La Figure II.9 montre un exemple de réseau DSN.
Chaque bloc DSN est un perceptron multicouche simplifié avec une seule couche cachée. Il se compose d'une matrice de poids de couche supérieure U qui relie la couche cachée non linéaire h à la couche de sortie linéaire y, et une matrice de poids de couche inférieure W qui relie la couche d'entrée linéaire et la couche non linéaire cachée.

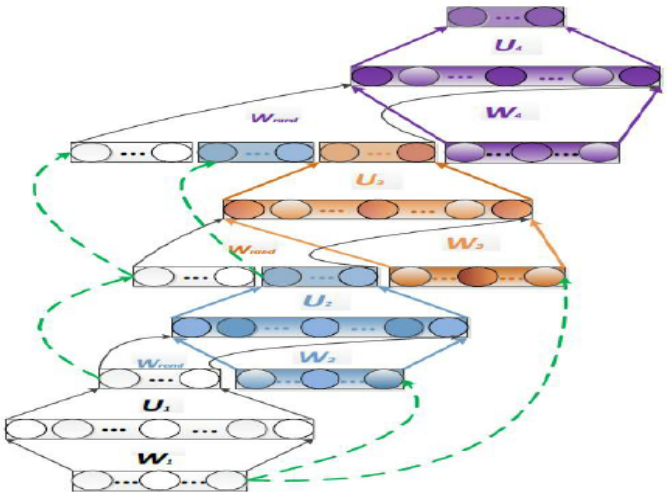


Figure II.10: Architecture d'un DSN contenant 4 modules. [Perwej, 2015]

- Soit le vecteur de données d'entrainement suivant $X = [x_1, .., x_i, .., x_N]$ avec $x_i = [x_{1i}, .., x_{ji}, .., x_{Di}]^T$ où D est la dimension d'un vecteur d'entrée et N la taille de l'ensemble d'entrainement ;
- Soit L le nombre de couches cachées et C la taille du vecteur de sortie ; $Y_i = U^T h_i$ est le vecteur de sortie du DSN avec $h_i = \sigma\,(W^T x_i)$ est le vecteur de couche cachée pour l'échantillon i, supposons que les poids de couche inférieure W sont connus. La fonction σ exécute l'opération sigmoïde logistique élément par élément tel que: $\sigma(x) = (1\ \ \exp(-x))$;
- Soit le vecteur cible $T = [t_1, .., t_i, .., t_N]$ où chaque vecteur est ainsi $t_i = [t_{1i}, .., t_{ji}, .., t_{Ci}]^T$;
- Les matrices U et W dans un bloc DSN peuvent être optimisées à l'aide d'un algorithme de descente de gradient accélérée pour minimiser l'erreur calculée ainsi: $E = \frac{1}{2}\sum_i \|y_i - t_i\|^2 = \frac{1}{2} Tr[(Y - T)(Y - T)^T]$; La sortie est alors définie par: $y_i = U^T \sigma(W^T x_i) = G_i(UW)$;
- Soit H le comportement (l'ensemble des sorties associées à chacune des entrées) du réseau est connu $H = [h_1, .., h_i, .., h_N]$, donc U est donnée par :

$$U\ =\ (HH^T)^{-1} H^T\ T\ =\ F(W)$$

II.3.3.7 Recurrent neural network (RNN)

Un réseau neuronal récurrent est un type de réseau neuronal qui contient des boucles permettant de stocker des informations dans le réseau. En bref, les réseaux de neurones récurrents utilisent leur raisonnement des expériences précédentes pour informer les événements à venir. Les modèles récurrents sont précieux dans leur capacité à séquencer les vecteurs, ce qui ouvre l'API[6] à l'exécution de tâches plus complexes.
Un exemple courant de réseaux neuronaux récurrents est la traduction automatique. Par exemple, un réseau neuronal peut prendre une phrase d'entrée en espagnol et la traduire en

[6] API est l'acronyme d'Application Programming Interface, que l'on traduit en français par interface de programmation applicative ou interface de programmation d'application.

une phrase en anglais. Le réseau détermine la probabilité de chaque mot dans la phrase de sortie sur la base du mot lui-même et de la séquence de sortie précédente.

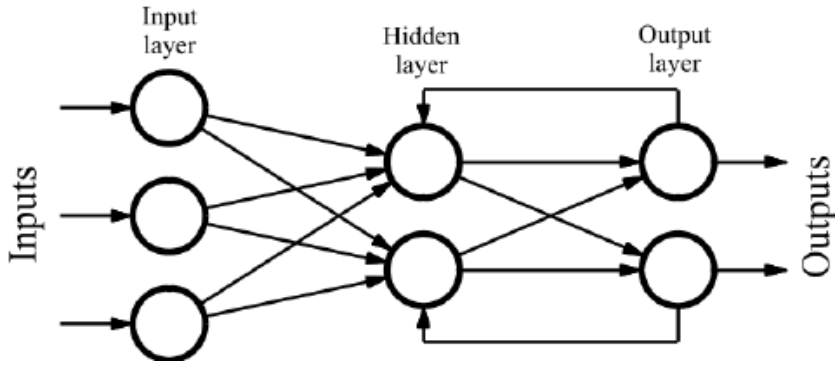


Figure II.11: Un graphe d'un réseau neuronal récurrent. [Quiza et Davim, 2009]

II.3.3.8 Convolutional neural network (CNN)

Un réseau de neurones convolutifs est un des modèles de classification d'images réputés être les plus performant du deep learning. Il permet d'attribuer de façon automatique à chaque image fournie en entrée sous forme d'une matrice de pixels, une étiquette correspondant à sa classe d'appartenance.
L'architecture du CNN possède deux parties :

- **Partie convolutive :** son objectif est d'extraire des caractéristiques propres à chaque image appelées 'features' en appliquant des opérations de filtrage par convolution. Ce processus de filtrage engendre de nouvelles images dites cartes de convultions, en réitérant l'opération on obtient des features dont l'image est réduite par rapport à sa taille initiale. Ces valeurs des dernières features sont concaténées dans un vecteur nommé code CNN.
  Une image se représente en 3 dimensions :
  - Deux dimensions pour une image en niveaux de gris qui correspondent à la largeur et à la hauteur de l'image**.**
  - Une troisième dimension, de profondeur 3 pour représenter les couleurs fondamentales (Rouge, Vert, Bleu).
- **Partie classification :** le vecteur obtenu dans la partie précédente est fourni comme entrée à ce bloc qui est constitué d'un réseau multicouche. Les valeurs du vecteur en entrée sont transformées à l'aide de plusieurs combinaisons linéaires et fonctions d'activation afin de renvoyer un nouveau vecteur en sortie.

Le but du CNN est de résoudre le problème des réseaux de neurones traditionnels qui souffrent du problème de dimension de ses couches [Belkacem, 2016].

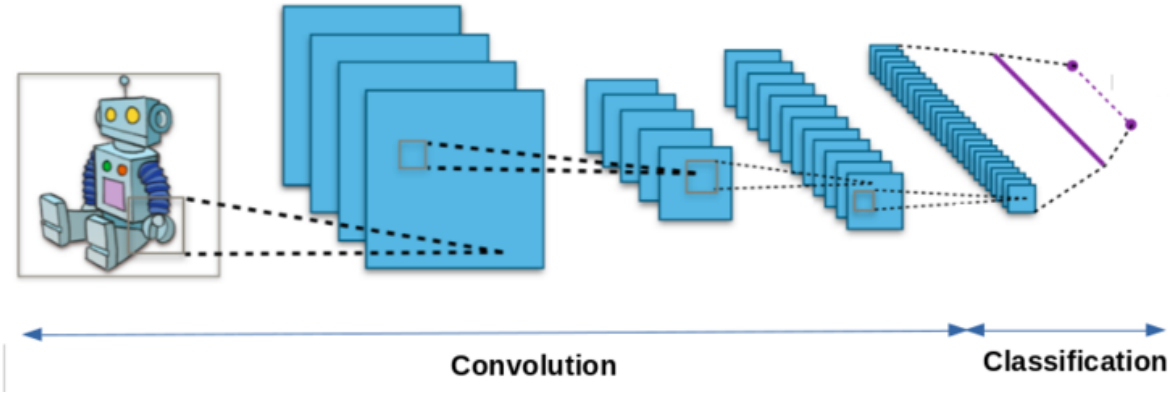


Figure II.12: Architecture d'un réseau de neurones convolutif. [Dejasmin, 2018]

Conclusion

Dans ce chapitre nous avons défini les deux concepts suivants : les réseaux de neurones et les réseaux de neurones profonds tout en mentionnant quelques modèles d'apprentissage de ce dernier, qui servent à résoudre des problèmes en se basant sur l'imitation du fonctionnement du cerveau humain.

Nous nous sommes intéressés dans ce chapitre au deep learning qui ouvre de nouvelles possibilités afin de mieux répondre aux questions, en utilisant des algorithmes d'apprentissage qui permettent d'entrainer un réseau de neurones de manière plus précise à assimiler un traitement d'information, dont le but est de le reproduire sur des données qui lui seront présentées où parfois des données sont cachées ou des relations entre des données sont souvent impossibles à identifier par l'homme dans une large base de documentation comme le cas de la recherche d'information.

Dans le chapitre suivant, nous allons présenter un ensemble de travaux réalisés sur la recherche d'information en utilisant les méthodes d'apprentissage du deep learning.

Chapitre III

Etat de l'art Deep Learning en RI

Introduction

De plus en plus, le deep learning émerge en force dans le domaine de la recherche d'information (RI) ; les approches des réseaux de neurones profonds sont largement exploitées afin d'accomplir de manière plus efficace les tâches du processus de la RI tel que l'amélioration de représentation des textes injectés à l'entrée de ces réseaux en corrigeant ses failles à savoir la représentation lexicale des documents et des requêtes (sac de mots) en procédant avec une représentation sémantique vectorielle. En outre, l'apprentissage de ces réseaux profonds permet d'améliorer les résultats d'appariement et ainsi la qualité des documents pertinents retournés à l'utilisateur.
Dans ce chapitre nous allons citer une liste de travaux basés sur l'utilisation des approches du deep learning dans les divers processus de la recherche d'information.

III.1 Les travaux liés

Les travaux abordés dans cette section sont un ensemble de modèles qui s'appuient sur des réseaux de neurones profonds pour la réalisation des tâches de la recherche d'information telles que la représentation et l'appariement. Dans ce qui suit, nous allons présenter un ensemble de modèles se rapportant à chacune de ces tâches.

III.1.1 Modèles basés sur la représentation

Dans ce type de modèles, les requêtes et les documents sont traités séparément à travers des réseaux de neurones profonds (avec des paramètres partagés) et les scores de similarité sont calculés à partir des représentations latentes des sorties respectives ; puis procèdent à l'appariement. Parmi ces modèles :

III.1.1.1 Deep Learning for Word Embedding [Mikolov et al., 2013a]

L'un des algorithmes les plus connus qui utilisent l'approche du deep learning pour le plongement lexical1 (word embedding) est l'algorithme Word2Vec [Mikolov et al., 2013a], qui vise à compresser un corpus donné en un dictionnaire de vecteurs en créant un espace réduit qui rapproche des mots similaires sémantiquement et syntaxiquement (contexte ou voisinage).
Cet algorithme utilise un réseau de neurones à trois couches : une couche d'entrée, une couche cachée et une couche de sortie. L'algorithme crée d'abord un vocabulaire à partir des données de texte d'apprentissage, puis apprend les représentations vectorielles des mots. L'entraînement d'un réseau de neurones avec word2vec se fait de manière non-supervisée en utilisant deux architectures neuronales possibles :

1. **Continuous Bag of Words (CBOW) :** sac de mots continu en français ; consiste à injecter les termes qui entourent un mot en entrée (le contexte), et de prédire le mot en question en sortie.
2. **Skip-gram :** l'entrainement de ce modèle fonctionne inversement que le précédent, il reçoit en entrée un mot pour prédire son contexte en sortie.

[1] Plongement lexical : est une technique de représentation d'un mot par un vecteur numérique tout en tenant compte du contexte.

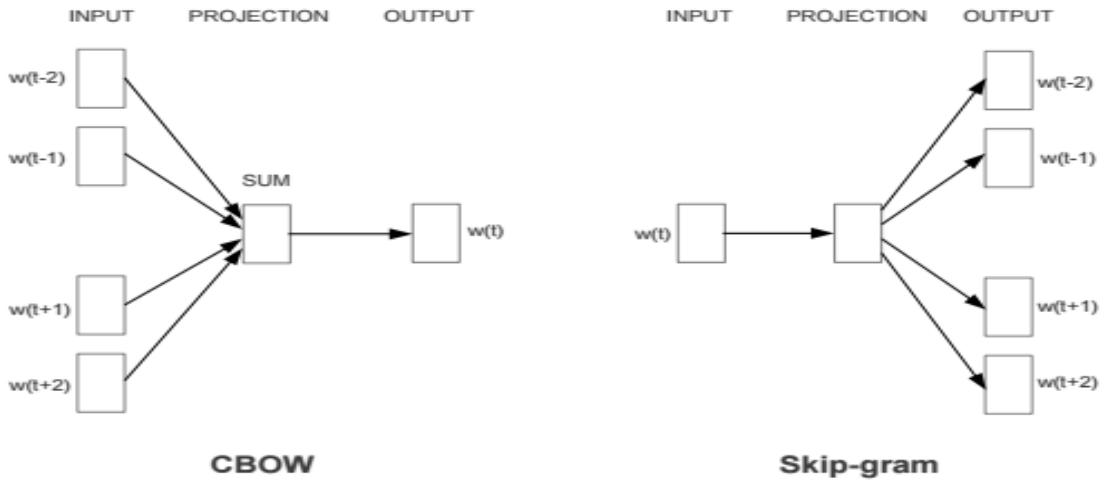


Figure III.1: Architecture de CBOW et Skip-gram. [Mikolov et al., 2013a]

L'entraînement du modèle vise à :

- Maximiser la moyenne des logs attribués aux voisins de mots *k* (en modifiant les poids pour réduire l'erreur de prédiction de l'algorithme) :

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-k}^{k} log\, P\left(w_{t+j}|w_t\right)\right]_{(j\neq 0)}$$

  Où : $\boldsymbol{U_w}$ et $\boldsymbol{V_w}$ sont les vecteurs d'un mot respectivement vecteur d'entrée (input) et vecteur de sortie (output) ;

- Calculer la probabilité reliant chaque mot aux mots du vocabulaire par l'équation suivante :

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

  Où $\boldsymbol{V}$ est la taille du vocabulaire.

La figure III.2 montre un exemple du word embedding qui capture la sémantique avec une représentation vectorielle d'un ensemble de termes selon sept documents :

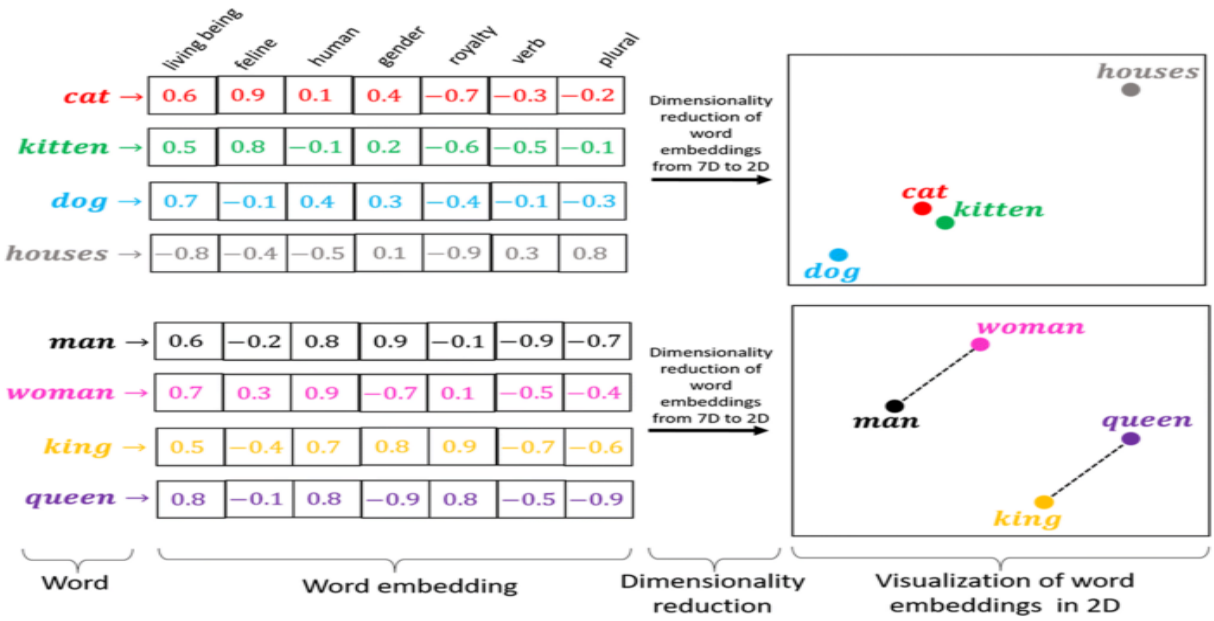


Figure III.2: Exemple du word embedding. [Hariom, 2019]

III.1.1.2 Exploiting Similarities among Languages for Machine Translation [Mikolov et al., 2013b]

Ce modèle se base sur un réseau de neurones récurrent pour apprendre les relations sémantiques entre les mots en utilisant leurs représentations continues (word embeddings). La figure III.3 donne une visualisation simple de vecteurs pour les nombres et les animaux en anglais et en espagnol, qui mène à déduire que ces concepts ont des arrangements géométriques similaires.

Etant donné un ensemble de paires de mots et leur représentation vectorielle associée$\{x_i z_i\}_{i=1}^{n}$, où $x_i \in \mathbb{R}^{d_1}$ est la représentation distribuée du mot i dans la langue source, et $z_i \in \mathbb{R}^{d_2}$ est le vecteur représentation de sa traduction.

En pratique, W peut être appris par le problème d'optimisation suivant :

$$\min w \sum_{i=1}^{n} \left\| w_{x_i} - z_i \right\|^2$$

Pour tout nouveau mot donnée t sa représentation vectorielle continue x, il est mappé à un autre espace linguistique en calculant $z = Wx$; le mot dont la représentation est la plus proche de z dans l'espace de la langue cible est la traduction du mot en entrée, en utilisant similarité cosinus comme métrique de distance.

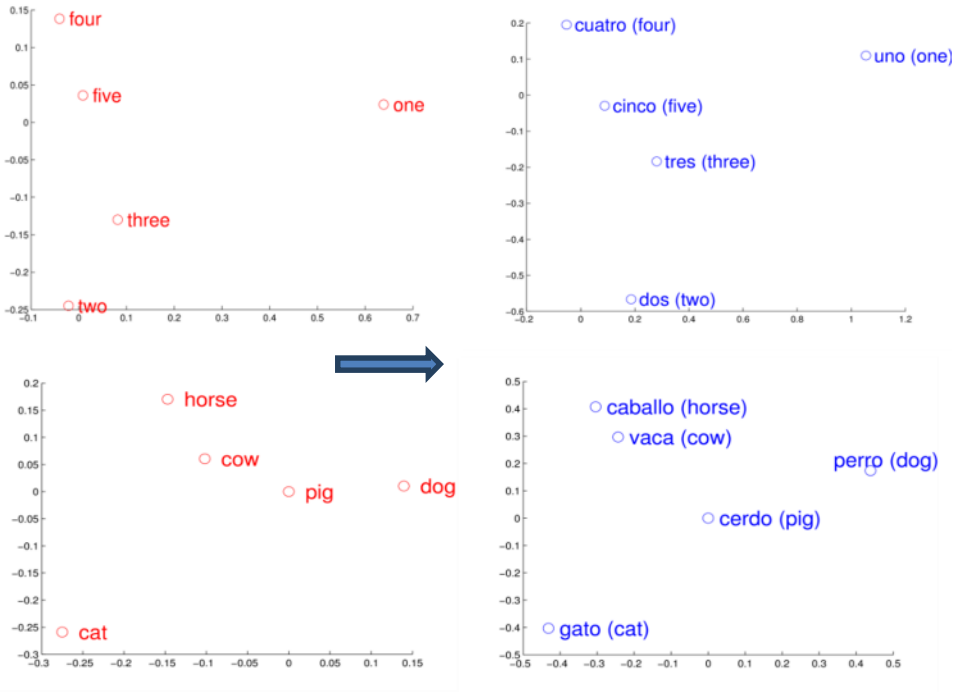


Figure III.3: Représentations de vecteurs de mots distribués de nombres et d'animaux en anglais (à gauche) et en espagnol (à droite). [Mikolov et al., 2013b]

III.1.1.3 Learning Deep Structured Semantic Models (DSSM) for Web Search Using Clickthrough Data [Huang et al., 2013]

Deep semantic structured model (DSSM) est un modèle connu pour être un des modèles les plus efficaces dans la tâche de recherche sur le web [Nguyen et al., 2017], utilise un réseau neuronal profond (DNN) dont l'entrée est un ensemble de vecteurs de termes *x* et la sortie est un vecteur de concept *y*, afin de classer un ensemble de documents pour une requête donnée comme suit :

- Une projection non-linéaire est effectuée sur la requête et les documents pour les représenter dans un espace sémantique en commun, en utilisant une méthode dans la première couche cachée du DNN appelée word hashing qui vise à réduire la dimensionnalité des vecteurs de termes de sac de mots en les représentant avec des lettres de trigramme, dont le but de résoudre la contrainte de la taille du vocabulaire très grande dans la recherche web ; par exemple :
  Le mot « good » va être représenté avec des lettres trigrammes de la manière suivante :
  - Ajouter la marque '# ' pour déterminer le début et la fin du mot : « #good# » ;
  - Diviser le mot sous des ensembles de 3 lettres afin d'obtenir le vecteur du terme : « #go, goo, ood, od# ».
- Pour des couches cachées $l_i, i = 1, \dots, N-1$, une matrice de poids Wi et un biais bi :

$$l_1 = W_1 x$$
$$l_1 = f(W_i\, l_{i-1}\, b_i)\; W_1 x, i = 2, \dots, N-1$$
$$y = f(W_N\, l_{N-1} + b_N)$$

Où : f est la fonction d'activation définie comme suit :

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- La pertinence de chaque document donné pour une requête est calculée comme la similitude cosinus entre leurs vecteurs dans cet espace sémantique.

$$R(Q, D) = \cos(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\| \|y_D\|}$$

Où : $\boldsymbol{y_Q}$ et $\boldsymbol{y_D}$ sont les vecteurs conceptuels de la requête et du document, respectivement.

L'entrainement de ce modèle consiste à apprendre ses paramètres : les matrices de poids W_i et les vecteurs de biais b_i dans le réseau neuronal comme partie essentielle du DSSM, de manière à :

- À maximiser la probabilité conditionnelle des documents cliqués pour une requête :

$$P(D|Q) = \frac{\exp(\gamma R(Q, D))}{\sum_{D' \epsilon D} \exp(\gamma R(Q, D'))}$$

Où : ***D*** **:** indique l'ensemble des documents candidats à classer, (D est l'ensemble des documents cliqués D+ en incluant quatre documents sélectionnés aléatoirement sans clic).

$\boldsymbol{\gamma}$ **:** est un facteur de lissage.

R(Q, D) : est le score de pertinence sémantique entre une requête Q et un document D.

- À minimiser la fonction d'erreur :

$$L(\Lambda) = -\log \prod_{(Q, D^+)} P(D^+|Q)$$

Où : $\boldsymbol{\Lambda}$ **:** désigne l'ensemble des paramètres du réseau de neurone {Wi, bi}.

À chaque itération, les poids W_i et les vecteurs b_i sont modifiés en utilisant les algorithmes d'optimisation numérique du gradient.

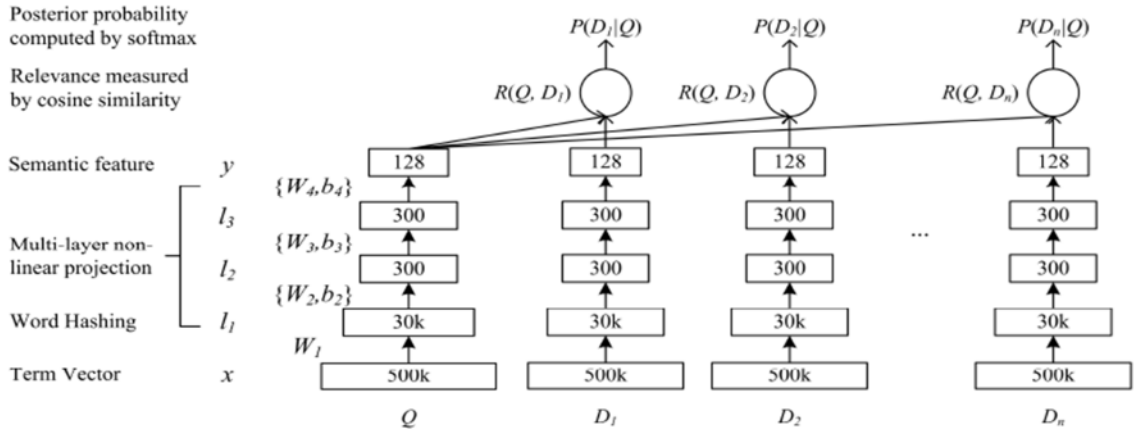


Figure III.4: Illustration du modèle DSSM utilisant un DNN. [Huang et al., 2013]

III.1.1.4 Convolutional Neural Network for Modelling Sentences [Kalchbrenner et al., 2014]

Ce travail a été proposé dans le but de développer une architecture d'un réseau de neurones convolutifs basée sur *Dynamic Convolutional Neural Network* (DCNN), pour la modélisation sémantique des phrases, en utilisant :

- **Dynamic k-Max Pooling :** c'est une k- opération de pooling max où k est la longueur de la phrase et de la profondeur du réseau. Le paramètre de regroupement est comme suit :

$$k_l = \max(k_{top} \left\lceil \frac{L-l}{L} s \right\rceil$$

Où : $\boldsymbol{l}$ **:** est le numéro de la couche convolutive courante à laquelle le regroupement est appliqué .

$\boldsymbol{L}$ **:** est le nombre total de couches convolutives dans le réseau.

$\boldsymbol{k_{top}}$ **:** est le paramètre de regroupement fixé pour la couche convolutive la plus élevée.

$\boldsymbol{s}$: est la longueur de la phrase en entrée.

- **Non-linear Feature Function :** appliquer sur le résulat (la matrice groupée) de dynamic k-max pooling un biais b unique pour chaque ligne et une fonction non linéaire g.

$$M = [diag(m{:},1), \ldots, diag(m{:},m)]$$

Où $\boldsymbol{m}$ **:** sont les poids des d filtres de la convolution.

Ensuite, après la première paire d'une couche convolutive et non linéaire, chaque colonne α dans la matrice α s'obtient comme suit, pour certains indices j.

$$a = g\left(M \begin{bmatrix} j \\ . \\ . \\ . \\ w_{j+m-1} \end{bmatrix} + b\right)$$

Où : $\boldsymbol{\alpha}$ **:** est une colonne de caractéristiques de premier ordre. Les caractéristiques du second ordre sont obtenues de la même manière en appliquant l'équation précédente à une séquence de caractéristiques de premier ordre.

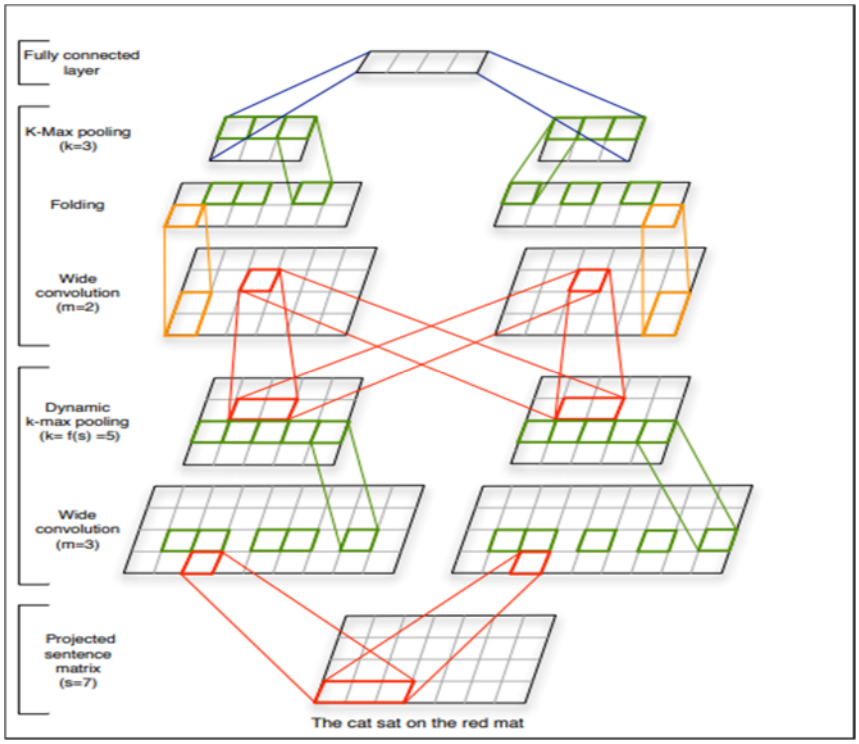


Figure III.5: Architecture d'un DCNN. [Kalchbrenner et al., 2014]

III.1.1.5 Neuronal Information Search Model Augmented by a Semantic Resource [Nguyen et al., 2017]

C'est un modèle neuronal pour la RI ad-hoc qui combine la sémantique relationnelle[2] dérivée des ressources externes et la sémantique distributionnelle[3] inférée à travers le corpus. Le réseau de neurones de ce modèle vise à projeter cette représentation combinée initiale du document ou de la requête sur un espace latent pour apprendre simultanément leurs représentations sémantiques latentes dans le but d'estimer leur pertinence.

Pour chaque branche du réseau, un vecteur d'entrée x_{input} du texte T est projeté dans un espace latent à l'aide de L couches cachées l_i $(i = 1, \dots, L)$ afin d'obtenir un vecteur sémantique latent y. Chaque couche cachée l_i et vecteur sémantique latent y sont respectivement obtenus par les transformations non linéaires suivantes :

$$l_i = x_{input}$$
$$l_i = f(W_{i-1} \cdot l_{i-1} + b_{i-1}) \; ; \qquad i = 1, \dots, L$$
$$y = f(W_L \cdot l_L + b_L)$$

Où : $\boldsymbol{W_i}$ et $\boldsymbol{b_i}$ sont respectivement la matrice de poids et le biais de la $i^{ème}$ couche.

$\boldsymbol{f(x)}$ **:** fonction d'activation ReLU (*Unité de Rectification Linéaire*).

[2] Sémantique relationnelle : la représentation sémantique latente des documents et des requêtes en bénéficiant des concepts et des relations (ex : synonymie).

[3] Sémantique distributionnelle : la représentation latente d'un mot par un vecteur appelé word embedding qui est capable de capturer la sémantique contextuelle des mots.

Après avoir obtenu les vecteurs sémantiques latents y_D et y_Q du document D et de la requête Q respectivement, par la transformation non linéaire des couches cachées, un score de similarité cosinus $R(D|Q)$ entre les vecteurs document et de requête est calculé.
Optimiser les paramètres du réseau de neurones en utilisant un coût d'ordonnancement relatif, basée sur la distance de similarité Δ :

$$\Delta = \sum_{p=1}^{n} \left[sim(Q, D^+) - sim\left(Q, D_p^-\right) \right]$$

Où : $(\boldsymbol{Q}, \boldsymbol{D^+})$: paires de document-requête pertinentes.
$\left(\boldsymbol{Q}, \boldsymbol{D_p^-}\right)$: paires de document-requête non pertinentes.
$\boldsymbol{sim}(.,.)$: est la sortie du réseau de neurones.
Ensuite, le réseau est entraîné pour maximiser la distance de similarité Δ entre les paires de document-requête pertinentes et non pertinents ; en utilisant la fonction de coût "en coude" (*hingeloss*) L :

$$L = max(0, \alpha - \Delta)$$

Où : $\boldsymbol{\alpha}$: est la marge de L, selon l'amplitude de Δ.

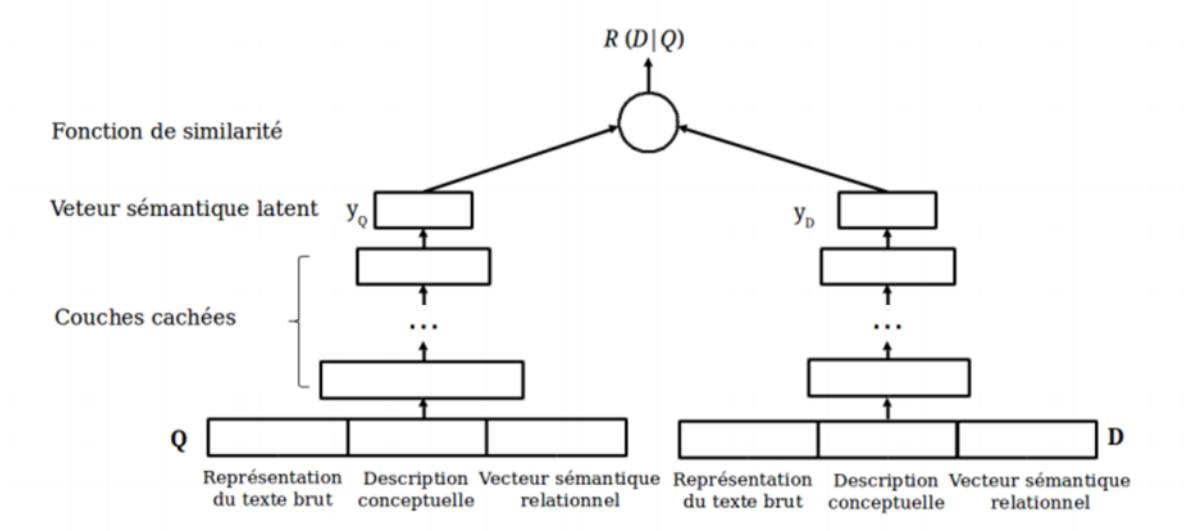


Figure III.6: Architecture du modèle neuronal de recherche d'information augmenté par une ressource sémantique. [Nguyen et al., 2017]

III.1.2 Modèles basés sur l'appariement

Ces modèles consistent en une approche dans laquelle une représentation commune pour les requêtes et les documents est construite tout en prenant en compte des interactions entres les entités dans les deux textes, puis un réseau de neurones profond est utilisé pour apprendre un appariement. Parmi ces modèles :

III.1.2.1 Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks (CDNN) [Severyn et Moschitti, 2015]

L'objectif de ce modèle basé sur des réseaux de neurones convolutifs est de générer une représentation intermédiaire optimale des paires document-requête de textes courts (phrases), et une fonction de similarité pour les relier de manière supervisée à partir des données d'apprentissage disponibles. L'architecture de ce modèle utilise pour calculer la similarité une approche qui apprend à extraire et à composer des n-grammes de degrés plus élevés, permettant ainsi de capturer des dépendances à plus longue portée.
Ce modèle d'apprentissage en profondeur vise à :

- Extraire les séquences de mots discriminatoires trouvées dans les phrases d'entrée qui sont communes dans les instances d'apprentissage en utilisant une matrice de filtres, puis les combiner aux biais b_i et une fonction d'activation.
- Produire comme sortie un vecteur C_{pooled} pour le document et la requête.
- Calculer la similarité entre les deux vecteurs pooled de la requête Xq et du document X_d avec la matrice de similarité M :

$$sim(X_q, X_d) = X_q^T M X_d$$

- Rajouter des features de similarité en plus pour construire un vecteur x_{join} .
- Calculer des interactions entre les différents éléments du vecteur
- Calculer des distributions de probabilités sur les différentes sorties obtenues avec softmax.

Ce modèle est entraîné de bout en bout, sert à minimiser l'entropie croisée de la fonction de coût en prenant en compte les différents paramètres (d'entrainement) à apprendre et qui sont optimisés en utilisant l'algorithme de rétro-propagation.

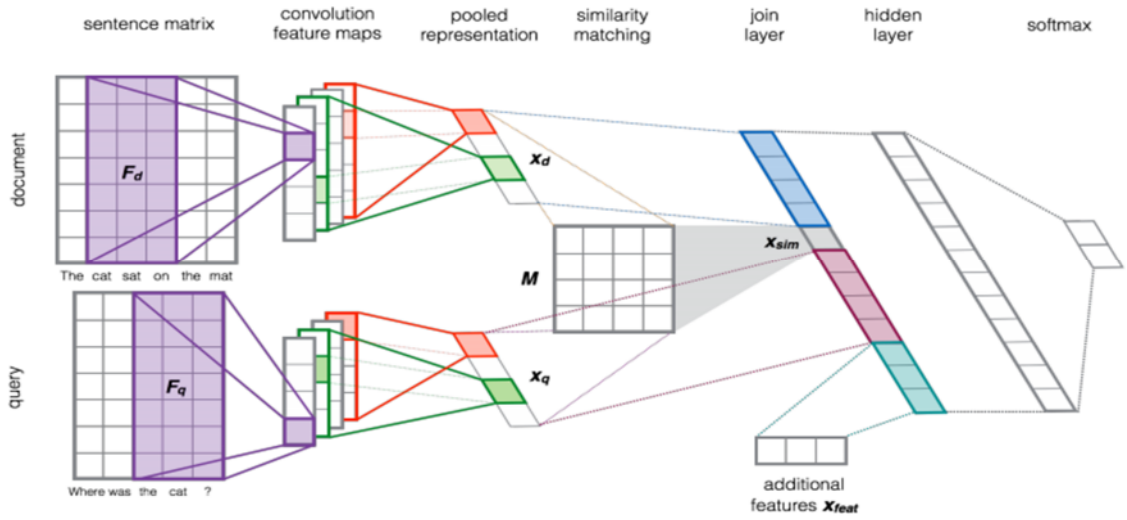


Figure III.7: L'architecture du CDNN pour le classement de paires de textes courts. [Severyn et Moschitti, 2015]

III.1.2.2 Ranking Short Answer Texts with Attention-Based Neural Matching Model (aNMM) [Yang et al., 2016]

C'est un modèle qui consiste à combiner différents signaux d'appariement et incorporer l'apprentissage de l'importance des termes de requête. Le modèle aNMM propose de construire une matrice de correspondance sémantique *QA* qui indique la force du signal d'appariement entre les paires document-requête pré-entrainée avec du word embedding, ensuite incorporer le schéma d'attention aux termes des requêtes en utilisant la fonction softmax afin de discriminer explicitement l'importance des termes de requête dans la même requête.

Ce modèle propose deux architectures :

1. **L'architecture aNMM-1 :** c'est l'architecture de base, dont la formule softmax est utilisée dans le processus de prédiction de propagation :

$$y = \sum_{j=1}^{M} g_j \cdot h_j = \sum_{j=1}^{M} \frac{exp(v \cdot q_j)}{\sum_{l=1}^{L} exp(v \cdot q_l)} \cdot h_j$$

Où $\boldsymbol{v}$ **:** indique un vecteur dimensionnel *P* qui est un paramètre de modèle.

$\boldsymbol{M}$ **:** la longueur de la requête.

$\boldsymbol{v \cdot q_j}$ *:* le produit scalaire du vecteur de mots de requête normalisé q_j à v (pour représenter l'importance du terme de requête).

L'apprentissage des paramètres du modèle aNMM-1 à partir des données d'entraînement est un apprentissage par paires qui vise à minimiser la descente du gradient stochastique suivante :

$$e\,(q, a^+, a^-; w, v) = max\,\big(0{,}1 - S\,(q, a^+) + S\,(q, a^-)\big)$$

Où : $\boldsymbol{S\,(q, a)}$ **:** désigne le score de correspondance prévu pour la paire *(q, a).*

2. **L'architecture aNMM-2 :** ce modèle est une extension du aNMM-1, il constitue une couche cachée de plus qui permet d'apprendre plusieurs vecteurs de poids comme *w.* Ainsi *w* devient une matrice bidimensionnelle.

La formule de prédiction de propagation vers l'avant est comme suit :

$$y = \sum_{j=1}^{M} \tau\big(v \cdot q_j\big) \cdot \delta\left(\sum_{k=0}^{K} r_t \cdot \delta\left(\sum_{t=0}^{T} w_{kt} \cdot x_{jk}\right)\right)$$

Où : $\boldsymbol{\tau\big(v \cdot q_j\big)} = \frac{exp(v \cdot q_j)}{\sum_{l=1}^{L} exp(v \cdot q_l)}$ et $\boldsymbol{\tau}$ est la fonction de porte softmax.

$\boldsymbol{T}$ **:** est le nombre de nœuds dans la couche cachée 1.

$\boldsymbol{r_t}$ **:** est le paramètre du modèle.

Le processus de rétro-propagation de ce modèle utilise trois paramètres de modèle différents d'une couche à une autre (w_{kt}, r_t et v_p) fournis aléatoirement au départ, ensuite, ils sont mis à jour avec l'algorithme de rétro-propagation. Lorsque le processus d'apprentissage converge, les paramètres du modèle appris pour la prédiction sont utilisés pour classer les textes de réponses courtes.

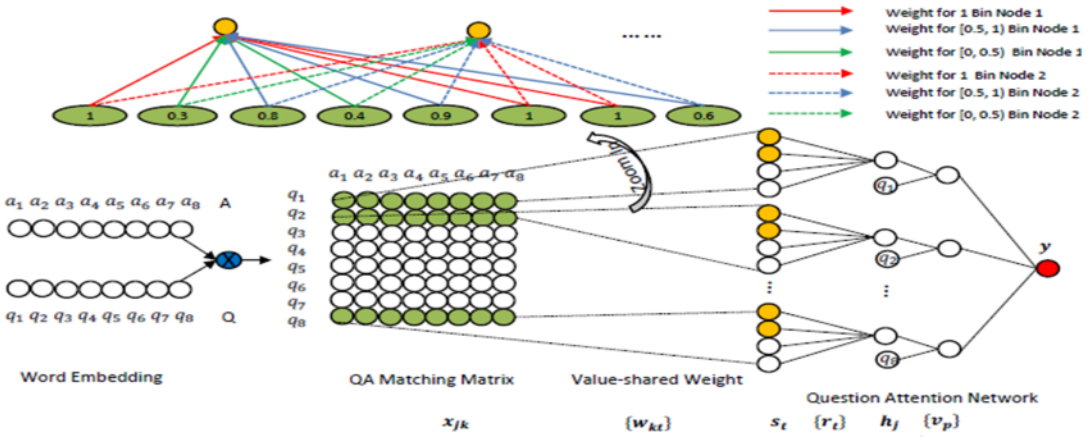


Figure III.8: L'architecture du modèle d'appariement neuronal basé sur l'attention (aNMM-2). [Yang et al., 2016]

III.1.2.3 MatchPyramid Models on Ad-hoc Retrieval [Pang et al., 2016]

L'objectif de ce modèle est de développer une architecture basée sur un réseau neuronal convolutif (CNN) pour apprendre des modèles d'appariement hiérarchiques. Le fonctionnement de ce modèle est comme suit :

- Construire une matrice de similarité à deux dimensions :

$$M_{ij} = w_i \otimes v_j$$

  Où : $\boldsymbol{M_{ij}}$ **:** est la similarité entre le $i^{ème}$ mot w_i dans le premier texte et le $j^{ème}$ mot v_j dans le deuxième texte.

  $\otimes$ **:** est une fonction de similarité.
- Extraire les modèles de correspondance de la matrice de similarité.
- Agréger les informations en un seul score de correspondance.

Dans la phase d'entrainement, ce modèle utilise la perte de classement par paires. étant donné un triple (q, d^+, d^-)) où le score de (q, d^+) est supérieur à celui de (q, d^-). La fonction de perte est définie comme suit :

$$L(q, d^+, d^-; \Theta) = \max\left(0, 1 - S(q, d^+) + S(q, d^-)\right)$$

Où : $\boldsymbol{S\,(q, d)}$ **:** désigne le score d'appariement prévu pour (q, d), et $\boldsymbol{\Theta}$ comprend les paramètres du réseau. L'optimisation est relativement simple avec la rétro-propagation standard.

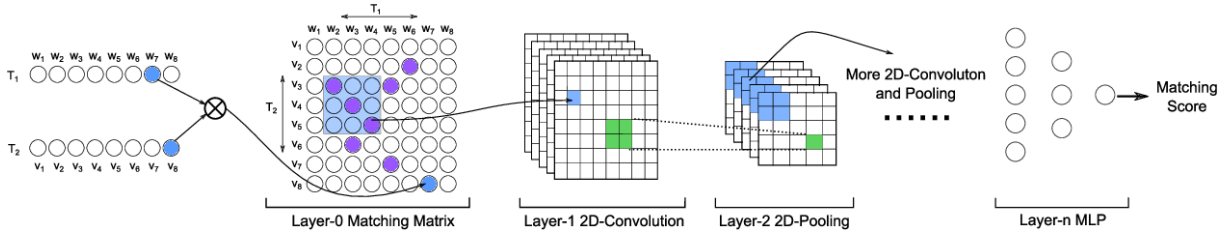


Figure III.9: Structure du modèle de MatchPyramid. [Pang et al., 2016]

III.1.2.4 Learning Semantic Representation with Deep Learning [Belkacem, 2016]

Ce travail propose une architecture à trois couches principales, son objectif est d'apprendre la pertinence des résultats pouvant être retournés pour une requête donnée, par apprentissage d'une matrice de similarité entre les deux vecteurs de la paire de comparaison.

- La couche représentation : sert à construire les vecteurs sémantiques des documents et requêtes en utilisant l'outil word2vec.
- La couche pertinence : sert à apprendre la pertinence des résultats dans une matrice de similarité *M* et minimiser la valeur de colinéarité document-requête (sémantiquement similaire). Son fonctionnement suit les étapes suivantes :

  $$Sim_c(X_q, X_d) = exp\left(-col(X_q, X_d)\right) = X_c$$

  Où : La valeur de colinéarité : $col(X_q, X_d) = \sum_{i=1,n}\sum_{j=1,n}(q_i d_j - q_j d_i)$

  $\boldsymbol{X_q} = q_i; i = 1, n$: le vecteur sémantique de la requête.

  $\boldsymbol{X_d} = d_i; i = 1, n$: le vecteur sémantique d'un document.
- La couche similarité : constituée d'un réseau de neurones à deux couches cachées et une sortie unique ; elle combine les différentes caractéristiques reliant la requête *Q* aux documents D_i, dans le but de calculer le score de similarité entre les entrées (la sortie finale *S* du réseau).

L'entrainement de ce modèle sert maximiser l'erreur *E* pour apprendre la pertinence des documents résultats associés à une requête, en se basant sur la liaison entre la requête et un résultat pertinent associé dans la matrice de similarité en apprenant les paramètres du modèle.

$$E(\theta) = \sum_{q \epsilon Q} S \log\left(\cos\left(X_q, X_d\right)\right)$$

Avec $\theta = \{W, b\}$ sont les paramètres d'entrainement et qui sont mis à jour de sorte à maximiser l'erreur *E*.

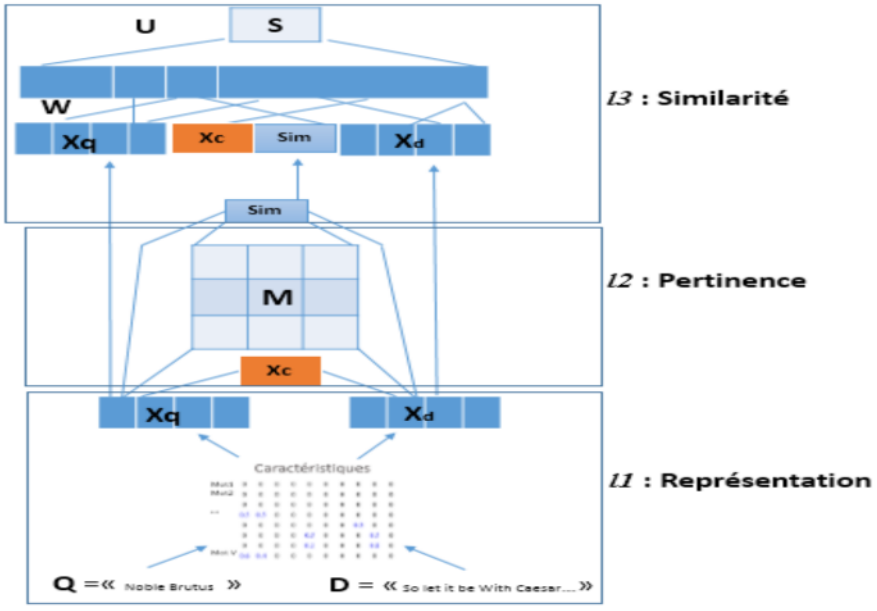


Figure III.10: Architecture du modèle de similarité par deep learning et word2vec. [Belkacem, 2016]

III.1.2.5 Neural Ranking Model with Weak Supervision [Dehghani et al., 2017]

L'objectif de cette approche est de classer et noter les documents pour chaque requête dans l'ensemble des requêtes d'entraînement avec des signaux générés à l'aide d'une approche d'apprentissage (pseudo-étiquetage).
Un simple réseau neuronal feed-forward est opté pour cette démarche ; composé de :

- Couche d'entrée z_0 : encode la requête d'entrée et le document *s* dans un vecteur d'une longueur fixe.
- $l-1$ couches cachées : chaque couche z_j est une couche entièrement connectée qui calcule la transformation suivante :

  $$z_i = \alpha(W_i \cdot z_{i-1} + b_i); 1 < i < l-1$$

  Où : $\boldsymbol{W_i}$ et $\boldsymbol{b_i}$ **:** désignent respectivement la matrice de poids et le terme de biais correspondant à la $i^{ème}$ couche cachée ;

  $\boldsymbol{\alpha}(.)$ **:** est la fonction d'activation.
- Couche de sortie z_l : est une couche entièrement connectée avec une seule sortie continue. La fonction d'activation de la sortie dépend de l'architecture de classement utilisée montrée dans la figure III.11.

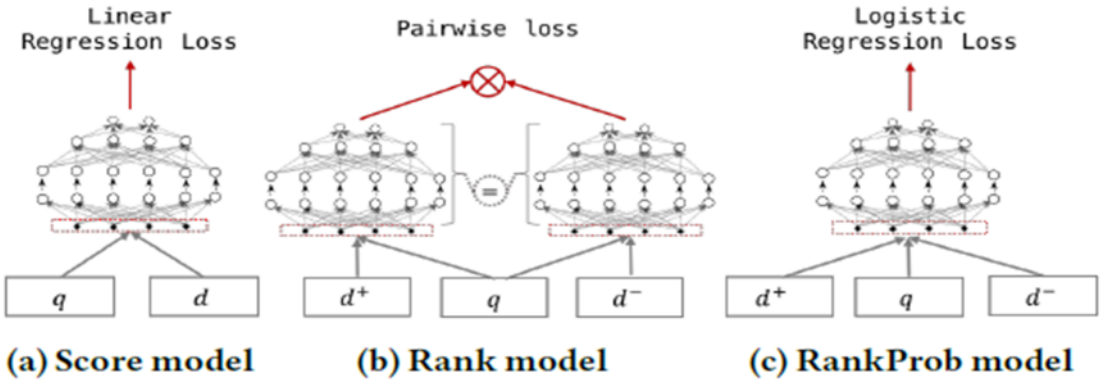


Figure III.12: Architecture de 3 modèles de classement. [Dehghani et al., 2017]

III.1.2.6 Modeling Label Ambiguity for Neural List-Wise Learning to Rank [Jagerman et al., 2017]

L'objectif de ce modèle est de résoudre l'ambiguïté des étiquettes de pertinence en utilisant un réseau de neurones profond à trois couches ; pour trouver une fonction *F* qui attribue des scores élevés aux documents qui ont une pertinence élevée et des scores faibles aux documents qui ont peu de pertinence.
L'idée de ce modèle est comme suit :

- Échantillonner une permutation π avec une probabilité de distribution *Plackett-Luce*[4] *(PL)* : $PL(\pi|\mathbb{D};\psi\mathbb{Y})$

  Où : $\boldsymbol{\pi}$ **:** est une permutation.

  $\mathbb{D}$ **:** l'ensemble des documents retournés.

[4] Distribution *Plackett-Luce (PL) :* une distribution de probabilité qui attribue une probabilité élevée aux permutations qui placent les éléments à score élevé en haut, tout en attribuant une faible probabilité aux permutations qui placent les éléments à score élevé en bas.

$\boldsymbol{\psi}\mathbb{Y}$ **:** un vecteur de score qui conserve l'ordre de pertinence des étiquettes.

- Utiliser cet échantillon pour calculer une perte stochastique à l'aide d'une méthode appelée *ListPL* :

$$\mathcal{L}(f(\mathbb{D}), \mathbb{Y}) = -\log\big(PL(\pi|\mathbb{D}; f)\big)$$
$$\pi \sim PL(\pi|\mathbb{D}; \psi\mathbb{Y})$$

Où : $\boldsymbol{PL(\pi|\mathbb{D}; f)}$ **:** distribution PL de la sortie du réseau *f*.

- Calculer la dérivation du gradient stochastique de cette fonction de perte (par rapport à la fonction d'activation) :

$$\sum_{i=k}^{n}\left(\frac{exp\big(f(d_{\pi k})\big)}{\sum_{j=1}^{n} exp\left(f\big(d_{\pi j}\big)\right)}\right) - 1$$

La perte résultante et le gradient correspondant peuvent ensuite être utilisés pour entraîner le réseau neuronal via la descente de gradient stochastique (SGD).

III.1.2.7 Situational Context for Ranking in Personal Search [Zamani et al., 2017]

Les auteurs de ce modèle proposent deux modèles d'appariement sémantique contextuels basés sur des réseaux de neurones profonds (DNN) en ajoutant le contexte situationnel, c'est-à-dire les caractéristiques contextuelles de la demande de recherche actuelle qui sont indépendantes du contenu de la requête et de l'historique de l'utilisateur ; afin d'améliorer la qualité de la recherche personnelle. Ces modèles prennent comme entrées : le contenu de la requête, le contexte situationnel (par exemple : l'heure et l'emplacement) et le contenu du document, et en sortie : un score de prédiction de la pertinence d'un document pour la requête donnée est calculé comme suit :

$$score\ (q, d) = f\big(\phi(q), \phi(d)\big)$$

Où : $\boldsymbol{\phi}$ **:** désigne une fonction qui mappe un texte sur une représentation vectorielle et $\boldsymbol{f}$ **:** est une fonction de correspondance qui calcule la similitude de deux représentations calculées.

Les couches cachées sont entièrement connectées et le $i^{ème}$ nœud de chaque couche est calculé comme suit :

$$h_i = \sigma(W_i^T \cdot X + b_i)$$

Où : $\boldsymbol{W_i}$ et $\boldsymbol{b_i}$ désignent respectivement la matrice de poids et la valeur du biais ;

$\boldsymbol{X}$ **:** est le vecteur de sortie de la précédente couche ;

$\boldsymbol{\sigma}$ **:** est la fonction d'activation.

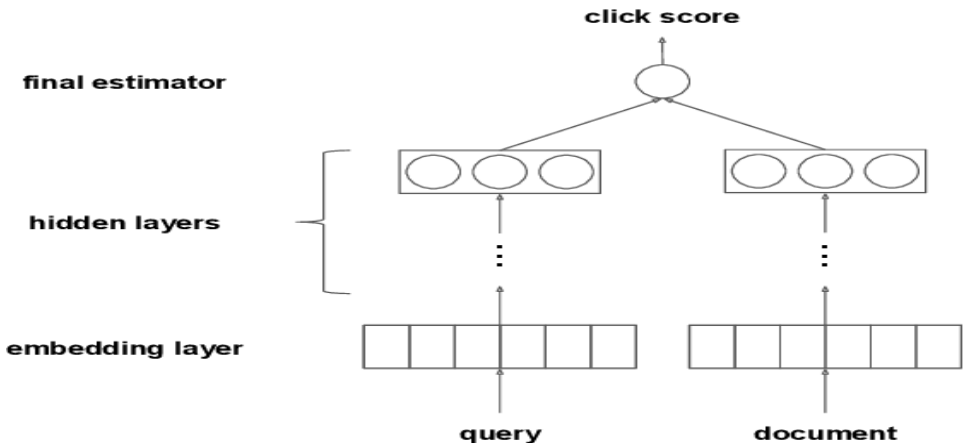


Figure III.13: L'architecture réseau du modèle d'appariement sémantique. [Zamani et al., 2017]

Les deux architectures de correspondance entre la requête donnée et les représentations abstraites du document sont :

1. **Context-Aware Deep Network Model (context-DNN) : c**e modèle utilise une couche cachée non linéaire pour modéliser la combinaison de toutes les caractéristiques contextuelles qui obtient la concaténation des représentations contextuelles en entrée.
2. **Context-Aware Wide and Deep Network Model (CWDNN) :** le modèle de réseau étendu et profond sensible au contexte est un modèle de réseau neuronal large et profond pour prendre en compte les fonctionnalités contextuelles pour modéliser des approches de mémorisation simples mais efficaces, lorsque de grandes quantités de données d'apprentissage sont disponibles.

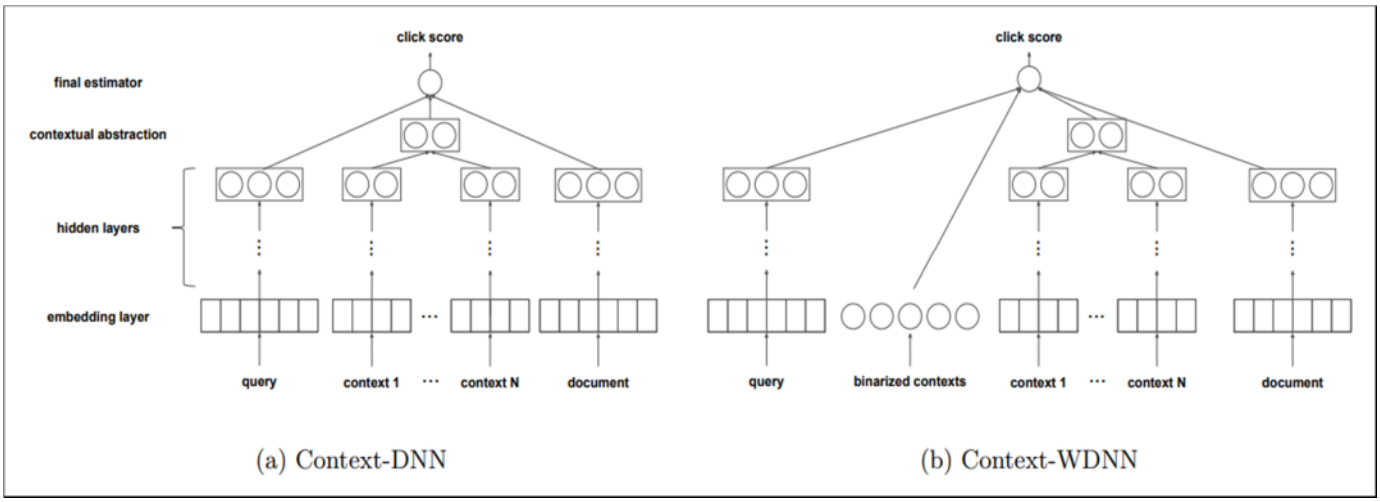


Figure III.14: Architectures de réseau des modèles Context-DNN et Context-WDNN. [Zamani et al., 2017]

III.2 Synthèses des travaux

Depuis plusieurs années, de nouveaux modèles et algorithmes sont mis au point pour traiter des données de plus en plus volumineuses et diverses. Parmi ces algorithmes, ceux du deep learning qui définissent un ensemble de méthodes d'apprentissage automatique conçues sur la

base de réseaux de neurones profonds, visant à mimer la profondeur des couches d'un cerveau humain. Ces algorithmes sont exploités dans plusieurs disciplines telles que : le traitement et la classification d'images, la reconnaissance d'objets et de la parole, notamment le traitement du langage naturel (TAL). L'efficacité de ces modèles a incité à les exploiter dans le processus de la recherche d'information à savoir : la représentation des documents/requêtes et l'appariement, afin de satisfaire une requête utilisateur ; l'information est donc traitée en couches hiérarchiques de neurones artificiels basée sur une architecture d'un réseau neuronal profond pour comprendre les représentations et les caractéristiques des données, pour calculer par la suite l'interaction entre des paires document/requête à l'aide d'une mesure de similarité. La force du deep learning réside dans le processus d'entrainement des réseaux de neurones profonds qui permet à une machine d'extraire des caractéristiques et apprendre toute seule. Cet entrainement revient à minimiser ou maximiser une fonction appelée fonction de coût [Nguyen et al., 2017] appelée aussi fonction objective [Yang et al., 2016], fonction d'erreur [Huang et al., 2013] ou fonction de perte dans [Jagerman et al., 2017] et [Pang et al., 2016] en appliquant l'algorithme d'optimisation par la descente du gradient (la rétro-propagation) dans le but d'apprendre les paramètres du modèle neuronal.

Les modèles du deep learning exploités pour les représentations des documents/requêtes ont la particularité de procéder avec une représentation continue vectorielle qui vise à exprimer la sémantique entre les termes, ainsi réduire le fossé sémantique de sac de mots (représentation lexicale), comme les deux architectures du modèle *word2vec* [Mikolov et al., 2013a] : *CBOW* et *Skip-gram* qui visent à réduire la dimensionnalité à travers le contexte en diminuant l'espace entre les termes similaires sémantiquement et syntaxiquement. L'efficacité de ces deux architectures a permis de les employer dans le processus de la *traduction* dans [Mikolov et al., 2013b] afin de déduire des arrangements entre les concepts d'une langue source et une autre langue cible à l'aide d'une métrique de distance. En outre, la représentation définie dans le modèle *DSSM* [Huang et al., 2013] ; basé sur un DNN, la particularité de ce modèle c'est qu'il vise à réduire la dimensionnalité avec une méthode dite hachage en décomposant des mots en lettres de trigramme, pour alléger la taille du vocabulaire d'entrée. Un autre modèle *DCNN* [Kalchbrenner et al., 2014] utilise un CNN pour représenter la sémantique des termes, en procédant à un regroupement des termes de phrases (de longueur variable) en entrée et induit un graphique de caractéristiques sur la phrase qui est capable de capturer explicitement des relations. Parmi tous ces modèles, un tout différent modèle [Nguyen et al., 2017], dont sa représentation sémantique d'un terme est une projection à travers une combinaison de deux représentations : les relations sémantiques explicites des termes dans une ressource externe, et les relations sémantiques implicites des termes dans le corpus. Tous ces modèles ont le point commun de faire l'appariement en calculant une mesure de similarité à l'aide d'une formule spécifique à chaque modèle. Une amélioration de cette mesure de similarité a été évoqué dans plusieurs modèles pour influencer sur la qualité d'appariement ; parmi ces modèles : le modèle *CDNN* pour le classement des paires de textes courts [Severyn et Moschitti, 2015] qui vise à injecter des features de similarité à un vecteur qui réunit la similarité entre un vecteur document et un vecteur requête, puis calculer les interactions entre les éléments de ce vecteur. Un autre modèle se focalise sur le classement des résultats de textes courts *aNMM* [Yang et al., 2016] en représentant la force des signaux d'appariement entre les paires document-requête en incorporant un schéma d'attention qui vise à discriminer l'importance des mots

dans une même requête. Dans le même contexte, les auteurs de [Pang et al., 2016] développent le *MatchPyramid* en agrégeant les informations de la matrice de correspondance en un seul score de correspondance en utilisant un CNN. L'auteure Belkacem dans [Belkacem, 2016] a choisi de minimiser la valeur de la sémantique entre les paires document-requête puis combiner leurs différentes caractéristiques reliant afin de calculer la similarité entre eux. Le modèle proposé par Dehghani et al. dans [Dehghani et al., 2017] consiste en trois architectures dont chacune un modèle de classement qui définit une fonction d'activation pour la sortie du réseau, en utilisant des signaux de faible supervision (par exemple : des données de clic) de la phase d'entrainement, c'est-à-dire sans implication humaine ou ressource externe ; dans le but de noter les documents pour chaque requête. Contrairement au modèle de classement pour une recherche personnelle proposé par [Zamani et al., 2017] qui n'exploite pas l'historique des clics mais plutôt le contexte caractérisant une requête, telles que : l'heure et l'emplacement lors de la recherche. [Jagerman et al., 2017] tente de résoudre le problème de l'ambiguïté des étiquettes de pertinence en trouvant une fonction qui attribue des scores élevés aux documents qui ont une pertinence élevée, et des scores faibles aux documents qui ont peu de pertinence en utilisant des probabilités dans un réseau à trois couches.

Le modèle *DSSM* [Huang et al., 2013] utilise une méthode dite hachage dans la première couche cachée, cette méthode prend en compte la cooccurrence[5] des termes et les décompose en trigramme ; l'inconvénient de cette méthode c'est qu'elle ne prend pas en charge les relations sémantiques entre les termes d'un document ou d'une requête, c'est-à-dire, la représentation de ce modèle ne calcule pas la similarité entre ces termes hachés avant de les représenter dans un espace vectoriel, il calcule uniquement la similarité entre les paires document/requête dans le processus d'appariement.

Notre contribution dans le chapitre 4 consiste à modéliser un modèle de représentation qui définit une autre version de l'architecture DSSM, qui exploitera le plongement lexical word2vec ; afin de bénéficier à la fois : d'une couche d'entrée allégée grâce à la méthode de hachage et d'un espace vectoriel de représentation sémantique qui prend en compte le contexte grâce à l'algorithme de word2vec.

Conclusion :

Dans ce chapitre, nous avons présenté un état de l'art sur un ensemble de modèles du deep learning exploités dans la recherche d'information (RI), dans le but d'améliorer la qualité des systèmes de recherche d'information. L'ensemble de ces modèles concerne soit la représentation des entrées document/requête, soit la similarité et le calcul de pertinence pour le classement. Parmi ces modèles, des modèles développés pour résoudre des anomalies de la RI comme, par exemple : la contrainte de taille du vocabulaire, en utilisant des architectures basées sur une représentation sémantique vectorielle.

Dans le prochain chapitre, nous allons présenter notre contribution, en détaillant l'architecture de notre modèle ainsi que l'algorithme de son fonctionnement.

[5] Cooccurrence : est la présence simultanée de deux ou de plusieurs termes dans le même énoncé.

Chapitre IV
Contribution

Introduction

Dans le chapitre précédent,nous avons présenté quelques travaux du deeplearningexploités dans la recherche d'information.Nous avons vu que les méthodes exploitant les réseaux de neurones et les techniques d'apprentissage profonddans la recherche d'information se basent sur les caractéristiques textuelles,les entrées pouvant être de simples séquences de mots ou des documents complets et des requêtes, selon l'objectif du modèle proposé.

Dans ce chapitre, nous présentons notre contribution qui se base sur les travaux de l'état de l'art, dont l'objectif est de construire un modèle de représentation pour une requête et un document,ce modèle est basé sur les représentations sémantiques construites à base des représentations continues des mots du vocabulaire.

IV.1 Motivation et idée de base

Afin de calculer la similarité document-requête, la majorité des modèles en recherche d'information (RI) représentent les documents et les requêtes sous forme de « sacs de mots » (bag of words) pondérés ou un sac de concepts, construits automatiquement par des techniques de types LSI[1]ou LDA[2]. Ce type de représentation ne permet pas de capturer le sens véhiculé par les mots et les relations potentielles entre ces mots, qui sont exprimés plus précisément par la structure et l'ordre dans lequel les termes sont présentés [Belkacem et al., 2017].

La représentation en sac de mots ou bag of words (BoW) présente deux problèmes majeurs :

- Le problèmede dimensionnalité car la dimension totale est la taille du vocabulaire. La solutionconsiste à utiliser une technique de réduction de dimensionnalité pour les données d'entrée.
- Elle ne tient pas compte des positions relatives des mots dans les documents (la relation sémantique des mots). En général, les mots voisins dans une phrase devraient être utiles pour prédire lemot cible.

Pour combler l'écart entre le vocabulaire utilisé dans la requête et celui présenté dans les documents,nous nous sommes intéressées à intégrer les relations sémantiques entre les mots par le *plongement lexical*[Vukotic et al., 2015][Braud, 2015] nommé *wordembedding*[Mikolov et al., 2013a][Mikolov et al., 2013b][Mikolov et al., 2013c] en anglais. Le plongement lexical des mots appelé également représentation continue des mots a été massivement exploités ces dernières années, notamment pour la tâche de recherche d'information ; c'estune représentation vectorielle des mots de type *word2vec*.L'approche *word2vec* propose de modéliser les termes sous forme de vecteurs (représentationdes motsdansun espace vectoriel) tout en tenant compte du contexte.Par exemple, un utilisateur intéressé par l'achat d'une voiture ; formule la requête « achat voiture », cependant, des documents pertinents peuvent utiliser « vente automobile » pour exprimer le même concept. Le plongement lexicalest l'une des techniques utilisée pour résoudre ce problème ; elle calcule la correspondance des mots similaires, telles que « automobile- voiture», « vente-achat ».

Dans [Mikolov et al., 2013a]le modèle représenté construit des représentations continues des mots en se basant sur leur contexte, en prenant en compte la fenêtre d'occurrence d'un mot pour construire une représentation permettant de déduire des relations sémantiques entre des termes différents. Basé sur cette idée, plusieurs travaux ont été développés dans le même

[1]LSI : Latent Semantic Indexation.

[2] LDA : Latent Dirichlet Allocation.

contexte,afin de construire des représentations continues des séquences de mots à partir de celles des mots la constituant.

Notre travail est inspiré des travaux de l'état de l'art, notamment les travaux de Mikolov sur le*plongement lexical*de mots dans[Mikolov et al., 2013a],et le modèle *deepstructuredsemanticmodels* (DSSM)[Huang et al., 2013]pour la représentation des textes documents et requêtesafin de calculer la similarité entre les paires document-requête. Notrecontribution est donc basée sur la construction des représentations vectorielles permettant de résoudre le problème du sac de mots,en utilisantlesréseaux neuronaux profonds (DNN) afin de classer un ensemble de documents pour une requête donnée.

IV.2 Notre contribution

Notre contribution se situe au niveau du modèle de représentation desdocuments (requêtes)dans le processus en U de la RI,elle s'articule autour de trois points principaux:

1. Nous avons appliquéla fonction de hachage *Hashing Trick*basée sur les bi-grammespour la première couche cachée du réseau de neurones profond (DNN) du modèle DSSM[Huang et al., 2013]. Cette fonction vise à réduire la dimensionnalité des vecteurs de termes de sac de mots en les représentant avec une séquence de deux mots tout en prenant compte de la cooccurrence des mots, et permet d'utiliser de très grands vocabulaires pour la représentation.
   L'objectif de l'utilisation de cette méthode de hachage est :
   - Traiter les nouveaux mots hors du vocabulaire d'entrainement du modèle sans avoir besoin de le recycler à partir de zéro, grâce à l'initialisation d'un nouveau vecteur pour chaque nouvelle donnée (document ou requête).
   - Le modèle DSSM [Huang et al., 2013] utilise une fonction de hachage basée sur des trigrammes (**décomposition en trois lettres**), ce qui rend la taille des vecteurs du vocabulairelors de chaque nouvel entrainement plus grande d'une part, d'une autre partcela provoque des collisions et une similarité entre des termes qui n'ont rien avoir entre eux, par exemple : le mot *"fièvre"* et *"fière"* (*fiè= fiè*). Nous utilisons la méthode de hachage sur des **bi-grammes de mots** pour diminuer la taille des vecteurs (éviter d'encombrer la première couche cachée), ainsi nous obtenons un espace vectoriel réduit avec moins de collisions.
2. Nous avons suggéré pour l'amélioration de la représentation du DSSM [Huang et al., 2013] l'utilisation de l'outil *word2vec* pour unereprésentation sémantiquedes mots dansun espace vectoriel. Cet outil prend en compte le contexte et les relations entre les termes dans un texte (document/requête).
3. Enfin,nous avons choisis le coefficient de Jaccard pour calculer la similarité entre deux vecteurs. D'après [Huang, 2008] et [Strehl et al., 2000] les auteurs ont montré que les performances de la similarité cosinus, du coefficient de Jaccard et du coefficient de Pearson sont très proches et qu'elles sont significativement meilleures que celles de la distance euclidienne.[Negre, 2013]

IV.3 La représentation bi-gramme

Les représentations vectorielles standards fonctionnant sur un principe de « sac de mots » conduisent à établir les similitudes fortes entre deux textes s'opposant sur un plan sémantique,mais elles ne tiennent pas compte de l'ordre des attributs. Pourtant, l'ordre des attributs peut se révéler un marqueur crucial à prendre en compte pour représenter efficacement et justement la sémantique du texte et ce particulièrement lorsque les attributs sont les mots des textes. [Trouvilliez, 2013]

Nous expliquons à travers l'exemple suivant, la représentationde texte, nous utilisons pour se faire les deux textes*T1* et *T2* :

T1 : « Les vêtements sont de bonne qualité mais vos prix ne sont pas corrects ».
T2 : « Vos prix sont corrects mais vos vêtements ne sont pas de bonne qualité ».

Dans le texte *T1*, le mot *"sont"* apparaît deux fois donc on note *(terme, nombre d'occurrences d'un mot donné dans le texte) = (sont, 2)*. De même pour tous les autres termes de*T1* et *T2*tout en supprimant les mots vides habituels (les, de, vos).
La représentation vectorielle standard des deux textes est :
{(vêtements, 1), (sont, 2), (bonne, 1), (qualité, 1), (mais, 1), (prix, 1), (ne, 1), (pas, 1), (corrects, 1)}.
Les deux textes dansl'exemple ont donc la même représentation vectorielle alors qu'elles ont des sens différents. Afin de contourner cette difficulté, il est usuel d'avoir recours à des représentations qui prennent en compte l'ordre des attributs afin de lever l'ambiguïté provoquée par le sac de mots.
Nous proposons une représentation bi-gramme.On désigne par bi-gramme, une suite de deux symboles linguistiques consécutifs.
Les deux textes qui avaient une représentation vectorielle standard identique ont cette fois des représentations par 2-grammes de mots différentes.

La représentation de T1 par bi-gramme :
{(vêtements, sont), (sont, bonne), (bonne qualité), (qualité, mais), (mais, prix), (prix, ne), (ne, sont), (sont, pas), (pas corrects)}.
La représentation du T2 par bi-gramme :
{(prix, sont), (sont, corrects), (corrects, mais), (mais, vêtements), (vêtements, ne), (ne, sont), (sont, pas), (pas, bonne), (bonne, qualité)}.

Sur les neuf 2-grammes de mots qui constituent les représentations des deux textes, seuls deux sont communs aux deux représentations. Les textes sont donc cette fois clairement dissociés.
Une fois que les documents textuels sont associés à des représentations vectorielles bi-grammes, elles vont servir au calcul des similarités pour extraire les correspondances entre leurs deux vecteurs respectifs.
Prenons par exemple ; les paires*(prix, raisonnables), (bon, prix)*sont classées dans le même concept noté*« prix abordable »*,et les paires *(bon, accueil), (accueil, souriant)* sont classés dans un même concept (contexte) dit *« bon accueil »* ; car ils sont sémantiquement similaires.

IV.4 Architecture du modèle

Notre modèleest composé d'une couche d'entrée, trois couches cachées, et une couche de sortiecomme le montre la figureIV.1.

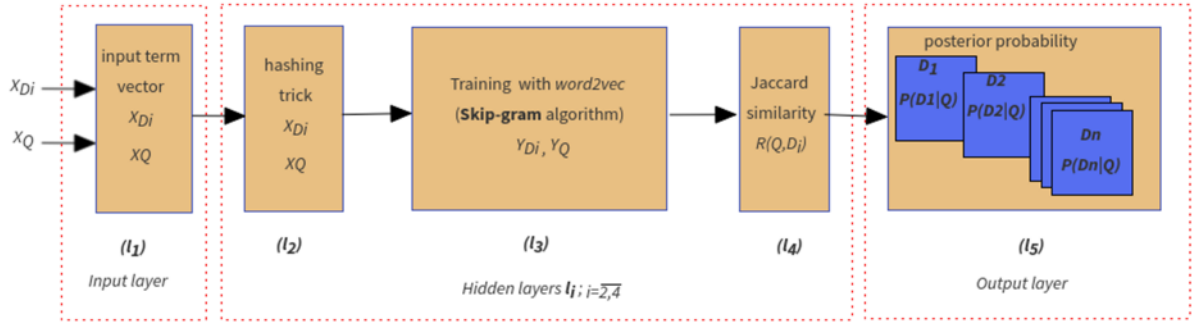


Figure IV.1: L'architecture de notre modèle avec une représentation vectorielle sémantique et une similarité de Jaccard.

- ***La couche d'entrée (l1) :***comme entrée du DNN,nous avons un ensemble de vecteurs de termes (requête ou document) de haute dimension, par exemple le nombre brut de termes dans un document ou une requête.

- ***La couche de hachage(l2) :***la première couche cachée, effectue le hachage de mots.Cette couche prend en entrée le vecteurdu documentX_{D_i} ou de la requête $X_Q$sous-forme de texte brut, et effectue la méthodede hachage *Hashing Trick*basée sur les bi-grammes(définie dans IV.3) pour produire en sortie des vecteurs hachés.Cette méthode vise à réduire la dimensionnalité des vecteurs de termes de sac de mots et les collisions en les représentantspar des séquences de deux termes.

- ***La couche de sémantique(l3) :***la deuxième couche cachée ;cette couche prend en entrée le vecteur du document ou de la requête sous-forme de texte hachéX_{D_i}et X_Q respectivement,puis produit la représentation vectorielle sémantique correspondante à la séquence d'entrée, ceci en utilisant la matrice du vocabulaire apprise en phase d'entrainement de cette couche etl'outil *word2vec*. La matrice du vocabulaire représente le nouvel espace vectoriel dans lequel les documents et les requêtes sont projetés, elle est apprisepar le *wordembedding*en utilisant l'algorithme *Skip-gram*(en français : algorithme de prédiction des grammes-voisins) qui reçoit en entrée les vecteurs des séquences de bi-grammes pour les placer dans des concepts, ainsi chaque document et requête est représenté par son vecteur de concept Y_{D_i}et Y_Q respectivement.

- ***La couchede similarité (l4):***la troisième couche cachée ;cette couche sert à calculer la similarité basée sur le coefficient de Jaccard entre les vecteurs de concepts construits par la couche précédente. Elle calcule la pertinence d'un document pour une requête avec la formule suivante :

$$R(Q,D) = Sim_{jaccard}(y_Q, y_D) = \frac{\overrightarrow{y_Q} \cdot \overrightarrow{y_D}}{\|\overrightarrow{y_Q}\|\|\overrightarrow{y_D}\| - \overrightarrow{y_Q} \cdot \overrightarrow{y_D}}$$

- ***La couche de sortie (l5) :***Après le calcul de similarité, les documents sont classés pour l'utilisateur en tenant compte de leurs degrés de pertinence, ce degré de pertinence est la probabilité postérieure qu'un document est pertinent sachant une requête donnée. Elle est calculée via une fonction *softmax* :

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \epsilon D} \exp(\gamma R(Q,D'))}$$

Où : $\boldsymbol{\gamma}$ **:** est un facteur de lissage qui est mis empiriquement sur un ensemble de données.

$\boldsymbol{D}$ **:**est une liste des documents candidats, construit par D^+ et un ensemble de quatre documents pris aléatoirement de la collection.

$\boldsymbol{R(Q,D)}$ **:**est le score de pertinence sémantique entre une requête *Q* et un document *D*.

IV.5 L'entrainement du modèle

L'entrainement du modèleest effectué dans la couche de sortie à base des documents cliqués pour les requêtes d'entrainement *clickthroughlogs* :

Dans un premier temps, sélectionner les documents D^+ cliqués pour une requête *Q*, ensuite nous calculons la probabilité postérieure d'un documentpour une requête donnée à partir du score de pertinence sémantique entre euxvia une fonction *softmax*.

Par ailleurs, nous devons minimiser la fonction d'erreur :

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$

où : $\boldsymbol{\Lambda}$ **:** désigne l'ensemble des paramètres du réseau de neurones $\{W_i, b_i\}$.

À chaque itération, les poids $W_i$et les vecteurs b_i sont modifiés en utilisant les algorithmes d'optimisation numérique du gradient.

IV.6 L'algorithme du modèle

Algorithme général

Entrée :$E = \{(Q, D)\}$// Ensemble des vecteurs des paires <requête, document>.

Sortie : D_i // Liste des documents pertinents.

Début

1) Injecter les documents et la requête dans la couche d'entrée sous forme de vecteurs.
2) Représenter les vecteurs documents et requête en bi-grammes à l'aide de l'algorithme1 de *Hashing trick*.
3) Représenter les relations sémantiques dans chaque vecteur haché en utilisant l'algorithme2 de *word2vec*.
4) Calculer la similarité entre chaque vecteur sémantique d'un document et celui de la requête avec *le coefficient de Jaccard*.
5) Classer les documents pertinents pour la requête, en calculant la probabilité postérieure de l'algorithme3 (l'algorithme d'entrainement).

Fin.

Algorithme 1 : Hashing trick

Entrée :$S = \{w_1, w_2, .., w_n\}, N$ // S :séquence de mots, et *N*:nombre de dimension.

Sortie : X:nouveau vecteur$[N]$ // Vecteur de sortie de dimension *N*.

Début

Pour une séquence de mot *S* faire

1) Construire un nouveau dictionnaire *D* de bi-grammes
2) Pour chaque bi-gramme$(w_i, w_{i+1}) \in D$ faire
   - Calculer sa valeur de hachage à l'aide d'une fonction de hachage hash_function ()

$$h \coloneqq hash_{function}(w_i, w_{i+1});$$

– Appliquer la fonction mod pour la valeur h puis affecter le résultat sur le nouveau vecteur X :

$X[h\ mod\ N]$+1 ; //Incrémenter pour chaque nouveau vecteur

Retourner X ;

Fait ;
Fin.

Algorithme 2 : Skip-gram

Entrée :$\{w_t, w_c\}\ et\ \theta = \{v_1, v_2, v_3, \dots v_N, u_1, u_2, u_3 \dots u_N\}$// L'ensemble des paramètres du modèle.
Sortie : (w, c)// w est le bi-gramme à prédire qui suit le contexte c.
Début
Initialisation aléatoire des paramètres du modèle.
Pour chaque bi-gramme à prédire $w_t \in T$ faire

1) Calculer la fonction de similarité avec la mesure cosinus :

$$cos(\vec{u}_t, \vec{v}_c) = \frac{\vec{u}_t \cdot \vec{v}_c}{\|\vec{u}_t\|\|\vec{v}_c\|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\sum_{i=1}^{N} u_i^2}\sqrt{\sum_{i=1}^{N} v_i^2}}$$

2) Construire la représentation des bi-grammes par concepts.
3) Maximiser la moyenne des logs de probabilités attribués de tout bi-gramme w_t du contexte sachant le bi-gramme w_{t+j}cible (en modifiant les poids pour réduire l'erreur de prédiction de l'algorithme) :

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-c}^{c} log\ P\left(w_{t+j}|w_t\right)\right]_{(j\neq 0)}$$

Fait ;
Fin.

Les paramètres du modèle$\{W_i, b_i\}$ sont initialisés (avant de lancer le processus d'entrainement) aléatoirement dans l'intervalle $\left[-\sqrt{6\ (fanin + fanout)}, +\sqrt{6\ (fanin + fanout)}\right]$avec $fanin$ et $fanout$sont respectivement le nombre d'entrées et de sorties.

Algorithme 3 : Entrainement du modèle

Entrée :$E = \{(Q, D)\}$// Ensemble des paires d'entrainement<requête, liste_doc_pertinents>.
Sortie :$\theta = \{W, b\}$et$M$// Paramètres du modèle et matrice de similarité apprise.
Début
Pour chaque paire(Q, D)faire

1) Calculer la similarité basée sur le coefficient de Jaccard entre les vecteursy_Q ety_D construits par la couche Skip-gram avec l'équation :

$$R(Q,D) = Sim_{jaccard}(y_Q, y_D) = \frac{\overrightarrow{y_Q} \cdot \overrightarrow{y_D}}{\|\overrightarrow{y_Q}\| \|\overrightarrow{y_D}\| - \overrightarrow{y_Q} \cdot \overrightarrow{y_D}}$$

2) Pour chaque document pertinent D_i faire
   - Sélectionner les documents D^+ cliqués pour une requête Q
   - Calculer la probabilité conditionnelle du document$D$ pour la requête$Q$ avec la formule :

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \epsilon D} \exp(\gamma R(Q,D'))}$$

   - Mettre à jour les paramètres du modèle en fonction de l'erreur par l'équation:

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$

Fait ;
Fait ;
Fin.

IV.7 Exemple

Nous allons dérouler sur un ensemble composé d'une requête$Q$ et de trois documents D_1,D_2 et D_3, le fonctionnement de notre modèle.
Les documents et la requête utilisés sont :
D_1: « *les vêtements sont de bonne qualité* »
D_2: « *vos prix ne sont pas corrects* »
D_3: « *la recherche d'information en informatique* »
Q: « *les vêtements sont simples et de bonne qualité* »
Le vocabulaire initiale de la collection des documents (D_1, D_2 *et* D_3) est :

$$V_i = \{vêtements, sont, bonne, qualité, vos, prix, ne, pas, corrects, recherche, information, informatique\}$$

$$Dim\ V_i = 12$$

Phase de hachage :

La représentation vectorielle deD_1,D_2, D_3 et Q par bi-grammes est donnée comme suit :
D_1 :*{(vêtements, sont),(sont, bonne),(bonne, qualité)}*
D_2 :*{(vos, prix), (prix, ne), (ne, sont),(sont, pas),(pas, corrects)}*
D_3 : *{(recherche, information), (information, informatique)}*
Q :*{(vêtements, sont),(sont, simples),(simples, bonne),(bonne, qualité)}*
Le vocabulaire de la collection des documents (D_1, D_2 *et* D_3) après le hachage avec bi-gramme est :

$$V_h = \{(vêtements, sont), (sont, bonne), (bonne, qualité), (vos, prix), (prix, ne), (ne, sont), (sont, pas), (pas, corrects), (recherche, information), (information, informatique)\}$$

$$Dim\ V_h = 10$$

Phase de Skip-gram :

1. L'initialisation aléatoire des vecteurs de bi-gramme :Pour $k = [1 ; 12]$on a :
   - v_k est le vecteur qui représente le bi-gramme w_k dans l'espace des concepts (nombre de bi-grammes initial = 12) ; et
   - $u_k$est le vecteur qui représente le bi-gramme w_k dans l'espace des contextes (au départ le vecteur des contextes est égal au vocabulaire de concepts =12).

Espace des concepts		Espace des contextes	
$vêtements, sont$	$(0.1, 1.1, 0.2, 1.0, 0)$	$vêtements, sont$	$(0.1, 1.1, 0.2, 1.0, 0)$
$sont, bonne$	$(1.1, 0.5, 0.2, 0.2, 0.3)$	$sont, bonne$	$(1.1, 0.5, 0.2, 0.2, 0.3)$
$bonne, qualité$	$(0.8, -0.2, 0.3, -0.5, -0.3)$	$bonne, qualité$	$(0.8, -0.2, 0.3, -0.5, -0.3)$
$vos, prix$	$(0.8, 0.2, -0.1, -1.0, 0.1)$	$vos, prix$	$(0.8, 0.2, -0.1, -1.0, 0.1)$
$prix, ne$	$(-0.2, -0.3, -0.2, -0.5, -0.5)$	$prix, ne$	$(-0.2, -0.3, -0.2, -0.5, -0.5)$
$ne, sont$	$(0.1, -1.1, -0.3, -1.0, 0.1)$	$ne, sont$	$(0.1, -1.1, -0.3, -1.0, 0.1)$
$sont, pas$	$(-0.6, 0.2, -0.1, -0.9, -0.3)$	$sont, pas$	$(-0.6, 0.2, -0.1, -0.9, -0.3)$
$pas, corrects$	$(-0.2, 0.2, 0.2, 0.7, -0.3)$	$pas, corrects$	$(-0.2, 0.2, 0.2, 0.7, -0.3)$
$sont, simples$	$(0.1, 0.1, -1.1, 0.1, 0.8)$	$sont, simples$	$(0.1, 0.1, -1.1, 0.1, 0.8)$
$simples, bonne$	$(0.3, -0.2, 0.7, 0.6, 0.3)$	$simples, bonne$	$(0.3, -0.2, 0.7, 0.6, 0.3)$
$recherche, information$	$(0.1, -1.1, -0.2, -1.0, 0)$	$recherche, information$	$(0.1, -1.1, -0.2, -1.0, 0)$
$information, informatique$	$(0.2, 1.0, -0.3, 0.7, -1; 1)$	$information, informatique$	$(0.2, 1.0, -0.3, 0.7, -1; 1)$

2. Calcul de similarité entre les bi-grammes avec la mesure cosinus :

$$cos(\vec{u}_t, \vec{v}_c) = \frac{\vec{u}_t \cdot \vec{v}_c}{\|\vec{u}_t\|\|\vec{v}_c\|} = \frac{\sum_{i=1}^{N} u_i v_i}{\sqrt{\sum_{i=1}^{N} u_i^2}\sqrt{\sum_{i=1}^{N} v_i^2}}$$

   - Calculons la similarité entre les vecteurs $\vec{u}_{sont,bonne}$ et $\vec{v}_{vêtements,sont}$:

$$cos(\vec{u}_{sont,bonne}, \vec{v}_{vêtements,sont}) = \frac{\vec{u}_{sont,bonne} \cdot \vec{v}_{vêtements,sont}}{\|\vec{u}_{sont,bonne}\|\|\vec{v}_{vêtements,sont}\|} = \frac{0.9}{1.50 \cdot 1.28} \simeq 0.47$$

   - Calculons la similarité entre les vecteurs $\vec{u}_{sont,bonne}$ et $\vec{v}_{recherche,information}$:

$$cos(\vec{u}_{sont,bonne}, \vec{v}_{recherche,information}) = \frac{\vec{u}_{sont,bonne} \cdot \vec{v}_{recherche,information}}{\|\vec{u}_{sont,bonne}\|\|\vec{v}_{recherche,information}\|}$$

$$= \frac{-0.68}{1.50 \cdot 3.26} \simeq -0.14$$

Le produit scalaire $\vec{u}_{sont,bonne} \cdot \vec{v}_{vêtements,sont}$ caractérise la similarité entre $\vec{u}_{sont,bonne}$ et $\vec{v}_{vêtements,sont}$.
Le produit scalaire $\vec{u}_{sont,bonne} \cdot \vec{v}_{recherche,information}$ caractérise la similarité entre $\vec{u}_{sont,bonne}$ et $\vec{v}_{recherche,information}$.
Globalement, plus le produit scalaire $\vec{u}_t \cdot \vec{v}_c$ est positif et grand, plus cela signifie que le $cos(\vec{u}_t \cdot \vec{v}_c)$est proche de 1, ainsi les deux vecteurs vont être dans le même concept.
Le $cos(\vec{u}_{sont,bonne}, \vec{v}_{vêtements,sont}) = 0.47 > 0$ donc $\vec{u}_{sont,bonne}$ et $\vec{v}_{vêtements,sont}$ vont être classés dans un même concept.
Le $cos(\vec{u}_{sont,bonne}, \vec{v}_{recherche,information}) = -0.14 < 0$ donc $\vec{u}_{sont,bonne}$ et $\vec{v}_{recherche,information}$ vont être classés dans deux concepts différents.

Après les calculs de similarité $cosine$entre chaque vecteur du contexte $\vec{u}_k$et vecteur de concept$\vec{v}_w$, les bi-grammes sont classés dans deux concepts distincts à savoir :

1. **Concept 1 :**$0 < cos < (\vec{u}_t \cdot \vec{v}_c) \leq 1$ que nous avons nommé« $produit$ » : contient tous les bi-grammes des documents D_1etD_2

$$\text{« } produit \text{ »} = \{D_1, D_2\}$$
$$= \{(vêtements, sont), (sont, bonne), (bonne, qualité), (vos, prix), (prix, ne), (ne, sont), (sont, pas), (pas, corrects)\}$$

2. **Concept 2 :**$-1 \leq cos < (\vec{u}_t \cdot \vec{v}_c) \leq 0$ nommé « $informatique$ » : contient les bi-grammes du document D_3.

$$\text{« } informatique \text{ »} = \{D_3\} = \{(recherche, information), (information, informatique)\}$$

<u>Phase d'appariement :</u>

Rappel :

$tf = freq(t, d)$: est la fréquence du terme t dans le document d .

$idf = \log\left(\frac{N}{n_t}\right)$: est la fréquence du terme t dans la collection.

Où : $\boldsymbol{N}$ est le nombre de documents de la collection, et $\boldsymbol{n_t}$ est le nombre de documents contenant le terme t .

$w(t, d) = tf * idf$ est le poids d'un terme t dans le document d.

Notre exemple contient trois documents donc $N = 3$. Par exemple : le bi-gramme *(bonne, qualité)* apparait une fois dans le document D_1 donc :

$$idf_{(bonne,qualité)} = \log\left(\frac{3}{1}\right) \simeq 0.48$$

Le calcul du $tf \cdot idf$ du bi-gramme *(bonne, qualité)* est donc :

$$tf \cdot idf_{(bonne,qualité)} = \frac{1}{3} \cdot \log(3) \simeq 0.16$$

De la même manière nous calculons le poids des autres bi-grammes pour construire la matrice de poids w_{ij}.

	vêtements, sont	sont, bonne	bonne, qualité	Vos, prix	Prix, ne	Ne, sont	Sont, pas	Pas, correct	Sont, simples	Simples, bonne	Recherche, information	Information, informatique
D_1	0.16	0.16	0.16	0	0	0	0	0	0	0	0	0
D_2	0	0	0	0.1	0.1	0.1	0.1	0.1	0	0	0	0
D_3	0	0	0	0	0	0	0	0	0	0	0.24	0.24

Une fois que nous avons calculé la matrice de poids, nous calculons la similarité entre les paires*(document, requête)* avec le coefficient de Jaccard.

Considérons le vecteur de termes pondérés de la requête Q :

$$Q(0.18, 0, 0.18, 0, 0, 0, 0, 0, 0.19, 0.19, 0, 0)$$
$$\|D_1\| \simeq 0.29$$
$$\|D_2\| \simeq 1.78$$
$$\|D_3\| \simeq 0.12$$
$$\|Q\| \simeq 0.37$$

Nous calculons la similarité de Jaccard entre la requête Qet les documents desdeux concepts« $produit$ » (D_1 et D_2) et « $informatique$ » (D_3) :

- Calcul de similarité entre la requête Q et le document D_1 :

$$Sim_{jaccard}(y_Q, y_{D_1}) = \frac{\overrightarrow{y_Q} \cdot \overrightarrow{y_{D_1}}}{\|\overrightarrow{y_Q}\|\|\overrightarrow{y_{D_1}}\| - \overrightarrow{y_Q} \cdot \overrightarrow{y_{D_1}}} \simeq \frac{0.05}{(0.37 \cdot 0.29) - 0.05} \simeq 0.83$$

- Calculde similarité entre la requête Q et le document D_2 :

$$Sim_{jaccard}(y_Q, y_{D_2}) = \frac{\overrightarrow{y_Q} \cdot \overrightarrow{y_{D_2}}}{\|\overrightarrow{y_Q}\|\|\overrightarrow{y_{D_2}}\| - \overrightarrow{y_Q} \cdot \overrightarrow{y_{D_2}}} \simeq \frac{0}{(0.31 \cdot 1.78) - 0} = 0$$

- Calcul de similarité entre la requête Q et le document D_3 :

$$Sim_{jaccard}(y_Q, y_{D_3}) = \frac{\overrightarrow{y_Q} \cdot \overrightarrow{y_{D_3}}}{\|\overrightarrow{y_Q}\|\|\overrightarrow{y_{D_3}}\| - \overrightarrow{y_Q} \cdot \overrightarrow{y_{D_3}}} \simeq \frac{0}{(0.31 \cdot 0.12) - 0} = 0$$

Par conséquent, selon le coefficient de Jaccard, le document D_1 et la requête Q sont les plus similaires, autrement dit, le document D_1 est plus pertinent pour la requête Q et son degré de pertinence est de 83%.
Donc étant donné la requête Q, les documents qui peuvent être pertinents pour la requête Q sont les documents qui appartiennent au concept « $produit$ » $= \{D_1, D_2\}$.

Phase classement :

Nous calculons les probabilités entre la requête et les documents du concept « $produit$ » $= \{D_1, D_2\}$:
Nous initialisons le facteur de lissage :$\gamma = 0.75$ et l'ensemble $D' = \{D_1, D_2, D_3\}$:

- Calcul deprobabilité entre la requête Q et le document D_1 :

$$P(D_1|Q) = \frac{\exp(\gamma R(Q, D_1))}{\sum_{D' \epsilon D} \exp(\gamma R(Q, D'))} = \frac{\exp(0.75 * 0.83)}{\exp(0.75 * 0.83) + \exp(0.75 * 0) + \exp(0.75 * 0)}$$

$$= 0.48$$
$$= \mathbf{48\%}$$

- Calcul deprobabilité entre la requête Q et le document D_2 :

$$P(D_2|Q) = \frac{\exp(\gamma R(Q, D_2))}{\sum_{D' \epsilon D} \exp(\gamma R(Q, D'))} = \frac{\exp(0.75 * 0)}{\exp(0.75 * 0.83) + \exp(0.75 * 0) + \exp(0.75 * 0)}$$

$$= 0.25$$
$$= \mathbf{25\%}$$

Donc le modèle va classer en premier lieu le document D_1 ensuite le document D_2.

Conclusion

Dans ce chapitre, nous avons décrit notre approche proposée pour le processus de la recherche d'information, elle consiste en un modèle de représentationen utilisant l'architectured'un MLP (Multi Layer Perceptron) ou Deep Neural Network (DNN) ; en se basant sur les représentations sémantiques des documents et requêtes construites par le modèle word2vec de Mikolov[Mikolov et al., 2013a] et la fonction de hachage *Hashing trick*basée sur les bi-grammes pour réduire la dimension des espaces vectoriels de représentation. Pour

l'appariement, notre architecture effectue une similarité de Jaccard sur les représentations obtenues.

Conclusion générale

Dans ce rapport, nous avons résumé le travail réalisé basé sur la recherche sur le web, notre thématique de base est l'exploitation du deeplearningdans la recherche d'information. L'objectif de ce travail est d'étudier d'abord les travaux de l'état de l'art concernant les réseaux de neurones et plus particulièrement l'apprentissage profond, ainsi que les approches de recherche d'information exploitant ces techniques, afin de s'en inspirer par la suite, pour proposerun modèle de recherche d'information exploitant les réseaux de neurones profonds. L'objectif principal est la résolution des problèmes liés à l'usage du bag of words (sacs de mots) posés en recherche d'information. En effet, ce type de représentation ne permet pas de capturer les sens des mots ni les concepts et thématique du document. Pour ce faire, nous nous sommes intéressées à l'exploitation d'autres approches permettant de capturer des représentations plus fines du contenu des documents et des requêtes. Nous avons alors proposé un ensemble de méthodes de construction de ces représentations, et l'exploitation de ces représentations dans le processus de similarité.

Perspectives :

Notre contribution peut bénéficier de plusieurs perspectives pour les futurs travaux, nous pensons à généraliser notre contribution selon les points suivants :

- Implémentation et expérimentation du modèle ; en l'évaluant sur la même collection de test que DSSM [Huang et al., 2013] afin de pouvoir comparer sa performance par rapport à ce dernier.
- Nous pensons à l'utilisation des autres sources d'information qui sont aussi porteuses de sémantique comme les multimédia (image et vidéo). Par exemple, avec le progrès important dans le domaine vision par ordinateur, les machines sont désormais capables de détecter les objets dans une image et de les annoter. Les images fournissent ainsi une compréhension globale du monde (par exemple via les cooccurrences entre objets) qui permet d'augmenter la sémantique latente capturée par les méthodes distributionnelles textuelles. Cette perspective fait référence au "languagegrounding". Dans cet esprit, certains travaux ont démontré la complémentarité du texte et de l'image en mettant en avant le biais existant entre ces deux modalités [Glenberg et Kaschak, 2002][Gordon et Van Durme, 2013]. En effet, les textes sont généralement porteurs de sémantique globale ou racontent généralement des évènements particuliers alors que les images portent des informations intrinsèques sur les objets et également les relations contextuelles qu'ils entretiennent avec les autres objets, donc l'adaptation de ce modèle à s'entrainer sur des données multimédia permettra d'accentuer la qualité de la sémantique, ainsi les relations dans le contenu des documents et des requêtes.
- Une perspective intéressante à ce mémoire serait d'envisager des approches sémantiques multi-sources capturant la sémantique exprimée à la fois dans le texte, les bases de connaissances et également d'autres modalités porteuses de sens commun, tels que les images et les vidéos. Une façon d'envisager cette perspective est d'étudier l'alignement entre ces différentes modalités afin de proposer des modèles de langue multimodaux à partir de ces différentes sources.

Bibliographie

Références bibliographiques

[Ackley et al., 1985] : DAVID H. ACKLEY, GEOFFREY E. HINTON, et and T. J. Sejnowski, A : *"Learning Algorithm for Boltzmann Machines"*, Computer Science Department Carnegie-Mellon University, TERRENCE J. SEJNOWSKI, Biophysics Department The Johns Hopkins University, Cognitive science 9, 147-169, 1985.

[Adamson et Boreham, 1974] : G.Adamson and J.Boreham, *"The Use Of an association measure based on character structure to identify semantically related pairs words and document titles"*, Information Storage and Retrieval, Vol.10, pp 253-263, 1974.

[Adil et al., 2015] : ADIL Toumouh, DEBAKLA Mohammed et ZOUANEB Imane : *"Un système de recherche d'information contextuel"*, Conférence : 2ème Conférence Internationale sur le Traitement de l'Information Multimédia CITIM'2015 At : Université Mustapaha Stambouli de Mascara – Algérie, Mai 2015.

[Aïmeur, 2011] : Esma Aîmeur : *"Introduction à l'intelligence artificielle"*, IFT6261, Université de Montréal département d'informatique et de recherche opérationnelle Montréal, Canada, hiver 2011. (http://www.iro.umontreal.ca/~aimeur/cours/ift6261/Ch1-Intro-IA-IFT6261-H-11.pdf).

[Allala et Mouas, 2017] : ALLALA Nourelhouda et MOUAS Zeyneb : *"Intégration d'une application d'indexation dans un environnement cloud computing"*, Mémoire de Fin d'études Master, Juin 2017.

[Belkacem, 2016] : Thiziri BELKACEM : *"Apprentissage de représentations sémantiques des documents et leur exploitation en RI"*, Exploitation du Deep Learning en Recherche d'Information, Mémoire de Master, Univérssité de Toulouse, 28 Juin 2016.

[Belkacem et al., 2017] : Thiziri Belkacem, Taoufiq Dkaki, José G. Moreno et Mohand Boughanem : *"Apprentissage de représentations de documents et leur exploitation en recherche d'information"*, 14e Conférence francophone en Recherche d'Information et Applications (CORIA 2017), Mar 2017, Marseille, France. pp.1-10. ffhal-02559775f.

[Bellin et al., 2016] : ISABELLE BELLIN, NICOLAS VAYATIS, JULIEN AUDIFFREN, SERGIO PEIGNIER ET ALICE NICOLAI : *"Auto-encodeur simple"*, 2016.

[Benbakreti, 2019] : BENBAKRETI SAMIR : *"Développement d'un système de reconnaissance en ligne du manuscrit Arabe"*, Thèse de doctorat, Spécialité : Électronique option : traitement de signal et de l'image, 23/07/2019.

[Boucham, 2009] : Boucham Souhila : *"Une approche basée ontologies pour l'indexation automatique et la recherche d'information multilingue (RIM)"*, 2009.

[Bourne et Anderson, 1979] : C**.**Bourne, and B.Anderson : *"Dialog: Labworkbook. In Lockheed Information Systems"*, pp 640–644. PaloAlto, Californie (USA), 1979.

[Breuleux, 2009] : Olivier Breuleux : *"Échantillonnage dynamique de champs markoviens"*, Université de Montréal, Département d'informatique et de recherche opérationnelle Faculté des arts et des sciences, en vue de l'obtention du grade de Maître ès sciences (M.Sc.) en informatique, Novembre, 2009.

[Chaudiron et Ihadjadene, 2002] : Stéphane Chaudiron et Madjid Ihadjadene : *"Quelle place pour l'usager dans l'évaluation des SRI ?"*, Recherches récentes en sciences de l'information : Convergences et dynamiques, Mar 2002, Toulouse, France. pp.211-231. ffhal-00331993f.

[Coulom, 2002] : Rémi Coulom : *"Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur"*, Interface homme-machine [cs.HC]. Institut National Polytechnique de Grenoble - INPG, 2002. Français. fftel-00004386f.

[Dehghani et al., 2017] : Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W.Bruce Cro. : *"Neural Ranking Models with Weak Supervision"*, In Proceedings of

SIGIR ’17, Shinjuku, Tokyo, Japan, August 07-11, 2017, 10 pages. DOI: 10.1145/3077136.3080832, 2017.
[Dejasmin, 2018] : Julien Dejasmin : *“Deep learning ia réseau de neurones connectés”* filed under Intelligence artificielle and tagged intelligence artificielle science scientifiques startup, Posted on April 17, 2018.
[Denjean, 1989] : P. Denjean : *“Interrogation d’un système videotex : l’indexation automatique des textes”*, Thèse de doctorat, Université Paul Sabatier, Toulouse, 1989.
[El Hachani, 1997] : Mabrouka EL HACHANI : *“L’indexation automatique”*, DEA Sciences de l’information et communication Option 3 : Système d’Information documentaire, Ecole Nationale Supérieure des Sciences de l’Information et des Bibliothèques (ENSSIB), Mars 1997.
[Elyaleb, 2009] : ELAYEB, Bilel : *“SARIPOD : Système multi-Agent de Recherche Intelligente Possibiliste de Documents Web”*, Thèse de doctorat, Institut National Polytechnique de Toulouse, 2009.
[Fellag, 2018] : Berchiche-Fellag Samia : *“Recherche d’information dans les documents XML : modèle basé sur une propagation sélective des termes”*, Thèse de Doctorat en Informatique, 2018.
[Fluhr et Debili, 1985] : C. Flurh and F. Debili : *“Interrogation en Langue Naturelle de Données Textuelles et Factuelles”*, Intelligent Multimadia Information Systems and Management (RIAO), Grenoble (France), pp 548-556, 1985.
[Glenberg et Kaschak, 2002] : Arthur M Glenberg and Michael P Kaschak : *“Grounding language in action”*, Psychonomic bulletin & review, 9(3) :558–565, 2002.
[Gordon et Van Durme, 2013] : Jonathan Gordon and Benjamin Van Durme : *“Reporting bias and knowledge acquisition”*, In Proceedings of the 2013 workshop on Automated knowledge base construction. ACM, 2013.
[Haddi et Messabih, 2019] : Haddi Miloud et Messabih Alwalid : *“Localisation des concepts dans les traces d’exécution : Une approche basée sur LSI”*, Mémoire de master, Option : RISR, Univérsité SAIDA, Juin 2019.
[Hammache, 2013] : Hammache Arezki : *“Recherche d’information : un modèle de langue combinant mots simples et mots composés”*, Thèse de doctorat en informatique, 2013.
[Haugeland, 1985] : John Haugeland : *“Artificial Intelligence”*, The Very Idea Bradford Book, (MIT Press, Cambridge, MA, 1985) ; 287 pp.
[Hariom, 2019] : Hariom Gautam : *“Word Embedding”*, https://medium.com/@hari4om/word-embedding-d816f643140, feb 16 2019.
[Huang, 2008] : Huang, A. : *“Similarity Measures for Text Document Clustering”*, In Holland, J., Nicholas, A., and Brignoli, D., editors, New Zeal and Computer Science Research Student Conference, 2008.
[Huang et al., 2013] : Huang P.-S., He X., Gao J., Deng L., Acero A. and Heck L. : *“Learning deep structured semantic models for web search using clickthrough data”*, CIKM, 2013.
[Jaccard, 1901] : Jaccard, P. : *“Étude comparative de la distribution florale dans une portion des alpes et des jura”*, Bulletin de la Société Vaudoise des Sciences Naturelles, 37 :547-579, 1901.
[Jagerman et al., 2017] : R Jagerman, J Kiseleva and M de Rijke : *“Modeling label ambiguity for neural list-wise learning to rank”*, - arXiv preprint arXiv:1707.07493, 2017.
[Kalchbrenner et al., 2014] : Nal Kalchbrenner, Edward Grefenstette and Phil Blunsom : *“A Convolutional Neural Network for Modelling Sentences”*, Department of Computer Science University of Oxford .arXiv:1404.2188v1 [cs.CL], 8 Apr 2014.
[Lelu et François, 1992] : A.Lelu and C.François : *“Information retrieval based on a neural*

unsupervised extraction of thematic fussy clusters, neuro-nîmes 92 : les réseaux neuromimétiques et leurs applications", nîmes, France, 2-6 novembre 1992.
[Manning et al, 2008] : Christopher D. Manning , Prabhakar Raghavan et Hinrich Schütze : *"Introduction à la recherche d'informations"*, Cambridge University Press, 2008.
[Mechach, 2016] : MECHACH KHEIRA : *"Etude de l'impact des méthodes de localisation dans les systèmes d'information distribués"*, Juin 2016.
[Mikolov et al., 2013a] : Mikolov T., Chen K., Corrado G. and Dean J. : *"Efficient estimation of word representations in vector space"*, arXiv preprint arXiv :1301.3781, 2013a.
[Mikolov et al., 2013b] : Mikolov T., Le Q. V. and Sutskever I. : *"Exploiting similarities among languages for machine translation"*, arXiv preprint arXiv :1309.4168, 2013b.
[Mikolov et al., 2013c] : Mikolov T., Yih W.-t. and Zweig G. : "*Regularities in Continuous Space Word Representations*", HLT-NAACL, vol. 13, p. 746-751, 2013c.
[Msaaf et Belmajdoub, 2015] : Mohammed Msaaf et Fouad Belmajdoub : *"L'application des réseaux de neurone de type feedforward dans le diagnostic statique"*, Xème Conférence Internationale : Conception et Production Intégrées, Dec 2015, Tanger, Maroc. ffhal-01260830.
[Negre, 2013] : Elsa Negre : *"Comparaison de textes: quelques approches"*, 2013, ffhal-00874280f.
[Nguyen et al., 2017] : Nguyen Gia Hung, Soulier Laure, Tamine Lynda and Bricon-Souf, Nathalie : *"" Modèle Neuronal de Recherche d'Information Augmenté par une Ressource Sémantique.* (2017) In : 14eme Conference francophone en Recherche d'Information et Applications (CORIA 2017), 29 March 2017 - 31 March 2017 (Marseille, France).
[Pang et al., 2016] : Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu and Xueqi Cheng : *"A Study of MatchPyramid Models on Ad-hoc Retrieval"*, CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences, arXiv: 1606.04648v1 [cs.IR], 15 Jun 2016.
[Perwej, 2015] : Yusuf Perwej : *"International Journal of Advanced Research in Computer and Communication Engineering"*, Vol. 4, Issue 2, Copyright to IJARCCE DOI 10.17148/IJARCCE.2015.4203 10 An Evaluation of Deep Learning Miniature Concerning in Soft Computing, February 2015.
[Porter, 1980] : M.F Porter : *"An algorithm for Suffix Stripping"*, Program Vol.14, N.3, PP130-137, http://www.muscat.com/martin/stem.html, 1980.
[Pottiez et Duquesnoy, 2018] : Aurélien Pottiez et Marc Duquesnoy : *"Recherche Intelligence artificielle et apprentissage de réseaux connexionnistes"*, Université de Lille, 11 mai 2018.
[Quiza et Davim, 2009] : Ramón QUIZA and J. Paulo DAVIM, *"Chapter of Computational modeling of machining systems Chapter"*, January 2009.
[Rainero, 2017] : Pierre Rainero : *" Deep Belief Networks (DBN) - RBM avec momentum"*, AGH - Semestre d'hiver Deep learning, https://www.pierre-rainero.fr/lessons/235, 03/12/2017.
[Robertson et Walker, 1999] : S.E.Robertson and S.Walker : *"Okapi/keenbow at TREC-8"*, In Proceedings of the 8th Text REtrival Conference (TREC-8). NIST, Gaithersburg, Maryland, 1999.
[Rosenblatt,1957] : F. Rosenblatt : *"The Perceptron: A Perceiving and Recognizing Automaton"*, Report 85-60-1, Cornell Aeronautical Laboratory, Buffalo, New York, 1957.
[Salton, 1971] : Gerald Salton : *"The SMART Information retrieval system"*, Prentice-Hall, Englewood Cliffs, 1971.
[Severyn et Moschitti, 2015] : Aliaksei Severyn and Alessandro Moschitti : *"Learning to rank short text pairs with convolutional deep neural networks"*, In Proceedings of the 38th

International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 373_382. ACM, 2015.
[Smolensky, 1986] : Paul Smolensky : *"Information processing in dynamical systems : Foundations of harmony theory"*, University of Colorado at older DEPARTMENT OF COMPUTER SCIENCE, January 1986. Dans D. E. Rumelhart et J. L. McClelland, éditeurs, Parallel Distributed Processing, volume 1, chaptitre 6, pages 194_281. MIT Press, Cambridge, 1986.
[Sriti, 2012] : Ali SRITI : *"La recherche d'informations : Sémantique d'informations dans le cadre du web sémantique"*, Université Mohamed Khider Biskra, 25 Juin 2012.
[Strehl et al., 2000] : Strehl, A., Ghosh, J., and Mooney, R. : *" Impact of Similarity Measures on Web-page Clustering"*, In Proceedings of the 17th National Conference on Artificial Intelligence : Workshop of Artificial Intelligence for Web Search (AAAI 2000), Austin, Texas, USA, 30-31 July 2000.
[Stricker, 2000] : Mathieu Stricker : *"Apprentissage des réseaux de neurones et régularisation"*, l'ESPCI, (chap6) Theses_PS, www.neurones.espci.fr/Theses_PS/Stricker_M/ CHAP6.pdf, 2000.
[Touzet, 1992] : Claude Touzet : *"Les réseaux de neurones artificiels, introduction au connexionnisme"*, cours, exercices et travaux pratiques. EC2, 1992, Collection de l'EERIE, N. Giambiasi. ffhal-01338010f.
[Trouvilliez, 2013] : Trouvilliez Benoît : *"Similarité de données textuelles pour l'apprentissage de textes courts d'opinions et la recherche de produits"*, Pour obtenir le grade de docteur de l'Université d'Artois, École doctorale : Sciences Pour l'Ingénieur Université Lille Nord-de-France no 72, Unité de recherche : Centre de Recherche en Informatique de Lens, 13 mai 2013.
[Turing, 1950] : Alan Turing : *"Computing machinery and intelligence"*, *Mind* (en), Oxford University Press, vol. 59, n° 236,) DOI 10.1093/mind/LIX.236.433), octobre 1950.
[Varga et al., 2008] : Varga, Tamás, and Horst Bunke : *"Machine Learning in Document Analysis and Recognition"*, Edited by Simone Marinai and Hiromichi Fujisawa. Vol. 90. Studies in Computational Intelligence, Berlin, Heidelberg : Springer Berlin Heidelberg, 2008.
[Vukotic et al., 2015] : Vukotic Vedran, Raymond Christian and Gravier Guillaume : *"Is it time to Switch to word embedding and recurrent neural networks for spoken language understanding?"*, In *INTERSPEECH-2015*, 130-134, 2015.
[Werfelli, 2015] : Oussama Werfelli : *"Présentation générale des réseaux de neurones artificiels"*, Publié le 7 nov. 2015.
[Wong et al, 1985] : S. Wong, W. Ziarko and P. Wong : *" Generalized vector space model information retrieval"*, In proceeding of the annual international ACM SIGIR Conference on Research and development in information retrieval, pp 18-25, ACM, 1985.
[Yang et al., 2016] : Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft : *" anmm: Ranking short answer texts with attention-based neural matching model"*, In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pages 287–296. ACM, 2016.
[Zamani et al., 2017] : Hamed Zamani, Michael Bendersky, Xuanhui Wang and Mingyang Zhang : *" Situational Context for Ranking in Personal Search"*, Center for Intelligent Information Retrieval, University of Massachusetts Amherst, MA 01003, Google Inc., Mountain View, CA 94043, 2017.
[Zipf, 1949] : Zipf G. : *" Human Behaviour and the Principle of Least Effort"*, 1st edition, Addison Wesley, 1949.
[Zobel et Moffat, 2006] : ZOBEL JUSTIN and MOFFAT ALISTAIR : *" Inverted Files for Text Search Engines,ACM Computing Surveys"*, Vol. 38, No. 2, Article 6, Publication date: July 2006.