REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie
———————
UNIVERSITE DE YAOUDE I
Faculté des Sciences
———————
CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIE ET GEOSCIENCES
———————
UNITE DE RECHERCHE ET DE FORMATION DOCTORALE EN MATHEMATIQUES, INFORMATIQUE, BIOINFORMATIQUE ET APPLICATIONS
———————

REPUBLIC OF CAMEROON
Peace – Work – Fatherland
———————
UNIVERSITY OF YAOUNDE I
Faculty of Science
———————
POSTGRADUATE SCHOOL OF SCIENCE, TECHNOLOGY & GEOSCIENCES
———————
RESEARCH & TRAINING UNIT FOR DOCTORATE IN MATHEMATICS, COMPUTER SCIENCE AND APPLICATIONS
———————

LABORATOIRE D'INFORMATIQUE ET APPLICATIONS
LABORATORY OF COMPUTER SCIENCE AND APPLICATIONS

EXPLAINABLE DEEP NEURAL NETWORK FOR SKILLS PREDICTION FROM RESUMES

PhD Thesis
A dissertation submitted in partial fulfillment of the requirements for the PhD Degree in the Department of Computer Science at University of Yaounde I.

By:
JIECHIEU KAMENI FLORENTIN FLAMBEAU
Registration N°: 09U0095

Supervisor:
Professor Maurice TCHUENTE

Director:
Doctor Norbert TSOPZE

2021

Explainable Deep Neural Network for Skills Prediction from Resumes

PhD thesis

Jiechieu Kameni Florentin Flambeau

A dissertation submitted in partial fulfillment of the requirements for the PhD Degree in the Department of Computer Science at University of Yaounde I.

Supervisor : **Pr. Maurice TCHUENTE**
Director : **Dr. Norbert TSOPZE**

UNIVERSITE DE YAOUNDE I

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

BP : 812 YAOUNDE
Email : deptinfo@uy1.uninet.cm

UNIVERSITY OF YAOUNDE I

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BP : 812 YAOUNDE
Email : deptinfo@uy1.uninet.cm

ATTESTATION DE CORRECTION DE LA THESE

Je soussigné, **Laure Pauline FOTSO**, Professeure, Présidente du jury de la soutenance de thèse de doctorat/PhD en informatique soutenue par Monsieur **JIECHIEU KAMENI FLORENTIN FLAMBEAU**, Matricule **09U0095**, atteste que la thèse intitulée « **Explainable Deep Neural Network for Skills Prediction from Resumes** », présentée par le candidat a été corrigée conformément aux recommandations des membres du jury.

En foi de quoi la présente attestation lui est établie et délivrée pour servir et valoir ce que de droit.

Présidente : FOTSO Laure Pauline, Professeure, Université de Yaoundé I

Rapporteurs :

TCHUENTE Maurice, Professeur, Université de Yaoundé I

TSOPZE Norbert, Chargé de Cours, Université de Yaoundé I

Membres :

MEPHU NGUIFO Engelbert, Professeur, Université Clermont Auvergne, France

BOUETOU BOUETOU Thomas, Professeur, Université de Yaoundé I

NDOUNDAM Réné, Maître de Conférences, Université de Yaoundé I

KOUAMOU Georges-Edouard, Maître de Conférences, Université de Yaoundé I

Dedicaces

I dedicate this thesis *to my late grandmother **DJAMKOU Elisabeth**, in recognition of the encouragement, the guidance and the education she provided to me.*

Acknowledgements

I would like to thank all those who have contributed to the accomplishment of this work.

- ☞ First of all, I would like to thank Professor **Maurice Tchuente** for agreeing to supervise this work and for his availability and for his follow-up;
- ☞ I would like to thank my thesis director, **Dr. Norbert Tsopze**, lecturer in the Department of Computer Science of the University of Yaounde I for his availability, his advice, his critics and the meticulous follow-up he provided throughout my doctoral studies;
- ☞ I would like to thank the Head of the Department of Computer Science, **Dr. Aminou Halidou**, for setting up the appropriate conditions of study and research within the Department;
- ☞ I would like to thank all the former heads of the Department of Computer Science namely Professor Atsa, Professor Emvudu, and Doctor Louka for having set adequate conditions of study in the Computer Science Department which have conferred me the required knowledge needed to fullfill the various stages of my studies including the writing of this thesis;
- ☞ I would like to express my sincere thanks to all my teachers of the Department of Computer Science of the University of Yaounde I, and all those of the Department of Computer Engineering of the National Advanced School of Engineering Yaounde (ENSPY) for the quality of the training I received from them, which was of an undeniable contribution in achieving the results obtained during this work;
- ☞ I would like to thank the members of the Mathematics, Informatics, Bioinformatics and Applications (MIBA) research unit for their significant contributions during the doctoral seminars;
- ☞ I thank all the members of the IDASCO research team, for all the remarks, comments and criticics made during the seminars and which have help us improving the quality of this thesis;
- ☞ Special thanks to Kevin Jiokeng, Franck Quentin Nfotabong, Billy Zafack, Christelle Niguieu, Damien Goumai and Nourridine Siewe for their important contribution in proofreading our articles;
- ☞ I thank my colleagues and friends from the Department of Computer Science of the University of Yaoundé I and those from the National Advanced School of Engineering Yaoundé (Adrien Atibita, Audric Feuyan, Alex Djouontse, Armel Nzekon, Barnabas Kouami, Boris Tegantchouang, Cavour Tadjouteu, Chamir Feutcheping, Darius Tetsa, Edward Nanda, Evaris Fomekong, Francis Yuya, Leatitia Mouafo, Loic Youmbi, Mike Chirmeni, Samuel Nyobe, Stephane Tindjou, Steve Tueno, Reine Mouafo, Yacynth Ndonna, ...) for their support and encouragement;
- ☞ Finally I would like to thank very warmly my family members: My parents Kameni Ngoudjo Pierre and Tchiaga Rose, My brothers and sisters Ngoudjo Wilfried, Noubissie Sylvain, Siewe Audrey, Djamkou Reine for their unconditional support and encouragement for the writing of this thesis.

Remerciements

Je tiens à remercier toutes les personnes qui de près ou de loin ont consenti un effort dans l'accomplissement de ce travail.

- ☞ Tout d'abord, j'adresse mes remerciements au Professeur **Maurice Tchuente** pour avoir accepté de superviser ce travail et pour sa disponibilité et son suivi;
- ☞ Je tiens à remercier vivement mon directeur de thèse le Docteur **Norbert Tsopze**, enseignant chercheur au département informatique de l'Université de Yaoundé I pour sa disponibilité, ses conseils, ses critiques et le suivi minutieux qu'il a apporté tout au long de cette étude;
- ☞ Je remercie le Chef de Département d'Informatique, le Docteur Aminou Halidou pour la mise en place d'un cadre propice à la recherche au sein du département d'informatique de l'Université de Yaounde I;
- ☞ Je remercie les Chefs de Département de la filière Informatique: les Professeurs Atsa, Emvudu, et le Docteur Louka pour avoir chacun en sa position défini un cadre adéquat pour la conduite des études au sein du Département d'Informatique; lequel cadre m'aura permis de terminer les différentes étapes de ma formation;
- ☞ J'adresse de vifs remerciements à tous mes enseignants du département d'informatique de l'Université de Yaoundé I et à ceux du département de Génie Informatique de l'Ecole Nationale Supérieure Polytechnique de Yaoundé pour la qualité de la formation dont j'ai benéficiée de leur part et qui a été d'une contribution indéniable pour l'atteinte des résultats obtenus durant ces travaux;
- ☞ Je remercie les membres de l'unité de recherhe Mathématiques, Informatiques, Bioinformatiques et Application (MIBA) pour leurs contributions significatives lors des doctoriales;
- ☞ Je remercie tous les membres de l'équipe de recherche IDASCO, pour toutes les remarques, commentaires et critiques lors des séminaires et qui m'ont permis d'améliorer la qualité de ces travaux;
- ☞ Un merci particulier à l'endroit de Kevin Jiokeng, Franck Quentin Fotabong, Billy Zafack, Nourridine Siewe, Damien Goumai et Christelle Niguieu pour avoir relu mes articles;
- ☞ Je remercie mes camarades et amis du Département d'Informatique de l'Université de Yaoundé I et ceux de l'Ecole Nationale Supérieure Polytechnique de Yaoundé nommément: Adrien Atibita, Audric Feuyan, Alex Djouontse, Armel Nzekon, Barnabas Kouami, Boris Tegantchouang, Cavour Tadjouteu, Chamir Feutcheping, Darius Tetsa, Edouard Nanda, Evaris Fomekong, Felicité Gamgne, Francis Yuya, Leatitia Mouafo, Loic Youmbi, Mike Chirmeni, Samuel Nyobe, Stephane Tindjou, Steve Tueno, Reine Mouafo, Yancynth Ndonna, ... pour leurs soutiens et encouragements;
- ☞ Enfin je remercie grandement les membres de ma famille : Mes parents Kameni Ngoudjo Pierre et Tchiaga Rose, Mes frères et soeurs Ngoudjo Wilfried, Noumbissie Sylvain, Siewe Audrey, Djamkou Reine pour les soutiens multiformes et encouragements.

Abstract

The automatic identification of skills in text documents (CVs, job offers, articles, etc.) is a task of Natural Language Processing (NLP) that finds its application in the construction of job recommender systems or in the automatic identification of the professional qualifications of researchers, employees or job seekers; this, in order to bridge the "skills gap" that the ATD (Association for Talent Development) defines as a significant gap between the skills held by an organization's human resources and those it needs for its development.

Several researchers have proposed methods for automatic identification of skills in text documents. But those methods, for some of them, only allow the identification of skills explicitly mentioned in the documents, and for others lack of explainability.

The main objective of this thesis is to extend the scope of existing approaches by proposing an artificial intelligence model based on Convolutional Neural Networks (CNN) capable of identifying in resumes, a set of skills that we will refer to as high-level skills insofar as they can be explained by more basic skills. High level skills as perceived in this work are generally professional qualifications such as "web developer", "programmer analyst", "network administrator", and so forth. In addition, we propose to explain the decisions of the CNN model by illustrating the high-level skills predicted by the terms contained in the resume and that characterize those skills.

The first contribution of this thesis is thus, the design of a multi-label classification architecture based on CNN and which use the "binary relevance" approach to multi-label classification of resumes according to skills they contain. Input resumes are transformed into matrices using a self-trained word embedding model and the resulting matrices are submitted as inputs to the CNN. Experiments carried out on a corpus of 30,000 IT resumes collected from the Internet have demonstrated the effectiveness of the model which reaches 98.79% of recall and 91.34% of precision.

The second major contribution is at the level of the explainability of the CNN models. Globally, we propose a method to explain the predictions of CNN models built for any text classification problem. More precisely, we describe a method based on the principle of the LRP (Layer-wise Relevance Backpropagation) algorithm to compute the contributions of the terms selected by the convolution filters to the values predicted by the model. In addition, we show the limitations of the base LRP method and propose an adaptation of the formula that computes the contributions of input features. Finally, propose to identify sufficient and necessary features to simplify the explanations provided to users.

The distribution of the relevance obtained with our explanation method is similar to that of LIME, a well-known state of the art model; and the evaluation of the complexity of both methods shows that ours is significantly better than LIME. Furthermore, we show how LIME sometimes assigns a score to terms that have no influence on the output. However, LIME has the advantage to be model-agnostic.

Keywords: *Skills-Gap, Resume, Skills Identification, Multi-label Classification, Convolutional Neural Network, Explainable Artificial Intelligence.*

Résumé

L'identification automatique des compétences dans les documents textes (CV, offres d'emploi, articles, etc) est une tâche du traitement automatique du langage naturel qui trouve son application dans la construction des systèmes de recommandation des offres d'emploi ou encore dans l'identification automatique des qualifications professionelles des chercheurs, employers ou demandeurs d'emploi; ceci dans la perspective de combler le "skills gap" que l'ATD (Association for Talent Development) définit comme étant l'écart entre les compétences détenues par la resource humaine d'une organisation et celles dont elle a besoin pour son développement.

Plusieurs chercheurs ont proposé des méthodes pour identifier les compétences dans les documents textes. Mais ces méthodes pour certaines, ne permettent d'identifier que des compétences explicitement mentionnées dans les documents, et pour d'autres ne sont pas explicables.

L'objectif de cette thèse est de concevoir un modèle d'intelligence artificielle à base de Réseaux de Neurones Convolutifs (RNC) capable d'identifier un ensemble de compétences que nous qualifierons de compétences de haut niveau dans la mesure où elles peuvent s'expliquer par des compétences plus basiques. Les compétences de haut niveau telles que perçues dans ce travail sont généralement des qualifications professionnelles comme "Administrateur réseau", "Gestionnaire de projet", "Dévelopeur web", etc.

La première contribution de cette thèse est donc la conception d'une architecture de classification multi-étiquette basée sur les RNC et utilisant l'approche "binary relevance" pour prédire les compétences à partir des CV. Les CV en entrée du modèle sont transformés en matrices en utilisant un modèle de "word embedding" construit par nous mêmes et la matrice obtenue est soumise au RNC. Les expérimentations effectuées sur un corpus de 30000 CV d'informaticiens collectés et étiquetés automatiquement ont permis de démontrer l'effectivité de la méthode qui atteint 98,79% de rappel et 91,34% de précision.

La deuxième contribution majeure se situe au niveau de l'explicabilité des modèles de RNC. Globalement, nous proposons une méthode permettant d'expliquer les predictions des modèles de RNC construits pour tout problème de classification de texte. Plus précisément, nous décrivons une méthode basée sur le principe de l'algorithme LRP (Layer-wise Relevance Backpropagation) et permettant de calculer les contributions des termes selectionnés par les filtres convolutifs aux valeurs prédites en sortie du modèle. En outre, nous mettons en évidence les limites de la méthode LRP de base et proposons une adapation de la formule de calcul des contributions. Enfin, nous proposons d'identifier les n-grams suffisants et les n-grams nécessaires afin de simplifier l'explication à fournir aux utilisateurs du modèle.

La distribution des pertinences obtenues avec notre méthode est semblable à celle de LIME, un modèle de l'état de l'art très connu; et l'évaluation de la complexité des deux méthodes montre que la nôtre est nettement meilleure que celle de LIME. De plus, nous démontrons comment LIME attribue un score à des termes qui n'ont pourtant pas d'influence sur la sortie. Toutefois, LIME a l'avantage de s'appliquer indépendamment de la nature du modèle.

Mots clés: *Déficit de Compétence, CV, Identification des Compétences, Classification Multi-étiquette, Réseaux de Neurones Convolutifs, Intelligence Artificielle Explicable.*

Contents

List of Figures

List of Tables

Abbreviations

1D-CNN One-Dimensional Convolutional Neural Network. 33, 52, 103

AI Artificial Intelligence. 74

ANN Artificial Neural Network. 25, 26, 29, 46

ATD Association for Talent Development. 1, 118

BI-GRU Bidirectional Gated Recurrent Unit. 34

BI-LSTM Bidirectional Long Short-Term First. 34

BR Binary Relevance. 47, 48

CBOW Continuous Bag Of Words. 22, 24

CC Chain of Classifiers. 48, 67

CNN Convolutional Neural Network. 3, 4, 31, 33, 35

DNN Deep Neural Network. 25, 26, 28, 35, 69, 72

GRU Gated Recurrent Unit. 34

HTML HyperText Markup Language. 54

IDF Inverted Document Frequency. xii, 18, 19

k-NN k-Nearest Neighbours. 46

LIME Local Interpretable Model-Agnostic Explanations. 76

LP Label Powerset. 47

LRP Layer-wise Relevance Propagation. 76, 79

LSTM Long Short-Term First. 20, 34

MLP Multi-Layer Perceptron. 29

NEN Named Entity Normalisation. 43

NER Named Entity Recognition. 43, 45

NL Natural Language. 34, 40, 69, 72

NLG Natural Language Generation. 6

NLP Natural Language Processing. 5, 6, 8, 13, 15, 16, 22, 26

NLU Natural Language Understanding. 5, 6

NN Neural Network. 34

POS Part Of Speech Tagging. 45

QA Question Aswering. 4, 89

ReLU Rectified Linear Unit. 28

RNC Réseau de Neurones Convolutifs. v

RNN Recurrent Neural Network. x, 20, 33–35

SA Sentitiment Analysis. 4, 89

SGD Stochastic Gradient Descent. 35, 36

SVM Support Vector machine. 45, 46

TF Term Frequency. xii, 18

TF-IDF Term Frequency - Inverted Document Frequency. 17, 18

XAI Explainable Artificial Intelligence. 69, 73–75

INTRODUCTION

Context and Motivation

The Association for Talent Development (ATD) defines the “skills gap” as a significant gap between an organization’s current human capabilities and the skills it needs to achieve its goals and meet customer demand [1]. While more and more graduate students are unemployed, employers sometimes complain not being able to find appropriate human resources to perform critical tasks.

The US Economic Center of Research reported in a survey realized in 2018 that the “skills gap” costs about 160$ billion to companies a year in US [2]. Moreover, CarrierBuilder[1] reported in a study carried out in 2017 that 60% of employers have jobs that remain vacant for more than two weeks [2].

In Cameroon, the World Bank in a study realized in 2016 indicates that in order to achieve a real structural transformation, the country must identify the “skills gap” in the areas of technology and innovation [3]. The international institution argues that to end poverty among young people, it is necessary to bridge the “skills gap” [3]. To achieve this goal, it is necessary to be able to formally define the “skills gap” by identifying the skills held by young people and comparing them with those required by companies. Organizations bridge the “skills gap” by hiring candidates with specific skills to perform critical tasks.

Nowadays, hiring processes are often conducted through the Internet. Applications (resumes and cover letters) are sent via e-mails, web sites or job posting platforms (indeed.com, freelancer.com, upwork.com, ...). The number of applications for a particular job can be very important, making the candidates selection cumbersome.

To ease the recruitment process, job recommender systems also known as job-matching systems have emmerged and enable recruiters to quickly find suitable candidates for a particular job. Traditional job-matching algorithms required user input[13, 16, 27], that is, users should specify their skills as lists of keywords; as well, skills needed to perform the job were also listed. In that

[1] CarrierBuilder is an online hiring plateforme accessible at
https://hiring.careerbuilder.com/?_ga=2.130206863.1209475669.1606937649-429933661.1606937649

case, the matching consisted in computing a similarity score between the candidate's skills and those required by the recruiter, and ranking the candidates according to their scores.

However, things become more challenging when dealing with raw text resumes where it is difficult to distinguish between skill-terms from non skill-terms. In that case, a prior work generally consists in extracting topic words (skills in our context) from resumes before matching them to those extracted from jobs. By the way, it has been shown that using topic words in content-based recommendation is more accurate than using any other word [4].

A lot of works have been done to tackle the problem of skills extraction from resumes. Some researchers have proposed the use of ontology to identify skills from resumes [5, 6], while others have suggested to handle the skills extraction problem as a named entity recognition (NER) problem where skills are considered as named entities [7, 8]. Moreover, skills can be organized into different levels of abstraction. There are low-level skills which can be considered as basic skills (e.g., css, html, php, ...) and high-level ones which can be characterized by a set of low-levels skills(e.g., web developer, front-end developer, ...). A huge work has already been done by some researchers to build skills taxonomies [9, 10] which represent the hierarchical relationship between skills of several domains including IT.

While the majority of existing skills extraction methods [8, 11, 7] already perform remarkably well when extracting skill-related terms from resumes, they are limited by their inability to infer high-level skills not explicitly mentioned in these resumes. In IT, for example, the observation of html, css, javascript, etc. could lead a recruiter to deduce that the candidate is a front-end developer even if this was not explicitly mentioned in his resume. On the other hand, a recruiter could doubt that a candidate has a certain competence despite the fact that it has been mentioned in his resume; just because there were no sufficient elements to convince him that the candidate truly has that expertise.

Artificial intelligence models have evolved over the years and offer excellent performances in solving varieties of problems in different fields (job recommendations, image processing, word processing, medical analysis, etc.). Though these models are powerful in terms of the accuracy of the predictions, they often suffer from the problem of opacity: in fact, they operate more like "black boxes" providing results without being able to explain them. However, entrusting important decisions to a system that cannot be explained presents an obvious danger [12]. This is the main reason why the construction of explainable artificial intelligence models has been gaining increasing attention in recent years.

Moreover, Deep learning models especially Convolutional Neural Networks (CNN) has shown surprising results in text processing, achieving states of the art results in many classification tasks. One advantage of CNN is that they are able to automatically discover features from inputs to perform classification. This property could be used to automatically identify terms (low-level skills) describing high-level skills. In addition, many research works have been conducted in the sense of analyzing CNN in order to explain their predictions in text classification [13, 14].

Problem Statement

Let's suppose that a recruiter is looking for a candidate who masters structural programming. But, the skill structural programming is not explicitly mentioned in the candidate's resume but instead C, ada, Pascal which, according to the taxonomy elaborated by Faliagka et al. in [9] are characteristics of structural programming. Explicit skills extraction methods cannot identify those skills which are not explicitly referenced in the resume description.

Only few researchers have dealt with the identification of implicit skills, but none of them has considered the need for model explainability which could make the recruiter have trust in the overall system decisions.

The problem solved in this thesis can therefore be stated as follows : ***Given a raw text resume written in natural language, what are the set of high level skills that the resume expresses and how to convince a human that the resume actually expresses those skills ?***

Indeed, the fact that a resume can express multiple competences at the same time makes the skills prediction from resumes a multi-label classification problem where classes are skills. Therefore, solving this problem will consist in designing an accurate model capable of predicting a set of skills that a resume expresses, and derive a method to justify the predictions provided.

Objectives and Methodology

Considering the great performances of Convolutional Neural Network (CNN) in text classification [15] and the works done by Jacovi et al. [13] to explain CNN models built for text classification, we think that CNN could be the ideal classifier which could help solving the problem stated above. The aim of this study is then to build an accurate and explainable multi-label classifier based on CNN to predict competences from resumes. Explaining the model prediction will consist in high-

lighting the basic skills responsible for the model decisions. We believe that, such approach to skills prediction may be trustful to recruiters as they are provided with evidence which may justified the competences predicted for each resume.

The methodology that will be used to solve the problem involves the following steps:

- First we design a multi-label classification architecture for resume classification using CNN as based classifiers. At this level, the architecture is first described with its various components, then the methods used to preprocess resumes are explained;
- Second we describe a novel method to explain the predictions of the overall classification architecture. The explanation method described is applied on other text classification problems including Sentitiment Analysis (SA) and Question Aswering (QA) to assess its genericity. And thereafter, it is applied to the skills prediction model.

Contribution and reading guide

The main contribution of this thesis is the proposition of a deeplearning based method to identify high-level skills from resumes. Moreover, in order to justify the skills identified, we also proposed a general purpose method to explain CNN architectures built for text classification[15].

The rest of this thesis is organized as follows : chapter 1 presents the theoretical foundations of the overall thesis along with the concepts, methods and techniques used to process natural languages. The next two chapters will be dedicated to the explanation of the two major aspects of the methodology defined so far : the skills prediction aspect (chapter 2) and the model explanation aspect (chapter 3). Finally, the conclusions drawn from the overall study will be stated and future works outlined.

CHAPTER 1

LITERATURE REVIEW

The aim of this chapter is to provide the reader with theoretical background on the domain of the study which is Natural Language Processing (NLP). In this chapter, we described the various principle, techniques, methods and tools used to process and understand the natural language.

1.1 Natural Language Processing

Before describing what Natural Language Processing (NLP) is, let's start with the definition of the underlying concepts.

Definition 1. Natural language is the language used by humans to express their feelings or thoughts[16]. Sources of natural language include but are not limited to book contents, tweets, text documents, voice record, etc. Natural language is used in human communication, and examples of natural language include English, French, Bulu, Yemba, Bassa, Fe-Fe, ...

Definition 2. Natural Language Processing (NLP) is a subfield of artificial intelligence and computational linguistics devoted to make computers understand the statements or words written in human languages [17]. In particular, NLP is concerned with how to build sophisticated machine programs capable of processing and analyzing large amounts of natural language data.

NLP brings together all the methods, techniques and tools aiming at automatically producing and understanding linguistic statements for the purpose of communication.

NLP consists of two major components (Figure 1.1) :

- **The first component is Natural Language Understanding (NLU)** which goal is to confer to computers the capabilities to understand inputs made in the form of sentences in text format or speech format. NLU directly makes possible the interaction between humans and computers ;

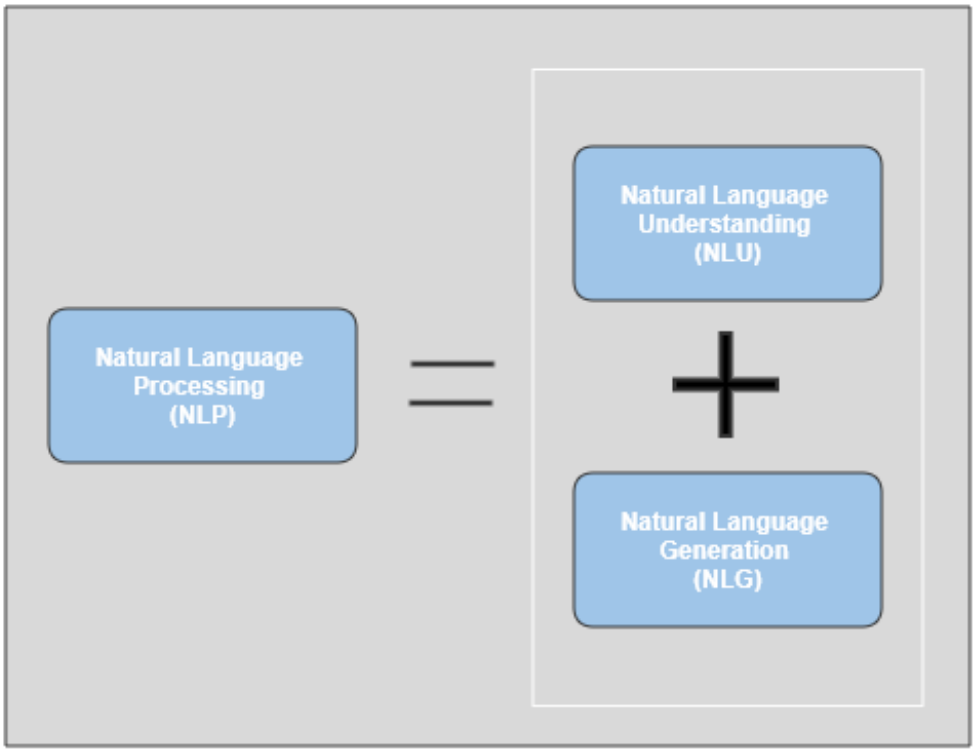


Figure 1.1: Components of NLP

- **The second component is Natural Language Generation (NLG)** which is concerned with the automatic production of syntactically and semantically correct linguistic statements.

In this chapter, we will mainly focus on NLU and its different tasks as the work developed in this thesis falls into this area.

Natural language understanding is considered as the first component of NLP. NLU involves various types of analysis at different levels of abstraction on natural language in order to convert it into useful representations and make the machine understand the underlying meaning of words, phrases, sentences or texts. Automatic language understanding involves the following analyses : Morphological analysis, Lexical analysis, Syntactic analysis, Semantic analysis, Handling ambiguity, Discourse integration, and Pragmatic analysis.

1.1.1 Morphological analysis

To understand morphological analysis, we must first define the main concepts involved in this type of analysis. Let's then start with some useful definitions.

Definition 3. Morphology is the branch of linguistic that studies the internal structure of words

and how they are formed [18].

Definition 4. A word is the smallest element of a sentence that bears meaning [16].

Definition 5. A morpheme is the smallest linguistic unit with a grammatical function in a given language [18].

While a word could be made-up of multiple morphemes, a morpheme cannot be further divided (e.g. “come”, “in”, “-ing”, . . .). The main difference between a morpheme and a word is that a morpheme sometimes does not stand alone while a word always stands alone. e.g. The word “boys” consists of two morphemes : “boy” and “-s”.

Table 1.1 shows the difference between a word and a morpheme.

Table 1.1: Morpheme versus word

Morpheme	**Word**
Morphemes can or cannot stand alone	Words always stand alone
Morphemes can be part of a word	A word consists of one or multiple morphemes
Morphemes can be affixes. When a morpheme stands alone it is called root	A word can be split into affixes and root

Definition 6. Morphological analysis refers to the grammatical analysis of how words are formed using morphemes. Morphological analysis is used in word segmentation, and Part Of Speech (POS) tagging.

Definition 7. A stem can be defined as a root to which an affix can be added [16]. E.g. the root “dog” is also a stem because, even though it contains no affix, an affix could be added to it to form a word, for example : “dog-s”. Every stem is a root but not all roots are stem. E.g. “of”, “or”, “I”, etc. are root but not stems because an affix cannot be added to any of them.

Example : Let’s consider the word “unrelated”. This word contains three morphemes : “un-”, “relat” and “-ed”. “un-” and “-ed” are the affixes (“un-” is the prefix and “-ed” is the suffix), “relat” is a stem.

Understanding the concepts of morphological analysis (root, word, morpheme, stem) is a step required to deal with tasks involved in lexical analysis such as tokenization, lemmatization and stemming.

1.1.2 Lexical analysis

Lexical analysis refers to the process of breaking down a text into words, phrases, and other meaningful elements [16]. Lexicon means the collection of words or phrases used in a given language [19].

Lexical analysis process involves : Tokenization, Lemmatization, Stemming, Part of speech tagging.

Tokenization

A Sentence consists of sequences of words and from a sentence we need to derive individual meaningful chunks which are called tokens and the process of deriving those tokens is called tokenization. Depending on the NLP problem we want to solve, the tokenization process can be done at different level of abstraction : paragraph-level, sentence-level or word-level.

If the analysis is at the word-level, then word tokenization should be applied, otherwise, if the analysis is at the sentence-level, then sentence tokenization and eventually word tokenization should be applied. A word-level tokenization typically consists in splitting a text based on punctuations and blank characters (spaces or tabulations).

> Example 2, tokenizing the sentence, "A pen costs FCFA 100 in Cameroon. Please buy me two of them. Thanks." may give the following tokens : ['A', 'pen', 'cost', 'FCFA', '100', 'in', 'Cameroon', '.', 'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']

Lemmatization and Stemming

Documents generally use different grammatical variations of the same word e.g. capital, capitals, capitalization. Those variations of the same word have similar meanings, and we may miss that if they are considered as different words. Another objective of stemming and lemmatization is to reduce inflectional or derivational related forms of a word to a common base form [20].

> Example of lemmatization : has, had, having -> have; are, is, am -> be.

While both stemming and lemmatization aim at representing the different forms of a word with a common base form, they both differ in the way they operate.

Stemming refers to an heuristic process that removes the end of words. The base form of the word resulting of a stemming operation is called a stem.

Most of the time, a stem is not identical to the morphological root form of the word. Stems may not be correct words of the language, but it is usually sufficient that related words map to the same stem, even if the stem itself is not a valid root word.

> For example, A stemming algorithm might reduce the words “fishing”, “fished”, and “fisher” to the stem “fish”.

J. Lovins was the first to create a stemmer algorithm in 1968 [20]. In 1980 Martin Porter wrote a stemmer (the porter stemmer [21]) which became a standard and one of the most famous stemming algorithm used for English language. Paice also proposed another stemming algorithm in 1990 [22].

> The Porter stemming algorithm will reduce “arguing”, “argued”, “argues”, “argus”, “argue” to the stem “argu”. As mentioned earlier, the stem “argu” is not a valid English word.

Unlike stemming, lemmatization tries do things properly by using a vocabulary and morphological analysis of words. The common base form obtained after lemmatization is a valid word in the language and is called the lemma.

Stemming applied to the token “saw” will return “s” while lemmatization will return a valid root word such as “see” or “saw” depending whether the token was used in the sentence as a noun or a verb. As well, applying lemmatization on the word “better” will give “good”, but we will lose this relation with a stemming algorithm which will still return “better”.

Stemming algorithms generally operate much more faster than lemmatizing algorithms as the latter ensure that the lemma obtained is a valid word by looking through a huge language dictionary/lexicon. Examples of lemmatizing algorithms include wordnet lemmatization [23] which uses wordnet lexicon as dictionary, and [24] a rule based approach to word lemmatization.

Part of speech Tagging

In corpus linguistics, part-of-speech tagging (POS tagging) is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech based on its definition and its context [16]. In other words, POS tagging is concerned with the identification of words that act as nouns, verbs, adverbs, prepositions, adjectives, etc.

Identifying POS tags is not an easy task because the same word may have a different part-of-speech tag in different contexts. Therefore it will be difficult if not impossible to have a generic

mapping scheme for POS tags. In addition, the manual identification of the POS tags of all the words in a huge corpus of documents is cumbersome. For that reason, machine-based POS tagging will be preferred to automatically tag words of a corpus of documents.

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	$
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	#
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Figure 1.2: List of 45-POS tags for english language (Source : [25]).

Figure 1.2 shows a list of 45-tags for the English language elaborated by Marcus et al. [25] and which has been used to label many English language corpora till now. POS tagging generally consists in placing the tag after each word. When applying stanford POS tagging[26] on the sentence of example 2 above we obtain the following :

> A/PRP pen/NN costs/VBP FCFA/NN 100/CD in/IN Cameroon/NN ./. Please/UH buy/VB, me/PRP, ./. two/CD of/IN them/PRP ./. Thanks/NNS ./.

POS-tagging can help to resolve syntactic ambiguity. For example, let's consider the sentence : “they refuse to permit us to obtain the refuse permit”. POS-tagging applied to this sentence gives the following :

> they/PRP refuse/VBP to/TO permit/VB us/PRP to/TO obtain/VB the/DT refuse/NN permit/NN.

We observe that the term “refuse” is being used twice in the sentence, the first time as a verb and the second time as a noun with a different meaning.

1.1.3 Syntactic analysis

Syntactic analysis refers to the analysis of grammar and structure of sentences by considering the phrases in those sentences. Syntactic analysis uses rules of grammar in order to define the logical meaning as well as correctness of the sentences. For example, the sentence “he car got a” is not grammatically correct. Syntactic analysis is the task of NLP that deals with the syntax of natural language.

1.1.4 Semantic analysis

The main goal of semantic analysis is to generate meaningful representations of natural language. While lexical analysis only focuses on the meaning of individual words, semantic analysis focuses on the meaning of larger chunks such as phrases, sentences, paragraphs and sometimes documents. Semantic analysis can be divided into two parts:

- Lexical semantics which is the study of words meaning. The main topics studied within lexical semantics involve either the internal semantic structure of words, or the semantic relations that occur within the vocabulary [27];
- The study of how individual words combine to provide meaningful sentences, phrases or paragraphs. For example, the sentence “white house is great” could mean the house whose color is white is great or the US white house is great.

Lexical semantics

Lexical semantics is the branch of linguistics which is concerned with the study of word meanings [28]. The study of lexical semantics includes the following points :

- Classification of lexical items. Lexical items can be classified based on whether their meaning derived from a common lexical unit;
- Decomposition of lexical items;
- Differences and similarities between various lexical structures;
- Relationship among lexical items, meaning of sentences, etc.

Some relations between lexical items include : Hyponymy, hypernyms, hyponyms, homonymy, and polysemy.

Hyponymy describes the relationship between a generic term and instances of the specified generic term. The generic term is called the hypernym, and its instances are called hyponyms. E.g., Fruit is a hypernym of orange, mango and guava. Hoponyms and hypernyms can be described using taxonomy.

Homonymy describes words that have the same syntax or same spelling or same form but differ in their meanings. The word "bank" for instance can mean a financial institution or a river bank;

Polysemy refers to a word or phrase which have different but related senses. They are also referred to as lexically ambiguous words.

Wordnet[23] is a well-known general English lexical database which implements the above concepts.

Word sense disambiguation is one of the major tasks in NLP where semantic analysis is heavily used. Moreover, word embedding is designed to be a word encoding scheme that captures the semantic of words. We will detail this latter on.

Handling ambiguity

We have seen that there are many cases that are too ambiguous for an NLP system to handle. Consequently, we need to identify the kinds of ambiguity and how we can handle them. A word, phrase, or sentence is ambiguous if it has more than one meaning. The word "light" for example can mean "not very heavy" or "not very dark". This is word level ambiguity. There are different types of ambiguity:

- **Lexical ambiguity.** As stated above, lexical ambiguity is word-level ambiguity. For example, in the sentence "call me when you arrive": here "call" is a verb. In the sentence "I received a call": here "call" is a noun. An accurate POS-tagger can help to solve lexical-ambiguity;
- **Syntactic ambiguity.** Syntactic ambiguity occurs when there can be different ways of interpreting the same sequence of words. Different grammar rules may apply for the same sequence and each structure has a different meaning. Example: The man called the girl on the phone. In fact, it is not clear whether the man called the girl, who was on the phone, or the man used a phone to call the girl. A probabilistic like solution can help get rid of this type of ambiguity;

- **Semantic ambiguity.** Semantic ambiguity occurs when the sense of a word can be misinterpreted. For example: “ABC head seeks arms”. Here the word “head” either refers to the Chief or the part of the body. This kind of ambiguity is referred to as semantic ambiguity. Word2vec [29] representation techniques are very useful when dealing with semantic ambiguities;

- **Pragmatic ambiguity.** Pragmatic ambiguity is when a phrase has different interpretation depending on the context. Pragmatic analysis helps users to discover this intended effect by applying a set of rules that characterize cooperative dialogues. E.g., “close the window” should be interpreted as a request instead of an order. Handling this kind of ambiguity is still a challenge.

1.1.5 Discourse integration

In discourse integration, the context is taken into account to interpret the meaning of a sentence. The meaning of a sentence not only depends on that sentence but also on the following sentence and sometimes the preceding one. For example, “he requested that” depends on the prior sentence to understand what the term “that” refers to.

Discourse integration is often used in Natural Language Generation Applications.

1.1.6 Pragmatic analysis

Pragmatic analysis refers to the integration of outside knowledge. That is a knowledge which is not necessarily found in the corpus of documents. The outside knowledge can be extracted from an ontology or a knowledge graph.

1.2 NLP development life cycle

Solving a natural language problem involves a successive set of steps, and the main raw material used is a dataset or a corpus.

Figure 1.3 shows the different steps one may go through to solve a NLP problem. These steps start with a deep understanding of the NLP problem and the disponibility of a corpus.

Before going through the different processes involved in the NLP development life cycle, let’s start by defining what a corpus is.

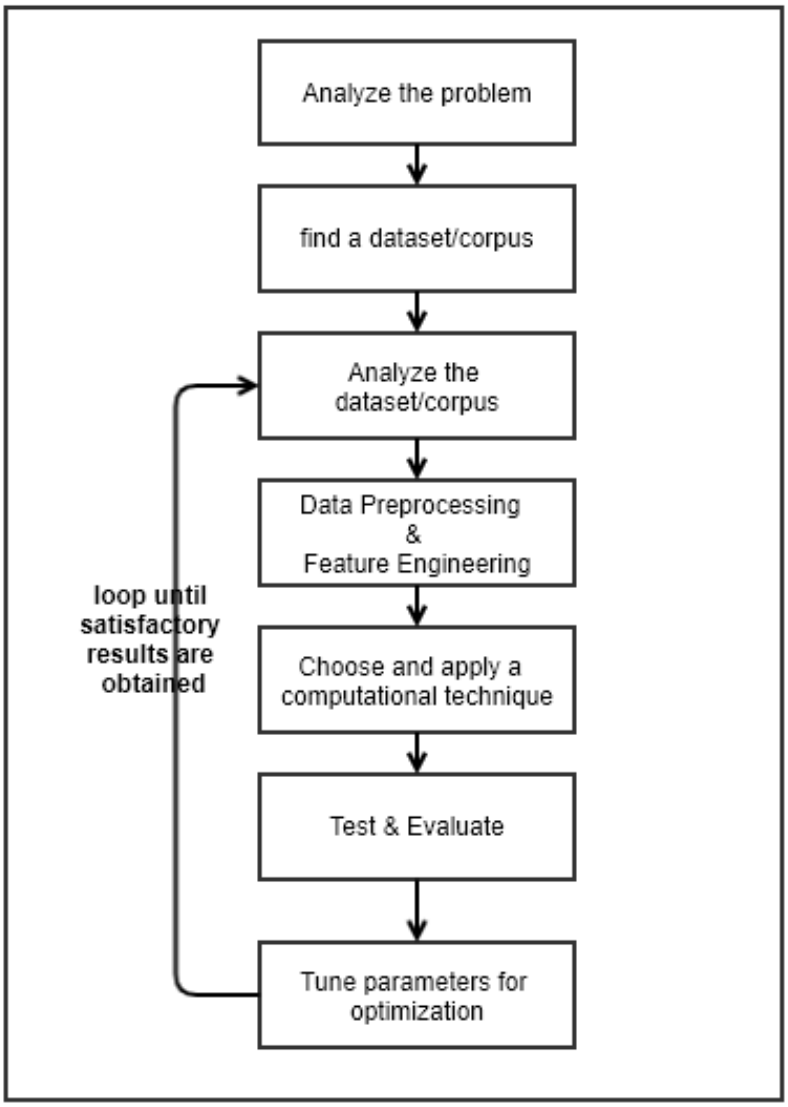


Figure 1.3: NLP development life cycle (Source [30])

1.2.1 Corpus and dataset

A corpus is the most critical and basic building block for any NLP-related application. Corpora provide quantitative and qualitative data needed to build NLP applications. Difficulties regarding creating a corpus for NLP applications include but are not limited to:

- What types of data do we need to solve the NLP problem?
- Data availability: is data available?
- Quality of data: is data accurate?
- Adequacy of data in terms of amount: Do we have enough data?

Cambridge dictionary [19] defines a corpus as a collection of written or spoken material stored on a computer and used to find out how language is used. Corpora are huge amount of data used for linguistic analysis or other NLP tasks. Data from corpora are unstructured, encoded in a specific dialect, presented in a specific format (text, audios, videos, or images containing text), which make them difficult to process.

1.2.2 Corpus data analysis

An important step before processing data in a corpus is to analyze the corpus in order to have a good understanding of data, their nature, how they are organized, and the statistical knowledge gathered inside the corpus. Corpus analysis for speech involves phonetic understanding, conversation analysis, etc, while corpus analysis for text corpus mainly concerns the analysis of word frequencies, co-occurrence analysis, determining the corpus vocabulary, etc. NLP needs a minimum of corpus analysis in order to have a good understanding of data which will guide the selection of appropriate methods and techniques to process these data.

1.2.3 Data selection, preprocessing and feature engineering

Before applying NLP techniques, data must be prepared and put in a convenient form, generally in form of vector of numerics. There are several data preparation methods mentioned in the literature and the choice of which combination of methods to use really depends on the problem we want to solve. Preparing data for the application of NLP techniques generally involves : selecting data, preprocessing data and transforming data.

Selecting data

The aim of data selection is to select the best training data to achieve the greatest possible performance when solving the problem. High quality data with limited numbers of labeled instances [31] may be ideal. Active learning is a technique that deals with the data selection problem by selecting the most informative data for training [32, 33]. Data selection also involves removing from the dataset, the data which are not relevant for the NLP application.

Feature Selection

The main objective of feature selection in NLP is to select a subset of terms from the training set that will be used as features for NLP algorithms. The feature selection process consists in selecting the features that we think will be relevant and what we think can be ignored. Feature selection serves two main purposes:

- First, it decreases the size of the vocabulary, thus reducing the amount of memory used and enhancing the efficiency of the NLP algorithms;
- Second, it increases NLP model's accuracy by eliminating noise features. Indeed, a noise feature is a feature that, when added to the input representation, increases the classification error on new data.

For good accuracy on feature selection, it is required to modify words, making them lower case, and choosing whether to do stemming or not. Features selection techniques includes the following steps:

Removing punctuation: In practice, punctuation does not bring too much information in NLP task, that is why they are generally removed. Generally removing punctuation from texts consists in replacing them by spaces. But this can lead to some problem in certain situations especially in cases where someone has hyphenated a word. For example, we would want cross-examination to become crossexamination. To deal with these situations, it would be preferable to replace punctuation with spaces and deal with hyphenated words as special cases.

Removing capitalization: The majority of time, capital letters often doesn't bring any value to the meaning of a sentence. More often, they just indicate the beginning of a new sentence. If we do not deal with capital letters, we risk ending up with a vocabulary containing multiple versions of the same word. To deal with that problem, the solution is to transform the whole text into lower case. This should obviously be done before removing stopwords as they are generally in lower case. Sometimes, transforming a word into capitals, could change its meaning. E.g. "US" and "us", the first stands for United States, while the second is a pronoun.

Choosing words to keep and those to drop. One approach of feature selection is to remove irrelevant words from texts. Another approach consists in selecting the words that give the most

meaning for the document. A combination of method can be used to achieve this task. These methods range from the use of a dictionary were relevant words are validated, to the use of a stop set which contains words that will be considered irrelevant for the task. The choice of which combination of methods and the order in which they may be applied tightly depends on the task to perform. Retaining only relevant words enables us to reduce the size of the vocabulary and consequently reducing the memory required to store the document model. It should also make things run faster.

Stopwords filtering. Probably the most commonly used method to remove irrelevant words is to create a list of stopwords that will be always ignored when processing the text. Evident stopwords generally include articles (the, a, an, etc), preprositions (of, for, ...), but the choice of which word to add in the stop list depends on the application. Indeed, a word which is relevant for sentiment analysis can be irrelevant for question answering and vice versa. There are standard stopwords for general text [34] that can be downloaded from the Internet. However, Some authors have proposed methods for automatic identification of stopwords [35, 36].

Word filtering based on frequency. Another filtering technique to get rid of non-relevant words is to take out those that appear less frequently or those that appear in almost all the documents. In fact words that appear in very few documents represent either typographical errors or words that are not related to the subject addressed in documents. Ignoring those words in many cases may not have an incidence in the final model accuracy. Conversely, words that appear in almost all documents are not discriminative enough. In fact, they cannot be used to distinguish a document from another since they appear with almost the same frequency in all documents.

TF-IDF based feature selection

Certainly, the most popular filtering method based on frequency is TF-IDF [37]. TF-IDF aims at determining the importance of a keyword or a phrase within a document. The TF-IDF weighting of a term i with respect to a document j, noted $tfidf_{ij}$ is defined by Equation 1.1

$$tfidf_{ij} = tf_{i,j} \times idf_i \tag{1.1}$$

$tf_{i,j}$ represents the Term Frequency which is the frequency of the term i in the document j and idf_i represents the Inverted Document Frequency, which is the inverse of the number of doc-

uments in which the term i appears. Idf acts as to penalizing terms which appear in almost all the documents as they are not enough discriminative.

Term frequency. In Equation 1.1 the term $tf_{i,j}$ can be calculated as the raw frequency of the term i in the document j which is the number of occurrences of the term i divided by the length of the document (the total number of occurrences of terms in document j). There are other ways of determining the Term Frequency (TF) that are summarized in Table 1.2

Table 1.2: Variants of TF

Weighting scheme	**formula**
Binary	0=absence 1=presence
Raw count	$f_{t,d}$ (number of occurrences of f in d)
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
Log scale	$log(1 + f_{t,d})$
Double normalization 0.5	$0.5 + 0.5 \times \frac{f_{t,d}}{\max_{(t' \in d)} f_{t',d}}$
Double normalization K	$K + (1 - K) \times \frac{f_{t,d}}{\max_{(t' \in d)} f_{t',d}}$

- The Binary frequency : $tf_{t,d} = 1$ if d contains the term t and 0 otherwise;
- frequency ratio : $tf_{t,d}$ = number of times t occurs in d / total number of terms in d;
- The log scaled frequency which acts as a normalizer. Higher frequency will be represented by the nearest values;
- The double normalization K which prevents biases when dealing with longer documents;

Inverted document frequency The inverted document frequency measures how much discriminative a term is, if it is common or rare across all documents. Idf is calculated by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that result (see Equation 1.2).

$$idf(t,d) = log(\frac{N}{|\{d \in D : t \in d\}|}) \tag{1.2}$$

Other variants of IDF are reported in Table 1.3.

In Table 1.3, N represents the document length and n_t represents the number of documents containing the term t. The unary weighting scheme for IDF masks the intervention of IDF in TF-

Table 1.3: Variants of IDF

Weighting scheme	formula
Unary	1
inverse document frequency	$log\frac{N}{n_t} = -\frac{n_t}{N}$
invert document frequency smooth	$log(\frac{N}{1+n_t}) + 1$
invert document frequency max	$log(\frac{\max_{t' \in d} n_{t'}}{1+n_t}) + 1$
probabilistic idf	$log(\frac{N-n_t}{n_t})$

IDF calculation (Equation 1.1). In other words, the importance of a term will only be determined by the frequency of that term.

Transforming the data

Machine cannot process words in their state. A prerequisite to apply any computational technique such as text classification or sentiment analysis is to transform words into numbers, and sentences into sequences of numbers. When we have numbers, we can now easily apply a machine learning algorithm for classification or clustering. The process of transforming sequences of words into sequences of numbers (vectors) is also known as text encoding.

There are few text encoding algorithms, each with its pros and cons and each suited for a particular task. The simplest encoding techniques lose information about words order while others don't. Some encoding techniques are fast and intuitive, but the size of the resulting document vectors grows quickly with the size of the dictionary, and the others optimize the vector dimension but lose in interpretability. Let's detail the most frequently used encoding techniques.

One-Hot or Frequency Document Vectorization (not ordered). A commonly used text encoding technique is the One-Hot encoding or document vectorization. This technique starts building a dictionary from all words available in the corpus of documents. Other preprocessing techniques such as lemmatization, stemming, tokenization will be used to build the dictionary. Each text will be encoded by a vector of 1s and 0s. Each word in the dictionary is associated to an index which corresponds to a position in the document vector. 1 encodes the presence of a word and 0 its absence. An element of the vector will be assigned the value 1 if the word of the dictionary corresponding to that index is present in the text to encode, and the value 0 else.

Figure 1.4 shows a simple example where the sentence "john likes ice cream" is encoded using one-hot encoding based on a simple dictionary containing the words *John, chocolate, cream, hates, ice, likes, hot.* This encoding does not take into account the order in which the words occur in

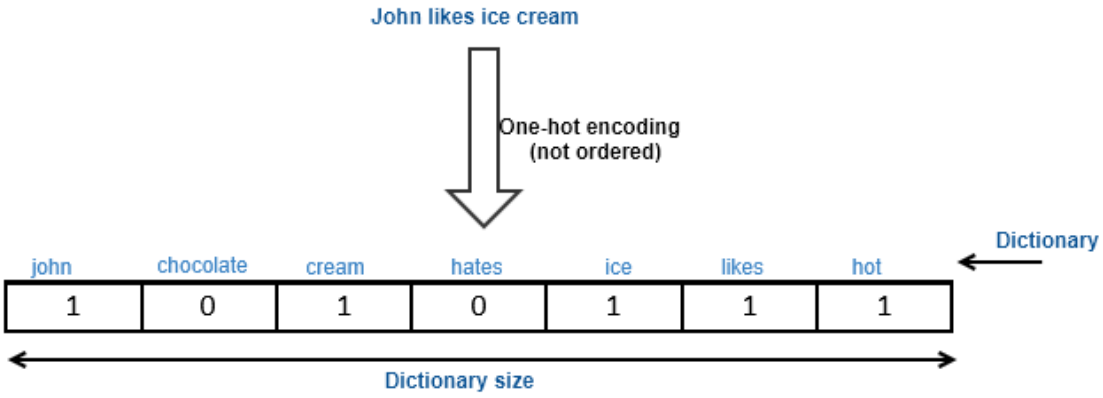


Figure 1.4: Illustration of one-hot encoding with no order.

the sentence. However, the order of words in a sentence is important, to take into account for example, negations or grammar structures. Another disadvantage of this encoding scheme is that the dimensionality of the final vector space grows rapidly with the size of the dictionary. This situation generally leads to sparse vector representations.

The solutions to the growing size of the dictionary include: cleaning and or extracting keywords from the text documents or using a predefined domain dictionary as discussed in section 1.2.3. Another variant of one-hot encoding enables to take word order into account.

One-Hot Encoding (ordered). Some machine learning algorithms build an internal representation of items in a sequence, like ordered words in a sentence. Recurrent Neural Network (RNN) and LSTM layers, for example, can exploit the sequence order for better classification results.

In this case, we need to move to a representation which takes the word order into account. The document is still one-hot encoded in a vector, but the words are fed sequentially into the model. When using the one-hot encoding technique, each document is represented by a tensor which may consists of a very long sequence of 0/1 vectors, leading to an extremely large and sparse representation of the documents.

Figure 1.5 shows an illustration of a text represented as a sequence of one-hot vectors. The order of words in the sequence corresponds to the order in which they appear in the sentence.

Index-based encoding. The index-based encoding is also an encoding technique which preserves the order of the words as they occur in the sentences. The principle of the method is to

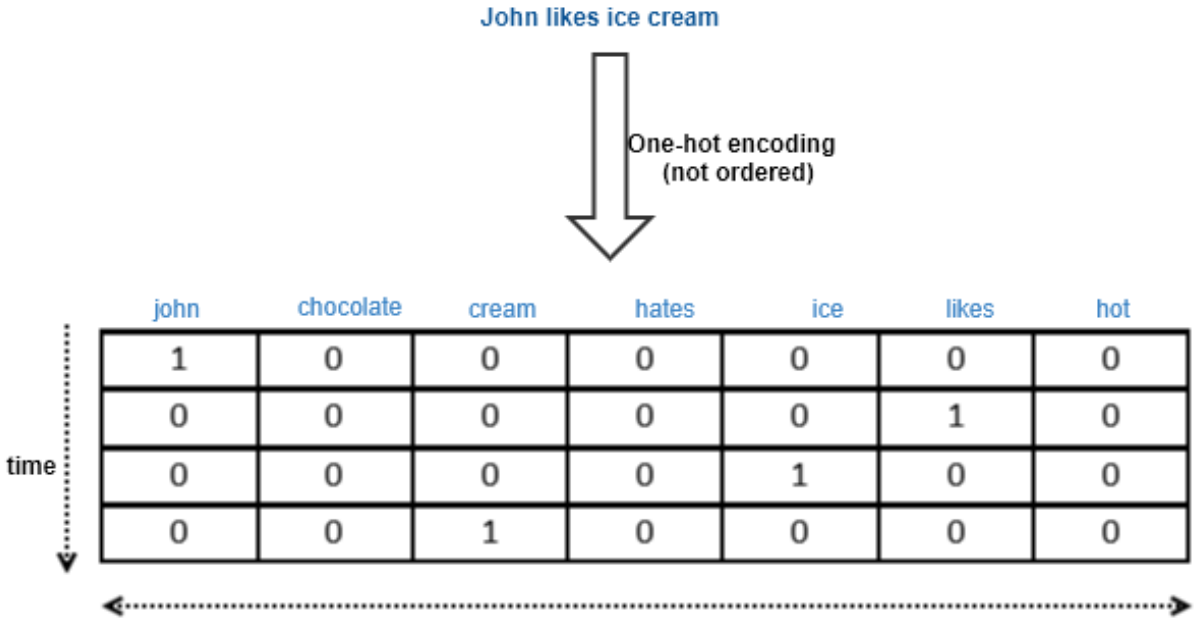


Figure 1.5: Illustration of one-hot representation with consideration of word order.

match each word with one index number. Like the one-hot encoding technique, the first step is to create a dictionary that maps words to indexes. Then, based on this dictionary, each document will be represented as a sequence of indexes (numbers), each number representing the dictionary index of the word at the corresponding position. The main disadvantage of index-based encoding is that it introduces a numerical distance between texts that doesn't really exist. Figure 1.6 shows an illustration of the representation of a sentence using index-based encoding.

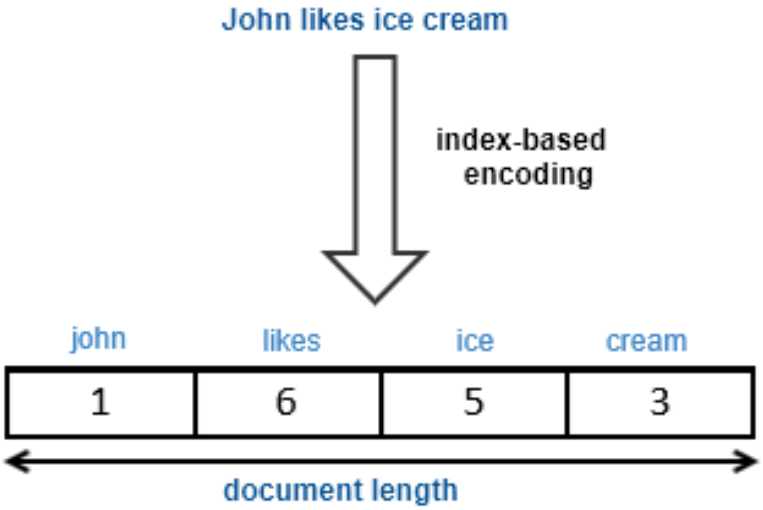


Figure 1.6: Illustration of index-based document encoding.

With index-based encoding, we can end-up with document vectors of different lengths. But,

the majority of machine learning algorithms require document vectors to have fixed length. The common approach to deal with that situation is to define a maximum length for documents. Documents that are shorter will be zero-padded and those that are longer will be truncated. Zero-padding means adding zeros sequence to reach the maximum number of words allowed. Truncating means pruning off all words after the maximum number of words has been reached.

Word Embeddings. Word embeddings are a family of NLP techniques which aims at mapping semantic meaning into a geometric space. The principle is to associate each word of the dictionary with a numeric vector such that the distance between any two words would capture the semantic relationship between them. The geometric space formed by these vectors is called an embedding space. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension.

Methods to generate mapping include : neural networks [29], dimensionality reduction on the word co-occurrence matrix [38, 39, 40], probabilistic models [41], explainable knowledge base method [42] and explicit representation based on the context in which words appear [43].

Word and phrase embeddings have been shown to boost the performance of NLP tasks such as syntactic parsing [44] and sentiment analysis [45]. The best-known word embedding techniques are word2vec [29] and Glove [46].

Training word embeddings : CBOW and Skip-gram

There are two main architectures to train and build word embeddings using neural networks. They are: Continuous Bag Of Words (CBOW) and Skip-gram. Both architectures are illustrated in Figure 1.7

Both methods proceed in different ways to compute the representations of words. In the CBOW model, the distributed representations of the context (surrounding words) are combined to predict the word in the middle while in the Skip-gram model, the distributed representation of the input word is used to predict the context.

To train the model, a “fake” task is defined. The goal is to learn the weights of the hidden layer that are the “word vectors”, and, we wont be interested in the inputs and outputs of this network.

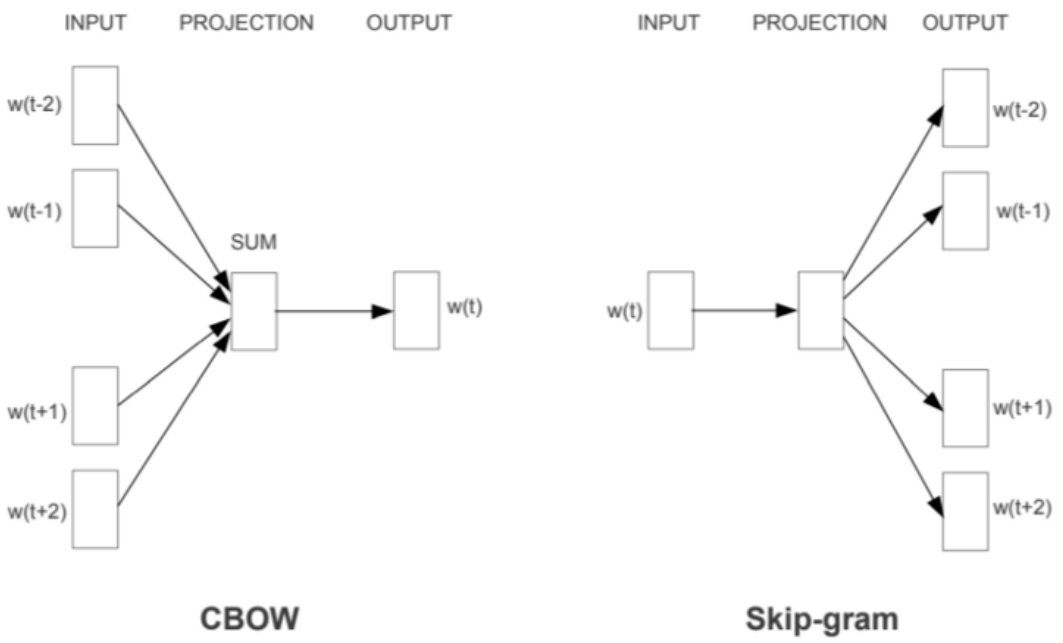


Figure 1.7: CBOW and Skip-gram model [47].

Skip-gram

In the case of Skip-gram, the fake task would be: given a word, we'll try to predict its context, that is the words which come before and after the given word. The size of this context will be called the windows size: a hyper-parameter. Let's consider the sentence "I will have orange juice and eggs for breakfast." and a window size of 2. If the target word is juice, its neighboring words will be (have, orange, and, eggs). Our input and target word pair would be (juice, have), (juice, orange), (juice, and), (juice, eggs).

Figure 1.8 shows the architecture of skip-gram which consists of an input word vector, a hidden layer and an output layer.

The input word is encoded in an n-dimension vector, where n is the number of words in the vocabulary. In other words the input vector is the one-hot representation of the word. The single hidden layer will have a dimension of $n \times d$ where d is the size of the word embedding, which is also a hyper-parameter. The output from the hidden layer would be a vector of dimension n, which will be fed into a softmax output layer. The dimensions of the output layer will be n, where each value in the vector will be the probability score of the target word at that position. The architecture is trained using the back-propagation algorithm which is done in one back pass for a single source word. Given an input word, a feed forward is complete to compute the output of the overall network for each target class. Then, the overall error vector is calculated as the sum of the

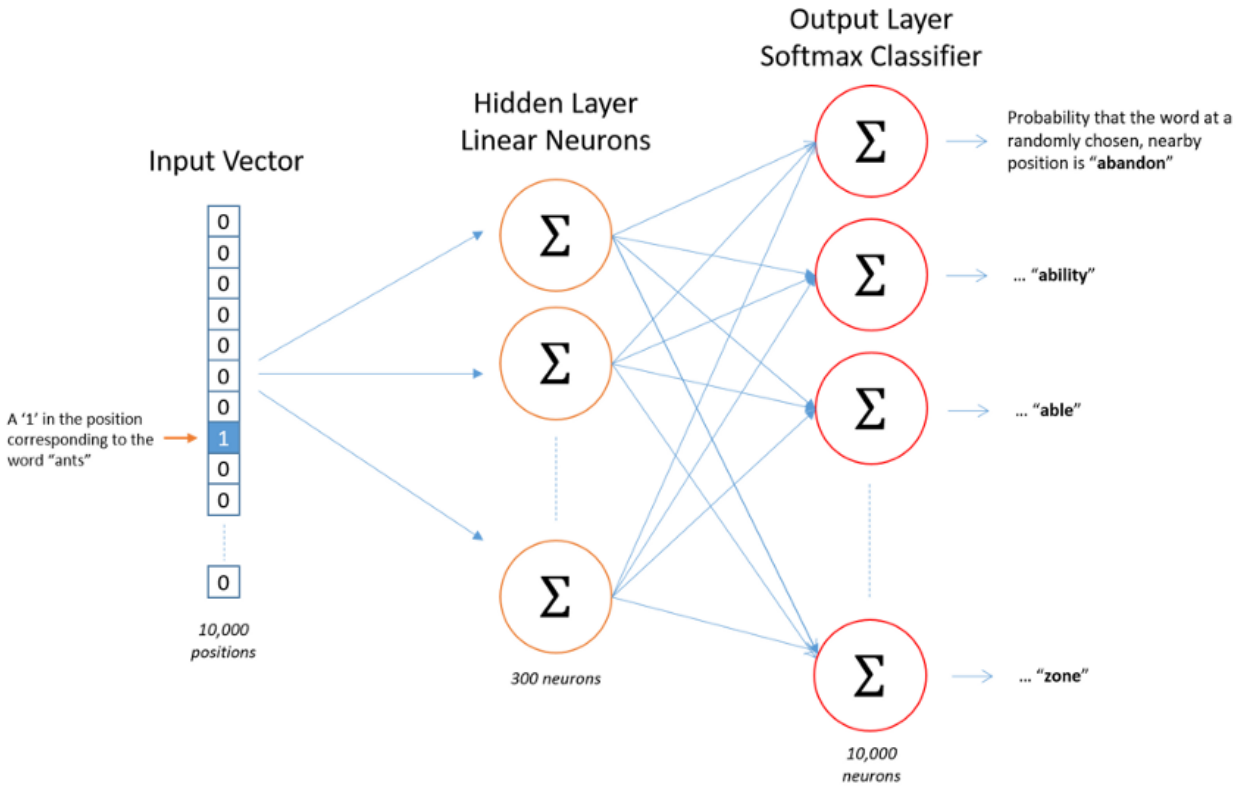


Figure 1.8: Skip-gram architecture [47].

individual error vector obtained for each target word. The weights of the hidden layer will then be updated based on this error vector.

Continuous Bag Of Words

The fake task in CBOW is slightly similar to that of skip-gram, in the sense that we still take pair of words and train the model to learn these co-occurrences but instead of adding the errors, the input words are added for the same target word. The dimension of the hidden layer and that of the output layer remain the same. Only the dimension of the input layer and the calculation of hidden layer activations will change. If the window size is 2 (i.e., 4 contexts words + the target word), then we will have $4 \times n$ input vectors, each corresponding to a context word. Each will be multiplied with the matrix of weights connecting the input units to the hidden units. The resulting products will be 4 d-dimension intermediate vectors. All the 4 intermediate vectors will be averaged element-wise to obtain the final activation which will then be fed into the softmax output layer.

Table 1.4: Comparison of Skip-gram and CBOW.

Skip-gram	CBOW
Context is used as input to predict a target word	The target word is used as input to predict the context
Works well with a small amount of training data	Needs more training data to perform well
Lower to train than CBOW	Faster to train than skip-gram
Represents well even rare words or phrases	Better accuracy for frequent words

Skip-gram versus CBOW

Table 1.4 shows a summary of the differences between both methods. Insights to build this table were found in [48, 49].

1.3 Deep learning for Natural Language Processing

Deep learning is a field of machine learning that has proven to be highly useful in various domains including text processing, image processing, speech processing and so forth. A majority of deep learning algorithms are based on the concept of Deep Neural Network (DNN), and the training of such algorithms today has been made easier with the availability of huge amount of data and important computation resources. It has been shown that the accuracy of a deep learning model keeps improving with additional data as shown in Figure 1.9. The term "deep" in Deep learning refers to the depth of the neural network architecture.

Figure 1.10 presents the difference between the deep and the shallow network architecture. Shallow neural network generally consists of a single hidden layer while deep neural network consists of several layers often of different types. Moreover, DNN are capable of learning features from raw data such as images, text documents or files (audio, video data, etc.). What differentiates any deep neural network from an ordinary artificial neural network is the way each use back-propagation. In an ordinary Artificial Neural Network (ANN), back-propagation trains end layers more efficiently than it trains initial (or former) layers. Indeed, errors become smaller and more diffused as we travel back into the network [50].

The term "deep" may impress more than one. However, there is not much difference between a shallow and deep neural network. Indeed, a DNN is simply a neural network with multiple hidden layers [50].

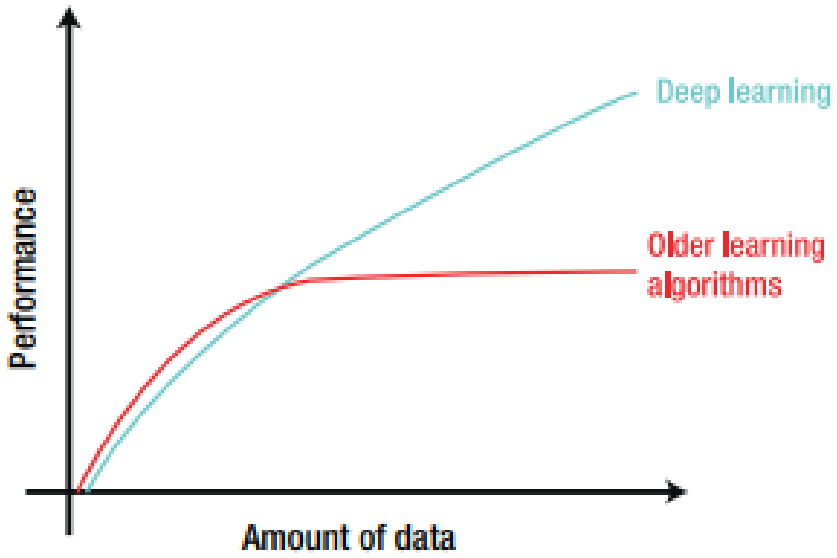
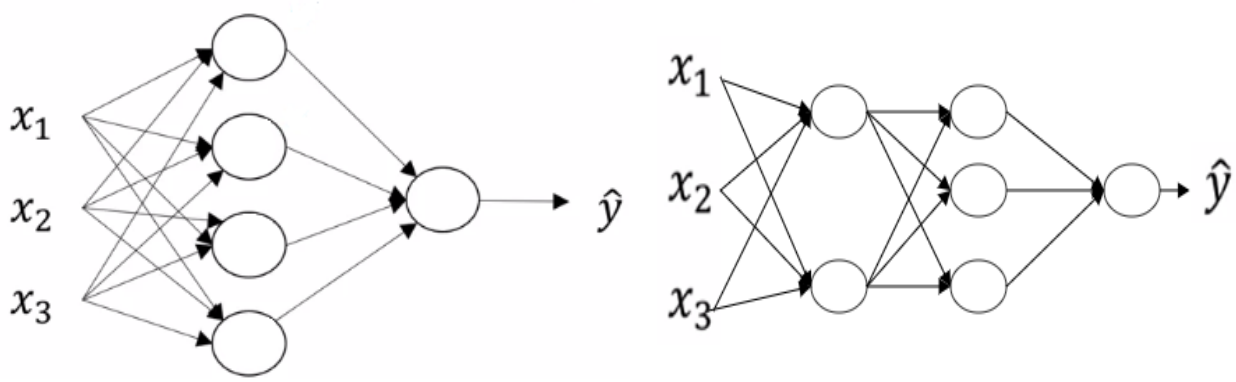


Figure 1.9: Source : Deep learning for natural language processing [50].



(a) Shallow neural network (b) Deep neural network

Figure 1.10: Shallow neural network versus deep neural network architecture.

Before presenting some popular deep neural networks, we will be revisiting the basics of Artificial Neural Network (ANN) as they are useful to understand how DNN work.

1.3.1 Artificial Neural networks

Artificial Neural Network (ANN) are a biologically inspired paradigm (inspired from the functioning of the human brain) that enables a computer to learn from observational data as human does. Neural networks have widely been used to solve several problems including: image recognition, optical character recognition, speech recognition, speech analysis and NLP. Neural networks help us to solve complex problems which require intelligence and where it is difficult to write a combinatory algorithm to solve the problem.

Structure of a simple neural network

A neural network consists of a collection of basic elements (cells), artificial neurons or perceptrons, that were first developed in the 1950s by Frank Rosenblatt. Each cell takes several binary input $x_1, x_2, \ldots, x_N$ and produces a single binary output. The neuron is said "fired" or activated if the weighted sum (Equation 1.3) is greater than the activation potential. The neurons that fire pass the signal to other neurons connected to their dendrites, which, in turn, will fire, if the activation potential is exceeded, thus producing a cascading effect. Figure 1.11 presents the architecture of a simple perceptron.

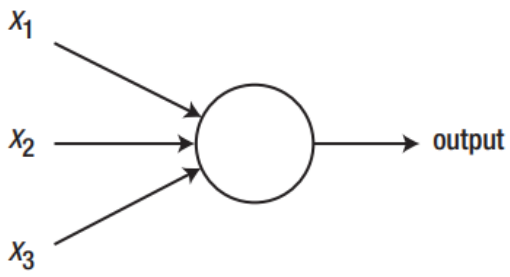


Figure 1.11: A perceptron.

Since every input does not have the same importance on the task to perform, weights are attached to each input, in order to allow the model to assign more importance to some input. Therefore, the outcome of the neuron will be 1 if the weighted sum is greater than the activation potential bias. This can be modeled by Equation 1.3 which behaves as the step function (Figure 1.12).

$$Output = \sum_j w_j x_j + Bias \tag{1.3}$$

In practice, the abrupt nature of the step function (Figure 1.12) makes it difficult to train the neuron well. That is the main reason why activation functions were introduced to make the model behave more predictably, such that small changes in weights and bias cause only a small change the in output.

Activation functions

There are several activation functions, each adapted to specific situations. Some of the most popular include :

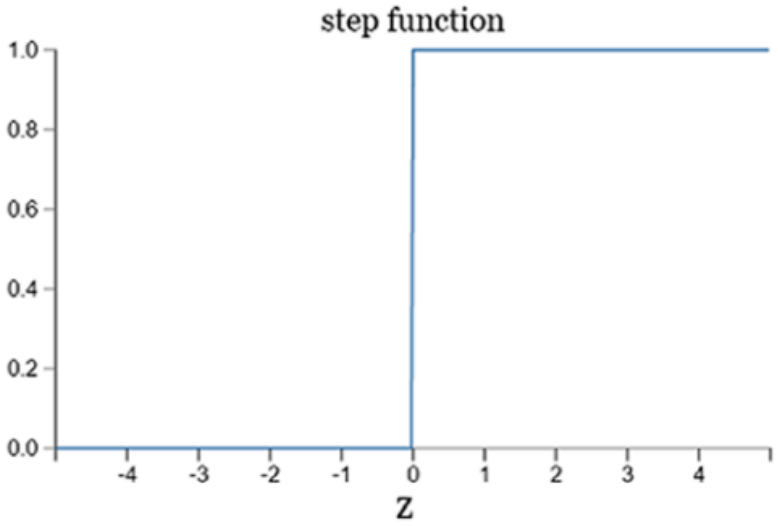


Figure 1.12: Step function curve.

The sigmoid activation which is computed using Equation 1.4, where x_j are inputs, w_j are weights, and b is the bias.

$$z_j = \frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{1.4}$$

Sigmoid is generally used as the output unit activation for two-class logistic regression.

The Rectified Linear Unit (ReLU) activation which keeps the activation of the neuron guarded at zero and is computed using Equation 1.5.

$$z_j = f_j(x_j) = max(0, x_j) \tag{1.5}$$

ReLU is generally used as activation of hidden units in Deep Neural Network (DNN).

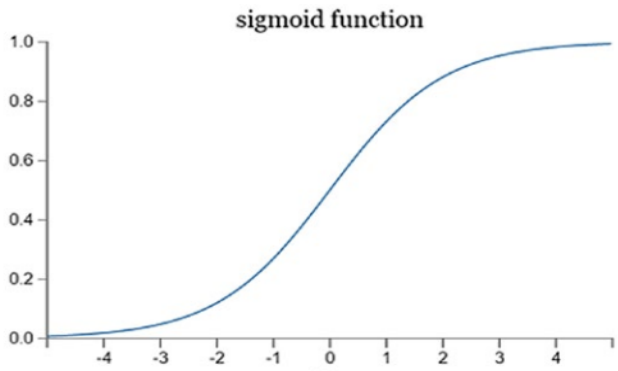
The SoftMax activation also referred to as normalized exponential function, transforms a set of given real values in the range of (0,1), such that the combined sum is 1. Softmax function is modeled by Equation 1.6

$$\sigma(Z)_j = e^{z_k} / \sum_{k=1}^{K} e^{z_k} \; for \; j = 1, ..., K \tag{1.6}$$

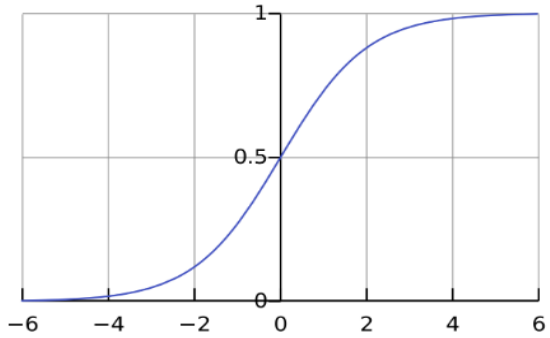
Softmax is generally used as activation in the output layer for multiclass logistic regression.

Tanh activation. It operates more like Sigmoid. The tanh function is defined by Equation 1.7.

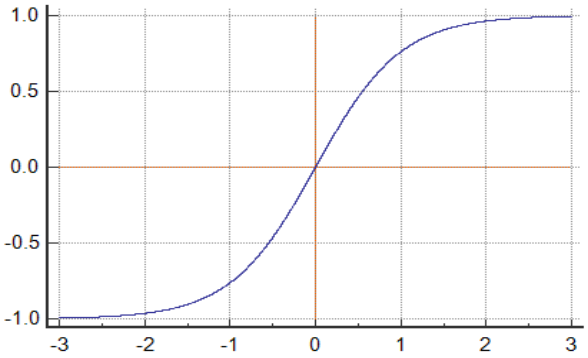
$$z_j = tanh(x_j) = \frac{2}{1 + e^{-2x}} - 1 \tag{1.7}$$



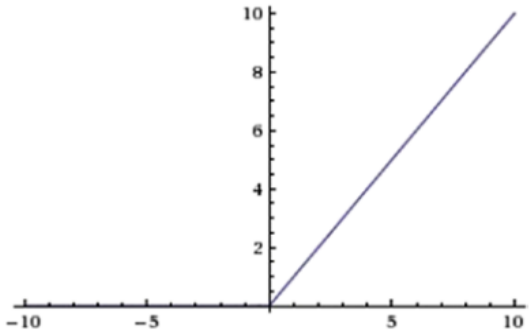
(a) Sigmoid



(b) Softmax



(c) tanh



(d) ReLU

Figure 1.13: Graphical representations of the 4 most popular activation functions.

Figure 1.13 shows the graphical representations of the various activation functions described above. We observe that sigmoid, softmax and tanh are all sigmoidal (s-shape). All the functions described are monotonic (increasing) and differentiable.

The fact that the activation function is differentiable allows the network to be trained easily with the gradient descent algorithm (Gradient Descent is covered in section 1.3.3). As in the human brain, neurons are organized into layers, with connections between neurons in the same layer and with neurons in adjacent layers, creating an Artificial Neural Network (ANN) or Multi-Layer Perceptron (MLP). The complexity of the architecture depends on the number of layers and the number of units per layer. The layers between the inputs and the outputs are referred to as hidden layers. A fully connected neural network (or simply dense neural network) is an ANN where all the neurons of layer L are connected to all those of layer $L+1$. Figure 1.14 shows an example of a fully connected neural network.

1.3.2 Types of neural networks

There are various types of neural networks currently being used in artificial intelligence. Neural Networks are classified based on their internal organization (layers, connections, ...) and how

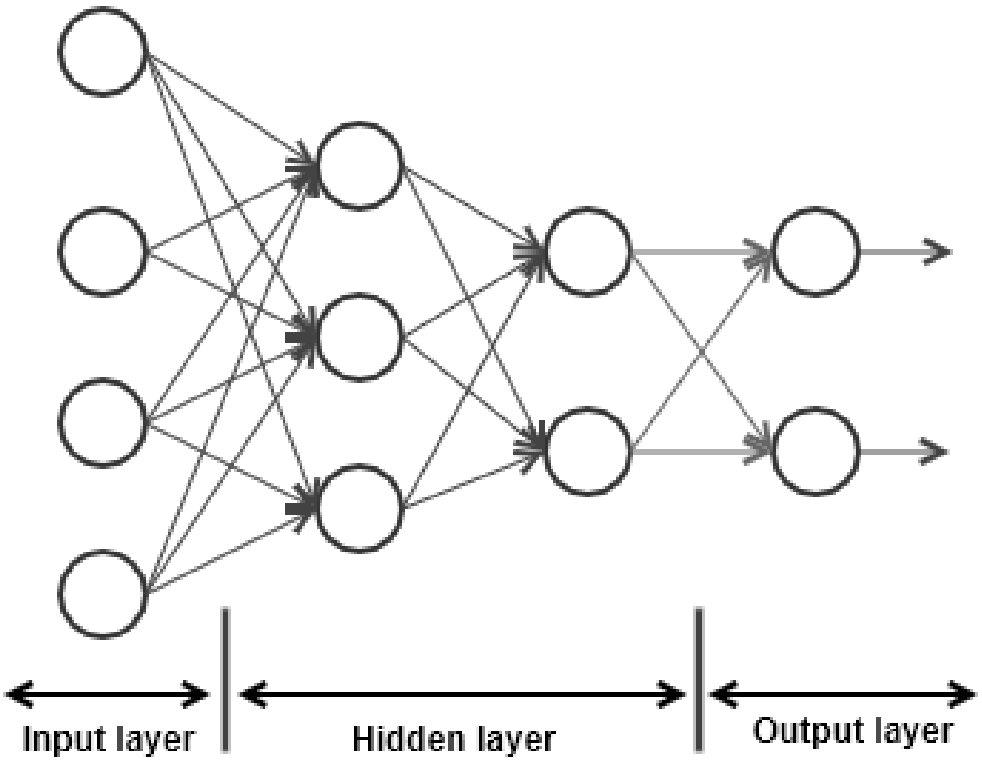


Figure 1.14: Fully connected neural network.

the information flow travels between layers. Some neural networks are organized in such a way that the connections between neurons can take the form of cycles. Recursive, or recurrent neural networks belong to this category. If however the connections between the neurons are acyclic, they form networks such as feed-forward neural networks.

Feed-forward neural network

Feed-forward neural networks can be considered as one of the simplest neural network architectures. In this type of network, information flow goes from the input layer to the output layer, and the outputs from one layer are the inputs of the next layer and there is no loop/cycle in the network. Figure 1.14 is an example of a feed forward neural network where the arrows indicate the sense of the circulation of information between the neurons.

Convolutional Neural Networks

The propagation of information flow in a Convolutional Neural Network (CNN) is similar to that of the feed-forward neural network. However, a Convolutional Neural Network consists of multiple layers of different types that realize different operations.

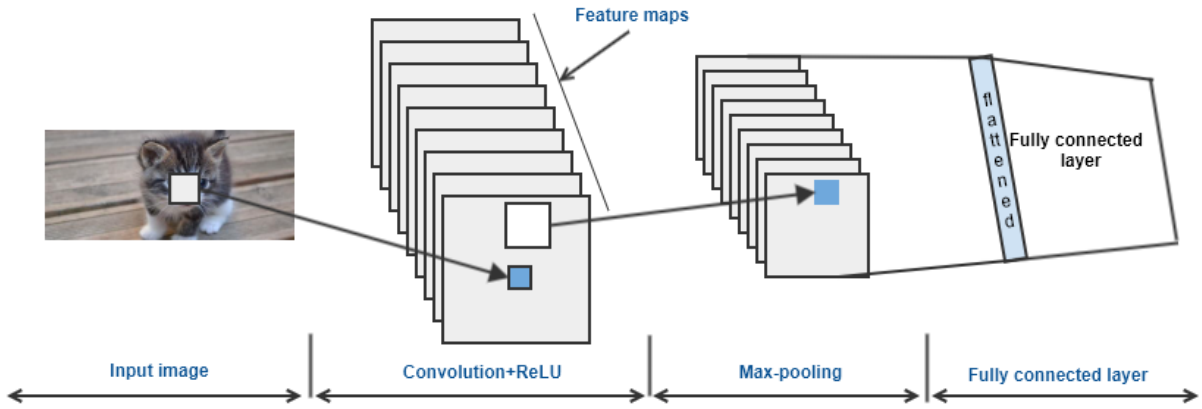


Figure 1.15: Simple architecture of a CNN.

Figure 1.15 shows the architecture of a classic convolutional neural network (ConvNet) with its basic building blocks. A typical convolutional neural network is made up of three types of layers which can be repeated any number of times.

The convolutional layer. The convolutional layer learns to detect or to identify spacial features in an input impage represented in the form of a matrix of numerics. The convolutional layer consists of multiple filters (also known as kernels) which applied on the input image, produce filtered output images that will be called feature maps. a Filter is generally represented as a small martix of numerics. The filter kernel size represents the dimension of the matrix ($k \times l$). Applying a filter on the input image can be seen as the filter matrix sliding over the input image and at each step, the convolution operation is performed between the filter matrix and the corresponding portion of the image at this step. If we consider a square filter K of size $k \times k$ and a square input image I of size $n \times n$, then the convolution of filter K with the image I results in a matrix O given by Equation 1.8.

$$O(x,y) = \sum_{i=0}^{i=k} \sum_{j=0}^{j=k} K(i,j) x I(x-1+j, y-1+i) \tag{1.8}$$

Figure 1.16 illustrates the application of a convolutional filter to an input image.

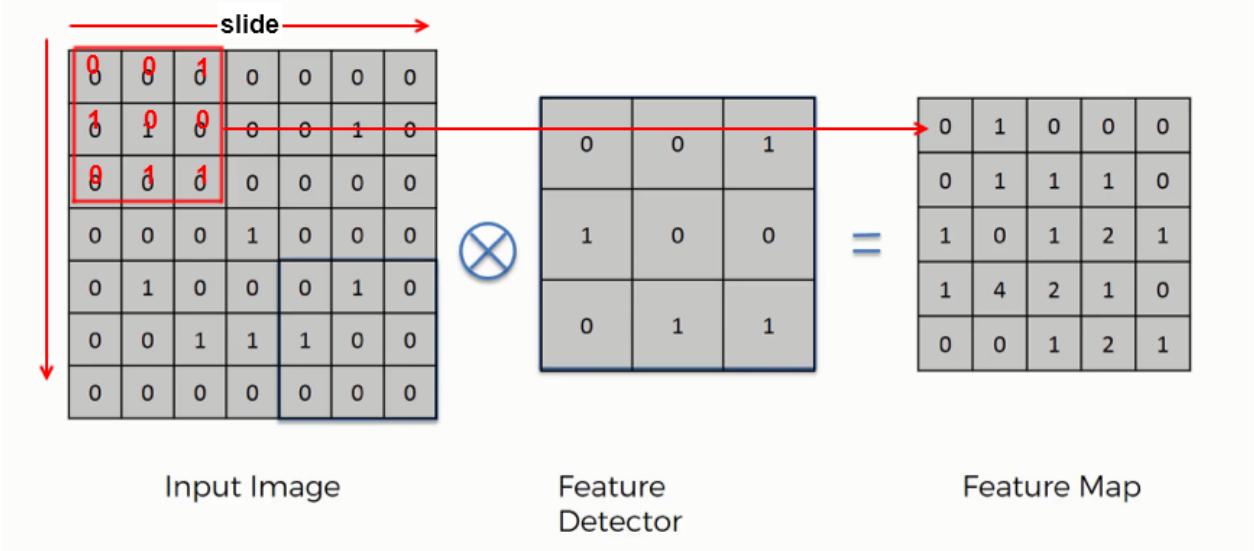


Figure 1.16: Illustration of the convolution operation

The results of the convolution are rectified using the ReLU activation before being fed as input to the max-pooling layer.

The Max-pooling layer. Immediately after the convolutional layer follows the max-pooling layer. The max-pooling layer takes as input the filtered images (feature maps) resulting from the convolutional layer and breaks the image into smaller patches, before maximum value are picked from each patch. The max-pooling is defined by two main parameters : the patch size (or windows size) and the stride. Applying max-pooling to an image can be thought as sliding a window of size $m \times n$ (the patch size) over the input image and selecting the maximum value in the region of the image covered at each step as illustrated in Figure 1.17. The windows moves by some stride across and down the image. Generally, the patch size is set to 2×2 and the stride is set to 2. A smaller stride than the patch size will see some overlap in patches while a larger stride would miss some pixels. That is why generally the stride is set equal to the patch size.

Max-pooling is used for two main reasons :

1. Dimensionality reduction (Downsampling) : If the patch size is let's say 2×2 and the stride is 2, then the input image size will be reduced by a factor of 2 (Figure 1.17);

2. It makes the network being resistant to small pixel value changes in the input image.

Fully connected neural network The output matrices of the last max-pooling layer are flattened and concatenated before going as input to the fully connected neural network. The fully con-

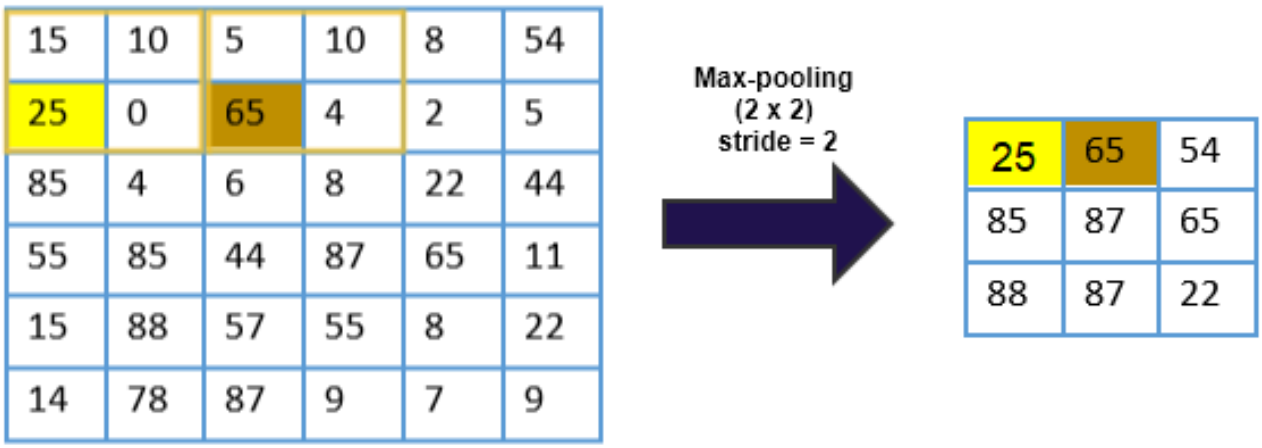


Figure 1.17: Illustration of the max-pooling operation.

nected neural network performs classification and produces the class scores based on the features extracted by the convolutional and the max-pooling layers. The fully connected neural network is generally a standard feed-forward neural network.

To summarize, we can say that : CNN consists of multiple layers :

- A series of convolutional layers + activation + max-pooling and a fully connected layer that can produce a set of probabilities associated to classes;
- The convolutional layers act as feature extractors;
- Features learned by convolutional layers are defined by weights that make up the convolutional kernels in the network;
- CNN learns the best weights during training using a process called back-propagation, which updates the weights backward in a way that reduces the classification error.

CNN was first designed for image processing. After, Kim has proposed an adaptation for text classification [15] where convolutional filters and max-pooling filters slide only on one dimension (The y dimension). That is why this type of CNN architecture has been named 1D-CNN. Apart from text classification, 1D-CNN are also heavily used for time series processing [51, 52, 53].

Recurrent Neural Network

Recurrent Neural Network (RNN) are suitable in situations where a data pattern changes over time [54]. A RNN applies the same layer to the input at each time step, using the output (i.e, the state of the previous time steps as inputs).

RNN is adapted for modeling sequence data such as time series or NL, because It produces its decision based on the actual state and on what it has encountered in the past. In other words, a RNN can take one or more input vector(s) and compute the corresponding output vector(s), and the output(s) produced are influenced not just by the weights applied on inputs like in classic NN, but also by a "hidden" state vector representing the context based on prior input(s)/output(s). This implies that the same input could produce a different output depending on previous inputs in the series. In NL for example, the meaning of a word depends upon the prior words and the next words.

Another property of RNN is that it is able to process series of inputs with no predetermined limit on size.

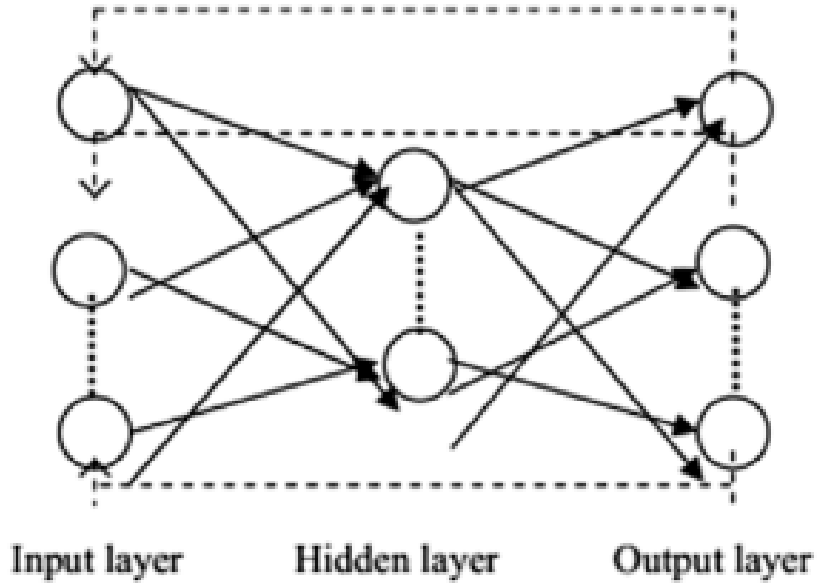


Figure 1.18: Typical RNN architecture.

Figure 1.18 shows typical RNN architecture where we can see backward connections between the output layer and the input layer.

Long Short-Term First (LSTM) is one of the most popular types of RNN used in deep learning, and was originally proposed by Boulanger-Lewandowski et al.[55]. LSTM introduces changes to how outputs and hidden states are computed using inputs. LSTM add additional gates and cells state such that it is able to address the problem of maintaining or resetting the context across sentences. Other variants of RNN include Gated Recurrent Unit (GRU) [56], Bidirectional Gated Recurrent Unit (BI-GRU), and Bidirectional Long Short-Term First (BI-LSTM) [57].

1.3.3 Training the Deep Neural Network

Now we have a better understanding of what a DNN is and what are the various types of DNN, we will now focus on the principal methods used to train Deep Neural Network (DNN).

The main purpose of any deep learning strategy is to find the best parameters (often weights and biases) which minimize the classification error. There are several learning methods which can be used to train DNN, and the majority of these methods can work with all the families of DNN especially : CNN, RNN, and Recursive Neural Network. We briefly present one of the most popular learning strategy: The Stochastic Gradient Descent (SGD) combined with the back-propagation algorithm.

Stochastic Gradient Descent

Before explaining what Stochastic Gradient Descent stands for, let's look at Gradient Descent first.

Gradient Descent. Gradient Descent is a popular optimization technique which can be used to train almost all the different types of Neural Networks. A gradient measures the degree of changes of a function with respect to a small variation of one or more of its variables. Mathematically, Gradient descent is a convex function whose output is the partial derivative of a set of its input parameters. Starting from an initial value, Gradient Descent is run iteratively to find the optimal values of the parameters which minimize the value of an objective also called the cost function. In deep learning, the cost function is usually the average of the loss function for each example in the training dataset. We assume that $E_i(\theta)$ $(i = 1, ..., n)$, is the loss function associated to a sample index i of a training dataset with n samples. Then, the objective function will be given by the formula 1.9 :

$$E(\theta) = \frac{1}{n}\sum_{i=1}^{n} E_i(\theta),\ where\, \theta\ is\ the\ model\ parameter. \tag{1.9}$$

The gradient of the objective function at θ noted $\nabla_\theta E(\theta)$ is given by Equation 1.10.

$$\nabla_\theta E(\theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta E_i(\theta) \tag{1.10}$$

The computing cost of the Gradient Descent algorithm per iteration for each independent parameter grows linearly with n. Consequently, when the model training dataset is large, the cost of

Gradient Descent for each iteration will be very high.

Stochastic Gradient Descent (SGD) reduces the computational cost of Gradient Descent by uniformly sample $x_{i\in\{1,2,..n\}}$ and compute the gradient $\nabla_\theta E(\theta)$ to update θ using Equation 1.11.

$$\theta_{t+1} \longleftarrow \theta_t - \eta\nabla_\theta E_i(\theta),\ \eta\ is\ the\ learning\ rate. \tag{1.11}$$

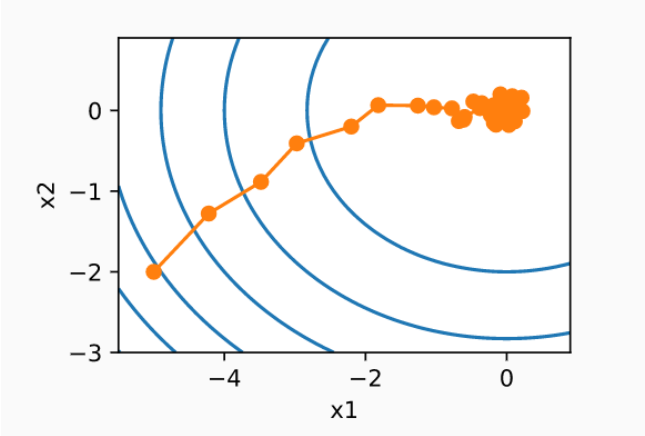


Figure 1.19: Trajectory of parameter in the SGD.

Figure 1.19 shows the trajectory of input parameters from the initial values to an optimal state. the learning rate will control the speed or length of the move from one parameter point to another. As we can observe in Figure 1.19, if the learning rate is too slow, the model will converge very slowly and if the learning rate is too high, the model is likely to diverge, going far beyond the optimal solution.

Generally, in deep learning, θ represents weights (w) or biases (b). Therefore, the updates of w and b will respectively be done following Equation 1.12 and Equation 1.13

$$w_{t+1} \longleftarrow w_t - \eta\nabla_\theta E_i(w) \tag{1.12}$$

$$b_{t+1} \longleftarrow b_t - \eta\nabla_\theta E_i(b) \tag{1.13}$$

The momentum was further proposed in [58] to avoid stepping on a local optima. SGD using the momentum is defined by Equation 1.14 [59].

$$\theta_{t+1} \longleftarrow \theta_t - \eta(\nabla_\theta)_i E_i(\theta) + \alpha\Delta\theta \tag{1.14}$$

α represents the momentum and $\Delta\theta$ is the previous update, i.e., $\Delta\theta = \theta_t - \theta_{t-1}$.

Depending on whether the gradient is computed at each iteration on the whole dataset, or just on a subset of the dataset, we distinguish :

- Batch gradient descent : all the training data is taken into consideration to take a single step. In other words, the mean of the gradient is computed for all the training data and the value obtained is used to update the model parameters;
- Mini-batch gradient descent : The dataset is devided into n subsets called batches. A batch is considered to take a single step, i.e., instead of computing the gradient on the whole dataset, it is computed for each batches as the average gradient of all the batch examples.
- Stochastic gradient descent : we consider just one example at a time to take a single step. gradient is computed and models parameters are updated after each sample passes through the model.

1.3.4 Back-Propagation algorithm

The back-propagation algorithm was originally introduced in 1960 and only became popular in 1989, thanks to the work done by Rumelhart, Hinton and Williams in [58]. The algorithm is used to train various kinds of neural networks through a process called chain rule. After each forward pass, through a network, the backpropagation algorithm performs a backward pass to update the model's paramaters (weights and biases).

Let's explain the backpropagation algorithm based on the sample network defined in Figure 1.20, where W^l represents the matrix of synaptic weights connecting layer $l-1$ to layer l, a^l_j represents the activation of unit j in layer l, z^l_j the preactivation of unit j of layer l and b^l is the vector of biases of layer l.

In general, the preactivation and the activation of neurons of layer l in matrix vector notation is given Equation 1.15 and Equation 1.16.

$$z^{(l)} = W^{(l)}x + b^{(l)}\ (the\ preactivation) \tag{1.15}$$

$$a^{(l)} = f(z^{(l)}),\ f\ is\ the\ activation\ function \tag{1.16}$$

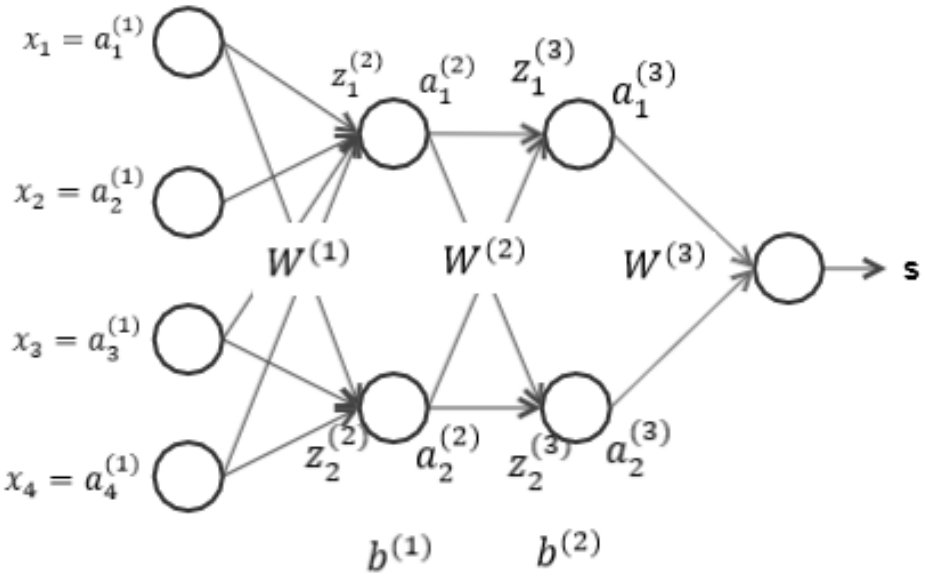


Figure 1.20: Sample network with 4 layers.

The forward pass applied to the neural network in Figure 1.20 will be as follows : $x = a^{(1)}$

$z^{(2)} = W^{(1)}x + b^{(1)}$ preactivation values at hidden layer 1.

$a^{(2)} = f(z^{(2)})$ activation value at hidden layer 1

$z^{(3)} = W^{(2)}a^{(2)} + b^{(2)}$ preactivation value at hidden layer 2.

$a^{(3)} = f(z^{(3)})$ activation value at hidden layer 2.

$s = W^{(3)}a^{(3)}$ Output layer prediction.

The backpropagation algorithm *repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector* [58].

The backpropagation relies on the gradient calculation. The gradient gives insights on how much a parameter needs to change to minimize the cost function.

However, only the output layer knows the classification error. Hidden layers and input layers are not aware of the overall model error. The back-propagation uses a chain rule to make the hidden layers then the input layer be aware of the gradient of the cost function.

For a single weight w_{jk}^l, the gradient is :

$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial E}{\partial z_j^l} \times \frac{\partial z_j^l}{\partial w_{jk}^l}$ (the chain rule)

$z_j^l = \sum_{k=1}^{m} w_{jk}^l a_k^{l-1} + b_j^l$ (by definition, m is the number of neurons in layer $l-1$)

$\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1}$ (by differentiation/calculating derivative)

$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} a_k^{l-1}$ (final value)

The same procedure can be applied to b_j^l :

$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l} \times \frac{\partial z_j^l}{\partial b_j^l}$ (the chain rule)

$\frac{\partial z_j^l}{\partial b_j^l} = 1$ (by differentiation /calculating derivative)

$\frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} 1$ (final value)

the partial derivative $\delta_j^l = \frac{\partial E}{\partial z_j^l}$ is called the local gradient.

The partial gradient can easily be computed in each layer using the chain rule, and knowing the partial gradient, we can then optimize the model parameters accordingly.

The principle of the optimizing algorithm is as follows :

while (end condition is not met) :

$w \longleftarrow w - \eta \frac{\partial E}{\partial w}$

$b \longleftarrow b - \eta \frac{\partial E}{\partial b}$

end

η is the learning rate.

- the values of parameters w and b are randomly initialized;
- w is the weight matrix and b the vector of biases;
- the algorithm ends when the objective function E is minimized.

Chapter Summary

In this chapter, we presented the theoretical knowledge of the domain surrounding our research, that is the Natural Language Processing especially Natural Language Understanding which is referred to as the first component of Natural Language Processing.

We described the various tasks of natural language processing including lemmatization, tokenization, POS-tagging, etc.

We also described the various method used to transform texts into meaningful representations to be easily processed by computers. Among this methods we described one-hot encoding with

its variants and word embedding representations (skip-gram and cbow).

Some deep learning tools used to process NL were also discussed, especially CNN and RNN which are often used for text processing. CNN are used for text classification and RNN are better suited when the sequence (order of appearance) of words matters.

The next two chapters will be dedicated to the two aspects of the methodology presented in the introduction namely : the skills prediction model and the explanation method.

CHAPTER 2

SKILLS PREDICTION MODEL

A resume is generally a document highlighting the experience and the qualifications/competences acquired by a person. This document is the joint point between the job posting and the application. Multiple competences can emerge from the qualifications, background and skills provided by the candidate, though they have not been explicitly stated in his resume. However, A recruiter can easily infer those competences from business knowledge and information extracted from the resume.

The aim of this chapter is to build a model capable of inferring those competences from features (terms) extracted from resumes. To achieve this goal, We use deep learning to train a model to recognize features which characterize each high-level competency. The overall architecture is built using a dataset of expert resumes labeled with their occupations which we will consider as high-level competencies. Since a resume may express multiple competencies at the same time, the problem of identifying those competencies can then be modeled as a multi-label classification task.

This chapter is organized as follows :

1. First, the related works about skills extraction from text documents are discussed;
2. After, a brief background on multi-label classification is presented with the various approaches used to solve multi-label classification problems;
3. Then, the architecture we proposed to solve the multi-label resume classification problem is described along with its components;
4. Finally, the methods used to “preprocess” the training data and build the components of the architecture are explained.

2.1 Related Work

Skills identification from resumes is a critical task when building job recommender systems. It is also useful for automatically building experts competence profiles and companies competencies knowledge-bases. The aim of skills identification is to automatically find the skills expressed in resumes by expert. The main challenges of skills extraction are :

1. A skill may be mentioned in different forms or in terms of synonyms (e.g. cpluplus, C++, programming, scripting, etc.) in resumes or job description;
2. There could be skills that may not be specified in a candidate's resume but can be deduced from business knowledge.

Several works have been proposed to tackle this problem. These methods can be grouped in three categories :

- Explicit skills identification methods which identify skills that are explicitly mentioned in resumes;
- Implicit skills identification methods which are capable to infer implicit skills from features extracted from resumes;
- And hybrid methods.

2.1.1 Explicit skills identification methods

Explicit skills identification methods or simply skills extraction methods parse the input resume and identify terms related to skills. This means that the resume must explicitly contains those terms. Many parsing algorithms have been developed to tackle the problem of skills extraction from resumes :

Shiqiang et al. [60] have proposed a personalized job-resume matching system, which could help job seekers to easily find appropriate jobs. They created a finite state transducer based information extraction library to extract models from resumes and job descriptions. Then, they defined a new statistical-based ontology similarity measure to match the resume models with the job models. The extracted model consists of degrees, majors and skills. The system developed by the authors uses information extraction techniques to parse job descriptions and resumes, and gets information such as skills, job titles, and education background. A domain specific ontology is used

to construct the knowledge base, which includes the taxonomies that support job-resume matching. The resume model includes specialities, working experiences and education background, all extracted from the resume. The job model is built from information extracted from job postings and consists of the same types of information extracted from the resume descriptions. The authors state that the preprocessing stage involves parsing, segmenting, and tokenizing to extract relevant content from jobs and resumes. The self-generated ontology is used to assign semantic labels to each token derived from the tokenization process. However, as stated by the authors, a word can have different meanings depending on the context. The method is said to disambiguate words but the disambiguation process is not explicitly described by the authors. In addition, the job model does not handle typos nor derivational relationships between words.

Men Zhao et al. [7] proposed a combination of Named Entity Recognition (NER) and Named Entity Normalisation (NEN) to identify skills from texts, considering them as named entities. The approach consists in annotating a set of resumes with skills they contain in order to build a dataset that will be used to train a Named Entity Recognition model capable of identifying the set of skills contained in a text resume. This approach has the potential to recognize all the skills mentioned inside the resume and can be used to build a kind of resume summary containing the list of skills held by the expert. But generally, Named Entities Recognition needs huge amount of annotated corpora to perform well; and it is difficult to manually build such datasets. To tackle this problem, they proposed a method to automatically annotate the resumes using Wikipedia Common Categories, which they consider as a taxonomy of skills. But, the problem now resides in the precision and the completeness of the labeling. Further in [8], the same authors added a skill entity word sense disambiguation component which infers the correct meaning of an identified skill by leveraging Markov Chain Monte Carlo (MCMC) algorithms. The proposed system handles name variations such as Artificial Intelligence which can be written in its plural form (Artificial Intelligences), in its abbreviated form (AI) or even written with typos (Artificially Intelligent). The System is also capable of recognizing unspecified skill entities by leveraging semantic context. The main problem of their approach is the accuracy of the taxonomy since its is built automatically. The authors acknowledged that fact and said that they will improve the taxonomy continually in order to improve the overall model performance.

Sayfullina et al. [11] have proposed a phrase-matching-based approach which differentiates between soft skill phrases referring to a candidate vs something else. According to Collins dictionnary, soft skills are desirable qualities for certain forms of employment that do not depend on acquired knowledge. E.g. common sense, ability to deal with people, and a positive flexible

attitude. The method proposed by the authors to identify soft skills is similar to the one used to recognize Named Entities. Indeed, soft skills are tagged in the corpus of resumes and a machine learning algorithm is used to learn a model capable of predicting new and unknown soft skills. A word2vec representation is used to encode each token derived from the tokenizing stage. The overall method was experimented using LSTM and CNN and the best performance was obtained with the CNN model. As for the preceding method, this approach needs to perform well, huge amount of accurate annotated corpora which is difficult in practice.

The common point with all the preceding methods is that they all parse input resumes and extract skills from them. This means that the resume must actually contain those skills. But in practice, some skills can be deduced by the recruiters when reading the resumes, even if they do not appear explicitly inside the resume description. For example, an expert's resume might contain basic skills such as *(css, html, javascript, ...)* which normally would lead a recruiter deducing that the expert has the profile of a "web developer". The approaches which consist in just parsing the resumes cannot be used to find implicit high-level skills.

2.1.2 Implicit skills identification method

Implicit skills extraction is concerned with the identification of non explicitly mentioned skills from resumes. Indeed, a resume may express certain skills that can be identified by an expert, though they were not explicitly mentioned by the owner.

Kiyimaki et al. [61] proposed *Elisit*, a graph-based approach to skills extraction from resumes. They use the hyperlink graph of *wikipedia* and skills extracted from *linkedIn* to associate skills to documents. They first analyse the input document with a vector space model in order to associate it to a *wikipedia* article. From this initial article, they use Spreading Activation Algorithm[62] on the hyperlink graph of *wikipedia* in order to find articles that correspond to *LinkedIn* skills and are related to the initial pages. As stated by the authors, developing a completely automatic optimization scheme for this model selection task is difficult because of the number of different parameters, the size of the Wikipedia graph and the heuristic nature of the whole methodology. That is why they evaluated their model manually.

2.1.3 Hybrid methods

We have proposed in [63] a hierarchical method for skills extraction from resumes in PDF format, which consists in segmenting a resume in its sections and use an appropriate method to extract

skills from each section. The techniques used to identify explicit skills include : n-grams selection, domain dictionary, and a grammatical rule. We also proposed a multi-label architecture to predict implicit skills. The multi-label model built consists of multiple Support Vector machine (SVM) that infers implicit skills from explicit skills extracted using the above mentioned NLP techniques. A limit of the proposed method is that the segmentation algorithm proposed does not perform well for PDF resumes in a tabular format, and the list of implicit skills must be known in advance. Moreover, the use of SVM requires a prior step of extracting basic features/skills.

Recently, Gugnani and Misra have proposed a method to extract both explicit and implicit skills from job and resume descriptions using Document Embedding model. The skills extracted from job descriptions are matched with those extracted from experts' resumes. The work done by the author is similar to what we did in [63] insofar as they combine a set of NLP techniques such as Named Entity Recognition (NER), Part Of Speech Tagging (POS), Word2vec and the use of a skill dictionary to identify skills from texts. However, the method to extract implicit skill diverge. To identify implicit skills, documents are encoded using a Document embedding technique. For each document (job or resume), the list of skills identify in the 10-most similar documents are retained. Skills that are not in the target document are considered as implicit skills to the target document. However, the main problem with this approach is that, there is no guarantee that skills extracted from all the 10-most similar documents are related to the target document. Indeed a document may appear in the list of the 10-most similar documents without actually being similar to that document. In addition implicit competences as defined by authors are not necessary high level competences; instead, they are competencies extracted from related documents.

Explicit skills extraction methods whether they are based on ontology, Named Entity Recognition or others are generally self explainable since a domain expert might easily assess the validity of the skills extracted while looking through the expert's resume.

Conversely, implicit skills extraction methods which are generally based on supervised learning can suffer from the problem of opacity. And from now none of the work explored in the literature has attempted to address the problem of explanability of skills identification models. We do think that an explainable skills identification model might be more trustful to recruiters since the hiring process is a critical process which helps an enterprise bridging the skills gap.

2.2 Background on Multi-label text classification

2.2.1 Problem statement

Definition 2.2.1. Feldman and Sanger define automatic text classification as a task which consists in categorizing an instance in a predefined set of classes. Given a set of classes (subject, labels, etc.) and a collection of text documents, classification is the process of assigning the right label to the right document [64].

Classify a data (or a sample) means to assign it a unique label (single-label classification) or more than one label (multi-label classification) [65]. Multi-label classification is then a variant of the classification problem where more than one label can be assigned to a single instance. Conversely, in single-label classification, an instance can be assigned to only one class. Mutli-label classification generalizes single-label classification and there is no constraint in how many of the classes an instance can be assigned to. More formally,

Definition 2.2.2. Given a dataset D and a set of classes Y, single-label classification consists in assigning to each instance x of X a unique label $y \in Y$. The single-label dataset D consisting of n samples $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where $(x_i, y_i) \in X \times Y$.

Models to solve the single-label classification problem includes Support Vector machine (SVM), k-Nearest Neighbours (k-NN), Artificial Neural Network (ANN), Decision Tree, ...

Definition 2.2.3. Given a set of N documents $X = \{x_1, x_2, ...x_n\}$ and $Y = \{y_1, y_2, ...y_n\}$, a set of n-labels. The multi-label classification task consists in learning a function $h : X \longrightarrow 2^Y$ from a training dataset $D = (x_i, y_i),\ 1 \leq i \leq N$ such that for each new instance $x \in X$ to classify, the multi-label classfier predicts $h(x) \subseteq Y$ as the set of classes to which x belongs.

2.2.2 Solving the multi-label classification problem

There are two main approaches to solve the mutli-label classification problem :

- The first one is problem transformation which consists in transforming the multi-label classification problem into one or more single-label classification problem(s) which can be solved by applying one of the techniques used to solve single-label classification tasks on each problem;

- The second approach consists in adapting the single-label algorithm to make it solve the multi-label classification task.

In this section, we will focus on some of the problem transformation techniques as they are widely used and are reported to be more effective to solve multi-label classification tasks [64]. According to Jesse in [64], the various multi-label problem transformation-based methods are inspired from two approaches namely : Binary Relevance (BR) and Label Powerset (LP).

Binary Relevance

The principle of the Binary Relevance approach is to transform the multi-label problem into n (n being the total number of labels) independent binary classification problems where each resulting problem concerns a single label taken from the set of labels in the dataset [64].

Therefore, a binary classifier is trained using a dataset corresponding to one of the labels and the outputs of each binary classifier are merged to form the overall multi-label output prediction.

Formally, for each class y_i, a Binary Relevance method starts by building a corresponding single-label training set D_j based on whether a sample belongs to the class y_j or not. D_j is given by Equation 2.1 where x_i is an instance of the training set and Y_i is the set of labels of x_i in the multi-label dataset.

$$D_j = (x_i, \phi(Y_i, y_i)), 1 \leq i \leq n \tag{2.1}$$

where $\phi(Y_i, y_i)$ is given by Equation 2.2.

$$\phi(Y_i, y_i) = \begin{cases} 1 & \text{if } y_i \in Y_i \\ \\ 0 & else. \end{cases} \tag{2.2}$$

The advantage of this method is that it is intuitive, simple to implement and the number of classifiers scales linearly with the number of classes. In addition, the predictions of each binary classifiers can be computed in parallel. The drawback is that a classifier has to be built for each class and each classifier is supposed to take its decision idependently of others.

Label Powerset

The principle of the Label Powerset approach is to transform the multi-label classification problem into a unique single-label classification problem by assigning a new label to each combination of labels in the label set.

If let's say we have the set of labels $Y = \{y_1, y_2, y_3\}$, the label powerset will form new labels based on the different combinations of labels in Y that is $l_1 = \{y_1\}$, $l_2 = \{y_2\}$, $l_3 = \{y3\}$, $l_4 = \{y_1, y_2\}$, $l_5 = \{y_1, y_3\}$, $l_6 = \{y_2, y_3\}$, $l_7 = \{y_1, y_2, y_3\}$.

This approach is also simple to implement and does not need to built many classifiers. However, the size of the label space scales exponentially with the number of actual classes. In addition, the model will only be capable of predicting only the combinations of labels which appear in the training set.

Chain of Classifiers

The Chain of Classifiers (CC) approach involves the training of n binary classifiers as Binary Relevance (BR) does. Unlike Binary Relevance (BR) classifiers are linked each to other to form a chain. The space of attribute for the class j is extended with the $j-1$ preceding outputs in order to preserve the correlations between the class l_j and the other classes.

The advantage of this approach is that the chaining serves as to forward informations between the classifiers, thus making the classifiers being aware of the correlations between classes. The drawback of this approach is the risk of propagating errors when the preceding classifiers have predicted the wrong output. In addition loosely correlated classes are put together to build classifiers, which might noisy the training. The chaining among the classifiers make it difficult to parallelize the computation of the different classifiers.

2.2.3 Evaluating a multi-label classifier

Let n be the number of instances in the test-set, A_i (resp. Z_i) the set of actual (resp. predicted) labels for the instance i. Three measures are generally used in the literature to evaluate a multi-label classifier [66]:

The accuracy

The accuracy for an instance i measures the proportion of labels which were predicted correctly to the total number (predicted and actual) of labels of that instance. Then the accuracy of the multi-label architecture is the average of accuracies calculated over all the instances. The global accuracy is calculated by the formula 2.3.

$$A = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \cap Z_i|}{|A_i \cup Z_i|}. \tag{2.3}$$

The precision

The precision for an instance i measures the proportion of labels which were predicted correctly by the model to the total number of predicted labels of that instance. The overall precision is the average of the precisions evaluated over all the instances. The global precision is calculated by the formula 2.4.

$$A = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \cap Z_i|}{|Z_i|}. \tag{2.4}$$

The recall

The recall for an instance measures the proportion of labels which were predicted correctly by the model to the total number of actual labels of that instance. The overall recall is the average of recalls evaluated over all the instances. The global recall is calculated by the formula 2.5.

$$A = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \cap Z_i|}{|A_i|}. \tag{2.5}$$

2.3 Multi-label classification architecture model for skills prediction

The first step in our method is to design a model capable of predicting a set of skills from an input resume. Since a resume might express multiple skills, we propose to handle the problem as a multi-label classification task. Figure 2.1 describes the architecture of the multi-label skills prediction model.

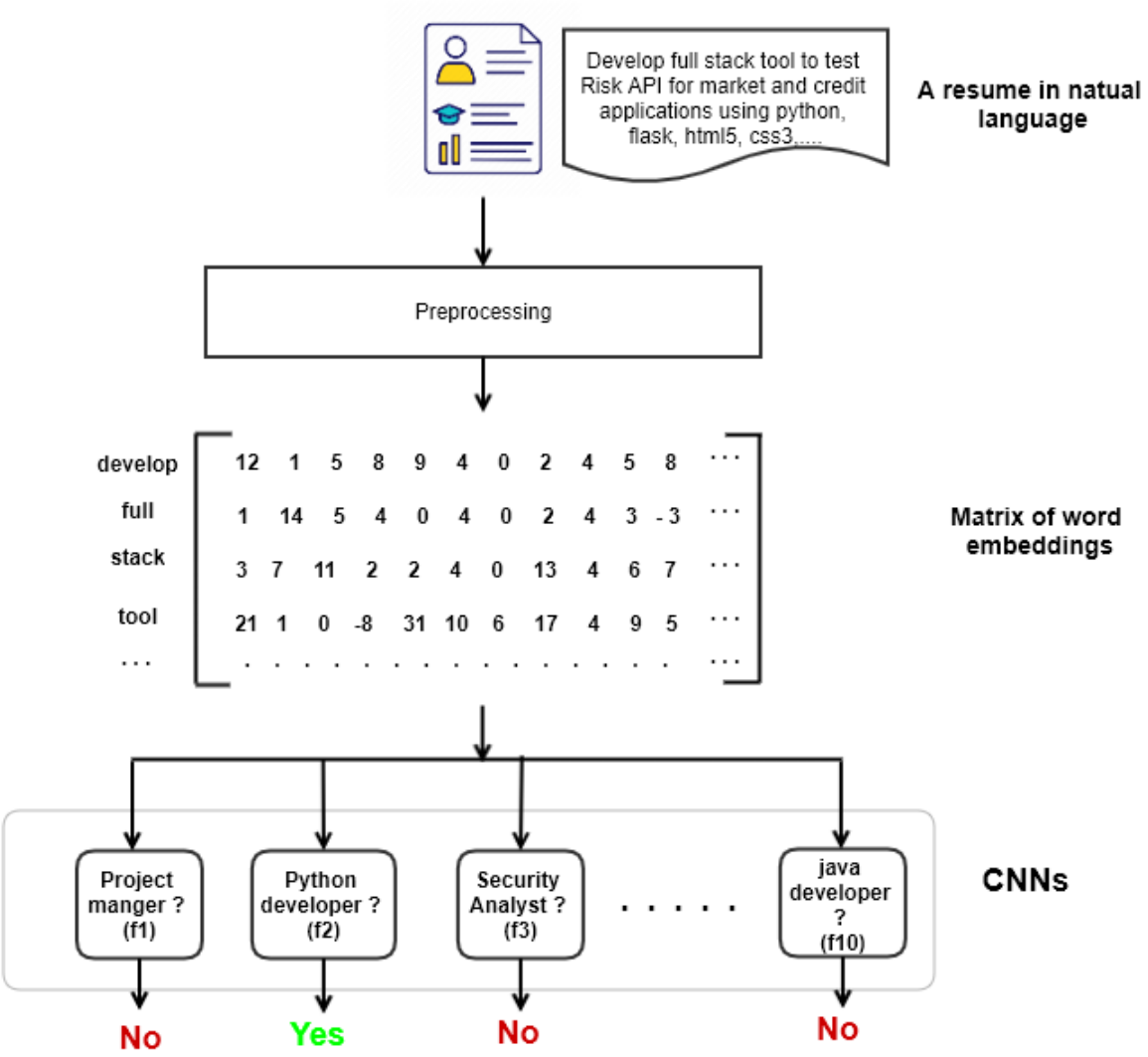


Figure 2.1: Multi-label skills prediction architecture model.

The architecture defined in Figure 2.1 consists of two main components :

1. A preprocessing component which goal is to transform a raw text-resume into a matrix which will serve as input to the CNN; and,

2. A multi-label classifier component consisting of a set of binary CNN classifiers.

2.3.1 Preprocessing component

When a raw text resume is passed as input to the model, the preprocessing component (preprocessor) transforms this resume into a matrix before passing it as input to the CNN classifiers. The

preprocessing consists in using a domain dictionary and a word embedding model to filter and embed the word vectors of a resume into a numerical matrix. We consider all the resume matrices to have the same length L which is defined empirically. The transformation of a resume from text to matrix operates as follows :

1. Initially, an empty row matrix M is created;
2. Repeat the followings for each term w in the resume, until the length of the matrix is reached or there is no other word to proceed :
   - if w is found in the domain dictionary and if there is an entry corresponding to the word w in the word embedding model then :
     (a) convert w into a numerical vector v using the embedding model;
     (b) append the corresponding vector v to the bottom of the matrix M.
3. If the length of the matrix is less than L then zero-padding is added to the bottom of the matrix to reach L.

At the end of these steps, we have the matrix of the resume which then goes as input to each base CNN classifier.

2.3.2 CNN classifiers

The second component of the architecture, consists of multiple independent binary CNN classifiers. Each CNN is used to classify resumes according to a single class or competence. In other words, a base CNN classifier f_j is designed to predict whether an input resume x belongs to the class c_j or not. The number n of base classifiers corresponds to the number of competences (classes) in the dataset (in practice, recruiters may know in advance, the sets of competences they would like to identify from resumes). In the rest of this work we will sometimes use the terms class and competence interchangeably since a class refers to a specific competence.

The prediction of the multi-label model then consists of the union of the competences predicted by the different base classifiers as specified in formula 2.6.

$$C(x) = \bigcup_{j=1}^{j=n} c_j[f_j = 1] \tag{2.6}$$

We choose CNN as base classifiers for two main reasons :

1. First, CNN inputs generally consist of raw datas which are less subject to information losses as compared to other models which generally require handcrafted features;

2. Secondly, convolutional filters can easily be analyzed for explanation purpose.

Each base classifier takes as input a matrix representation of a resume and predicts a value corresponding to the probability that the resume belongs to a target class or not.

Figure 2.2 shows a view of the architecture of a base CNN classifier. This architecture is inspired from the architecture proposed by Kim, Yoon[15] and consists of an input layer, a convolutional layer followed by a max-pooling layer, a ReLU layer and a fully connected layer.

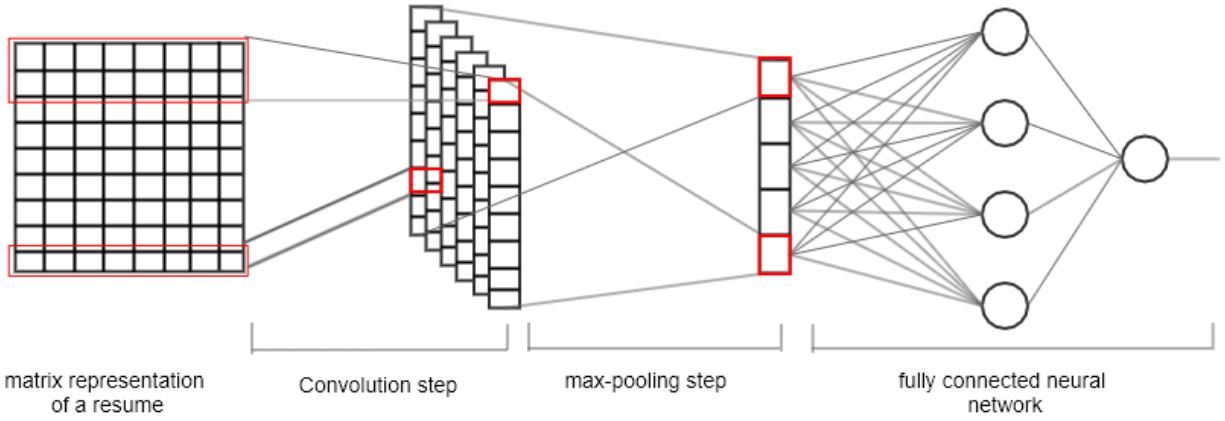


Figure 2.2: CNN architecture for text classification.

This architecture is also referred to as One-Dimensional Convolutional Neural Network (1D-CNN). Each layer of the 1D-CNN realizes specific operations.

The CNN Input layer The CNN takes as input a sequence of word embedding vectors representing the text to classify. A word w can be modeled as a d-dimension vector of numerics : $w \in R^d$. Consequently, a text of length n which is a sequence of n words can be modeled as a matrix M of dimension $n \times d$ ie $M = w_1, w2, ..., w_n \in R^{n \times d}$. An l-word n-gram $u_i =< w_i, .., w_{i+l-1} >$ $(0 \leq i \leq n-l)$ is a sequence of l consecutive words in the input text, $u_i \in R^{l \times d}$.

1D-Convolution layer The convolution layer consists of 1D-convolution filters. A 1-kernel size convolution filter f_j can be modeled as a $d - dimension$ vector (same shape as a word). The kernel size of a filter refers to the height of the filter's sliding window since the width is fixed and corresponds to the word embedding dimension. A filter of kernel size 2 will slide vertically over

two consecutive words at one time as shown in Figure 3.5. The convolution performs the scalar product $< u_i, f_j >$ between an n-gram u_i and a filter f_j. The convolutions of a particular filter f_j with the whole input matrix results in a column vector $F_{,j}$ called the feature map associated to f_j, and for the whole filters, the result is a matrix $F \in R^{(n-l+1)\times m}$ where m is the total number of filters and the columns of the matrix represent single feature maps associated to convolution filters. The values inside the features map are rectified by setting all negative values to zero (Rectified Linear Unit) before going as input to the max-pooling layer. To summarize, the convolution layer can be modeled as a multivariate function $c : R^{n\times d} \rightarrow R^{(n-l+1)\times m}$ which takes as input a matrix of word embeddings and returns a matrix of feature maps.

Max-pooling layer (Global Max-pooling). The global max-pooling filter picks the maximum value in each rectified feature map (each column of F), that is the value corresponding to the word which had the highest convolution score with the filter associated to that features map. The result of the global max-pooling is a $m - dimension$ vector. The max-pooling layer can then be modeled as function $p : R^{(n-l+1)\times m} \rightarrow R^m$ which takes a matrix of feature maps and returns a vector containing the maximum value in each column.

Fully connected neural network (FCNN). The FCNN takes as input the max-pooled vector P (vector resulting from the max-pooling operation) and produces an output vector corresponding to the activation of each output unit. Therefore, the FCNN can be modeled as a function $h : R^m \rightarrow R^c$, c being the number of output units. Let's denote by L the total number of layers in the FCNN, h^i the output vector of the $i\text{-}th$ layer, starting from $h^0 = P$ (the max-pooled vector), and finishing with a special output layer h^L which computes the output of the network. If g is the activation function of hidden units, then, the activations of units in layer k (in matrix-vector notation) is given by Equation 2.7 where b^k is the vector of biases and W^k is the matrix of weights connecting layer $k-1$ to layer k.

$$h^k = g(W^k h^{k-1} + b^k) \tag{2.7}$$

The activation of a single unit i in layer k is given by Equation 2.8

$$h_i^k = g(\sum_j W_{ij}^k h_j^{k-1} + b_i^k) \tag{2.8}$$

Then the activations of the output units will be given by Equation 2.9 (in matrix-vector nota-

tion):

$$h^L = g(W^L h^{L-1} + b^L) \tag{2.9}$$

The 1D-CNN mathematically can be modeled as a function f which realizes the composition of the functions c, m and h : $f = h \circ p \circ c$.

The overall network is trained using the stochastic gradient descent (SGD) algorithm described in chapter 1, section 1.3.3

After having described the overall architecture model and its main components, we now focus on how the various models composing this architecture are built (trained) to solve the multi-label classification problem.

2.3.3 Building the architecture models

To build and train the models composing the architecture described in the above section, we have assembled a set of resumes collected from the Internet. Each resume is labeled with the set of professional occupations of its owner.

The following processing steps are applied in order to build and train the models used in the architecture :

1. First of all, classes (competences) are normalized to have a common representation for competences;
2. Next, a word embedding model is trained from the corpus of resumes using the skip-gram algorithm ;
3. Then, resumes are transformed into matrices using the trained word embedding model;
4. After that, the original muti-label dataset is splitted into n single-label datasets; each used to train a single base classifier;
5. Finally, the base CNN classifiers are trained using an appropriate sub-datasets.

2.3.4 Classes normalization

The first step before building the models is to normalize the competences. In fact, input resumes consist of raw text resumes from the IT domain and are written in HTML format. We consider

occupation titles as high level skills (java developer, project manager, ...): the resumes have been labeled with those titles. However, occupation titles are not normalized: experts might use different expressions to represent the same occupation. For example: experts might use *DBA, database admin, database administrator or database administration to represent the title "Database administrator"* or *java programmer, java coder, java developer, java engineer, python/java developer to represent the title "java developer"*. Since we have dozens of thousands of resumes, it would be fastidious to normalize all those titles manually. To deal with that problem, we wrote an automated "normalizer" algorithm which operates with a hand written dictionary. The dictionary contains for each normalized competence, a list of expressions that are generally used to identify that competence. Table 2.1 presents an overview of the dictionary.

Table 2.1: Different expressions of occupation titles

Normalized class	Expressions
Python_Developer	python django, python flask, python developer, python programmer, python engineer, python software developer, python software engineer, django developer, django programmer, django engineer, django software developer, django software engineer, python web developer, python application developer, python application engineer
Systems_Administrator	sys admin, sysadmin, systems engineer, system administrator, system admin, systems admin, systems administrator, systems specialist, sys administrator, system engineer, systems engineer
Database_Administrator	database administrator, database engineer, database programmer, database developer, database admin, dba

The normalizing algorithm operates as follows:

1. Iterate over all the expressions in the dictionary and compare each of them to the occupation titles defined in the resume;

2. If an expression matches with one of the occupation titles of the resume then, the corresponding resume is labeled with the normalized competence associated to that expression.

Instead of comparing the whole expression, we compare words because expressions in the dictionary and those in the resume might not match exactly. For example, an expert with a title "developer (java, python)" will be labeled with the competences "java developer" and "python developer" since his occupation title contains all the words of the expression "java developer" and those of "python developer". Comparing words instead of the whole expressions also prevents us from putting inside the dictionary all the different expressions that experts might use to express the same occupation; what would be practically impossible.

In this step, we also consider the hierarchical relationship among the competences [9]. Figure 2.3 shows an example of the hierarchical relationship between the high-level competences: *java developer, python developer, front-end developer, web developer and software developer*. It also shows examples of low-level features which describe each competence. We recall that, if an expert is a java developer, then he is also a software developer. To handle the hierarchical relationship among competences, we use a simple taxonomy derived from [9] and restricted to the set of normalized competences that we have considered as the output classes. Using the skills taxonomy, the initial set of competences of a resume is extended by adding their parent competences. For example, based on taxonomy presented in Figure 2.3 "software developer" generalizes "python developer" and "java developer". Therefore, "Software Developer" skill will be added to the set of competences of resumes labeled with "java developer" or "python developer".

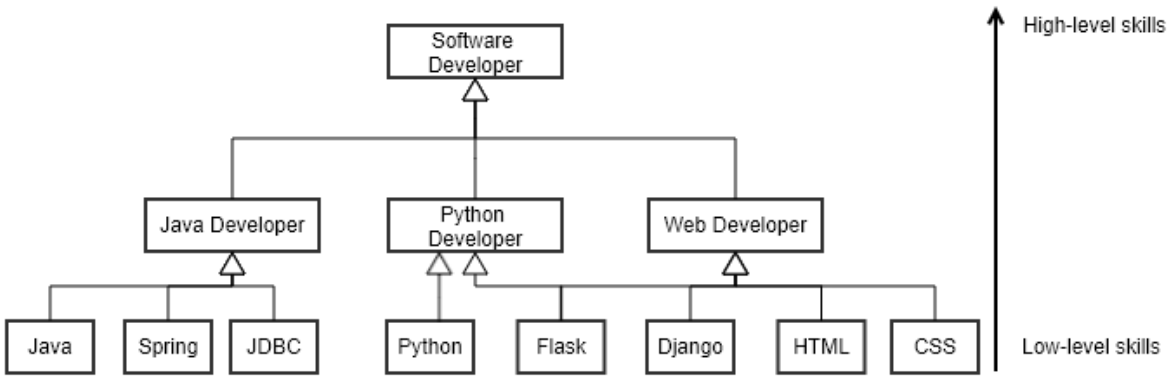


Figure 2.3: Example of hierarchical relationship among competences

2.3.5 Training the word embedding model

Word embeddings are used to build the matrix representation of resumes. The interest of using word embeddings mainly resides in the fact that it captures the semantic of words and preserves the semantic similarity between them. It also prevents the problem of data sparseness when it comes to representing text documents. The effectiveness of word vectors has been proven in natural language processing [67]. Embedding the semantic of words into vectors is particularly useful when working with CNN for text classification, because we want each filter to identify closely related words. Given a word vector, the cosine measure can be used to compute the similarity with other word vectors.

We use the skip-gram [68] algorithm to train our own word embedding model from the resume corpus. It has been proved that context-specific word embedding models better capture the relatedness between words than general models[69]. In fact, context-specific corpora are less subject to the problem of polysemous words[70, 71] : a word can have different meanings depending on the context. For Example the term “java” can mean a coffee or a programming language depending on the context.

Generally, when training word embedding models, two main parameters are considered: the dimension of the word vector and the windows size which refers to the number of words to consider before and after the target word to represent its context. The word embedding model used in this work was built using a dimension of 100 and a windows size of 5. The size of the dimension and the windows size were defined empirically. We observed satisfactory performances using those values. Example 1 shows the most similar words to the term *angular* along with their similarity values. From this example, we observe that the most similar terms to angular (angularjs, zepto, vue, angular4, backbone, react, ...) out of hundreds of thousands of words in the vocabulary are either other written forms of angular (angularjs, angular2, angular4) or other javascript frameworks (zepto, backbone, react, ...). This illustrates the effectiveness of our self-trained word embedding model.

Example 1. Most similar words to the word angular in the resume corpus:

[('angularjs', 0.720), ('angular2', 0.718, ('zepto', 0.7039), ('vue', 0.686), ('angular4', 0.6773), ('backbone', 0.666), ('react', 0.663), ('skrollr', 0.658), ('angualr', 0.656), ('colorbox', 0.654)].

2.3.6 Words filtering and resumes transformation into matrices

CNN were initially designed for images which are represented as 2-dimensional matrices. If we want to use CNN to classify texts, then we will have to convert input texts to matrices. The common method to convert a text into a matrix is to convert each of its words into a vector and assemble them to form a matrix. The detail method is described in sub-section 2.3.1.

In many text classification tasks, the length of the resulting matrix is generally fixed to the length of the resume which has the maximum number of words. If that method is practical when dealing with small documents such as tweets, it can be problematic when dealing with huge documents in terms of number of words. Indeed, a resume could have up to 12 000 words. If we consider that the real numbers in the word vectors are represented in simple precision, and that

the dimension of each word vector is set to 100, then we will need about 4.8Mo of memory to represent a resume matrix and 144Go to represent a dataset of about 30 000 resumes.

To reduce the total amount of memory used, common methods suggest to fix the length of the resume to a smaller value than the maximum length. Therefore, if a resume has a length greater than that value, then extra words will be removed from it to reach the fixed size. If instead, its length is less than the fixed value, then 0-padding will be added at the bottom of the resume to reach the fixed length. However, the problem with just setting the length is that, many relevant terms can be ignored. Instead, in this thesis, we empirically fixed the size of the resume matrices and considered a domain dictionary where relevant words are selected. The dictionary was established using IT skills gathered from *dice.com*[1] and others collected from the corpus of resumes.

2.3.7 Building sub-datasets to train the base classifiers

To form the training set for each base classifier, the original multi-labeled dataset is divided into n single-labeled datasets (n being the total number of classes in the original dataset). Each resulting sub-dataset corresponds to a binary classification problem focused on a unique competence.

To build the single-label dataset D_i, for a target class c_i, the class of each resume example x_j is set to *positive* (1) if in the original dataset the input x_j belongs to the class c_i and to *negative* (encoded by 0) else. This procedure corresponds to the Binary Relevance approach to multi-label classification described in Section 2.2.2). The Binary Relevance approach applied on the problem of skills prediction from resumes is illustrated in Figure 2.4.

Moreover, assuming that we have n classes, and that resumes are equally distributed among them, the dataset used to train a base classifier, will have in average $1/n$ of examples belonging to the target class and the rest belonging to the other classes. For example, if $n = 10$ then, 10% of the examples will belong to the target class, and 90% will be of the other classes. That unequal distribution of examples between the positive class (target class) and the negative class (other classes except target class) can lead the model to a kind of overfitting: the model specializes in learning the features of the negative class in detriment to those of the positive class. Indeed, it has been proved that, learning from class-imbalanced data has a negative effect on the classifier performance [72].

To deal with that problem, we balance the dataset accordingly: all the examples belonging to the target competence are retained and an equal number of examples is randomly selected among the examples of the other classes. The sub-dataset formed is then shuffled before training the

[1] https://www.dice.com/skills

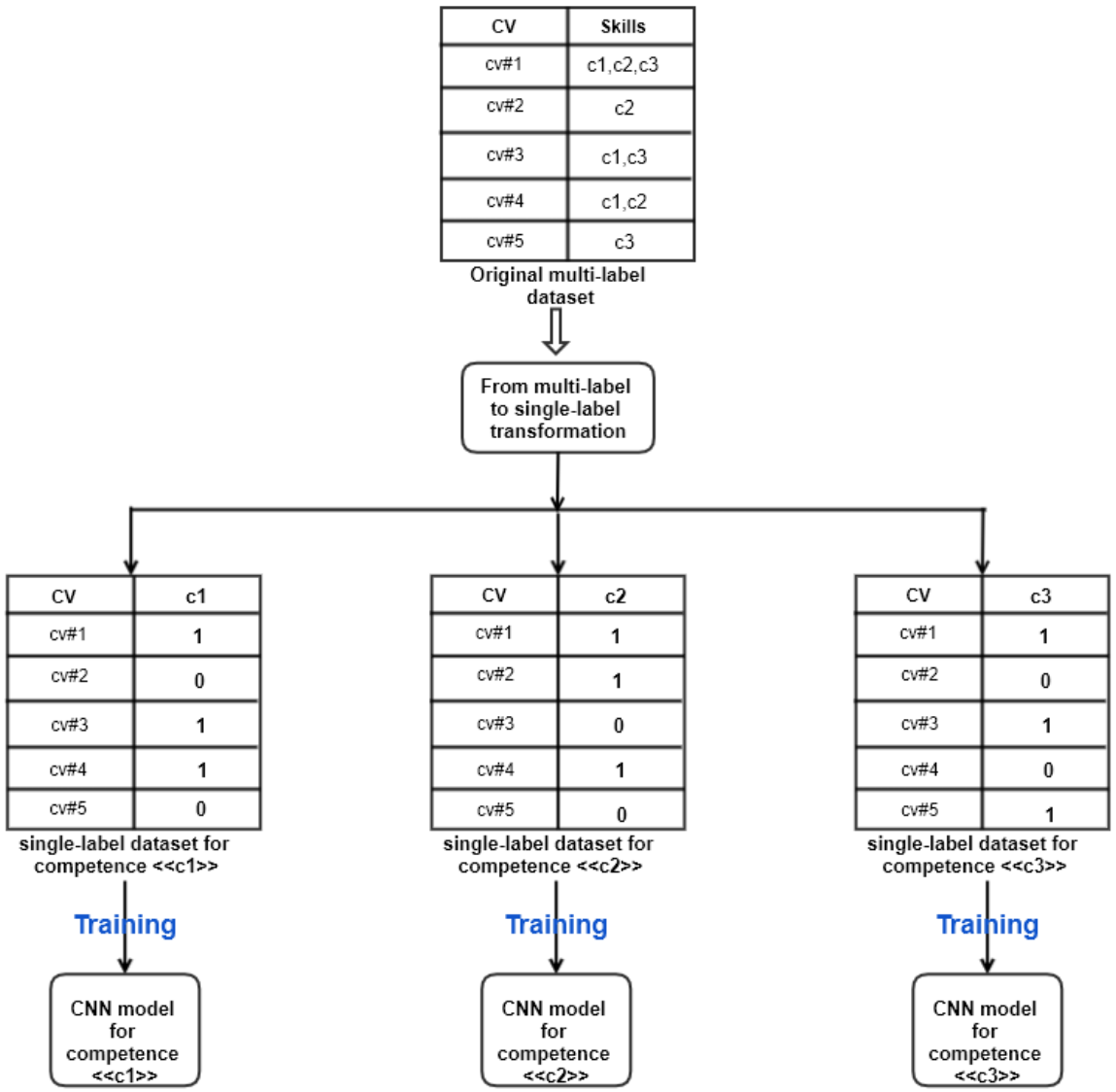


Figure 2.4: From multi-label dataset to single-label datasets.

corresponding base classifier.

2.4 Results and Discussion

In this section, we first present the analysis and description of data before discussing the results of the experiments.

2.4.1 Descriptive data analysis

The experiments were carried out on a collection of 28 707 anonymous resumes available publicly on *indeed.com*[2] website and distributed among 10 classes with the proportions in Figure 2.5. The

[2]https://www.indeed.com (consulted on the 21^{th} *Aug* 2019)

number of pages of the resumes ranges from 1 to 6. The extracted resumes are all from the IT domain and are related to the competences : *Software Developer, Front-End Developer, Network Administrator, Web Developer, Project Manager, Database Administrator, Security Analyst, Systems Administrator, Python Developer, Java Developer*. As shown in Figure 2.5, the skill *Software developer* is the most represented, while *Python developer* is the least represented. We recall that *Software Developer* includes (*Java Developers, Python Developers, Web Developers* and *Front-End Developer*).

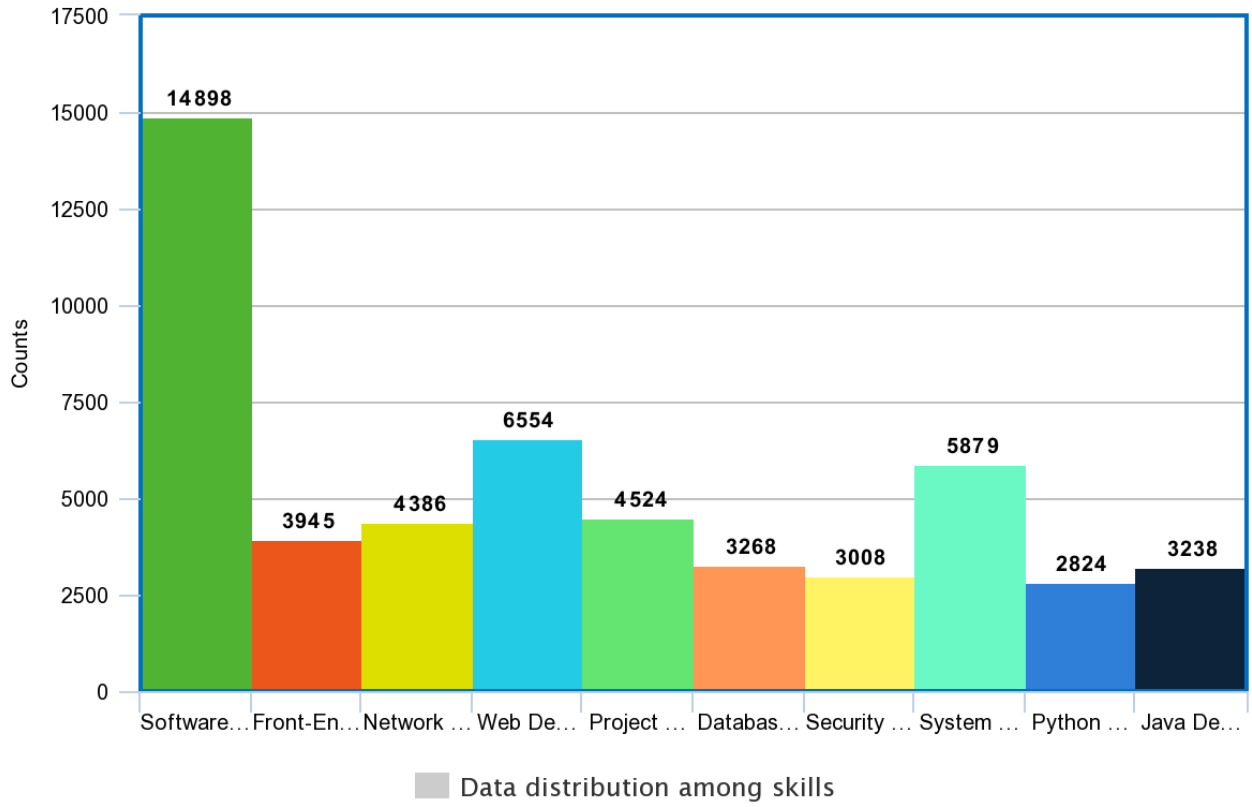


Figure 2.5: Classes distribution in the dataset

A descriptive text analysis has also been done on resumes of each class to find out the most relevant terms describing each class. To achieve this goal, we evaluated the importance of each term with respect to a competence. TF-IDF (formula 2.10) was used to evaluate the importance of a term with respect to a competence. The importance of a term w with respect to a competence c, is calculated by Equation 2.10. The intuition behind this method of evaluating the importance of terms is that, an important term for a competence is a term which appears frequently in resumes labeled with that competence and less frequently in resumes labeled with other competences. In fact, terms which appear frequently in all resumes, are not discriminant enough; that is why the standard frequency is not appropriate since it will highlight several non discriminant terms.

Table 2.2: 10 most important terms per classes.

Classes	The 10-Most important terms
Software_Developer	('jquery', 0.709), ('git', 0.684), ('api', 0.679), ('ui', 0.679), ('css', 0.674), ('bootstrap', 0.673), ('html5', 0.672), ('front', 0.666), ('javascript', 0.663), ('ajax', 0.657)
Front_End_Developer	('responsive', 1.111), ('front', 1.047), ('sass', 1.031), ('css3', 1.026), ('photoshop', 0.976), ('browser', 0.945), ('html5', 0 .941), ('bootstrap', 0.932), ('react', 0.920), ('ui', 0.917)
Network_Admin	('switch', 1.050), ('cisco', 1.036), ('firewall', 0.953), ('lan', 0.937), ('router', 0.891), ('directory', 0.888), ('wan', 0.888), ('vpn', 0.848), ('vmware', 0.842), ('equipment', 0.805)
Web_Developer	('jquery', 0.813), ('html5', 0.789), ('bootstrap', 0.784), ('css3', 0.765), ('front', 0.754), ('responsive', 0.745), ('css', 0.744), ('website', 0.740), ('page', 0.736), ('wordpress', 0.724)
Database_Administrator	('dba', 1.303), ('tune', 1.015), ('rman', 0.967), ('tuning', 0.942), ('replication', 0.940), ('recovery', 0.908), ('index', 0.880), ('backup', 0.859), ('rac', 0.854), ('administrator', 0.799)
Security_Analyst	('vulnerability', 1.339), ('cyber', 1.141), ('threat', 1.119), ('nist', 1.077), ('analyst', 1.074), ('incident', 1.070), ('remediation', 1.021), ('assessment', 0.987), ('risk', 0.971), ('compliance', 0.970)
Systems_Administrator	('vmware', 0.935), ('directory', 0.918), ('active', 0.801), ('hardware', 0.791), ('administrator', 0.787), ('cisco', 0.784), (' switch', 0.761), ('dns', 0.751), ('backup', 0.738), ('firewall', 0.721)
Python_Developer	('django', 1.815), ('flask', 1.618), ('panda', 1.581), ('numpy', 1.546), ('python', 1.412), ('pycharm', 1.409), ('matplotlib', 1 .395), ('scipy', 1.258), ('2.7', 1.229), ('postgresql', 1.227)
Java_Developer	('hibernate', 1.647), ('j2ee', 1.629), ('jdbc', 1.566), ('servlet', 1.560), ('junit', 1.556), ('jsp', 1.522), ('strut', 1.495), ('maven', 1.490), ('log4j', 1.456), ('spring', 1.450)
Project_Manager	('budget', 1.025), ('leadership', 0.738), ('manager', 0.732), ('vendor', 0.730), ('strategic', 0.717), ('stakeholder', 0.708), ('cost', 0.705), ('risk', 0.693), ('scope', 0.684), ('executive', 0.683)

Let D be the corpus of resumes. Let D_i be the corpus of documents which belong to the class i. We consider the following definitions :

Definition 2.4.1. The importance of a term $w \in D$ with respect to a class i is the product of the frequency of that term with respect to the class i, and the inverted document frequency of the term with respect to the class i.

$$TF - IDF_i[w] = TF_i[w] * IDF_i[w]. \tag{2.10}$$

Definition 2.4.2. The frequency of the term w with respect to the class i ($TF_i[w]$) is the frequency of w in the restricted corpus of resumes labeled with the class i. $TF_i[w]$ is calculated by the formula 2.11.

Table 2.3: Overlapping of most important terms describing competences

	Software Developer	Front-End Developer	Network Admin	Web Developer	Project Manager	Database Admin	Security Analyst	Systems Administrator	Python Developer	Java Developer
Software Developer	100	45	0	76	0	1	0	0	37	37
Front-End Developer	45	100	0	65	1	0	0	0	13	10
Network Admin	0	0	100	0	14	13	21	81	0	0
Web Developer	76	65	0	100	1	0	0	0	30	28
Project Manager	0	1	14	1	100	3	17	15	0	0
Database Admin	1	0	13	0	3	100	3	16	1	3
Security Analyst	0	0	21	0	17	3	100	22	0	0
Systems Admin	0	0	81	0	15	16	22	100	0	0
Python Developer	37	13	0	30	0	1	0	0	100	17
Java Developer	37	10	0	28	0	3	0	0	17	100

$$TF_i[w] = \frac{Number\ of\ times\ w\ appears\ in\ the\ corpus\ D_i}{Total\ number\ of\ terms\ in\ D_i}. \tag{2.11}$$

Definition 2.4.3. The inverted document frequency of the term w with respect to the class i is calculated by the logarithm of the inverse of the document frequency of the term w in the counter corpus (corpus of documents that are not labeled with the class i).

IDF_i is calculated with respect to the counter corpus of D_i in order to penalize terms which appear frequently in resumes that are not of the target class i.

$$IDF_i[w] = log(\frac{|D|}{|d, d \in D \setminus D_i|}). \tag{2.12}$$

Table 2.2 shows an overview of the 10 most important terms per class out of about 56000 terms. From this table, we observe that *System_Administrator* and *Network_Administrator* are closely related competences because several of their most important terms (*switch, cisco, firewall, directory, vmware*) are similar. On the other side, we observed that the top-words describing competences *python developer, java developer, security analyst* and *database administrator* do not overlap much with those of other competences. But, the competences *Front_End_Developer* and *Web_Developer* have similar top-terms *(css3, responsive, html5, bootstrap*). These observations are confirmed by Table 2.3 which shows the degree of co-occurrences of the 100 most important terms of each competence. From this table, we can observe that 65 out of the 100 most important terms describ-

ing *Front_End_Developer* and *Web_developer* are the same; and 81 out of the 100 most important terms describing *Network_Administrator* and those describing *Systems_Administrator* are the same. *Software_Developer* has common descriptive terms with other developers related competences because it generalizes them.

2.4.2 Evaluation of each CNN model

First, the accuracies of the base classifiers are evaluated separately. Thereafter, the overall multi-label model is evaluated using appropriate measures. Table 2.4 shows the accuracy of each individual base classifier. The aforementioned results were obtained with the following parameters: the number of filters set to 100; the filters kernel size set to 1; the stochastic gradient descent used as the optimizer; the batch size equal to 64 with 100 epochs, 0.01 as the learning rate and a momentum of 0.1. Those parameters were found through experiments and they produced satisfactory results.

Table 2.4: Accuracy of each base classifier

Classifiers	**Accuracy**
Java Developer	99.18%
Python Developer	99.20%
Web Developer	96.30%
Front-End Developer	95.18%
Network Administrator	94.88%
Systems Administrator	95.65%
Software Developer	99.08%
Security Analyst	99.30%
Database administrator	99.11%
Project Manager	99.29%

The results reported in Table 2.4 show that the model performs very well with an accuracy of about 99% for the competences : *Java Developer, Python Developer, Security analyst, Database administrator*, and *Project Manager*. In the same time, the model predicts the competences : *Web developer, Front-end Developer, Network administrator*, and *System administrator* with an accuracy of around 95%. The high accuracy of the first group can be explained by the fact that those competences can easily be distinguished from others because their low-level features do not overlap much with that of other competences (as discussed in sub-section 2.4.1). Indeed, many of the terms describing java developers are not the same as those describing python developers and others. But many of the terms (81 out of the 100 most important terms) describing network administrators overlap with those describing System administrators because those competences are

closely related (see Table 2.3). As well, terms describing web developers are much the same as those describing front-end developers. Following these observations, some *network administrators* will likely be classified as *systems administrators* and vice versa because both competences involve almost the same basic skills. The same reasoning applies for *web developers* and *front-end developers*. That situation will inevitably lead to ambiguities in both the learning and the evaluation processes because, some examples will actually have the characteristics of a competence but not labeled as such. In that case, the prediction will be evaluated as incorrect even if in fact it could be correct.

2.4.3 Evaluation of the overall multi-label model

To evaluate the skills prediction model, we use the three measures described in section 2.2.3 namely : the precision, the accuracy and the recall. The precision will estimate the probability that a skill predicted by the model for a given instance is correct while the recall will predict the ability of the model to predict all the competences that an input resume actually expresses. The accuracy takes into account both the semantic of the recall and the precision.

Table 2.5: Comparison of our method with other text encoding methods applied on resumes

	Recall	**Precision**	**Accuracy**
word2vec+CNN	98,79%	91,34%	90,22%
doc2vec+LR	94,68%	81,45%	78,63%
one-hot+LR	92,28%	80,11%	78,07%

.

Table 2.5 shows a comparison of the results obtained with our multi-label classification model with those obtained from other models based on different resumes' encoding methods. The recall of 98,79% indicates that our model almost always predicts all the expected competencies. On the other hand, the value of the precision shows that 91,34% of competences are well predicted by the model.

However, if we can rely on the fact that when a resume is labeled by a competence, it means its owner really has that competence, the opposite is not always true. More precisely, the fact that a resume has not been labeled with a particular competence does not necessarily mean that it does not express that competence. An expert can be labeled as a *system administrator* but actually has skills of a *network administrator* or *project manager*. This could explain the gap between the recall and the precision.

Indeed, in many cases, the model predicts a competence based on the words it has identified in the resume. But just because the resume was not labeled with that competence, the evaluation

will mark it as incorrect. So, in practice the precision of the model would be greater than that we evaluated on the test set due to incomplete resumes labeling by experts. In addition, even though a high recall and a high accuracy are preferable in all cases, in the context of job-resume matching, the recall seems to be more important than the precision because job-matching algorithms are not meant to recruit, but to produce a *short-list* of resumes in order to ease the recruiter's work. Thus, the fact that a job recommender system fails to select good candidates is more a major concern than the fact that it accidentally selects bad ones. In the latter case, we can still rely on human to filter and eliminate unfit candidates.

In Table 2.5, our model (word2vec+CNN) is compared to some models built using other resumes' encoding methods and logistic regression as classifier : (1) one-hot encoding + Logistic Regression (LR) as base classifiers and (2) doc2vec + Logistic Regression (LR) as base classifiers. We observe that our model outperforms the model based on "doc2vec encoding" where resumes are transformed into vectors using a self-trained doc2vec model. Our model also outperforms the one based on "one-hot encoding". This is likely because, the matrices of words' vectors better preserve the semantic of words and reduce information loss as compared to "doc2vec encoding" where the whole semantic of the resume is embedded into a single vector or to the "one-hot encoding" where the semantic relationship between words is not taken into account. The classifier used (CNN vs LR) also plays a significant role in the model performance.

These results demonstrate the effectiveness of embedding resumes into matrices and using CNN as base classifiers in the multi-label architecture.

2.4.4 Contribution of word filtering

Here we analyse the contribution of word filtering in the model performance. When describing the proposed method, we said that instead of selecting any word from the resume, we selected domain related words because they might be more informative than common language words. We will call the method which consists in selecting domain words **DWS** which stands for Domain Words Selection in opposition to the standard method where only common stop words are removed.

Figures 2.6 and 2.7 show respectively the comparisons of the evolution of the overall recall and the precision of both methods (DWS and standard) with respect to the length of the resume. From these figures, we observe that when using the standard filtering method the recall is high while the precision is very low. This can be explained by the fact that many common words were selected to encode the resume into a matrix. This assumption is confirmed by Table 2.6 which

shows the number of words per resumes when no filtering is applied, after stopwords filtering is applied, and after only domain words are selected. Those common words a priori cannot suitably discriminate resumes according to their competences; instead, they might introduce noises and classification errors when training the base classifiers. We also observe that the performances of both method increases with the length of the resume : that means a resume representation with much words increases the number of features used for classification thus, improving the overall model performance. In conclusion, we can say that the DWS filtering method presents a better trade-off between the precision and the recall than the standard filtering method.

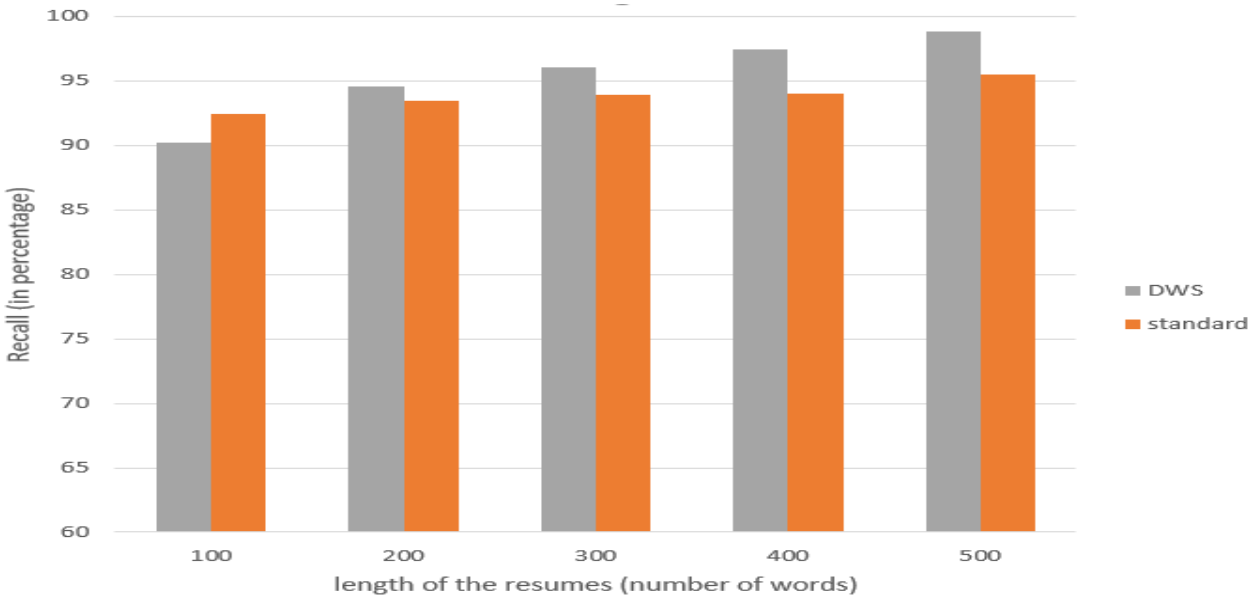


Figure 2.6: Recall of DWS filtering vs that of standard filtering.

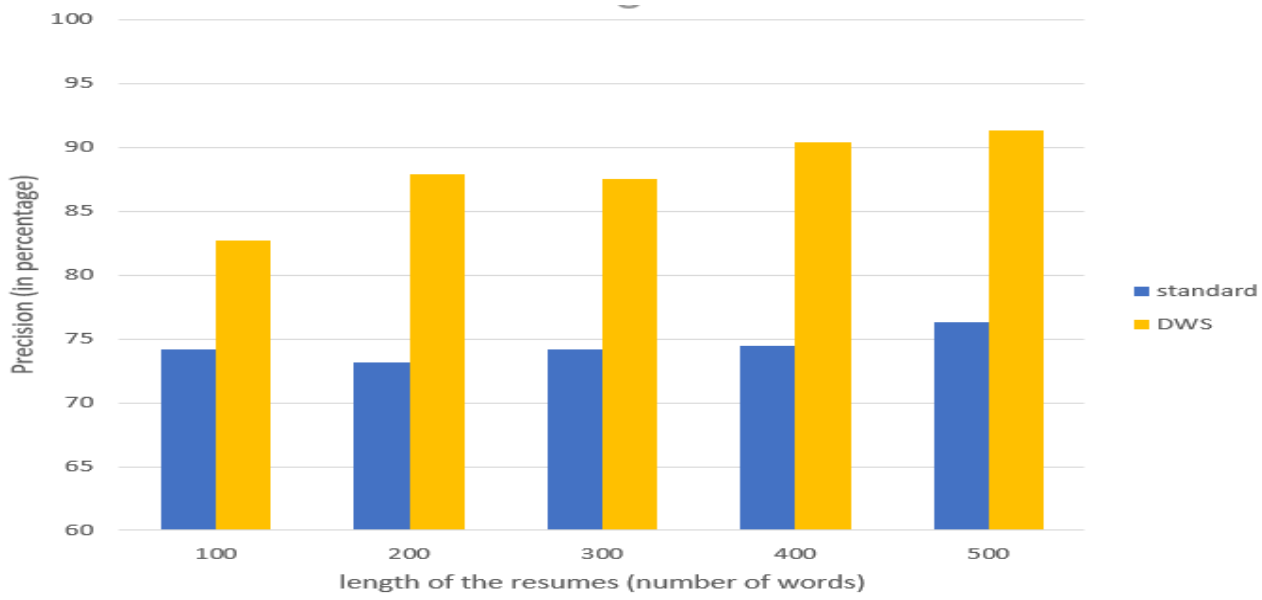


Figure 2.7: Precision of DWS filtering vs that of standard filtering.

Table 2.6: Word counts (wc) per resume after filtering.

Filtering method	min wc	max wc	avg wc
no filtering applied	2	12479	948
stopwords filtering	0	9809	724
domain words selection	0	5526	450

Chapter summary

The skills prediction problem described in this chapter is by definition a multi-label classification problem. The approach used to deal with this problem is the Binary Relevance approach described in section 2.2.2. The choice of binary relevance is due to the simplicity and the intuitiveness of its implementation. In addition, even though the models built are independent each of others, we assume that the correlations between the various competences are captured in the features (terms) which characterize them. In other words, two competences are correlated if they have similar terms describing them. In that case, it would be unnecessary to chain the classifiers using the Chain of Classifiers (CC) approach as the correlations between classes are captured by features used to train each base classifier.

The base binary classifiers used consist of CNN as they have demonstrated impressive performances in text classification. In addition, The functioning of the convolutional filters can easily be interpreted for explanation purposes.

The use of CNN as base classifiers suggests that input resumes must be transformed into matrices which are the convenient input format for CNN. Resumes are transformed into matrices using the Skip-gram algorithm described in chapter 1, section 1.2.3.

The word vectors of each resume are appended vertically to form the matrix of a resume. But resumes in the dataset can have up to 2000 words which will lead to huge matrices, thus decreasing the performances of learning algorithms and increasing the memory required to store the input matrices. To deal with this problem, an appropriate filtering method based on a domain dictionary was used to select only relevant words for the classification. The filtering method has permitted us to reduce the average size of resume matrices by a factor of 50%.

The evaluations realized have demonstrated the effectiveness of the overall skills prediction model reaching 98,79% of recall and 91,34% of precision and achieving more that 99% of accuracy for certain competences.

CHAPTER 3

MODEL EXPLANATION

In human understanding, a decision should be explained. Medical practitioners use diagnostic results to explain medical prescriptions to their patients, while lawyers explain court decisions using facts. Artificial intelligence models have evolved over the years, becoming more and more sophisticated and providing outstanding results in solving complex problems in a wide range of applications such as: text processing, image processing, medical analysis, etc.

In the beginning, researchers were just concerned about improving the accuracy of artificial neural models. Then, deep neural networks (DNN) have emerged providing better results than the past models. Even though the results provided by DNN are satisfactory enough, they still suffer from the problem of opacity. Indeed, they are often considered as black boxes, providing results without being able to explain them. But, entrusting important decisions to a system that cannot be explained, presents obvious risks [12].

The problem of model interpretability has become a major concern in machine learning research communities since some machine learning programs were reported being racist, making social discrimination despite having good results on test data [73, 74]; other programs were found using inappropriate features (background of an image for example) to compute their decisions [73, 75].

This chapter focuses on deep neural model explanation. Model explanation presents more than one benefit, among them: it eases the acceptability of artificial models in critical domains such as medicine, finance, etc.; it provides insights to improve learning algorithms and thus, models accuracy. Models explanation can also provide knowledge in the absence of domain theory.

Techniques to explain artificial models vary from one kind of model to another. While it is easy to interpret a Decision Tree or a model based on Association Rules for instance, the explanation of deep neural networks can be challenging.

In this chapter, we will first present the theoretical knowledge (concepts, techniques and meth-

ods) needed to understand XAI in general and XAI for natural language processing in particular. Thereafter we will present the method that we proposed to explain a CNN model built for text classification.

3.1 Background on XAI for Natural language Processing

Traditional models used to process and analyse Natural Language (NL) were inherently explainable. Those models include: Association Rules, Decision Tree, Hidden Markov Model, Logistic Regression and so forth. However, with the advent of black-box models such as Deep Neural Network (DNN) and word embeddings used as features, we observed an increase in the models accuracy, and sometimes at the expense of interpretability.

The lack of interpretability or transparency in a model decision can be problematic as the resulting decision may not be trusted by the end-user because no justification was provided.

The need to understand and explain artificial model decisions has led to the creation of an emerging field called Explainable Artificial Intelligence (XAI).

The majority of works on the field of XAI reported in [76, 73] has been done for image processing. Only few research works are concerned with text processing. Interpretability methods can be grouped into different categories based on different considerations:

1. Depending whether the explanation is for an individual prediction (local interpretability) or for the whole model prediction process (global interpretability);

2. Depending whether the explanation method is part of the prediction process (self-explaining) or whether a post processing is required to provide the explanation (post-hoc).

Despite some attempts to distinguish model interpretability from model explanability [77], there seems to be not yet a consensus on the difference between both concepts as the majority of methods surveyed use both concepts interchangeably [78, 79]. We will also use both concepts interchangeably in the following developments.

3.1.1 Local versus Global interpretability methods

A local explanation method provides a justification for a model prediction for a specific input [80]. Papers [13, 14] are examples of local explanation methods.

Conversely, a global explanation method aims at understanding the whole predictive behavior of the model, not just for a single instance.

3.1.2 Self-Explaining versus Post-hoc

Explanation methods can also be categorized depending on whether they are part of the prediction process (self-explaining) or whether their generation required post processing.

A self explaining approach generates explanation at the same time as the prediction, using information provided by the model as a result of the process of making that decision. Decision trees and association rules are examples of self-explaining models.

In contrast to self-explaining models, post-hoc methods require additional operation to be done after the predictions are made. According to [80], LIME [14] is an example of a post-hoc method which produces a local explanation using a surrogate model. Table 3.1 shows an overview of the high-level categories of explanation techniques.

Table 3.1: Overview of the high-level categories of explanations (Source: [80]).

Local post-hoc	Explains a single prediction by performing additional operations (after the model has emitted a prediction).
local self-explaining	Explains a single prediction using the model itself (calculated from information made available from the model as part of making the prediction).
Global post-hoc	Performs additional operations to explain the entire model's predictive reasoning.
Global self-Explaining	Uses the predictive model itself to explain the entire model's predictive reasoning (a.k.a directly interpretable model).

3.1.3 Explanability methods

According to [81, 82] and regardless the classification done so far (in Table 3.1) there is at least five techniques used to explain black-box models, and they differ by the way they operate to present the justifications for the model decisions.

Saliency methods. Saliency methods explain the decision of an algorithm by assigning values that reflect the importance of input components based on their contribution to the classifier decision. Works using this method include [83, 84, 85, 86, 87, 88] for image processing and [14, 89, 13] for text processing.

The signal method. Signal methods observe the stimulation of neurons or a collection of neurons. Articles using this method includes [90, 91, 92]. Signal methods are often used for global interpretation.

Verbal method. With verbal methods, phrase, chunks or sentences are provided to the user as explanation [93, 93, 94]. Verbal methods are also post-hoc as they required additional processing to provide sentences as explanation to the end user.

Interpretability by mathematical structures. Interpretability by mathematical structures concerns the use of mathematical models to reveal the inner mechanism of machine learning and neural network models. Papers using this approach to model interpretation include : [95, 96, 97, 98, 99, 100].

Example-driven. Such approaches perform local interpretation by identifying and presenting other instances, usually from available labeled data, that are semantically similar to the input instance. They are similar in the logic to nearest neighbor-based approaches [101] and have been applied to different NLP taks including text classification [102] and question answering [30].

3.1.4 Visualization techniques

There are various ways to present explanations to the end user. The method used to present explanations may depend on the type of the audience. Explanations given to a non expert should be human-readable not technical. Several visualization techniques are reported in the literature.

Saliency Map/ Heat Map

Saliency map provides a visual representation of how much each input feature contributed to the value predicted. It represents graphically the relevance of each feature. In text classification, a heat

map table or simply heatmap shows the importance of each input feature on the score predicted for each output classes. Figure 3.1 is an example of a heatmap.

	DESC	ENTY	ABBR	HUM	NUM	LOC	REL-HUM
senate	0.02	-0.02	-0.00	-0.00	0.01	-0.00	-0.00
witeness	0.01	-0.04	-0.01	-0.00	0.04	-0.01	-0.00
hearings	0.03	0.03	0.02	0.00	-0.03	-0.03	0.01
water-gate	0.01	0.01	0.01	0.00	-0.02	-0.03	0.01
who	-0.15	0.07	-0.17	0.37	-0.28	-0.16	0.51
at	-0.01	0.01	0.00	-0.00	-0.01	-0.00	-0.00
the	0.00	-0.03	-0.07	0.08	-0.02	-0.02	0.11
was	-0.01	-0.02	-0.02	0.04	0.00	-0.01	0.05

Figure 3.1: Sample heatmap.

Saliency projection

Saliency projection is a human friendly way to present justifications for the model predictions. The goal is to highlight on the input text, the most relevant terms which may help the user understanding the model decision. Figure 3.2 shows salient terms projected on a sample resume.

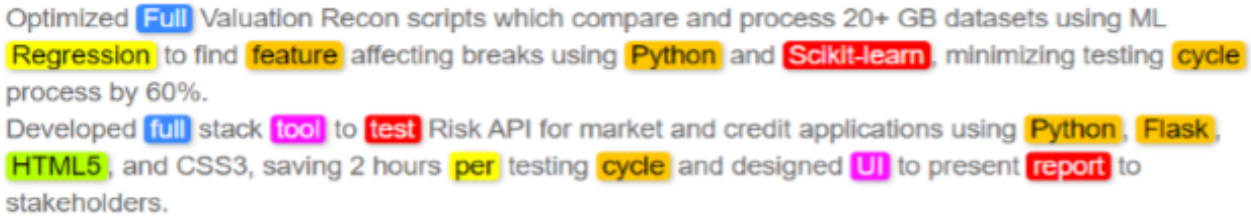


Figure 3.2: Important words projected on an input resume

Natural language explanation

In this category, explanations are given to users in forms of verbal chunks in human-comprehensible natural language. The NL can be generated using appropriate DNN, e.g. by training a model with natural language explanations coupled with a deep generative model [103] or by using simple template-based approaches [30]. Figure 3.3 shows an example of a template-based explanation using natural language.

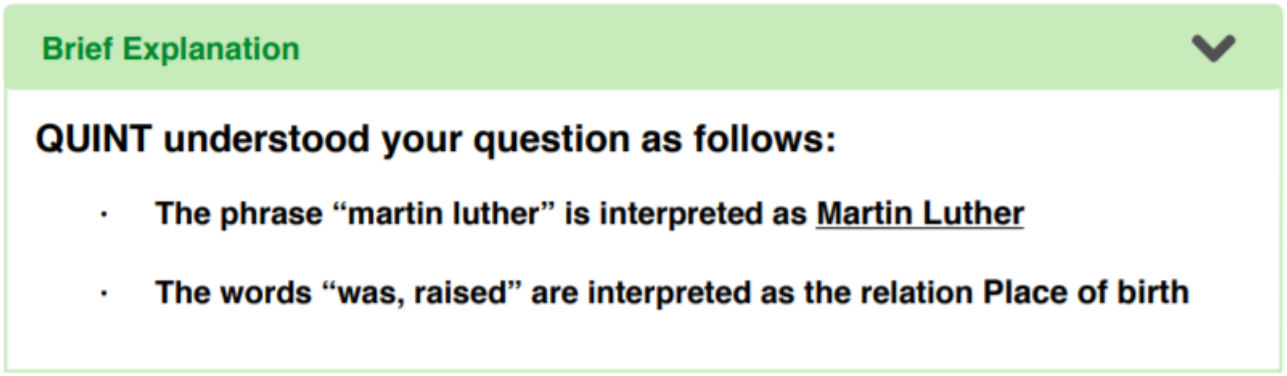


Figure 3.3: Illustration of template-based natural language explanation for QA [30].

Explanation by analogy

Another way to present explanations to users consists in providing them with similar cases where the same decision has been provided. This approach is also referred to as example-driven explanation [104, 102]. Figure 3.4 illustrates an example-driven explanation.

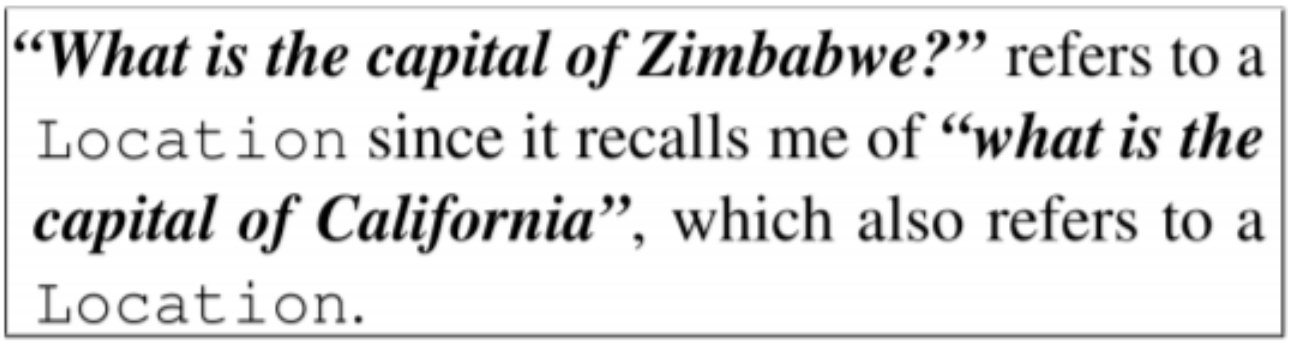


Figure 3.4: Explanation by analogy [102].

3.1.5 Evaluate the explanation

Measuring the quality of an explanation is such a difficult task. Given the young age of the field, there is no standard on how explanations should be evaluated. According to [80], the majority of works in XAI have done only an informal evaluation, while a smaller number of studies looked at more formal evaluation including leveraging ground truth approaches and human evaluation.

Informal examination of explanations

Informal evaluation takes the form of high-level discussions of how the generated explanations align with human perception, knowledge or intuition [80]. This includes cases where results of a single explanability approach is examined [13, 105] as well as when explanations are compared to

those obtained with other approaches [106] such as LIME which is often used as baseline.

Comparison with ground truth

Several works evaluate the explanations provided by comparing them to ground truth data in order to quantify the performance of the explanability method used. However, care should be taken about the quality of ground truth data acquired. In addition, not only one explanation may be valid. Works using this type of evaluation include [107, 108, 109].

Human evaluation

Another way to assess the explanation quality is to ask humans to evaluate the effectiveness of the generated explanations. One interest of this method is that we avoid the assumption that there is only one good explanation that could serve as ground truth. However, to have a precise evaluation, one must have many evaluators to deal with human subjectivity and divergence in annotation. [110, 111, 112] are examples of papers using this technique.

3.1.6 Challenges

As stated earlier XAI is an emerging field of Artificial Intelligence (AI). Researchers and practitioners are divided between the construction of accurate and directly interpretable (self-explaining) model or the explanation of existing black box models. No matter the orientation, building interpretable models is still a topical issue and involves several challenges.

How to explain ? Miller in [75] argued that the main challenge of explainable AI is to explain model the same way as human does. That is why he suggests to inspire from social sciences. He highlighted four major findings that he believes most researchers and practitioners are unaware of :

- Explanations are contrasting : that is people do not ask why event P happened but why event P instead of some event Q. This consideration has important social and computational consequences;
- Explanations are selected : People do not expect explanations that consist of an actual and complete cause of an event. Humans often select one or two causes from sometimes an infinite number of causes to be the explanation;

- Probability probably doesn't matter : While truth and likelihood are important in explanation and probabilities really do matter, referring to probabilities or statistical relationships in explanation is not as effective as referring to causes. According to Miller, the most likely explanation is not always the best explanation for a person, and importantly ;
- Explanations are social : they are a transfer of knowledge presented as part of conversation or interaction and are thus presented relative to the explainer's beliefs about the explainee's beliefs. However, the author precises that this does not imply that explanations must be given in natural language.

Taking into account the cognitive dimension of the explanation along with the social interaction between the explainer and the explainee is a major challenge in XAI.

How to present explanations ? The way an explanation must be provided to users is also an issue. The majority of existing methods score features based on their contribution to the value predicted by the model. But it is still technical and could not be clearly perceived by a non expert.

How to evaluate an explanation As stated earlier, there is no consensus of what a good explanation is or how an explanation can be evaluated. Until now the definition of a good explanation relies on the researcher intuition. However, we do think that one of the key evaluation criterion of an explanation should be the fidelity. That is an explanation must be true to what the model actually learned. This means that an explanation can be unconvincing but actually reflects the inner reasoning of the model. How to assess the fidelity of an explanation is also a major challenge.

3.2 Related Work

The explanation of deep learning models often presented as black box models has become a topical issue in recent years. Most of the works about model explanation reported in these surveys [81, 73] are concerned with image processing and they cannot easily be transposed to text processing due to the discrete nature of texts [13]. Works aiming to interpret machine learning algorithm can be grouped in two major categories according to [81]: interpretability by mathematical structures which concerns the use of these structures to reveal the mechanism of machine learning and neural network algorithms [95, 96, 97, 98, 99, 100]; and the perceptive algorithm which tends to present the user with explanatory elements that can be humanly perceived. The perceptive explainability includes : the saliency method, which explains the decision of an algorithm by

assigning values that reflect the importance of input components based on their contribution to the classifier decision [83, 84, 85, 86, 87, 88]; the signal method that observes the stimulation of neurons or a collection of neurons [90, 91, 92]; and the verbal method in which verbal chunks or sentences are provided to the user as explanation [93, 93, 94]. In this section, we will focus on the saliency method as the explanation method proposed in this thesis falls into that category.

LIME [14] is a model independent interpretability method, in the sense that it is able to explain without needing to explore the inner functioning of the model. The principle of LIME method is to perturb the input around its neighborhood and observe the changes in the model predictions. For example, suppose we want to explain the prediction for the sentence "I hate this movie". Lime will perturb that sentence and get the output predictions of resulting sentences such as : "I this movie", "hate this movie", "hate movie", etc. From these perturbations, the relevancy of input words are determined. Even though LIME produces good results in practice, it does not "unblack-box" the model and the fidelity of the explanation to the exact inner model functioning can still be questioned. Another limit of LIME is that, to explain a single instance, it computes the outcome of multiple other instances obtained by varying the initial instance : this introduces a supplementary cost in terms of time complexity.

Jacovi et al. [13] have developed a method to understand how the CNN classifies a text. They examined the CNN parameters and showed that filters used in the convolutional layer may capture several different semantic classes of n-gram by using different activation patterns: Informative n-grams selected by the pooling and used to classify the text, and, uninformative n-grams eliminated by the pooling. They also distinguish between deliberate and accidental n-grams. Deliberate n-grams are effectively informative with higher score regarding the final decision, while accidental n-grams are n-grams selected by the pooling despite having a low score, because no other n-gram features scored higher than them. However, their method applies only to a limited range of CNN architectures, those with only one layer in the fully connected stage. In addition, the fact that different convolutional filters may select the same n-gram is not taken into account when scoring the n-gram features.

The Layer-wise Relevance Propagation (LRP) was firstly introduced in [89]. In this method, the relevance of input features is evaluated by computing their contribution to the output of a particular neuron. The contributions are evaluated such that, the sum of contributions of units in a layer to a unit in the next layer is equal to the output of the latter. Other articles using this method include [113, 113, 114, 115].

After having described the background on the explanation of black-box models, we will now

focus on the description our contribution to this young and emerging field.

3.3 A method to explain 1D-CNN

In this section, we describe the method that we proposed to explain 1D-CNN for text classification. The method proposed is a general purpose method that can be used for the local explanation of any 1D-CNN built for text classification.

Following the classification of the interpretability methods presented in section 3.1, our method is a saliency based method which performs local interpretation. It is also a post-hoc method since a post-processing is needed after the model decision to determine what to produce as a justification for that decision.

The idea of the prediction explanation method as described in this section is to highlight key features which have led to the classifier decision and that can be perceived by a human as a reason for that decision.

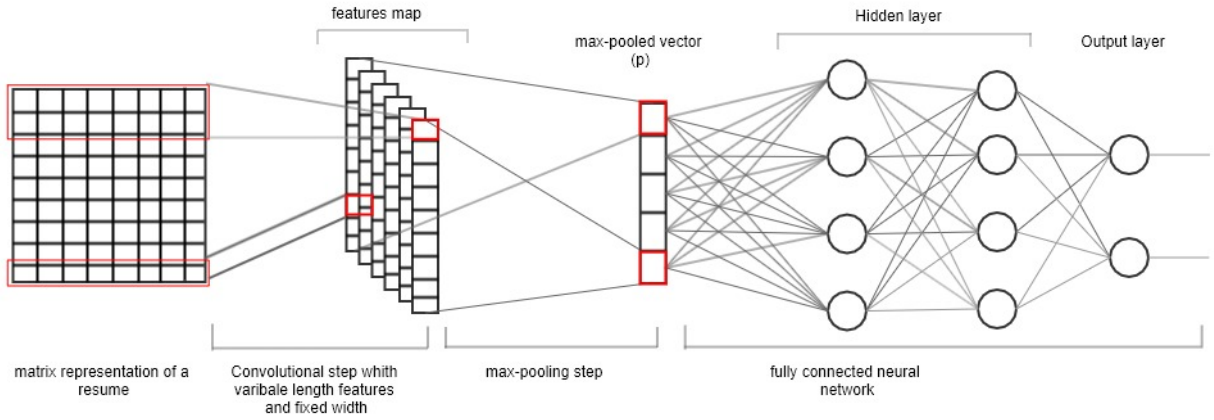


Figure 3.5: Text classification using CNN.

Figure 3.5 presents the architecture of a text CNN consisting of one convolutional layer with variable length filters, a max-pooling layer and a fully connected layer. Detail description of the functioning of the 1D-CNN can be found in chapter 2 section 2.3.2.

To sum up, a 1D-CNN takes as input a sequence of word embeddings which can be modeled as a 2-dimensional matrix. The convolutional filters, slide vertically over the input matrix and perform a scalar product with each embedding vector or sequence of embedding vectors. The convolution of a filter with the input matrix results in a vector (called feature map) where each component is the result of the scalar product between the filter and the corresponding word vec-

tor or sequence of word vectors. The max-pooling filter goes over each feature map and picks the maximum value. This value is associated to the n-gram which has produced the maximum convolution score with the filter associated to the that feature map. Applying the max-pooling over all the feature maps, results in a vector (the max-pooled vector) which is passed as input to the fully connected neural network.

Given the above description, we can figure out that values which were not selected by the max-pooling filter do not influence the classifier decision. Consequently, n-grams associated to the values which were not selected by the max-pooling filter had no effect on the classifier output value. Only the n-grams which were selected by the max-pooling filters had an influence on the classifier final decision [13].

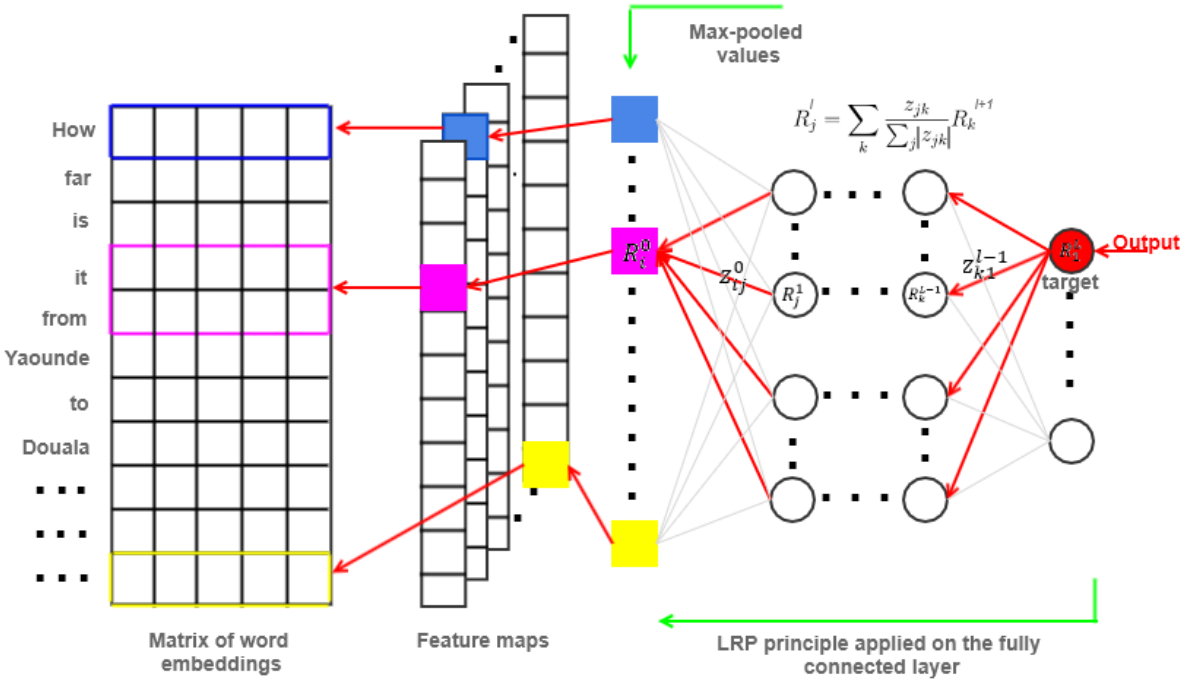


Figure 3.6: Overview of the explanation process.

Based on the later assumption, the explanation method described in this section and illustrated in Figure 3.6 involves the following steps :

1. First, a text is presented as input to the CNN and the output $f(x)$ is determined (Classification);

2. Then, an adaptation of the LRP algorithm is used to compute the relevance of each component of the max-pooled vector with respect to each output unit value.

3. After, n-grams associated to each of those components are recovered and their relevance are computed.

4. Finally, a greedy-like approach is used to compute sufficient and necessary feature sets.

Recovering the n-gram selected by a filter simply consists in selecting the n-gram which has produced the maximum convolution score with that filter.

The first step is described in detail in chapter 2 section 2.3.2. This section focuses on the prediction explanation method.

3.3.1 Layer-wise Relevance Propagation (LRP)

Layer-wise Relevance Propagation (LRP) [89] is a technique used to evaluate the contribution of input features to the neural network's output. The principle of LRP is to compute the contributions of units in the last hidden layer, then back-propagate them up to the input features. In this section, we will show how LRP is used to evaluate the contribution of the components of the max-pooled vector (see Figure 3.5).

Let's denote by $f : R^l \rightarrow R^c$ the multivariable vector-valued function representing the mapping between the inputs features and the output of the network, where l represents the size of the input vector (size of the max-pooled vector) and c represents the number of output classes. We denote by $z_{ij}^l = w_{ij}^{(l+1)} h_i^{(l)}$ the product of the activation of the unit i in layer l with the synaptic weight of the connection between the unit i and the unit j in layer $l+1$.

We denote by $z_j^l = \sum_i w_{ij}^l h_i^{(l-1)} + b_j^l = \sum_i z_{ij}^l + b_j^l$ $(0 <= l < L)$ the preactivation of the neuron j in layer l.

Definition 8. The contribution of a unit i of the $l-th$ layer to a unit j of the $(l+1)-th$ layer $(0 <= l < L)$ denoted by $R_{ij}^{(l,l+1)}$ is the extent to which the unit i has contributed to the preactivation value of the unit j.

Formally the contribution of the unit i to the output unit j is calculated using Equation 3.1 where w_{ij}^l is the synaptic weight of the connection between the neuron i of the layer l and the neuron j of the layer $l+1$; and h_i^l the activation of the unit i.

$$R_{ij}^{(l,l+1)} = \frac{z_{ij}^l}{z_j^{(l+1)}} = \frac{h_i^l * w_{ij}^{(l+1)}}{\sum_k h_k^l w_{kj}^{(l+1)} + b_j^{(l+1)}} \tag{3.1}$$

To prevent $R_{ij}^{(l,l+1)}$ to take underbounded values due to potential small values of $z_j^{(l+1)}$ Equation

3.2 will be considered instead, where ϵ is the stabilizer.

$$R_{ij}^{(l,l+1)} = \begin{cases} \frac{z_{ij}^{l}}{z_{j}^{(l+1)}+\epsilon} & \text{if } z_{j}^{(l+1)} \geq 0 \\ \\ \frac{z_{ij}^{l}}{z_{j}^{(l+1)}-\epsilon} & \text{if } z_{j}^{(l+1)} \leq 0. \end{cases} \tag{3.2}$$

Knowing the contribution of units of the $(l+1)-th$ layer ($0 \leq l < k-1$) to the value predicted by the output neuron j, we can evaluate the contribution R_{ij}^{l} of each neuron i in layer l to the value predicted by the output neuron j using Equation 3.3.

$$R_{ij}^{l} = \sum_{k} R_{ik}^{(l,l+1)} * R_{kj}^{l} \tag{3.3}$$

where $R_{jj}^{L} = f_j(x)$ and R_{ij}^{0} represents the contribution of the input feature x_i to the output neuron j.

The mathematical foundations of LRP method are discussed in [89]. The LRP algorithm operates as follows :

We start by computing the contribution of each unit i in the (L-1)th layer to the value predicted for each unit j in the output layer (L-th layer) using Equation 3.4

$$R_{ij}^{L-1} = R_{ij}^{(L-1,L)} * f_j(x) \tag{3.4}$$

$R^{(L-1,L)}$ is calculated using Equation 3.2.

Thereafter, the contributions of units in layer $L-2$ to layer $L-1$ are calculated using the recurrent equation 3.3. The process is iterated until the contributions of input features to the output values predicted by the network are determined.

3.3.2 LRP ratio Adaptation (LRP-A)

Section 3.3.1 describes the LRP method and its variants (LRP-ϵ, LRP-0). However, we found a problem with either of these methods.

Let's consider the simplified neural network presented in Figure 3.7. Generally, activation functions in neural networks are increasing functions. This means that the greater the input of the function, the greater its output value. However, based on the architecture presented in Figure 3.7,

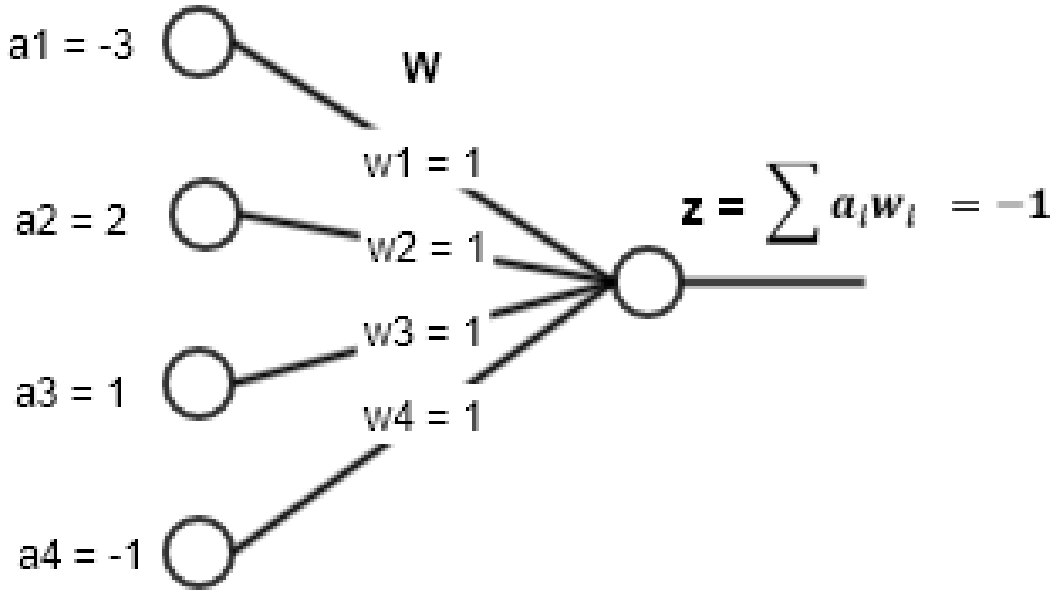


Figure 3.7: Simplified neural network.

the contribution ratio of the input unit a_1 to the linear sum of product z will be $a_1 w_1 / \sum a_i w_i = -3/-1 = 3$, and the contributions of a_2, a_3 and a_4 will respectively be -2, -1, and 1. This means that a_1 will be considered as the highest contributing input feature though it has a negative effect on the weighted sum z because it contributes to reduce the value of z, thus reducing the activation of the output unit. On the other hand, a_2 which actually contributes to increase the value of the weighted sum z will be considered having a negative contribution.

Situations where the weighted sum z is negative, could distort the interpretation of some computed relevances when using Equation 3.1. To avoid that behavior, the contribution ratio of a unit i is calculated by dividing the product $a_i w_i$ by the sum of the absolute values of the products $a_j w_j$, $z' = \sum |a_j w_j|$, which leads to Equation 3.5. This operation acts as normalizing the vector of contributions $a_i w_j$ using L1-Norm. Using Equation 3.5, the contributions of a_1, a_2, a_3 and a_4 will respectively be -0.43, 0.28, 0.14 and -0.14; which better reflects the effect of each input variable on the output unit pre-activation function.

$$R_{ij}^{(l,l+1)} = \frac{z_{ij}^{l}}{z_{j}^{(l+1)}} = \frac{a_i * w_i}{\sum_k |a_k w_k|} \tag{3.5}$$

3.3.3 Contribution of n-gram features

Having the output of the overall neural network for an input x, the first step in the explanation method is to calculate the contribution of each component of the max-pooled vector using the LRP (Section 3.3.1) method. We end up with a matrix representing the contribution of each filter to each class. After that, the next step is to compute the contribution of each n-gram to the output units.

We recall that each component of the max-pooled vector corresponds to the maximum value of the convolutions of a filter with every word vector (or sequence of word vectors) of the embedding input matrix. A filter f_j selects an n-gram u_i if the convolution of that filter with the given n-gram has produced the maximum value. Moreover, different filters may select the same n-gram for the same input text. Therefore, the contribution or the relevance of an n-gram u_i noted R_{u_i} is calculated as the sum of the contributions of the filters (R_f) that select that n-gram (Equation 3.6).

$$R_{u_i} = \sum_{f_j \in A_i} R_{f_j} \tag{3.6}$$

R_{f_j} is the relevance of the max-pooled value associated to the filter f_j which also defines the relevance of that filter. R_{f_j} is calculated using the LRP method described in section 3.3.1. A_i is the adjacency list associated to the n-gram u_i, which is the set of filters that selects the n-gram u_i.

Formally, the n-gram selected by a filter f_j is the n-gram u_j^* which maximizes the scalar product $< u_i, f_j >$ $(1 \leq i \leq n-l+1,\ 1 \leq j \leq m)$. where n is the number of words in the input text, l the number of words of the n-gram and m the total number of filters. u_j^* is determined by the formula 3.7 [116], where U represents the set of l-word n-grams.

$$u_j^* = \arg\max_u (< u_i, f_j >),\ u_i \in U \tag{3.7}$$

With (Equation 3.6) it is possible to evaluate the contribution of each n-gram to each output independently. We could then consider that: the greater the value of the contribution of an n-gram to the value predicted for an output neuron j, the more that n-gram is relevant to explain the output of that neuron. However, when using softmax activation in the output layer, the activation of a unit depends on the output values of the other units. In that case, the relevance of an n-gram for a target class must also take into account its contribution to other classes. Therefore it would be fair to refine the latter assumption to: a relevant n-gram to a given class c_j is an n-gram which highly

contributes to the target class c_j and contributes little to other classes. Based on this assumption, the relevance of an n-gram u_i with respect to an output class c_j can be computed as the difference between the contribution of that n-gram to the class c_j and the mean of its contributions to other classes. This means that: the more an n-gram contributes to the class c_j and the less it contributes to the other classes, the more it is relevant to the target class c_j.

$$R_j^{'i} = R_j^i - \frac{\sum_{p \neq j} R_p^i}{k-1} \tag{3.8}$$

k is the number of classes. For example, Table 3.2 shows the contributions of some n-grams to

Table 3.2: Contribution of some n-grams to the sentiment predicted for the sentence : *"great pocket pc phone combination"*

	NEG	POS	REL-POS
great	-0.0971	0.0822	0.1794
great pocket pc	-0.1132	0.0789	0.1922
phone combination	0.0019	0.0014	-0.0005

the sentiment predicted by a sentiment analysis model for the sentence *"great pocket pc phone combination"*. The model predicted the "positive" class (**POS**). From this table, we can observe that the contribution of the 3-gram *"great pocket pc"* to the positive class (**POS**) was less than that of the 1-gram *"great"*. However, we will consider the 3-gram *"great pocket pc"* as more relevant to the "positive" class because it also strongly negatively contributes to the "negative" class (**NEG**), compared to the word *great*.

For logistic output units, where the activation of a unit only depends on its output value we can consider that $R_j^{'i} = R_j^i$.

The algorithm 1 summarizes the method to compute the relevance of the n-gram features detected by the convolutional filters and described above.

3.3.4 n-gram polarity

Definition 9. Let u_i be the n-gram identified by the filter i and R^i the contribution vector of u_i to the output units, ie R_j^i is the contribution of u_i to the output class c_j. The class explained by u_i is the class c_j^* where u_i contributes the most. Formally,

$$c_{j^*} = \arg\max_j(R_j^i), 2 \leq j \leq k \tag{3.9}$$

Algorithm 1 SF-set : Compute N-gram Contributions

Require: A 1D-CNN model f ; An n-words input sequence X;
Ensure: R = The relevance of each n-gram detected by convolutional filters
Ensure: A = The adjacency lists associated to each n-gram i.e. A[i] represents the list of filters that detects the i-th n-gram feature.
1: Apply LRP on the fully connected layer of the CNN model f and get the contributions of the max-pooled values which are the inputs of the fully connected layer.
2: Let C be the vector of contributions obtained in the preceding step.
3: $F = f.Conv1D(X)$ ▷ The result of the convolutional layer (feature maps).
4: Initialize A to an empty dictionary, i.e. A = {}
5: Initialize R with zeros
6: $U = \arg\max_j F_{ij}$ ▷ Indices of n-gram features detected by convolutional filters
7: **for** `j from 1 to m` **do**
8: $A[U[j]] = A[U[j]] + \{j\}$ ▷ add filter j into the adjacency list associated to the n-gram U[j]
9: $R[U[j]] = R[U[j]] + C[j]$ ▷ C[j] is the contribution of the max-pooled value associated to the filter j
10: **end for**
11: **return** U, R, A

k being the total number of classes.

Definition 10. A positive n-gram for a target class c_j is an n-gram that contributes positively to c_j (i.e. $R_j^{'i} > 0$).

An n-gram with negative contribution ($R_j^{'i} < 0$) will be called a negative n-gram; and an n-gram with null contribution ($R_j^{'i} = 0$) does not influence the outcome of the target output unit.

Positive n-grams for a predicted class are n-grams which can likely be used to explain the prediction of the classifier because they actually contribute to increase the output value of the predicted class. But not all positive n-grams are necessary to explain the target class. We assume that only a subset of n-gram features is sufficient to explain a prediction. They are key n-gram features on which the model mainly relied to produce its decision.

3.3.5 Sufficient feature-sets

let $X = (x_1, x_2, ..., x_n)$ an input instance such that $f(X) = y$. Let $U = \{u_i\}_{i=1}^{i=n}$ the set of n-gram features selected by convolutional filters, $U^+ \subset U$ and $U^- \subset U$ (such that $U = U^+ \cup U^-$), the subsets of positive respectively negative n-gram features selected by convolutional filters with regards to the classifier output decision.

Definition 11. A sufficient feature-set is a subset $S \subset U^+$ of positive n-gram features such that the inhibition of all other positive features except S-features does not change the final classifier

decision for an input text x. Inhibiting an n-gram feature consists in inhibiting all the filters that select that n-gram; and inhibiting a filter consists in setting the corresponding max-pooled value to zero or setting the synaptic weights connecting the max-pooled value associated to that filter to zero.

A sufficient set of features S is minimum if there is no other sufficient set of features S' such that $S' \subset S$.

Definition 12. The relevance of a sufficient feature-set S with respect to a target class c, is the sum of the contributions of the n-gram features of S to class c.

Algorithm 2 SF-set : Get Sufficient feature-set

Require: A 1D-CNN model f ; An n-words input sequence X;
Ensure: S = A minimum sufficient feature-set with maximum relevance

$p = f(X)$ ▷ The index of the class predicted for the input X
$U, R, A = ComputeN-gramContributions(f, X)$
$U^+ = \{u_i \in U \mid R_p^i > 0\}$
$sort\ U^+\ elements\ in\ ascending\ order\ of\ their\ contribution$
$y = p,\ i = 0,\ S = U^+$
while p == y **do**
    $L = A[U[i]]$ ▷ List of filters selecting U[i]
    f' = the model obtained by setting in f the weights connecting the FCNN and the max-pooled value corresponding to filters in L to zero.
    $y = f'(X)$
    if y == p **then**
        Remove $U[i]$ from S ▷ U[i] is not necessary
    end if
    $i = i + 1$
end while
return S

We are interested in finding the minimum sufficient feature-set with maximum relevance. The aim of finding sufficient feature-sets is to simplify the explanation by providing the user with key features on which the model has mainly relied to produce its outcome. Sufficient feature-sets can also provide insights to improve model accuracy, for example, if we find out that the model just relied on a subset of features to produce its decision, yet in fact those features was not sufficient. The algorithm

The algorithm 2 is a greedy-like algorithm to determine sufficient features set. The principle of the algorithm is to sort the subset of positive n-grams in the ascending order of their contribution, and thereafter inhibit successively unnecessary n-grams until the classifier decision changes. The remaining n-grams are considered as sufficient n-grams. During the process, algorithm 1 is used to determine the contributions of the n-grams detected by convolutional filters.

3.3.6 Necessary features

let $X = (x_1, x_2, ..., x_n)$ an input instance such that $f(X) = y$. Let $U = \{u_i\}_{i=1}^{i=n}$ the set of n-gram features selected by convolutional filters, $U^+ \subset U$ and $U^- \subset U$ such that $U = U^+ \cup U^-$, the subset of positive respectively negative n-gram features selected by convolutional filters with regards to the classifier output decision.

Definition 13. A necessary n-gram feature is a positive n-gram $u \in U^+$ such that the inhibition of filters that select u will change the classifier decision for the input text X.

Algorithm 3 NF-set: Get Necessary features-set

Require: A 1D-CNN model f; an n-words input sequence X;
Ensure: N = a set of necessary n-gram features

$p = f(X)$ ▷ The class predicted for input X
$U, R, A = Compute\ N-gram\ Contributions(f, X)$
$U^+ = \{u_i \in U \mid R_p^i > 0\}$
$sort\ U^+\ elements\ in\ descending\ order\ of\ their\ contribution$
$y = p,\ i = 0,\ N = \{\}$
while $p \neq y$ **do**
    $L = A[U[i]]$
    f' = the model obtained by setting in f the weights connecting the FCNN and the max-pooled values associated to filters in L to zero.
    $y = f'(X)$
    if $y \neq p$ **then**
        $N = N + U[i]$ ▷ U[i] is necessary
    end if
    $i = i + 1$
end while
return N

The necessary feature-set is the set of necessary features. Necessary features present a major importance in explanation because according to the model, they were required to produce the decision given by the model.

Algorithm 3 is a greedy-like algorithm to determine the set of necessary features. Unlike sufficient feature set, necessary features are taken individually. The principle of the algorithm is to sort the set of positive features in the descending order of their contribution to the class predicted by the classifier. Thereafter, each individual feature is considered necessary if its inhibition changes the classifier decision. We assume that if a feature is not necessary then, neither will be any feature that contributes less than it.

3.3.7 Computational Complexity analysis

Let's denote by n the number of rows of the CNN input matrix, l the number of filters, d the dimension of the word embeddings and k the filter kernel size.

Let's denote by $t_f(n)$ the time complexity of a forward pass through the FCNN, $t_c(n)$ the computational complexity of the convolutional layer, t_p the computational complexity of the pooling layer. Then the computational complexity of a forward pass through the 1D-CNN will be $t = t_c + t_p + t_f$.

In [117] authors have approximated the complexity of the 1D-Convolutional layer to $t_c(n) = O(k \cdot n \cdot d^2)$. Indeed, when a filter of shape $k \times d$ slides over the input matrix, at each step it performs $O(k \times d)$ multiplications. Since the matrix is of size n, the total number of multiplications will then be $k \cdot d \cdot (n-k+1)$ where $n-k+1$ represents the total number of moves of a single filter along the input matrix (the number of partial convolutions). Generally $k << n$, hence the operations performed by one filter could be estimated to $O(k \cdot n \cdot d)$. For all the filters in the convolutional layer it would be $O(k \cdot n \cdot d \cdot l)$. The authors assumed that $d \approx l$ and obtained $t_c(n) = O(k \cdot n \cdot d^2)$.

The complexity of the max-pooling is $t_p(n) = O(n \cdot l)$. Indeed, the max-pooling filter picks the maximum value in each feature map. And, a feature map has a length of $n-k+1$. For all the features maps, the complexity of the max-pooling layer will be $O(l \cdot (n-k+1))$ which will leads to $t_p(n) = O(n \cdot l)$ since $k << n$.

The input vector of the fully connected layer has a length of l which corresponds to the total number of filters in the convolutional layer. If we assume that each layer in the FCNN has the same number of neurons which is equal to the number of input features, then the number of multiplications during a forward pass through the FCNN will be $O(n\_layers \times l^2)$, where n_layers represents the number of layers in the FCNN. Assuming that $n_layers = l$, we will have $t_f = O(l^3)$. and Based on the assumption that $l \approx d$, we will have $t_p(n) = O(n \cdot d)$ and $t_f = O(d^3)$.

Consequently, The complexity of a forward pass through the 1D-CNN will then be $t(n) = O(k.n.d^2 + n.d + d^3) = O(k.n.d^2)$ if we assume that $n > d$.

Contribution of n-grams The calculation of the contributions of the max-pooled components has the same complexity as a forward pass through the FCNN that is t_f. Determining the n-gram associated to each filter takes $O(n)$. For all the filters it will be $O(n \cdot l)$. Then determining the relevance of all the n-gram features by summing up the relevance of filters selecting those n-grams will take $O(l)$. The total computational complexity of the algorithm which determines the relevance of

n-gram features is therefore : $c(n) = O(l^3 + n \cdot l + l) = O(l^3) = O(d^3)$ if we assume that $d^2 > n$.

In conclusion, the algorithm which determines the relevance of n-gram features, has almost the same computational cost as a forward pass through the fully connected neural network.

Sufficient feature set and Necessary features The first step before determining sufficient feature set is to compute the contribution of n-gram features that we have estimated to have a complexity $c(n) = O(d^3)$. Thereafter, the list of positive features is determined and sorted. This operation takes $O(n \cdot log n)$. The while loop which computes sufficient features takes $O(n \cdot (n + t(n))) = O(n^2 + n \cdot t(n)) = O(n \cdot t(n))$. The time complexity of the algorithm that determines sufficient features is therefore $O(n \cdot t(n))$

When applying the same reasoning on the algorithm which determines necessary features, we obtain a similar computational cost, i.e. $O(n \cdot t(n))$.

3.3.8 Comparison with LIME

Here we analyze the computational time needed to compute the n-gram contributions using LIME algorithm. To explain a single text instance, LIME builds multiple instances by varying the original text (see section 3.2). In the worse case, the number of resulting text instances will be 2^n which corresponds to the total number of variations that can be performed on the input text. In that case, the total complexity of the LIME method will be $O(2^n \cdot t(n))$ where $t(n)$ is the computational cost of a forward pass through the CNN. To avoid computational explosion, LIME has defined a parameter $s < 2^n$ which limits the number of samples to constitute. The computational cost therefore becomes $O(s \cdot t(n)) = O(s \times k \times n \times d^2)$ while our method performs in just $O(d^3)$.

3.4 Experiments and Results

The experiments[1] have been conducted on a set of multiple channel CNN models based on Kim's architecture [15] plus a trainable embedding layer. The explanation method has been tested on architectures consisting of 1 to 3 channels with filters of kernel sizes 1, 2 and 3. The word embedding layer consists of 50-dimension word vectors and the input texts were padded to a maximum length equals to 50.

Since the explanation method developed in this thesis is a general purpose interpretability

[1] https://github.com/florex/xai-cnn-lrp

method for 1D-CNN, we tested it on multiple 1D-CNN models of different NLP tasks including Question Aswering (QA), Sentitiment Analysis (SA) to assess the validity of the overall method. Thereafter, we applied it on the model developed for skills prediction

3.4.1 Datasets

The method has been experimented on the following datasets :

- **IMDB :** A movie review dataset for binary sentiment classification [118];
- **sentiment140 :** A dataset containing 1,600,000 tweets extracted using the twitter api for the sentiment analysis task [119];
- **TREC-QA_5500 :** A dataset for Question Answering Track with 5500 train samples [120]
- **TREC-QA_1000 :** A dataset for Question Answering Track with 1000 train samples [120];
- Few already pretraited training data from **Imdb**, **Amazon** and **yelp** were also provided by [121].

Table 3.3 describes the classes of the above mentioned datasets.

Table 3.3: Dataset classes description

dataset name	**number of classes**	**Classes description**
IMDB	2	0 = Negative 1 = positive
sent140	3	0 = Negative 2 = Neutral 4 = Positive
TREC-QA	6	DESC=DESCRIPTION ENTY=ENTITY HUM=HUMAN NUM=NUMBER LOC=LOCATION ABBR=ABBREVIATION

A model was built for each task described above. Table 3.4 shows a summary description of the models used to test the explanation method described in this chapter. The model name is related to the name of the dataset used to train the model.

Table 3.4: Description of models built to test the explanation method

model name	number of channels	kernel sizes per channel	filters per channel	model accuracy
IMDB	3	[1,2,3]	[40,40,40]	83.6%
sent140	3	[1,2,3]	[40,40,40]	88,9%
TREC-QA_5500_1ch	1	[1]	[40]	78.14%
TREC-QA_5500_3ch	3	[1,2,3]	[40,40,40]	84.52%
TREC-QA_1000_3ch	3	[1,2,3]	[40,40,40]	74.80%

Table 3.4 shows that the model built for sentiment classification using **Imdb** dataset has 3 channels with filters of kernel size respectively of 1, 2, 3 and each of these channels has 40 filters; this makes a total of 120 filters in the convolution layer.

3.4.2 Quantitative Evaluation

We will use two measures namely the fidelity and the stability/coherence to quantitatively assess the effectiveness of the interpretation method proposed.

Fidelity

The overall principle of the explanability method described in this thesis relies on the quality of the relevance computed by the adapted LRP. The fidelity evaluates the extent to which the relevances of n-grams computed using our method are faithful to the black box model [82, 122].

To evaluate the fidelity of our method adapted from LRP (LRP-A), we build a surrogate model which performs a linear regression over the n-gram features detected. More precisely, if R_{ij} is the contribution (relevance) of the n-gram i with regards to output value of the output unit j in the black box model, then the value predicted by the surrogate model for the output unit j noted y_j is the sum of the contributions of the n-gram features to that output unit (Equation 3.10).

$$y_j = \sum_i R_{ij} \tag{3.10}$$

And, the class predicted by the surrogate model for an input text is the class associated to the output unit which have the maximum output value (Equation 3.11).

$$j* = \arg\max_j y_j,\ (1 <= j <= n\_class) \tag{3.11}$$

The fidelity of the surrogate model s with regard to the black box model b can therefore be computed using Equation 3.12 which evaluates how well the surrogate model reproduces the behavior of the black box model.

$$Fidelity = \frac{|x \in X \mid s(x) = b(x)|}{|X|},\ X\ being\ a\ set\ of\ text\ instances \tag{3.12}$$

Table 3.5: Evaluation of the fidelity

Fidelity/models	**QA (6 classes)**	**SA (2 classes)**	**Resume classification (10 classes)**
LRP-A	95%	95.48%	94.38%
LRP-A without Norm	93,9%	91.41%	89.17%
Standard LRP	92,71%	86,15%	86.22%
LIME	94.7%	95.81%	95.31%

Table 3.5 shows a comparison of the fidelity of our method with the fidelity of LIME and LRP on various classification tasks. These results show that the relevance obtained with the standard LRP are less faithfull to the black box model, which confirms the analysis done in section 3.3.2. In addition, when applying LRP-A without normalizing the vector of contributions, the fidelity is less than the normalized variant, but still better than that obtained with the standard LRP. These results also show that the fidelity obtained with our method is comparable to that of LIME a well-known state of the art explanability method. However, the problem with LIME is certainly the time complexity needed to compute the explanation of one instance.

Figure 3.8 shows the evolution of the computational time required by LIME to compute the explanation of a text instance with respect to the number of variations k.

Stability/Coherence

The stability is a key requirement which impact heavily users' trust in explanability methods [77]. A way to evaluate the stability is to measure if similar input texts have similar explanations in terms of sufficient and necessary features. To evaluate the stability, we built a dataset consisting of 500 subsets of 10 similar sentences each. The distance similarity between two texts $s1$ and $s2$ noted $d(s1, s2)$ is computed as the cosine similarity between the average word embedding vectors of s1 and that of $s2$.

For a given text x in the test set, we consider its closest text x^c and its k-th closest text x^f, again in the test set. The stability would require that the distance between the explanations $e(x)$ and $e(x^f)$, normalized by the distance between x and x^f, should not be much different from the

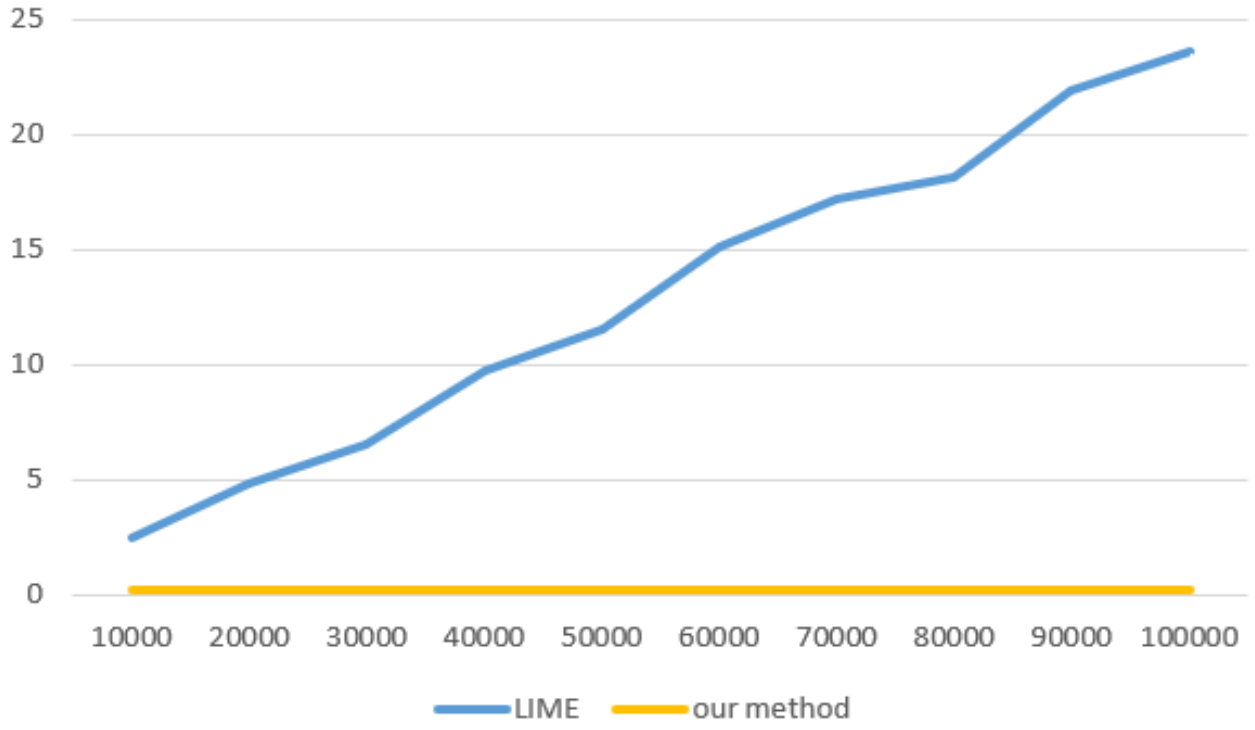


Figure 3.8: Evolution of LIME computational time versus LRP.

distance between the explanations $e(x)$ and $e(x^c)$, again normalized by the distance between x and x^c [122]. Stated in words, normalized distances between explanations should be as similar as possible. This can be computed using the coherence index defined in Equation 3.13.

$$C_x = \frac{d(e(x^f), e(x))/d(x^f, x)}{d(e(x^c), e(x)/d(x^c, x)} \tag{3.13}$$

Table 3.6: Mean and standard deviation of the coherence index (The closer to 1 the better)

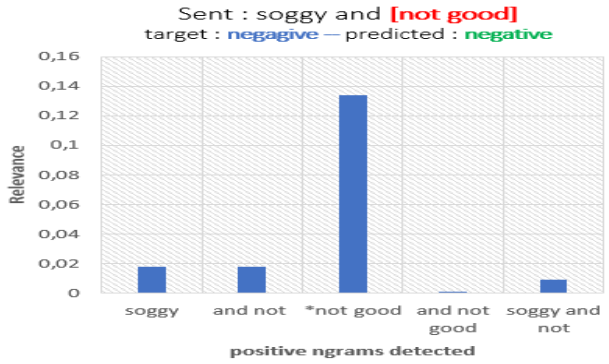
	QA (6 classes)	**SA (2 classes)**	**Resume classification (10 classes)**
Stability	1.12 ± 0.98	1.002 ± 1.14	0.999 ± 1.72

Table 3.6 presents the coherence-index of sufficient and necessary feature sets. The mean and the standard deviation show that the coherence computed on each input text is almost close to 1. This means that similar sentences have almost similar sufficient and necessary feature sets.

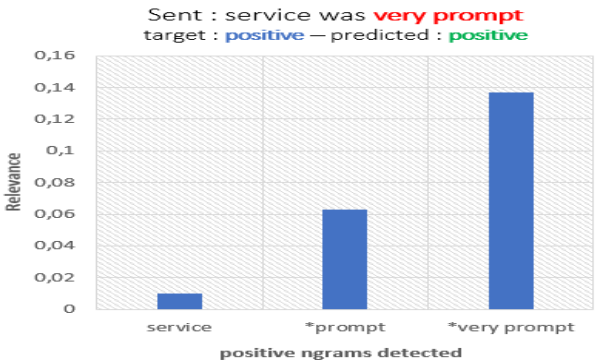
3.4.3 Qualitative Evaluation

In this section, we will conduct high-level s on the results of the explanability method proposed.

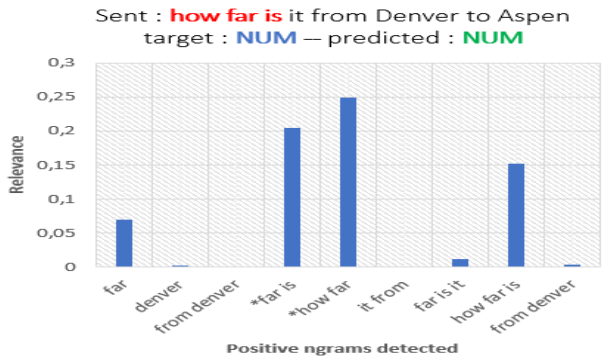
The explanation method was tested on the models described in Table 3.4. Figure 3.9 shows the distribution of relevance among positive n-grams for random sentences picked from the datasets



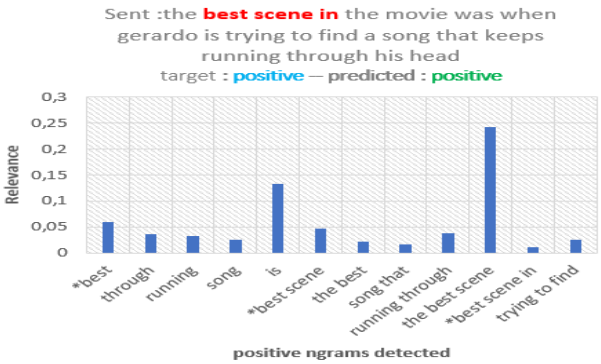
(a) A sample sentence from sent140



(b) A sample sentence from sent140



(c) A sample sentence from TREC-QA_5500



(d) A sample sentence from IMDB

Figure 3.9: Relevance of positive n-grams detected by filters.

described in section 3.4.1. Labels with asterix (*) represent sufficient features sets detected by algorithm 2. They are also highlighted in red in the original sentence. Necessary features are terms inside the brackets. In Figure 3.9c the model classified the question according to the type of the expected answer. Indeed, the sentence *"how far is it from Denver to Aspen?"* is classified as **"NUM"** because the expected answer is a number representing the distance between Denver and Aspen. Several positive contributing n-grams *(far, denver, from denver, ...)* were identified, but only the n-grams *"far is"* and *"how far is"* were sufficient to produce the same decision. This means that if all the filters that select other positive n-grams were inhibited except those selecting *"far is"* and *"how far is"*, the decision would have been the same. Likewise, Figure 3.9d has multiple positive n-gram features, however only three n-gram features contributed enough such that they were sufficient to produce the final decision according to the model. They are : *"best"*, *"best scene"*, *"best scene in"*; and when projected on the sentence it gives *"the best scene in the movie was when gerardo is..."*, which greatly simplifies human perception of the explanation provided.

In Figure 3.9a, the n-gram *"not good"* is at the same time sufficient and necessary, which means that it has greatly contributed to classify the sentence *soggy and [not good]* as *NEGATIVE*.

Table 3.7 shows the heatmap of a one-channel model using the TREC-QA dataset (TREC-QA_5500_1ch

Table 3.7: Relevance of n-grams to the class predicted for the question : *"who was the star witness at the senate watergate hearings?"*

	DESC	ENTY	ABBR	HUM	NUM	LOC	REL-HUM
senate	0.02	-0.02	-0.00	-0.00	0.01	-0.00	-0.00
witeness	0.01	-0.04	-0.01	-0.00	0.04	-0.01	-0.00
hearings	0.03	0.03	0.02	0.00	-0.03	-0.03	0.01
water-gate	0.01	0.01	0.01	0.00	-0.02	-0.03	0.01
who	-0.15	0.07	-0.17	0.37	-0.28	-0.16	0.51
at	-0.01	0.01	0.00	-0.00	-0.01	-0.00	-0.00
the	0.00	-0.03	-0.07	0.08	-0.02	-0.02	0.11
was	-0.01	-0.02	-0.02	0.04	0.00	-0.01	0.05

model described in Table 3.4). The table cells represent the contributions of the words labeling the table rows to the classes labeling the table columns. The intensity of a cell's color indicates the degree of the contribution of the corresponding word to the corresponding class. The dark red color indicates a strong negative contribution while the dark green color indicates a strong positive contribution to the class and the white color indicates almost null contribution. The last column (REL-HUM) represents the relevance to the predicted class (**HUM**) as the difference between the contribution of a word to the class **HUM** and the mean of its contribution to the other classes as explained in section 3.3.3, Equation 3.8. From Table 3.7, we can figure out that the term *"who"* did not only contributed highly to the class **HUM** but also contributed negatively to other classes except to **ENTY** where it contributed positively. By observing the relevance of each word to the predicted class, we can deduce that *"who was the"* are the main terms that were responsible for the prediction of the class **HUM** (**HUMAN**), which is the expected type of the answer to that question.

Discussion on sufficient and necessary feature-sets

Figure 3.10 shows a list of sentences taken from a question answering dataset (**TREC-QA**) and from sentiment analysis datasets (**imdb**, **sent140**). In each sentence the value after the dashes "–" represents the class predicted by the model. When this value is colored in green, it means that the model has correctly predicted the class, else the prediction is incorrect. The words colored in red in each sentence represent the sufficient n-gram features and those inside brackets represent necessary n-gram features. In sentence 1, the 2-gram *"When did"* was at the same time sufficient and necessary to predict the class **NUMBER** as the expected type of the answer to that question. In sentence 2, *"how tall"* was considered sufficient to predict the class **NUMBER** and there is no necessary feature.

Question-Answering :

1) [When did] hawaii become a state ? – NUM

2) How tall is the building ? – NUM

3) Why does the moon turn orange ? – DESC

Sentiment Analysis :

4) It was either too cold not enough flavor or just bad – NEGATIVE

5) Not frightening in the least and barely comprehensible – NEGATIVE

6) Lately they have been extremely [nice] and helpful on the phone – POSITIVE

7) [Not] frightening in the least and very realistic – NEGATIVE

8) [Not] frightening in the least and [very realistic] – NEGATIVE

9) Not bad and very comprehensible – NEGATIVE

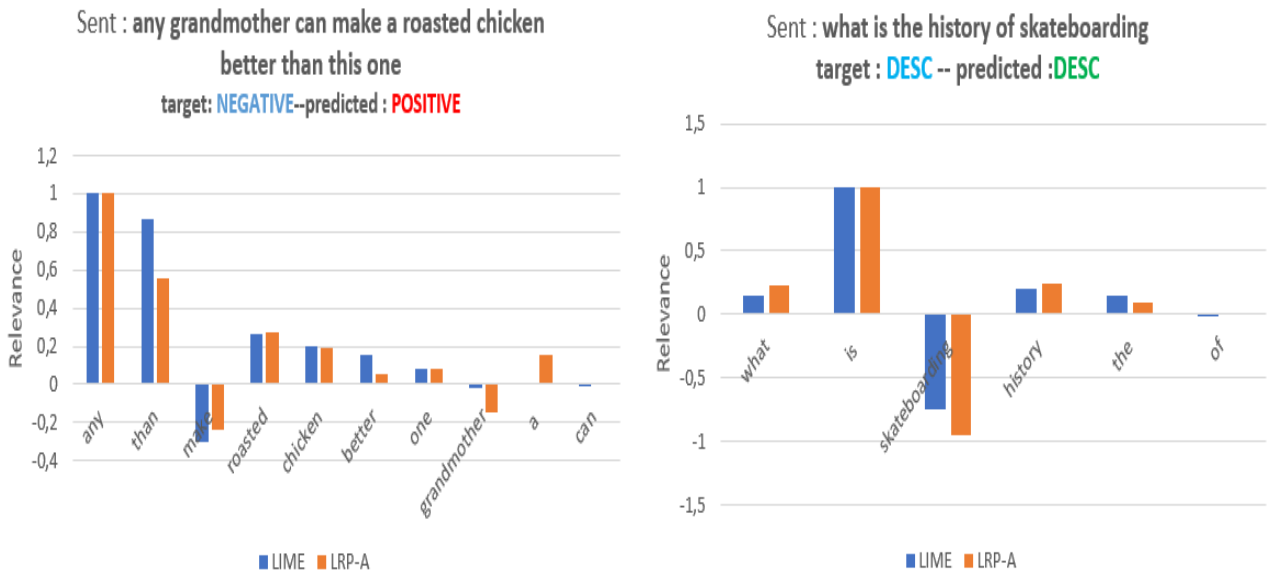
10) frightening in the least and [very realistic] – POSITIVE

Figure 3.10: Sufficient and necessary features.

In sentence 5, the term *"not"* was considered by the model as sufficient to predict the sentiment **NEGATIVE**. We performed little variations on that latter sentence, to change the underlying sentiment to POSITIVE, which have led to sentences 7, 8, 9, 10. We observe that the model has still predicted the sentiment NEGATIVE (for sentences 7, 8, 9) while still relying on the term *"not"*. However, when the term *"not"* is removed (sentence 10) the sentiment changes to POSITIVE, and the n-gram "very realistic" is determined as both sufficient and necessary to explain this new output. These examples show that in the case of sentences 7, 8, 9 the model has wrongly relied on the term *"not"* to produce its decision. This is most likely due to insufficient training samples. Indeed, the model was built with a training set of 2250 sentences. A solution track would be to add in the training set, sentences which contain the term *"not"* with similar contexts to those of sentences where the prediction was marked as incorrect.

Comparison with LIME

Since LIME is a statistical model agnostic method for explanation which also evaluates the relevance of each input feature to the output classes, we thought of comparing the distribution of relevance using our method with that obtained using LIME. Figure 3.11 shows the distributions



(a) LIME and LRP-A on Sentiment Analysis

(b) LIME and LRP-A on TREC-QA dataset

Figure 3.11: LIME versus CLRP

of relevance using LIME (in blue) and using our method (LRP-A) for two sentences. The first sentence (Figure 3.11a) is from **imdb** corpus and the second one (Figure 3.11b) is from the **TREC-QA** corpus. Data have been normalized using Decimal Scaling Normalization [123] in order to have both distributions in a comparable scale and to preserve the sign of each contribution. We observe that LIME and LRP-A (our method) have almost the same distribution of relevance among words for both sentences, and the most important terms highlighted by both methods are the same.

However, we can observe that no filter has selected the term *"can"* with our method but LIME has still scored it though with low relevance. Since our method is a white box method, it is easy to verify that the term *"can"* had no impact on the output since it was not selected by any max-pooling filter.

Another interesting fact to notice is that in Figure 3.11a, the model failed to predict the right class. Indeed, the model predicted the class **POSITIVE** while the expected class was the **NEGATIVE** class. When observing the distribution of relevance of words to the predicted class (**POSITIVE**) we can figure out that the model mainly relied on the features *(any, than)* which cannot be perceived as a correct justification for that prediction. Therefore, we can deduce that the model relied on wrong features to predict the class POSITIVE. This is likely due to an insufficient number of samples when training the model. This knowledge could provide insights to improve the model accuracy. For example, by providing negative samples where the terms *(any, than)* appear to tell the model not to associate them with the class **POSITIVE**.

Impact of the LRP adaptation

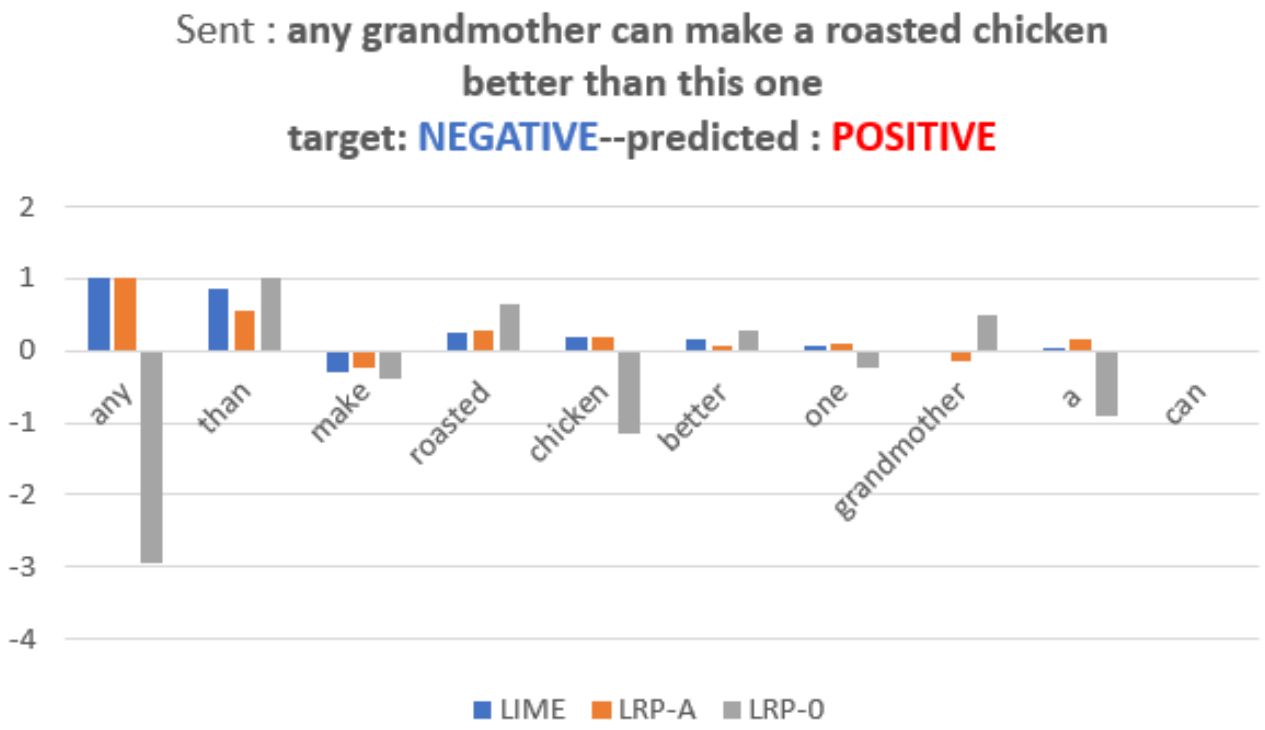


Figure 3.12: Impact of the new contribution ratio formula

As described in Section 3.3.2, to prevent undesired effects when the denominator in the LRP ratio (Equation 3.1) is negative, we proposed to use Equation 3.5 to compute the ratio. Figure 3.12 presents a comparison between the distribution obtained using LIME, the standard LRP-0 and LRP-A. We observe that while LIME and LRP-A have almost the same distribution, the sign of the contribution of certain words when using LRP-0 diverge. This confirms the analysis done so far in section 3.3.2

3.5 Application to the Skills Prediction Model

In this section, we use our explanation method to analyze the convolutional filters to understand the predictions of each single base CNN classifier. Two different levels of analysis are considered :

1. First, a filter-level analysis is done to understand which words each filter is specialized in identifying;

2. Secondly, an analysis at the resume level is performed to understand which terms were responsible for the classifier decision for an input resume.

3.5.1 Filter-level analysis

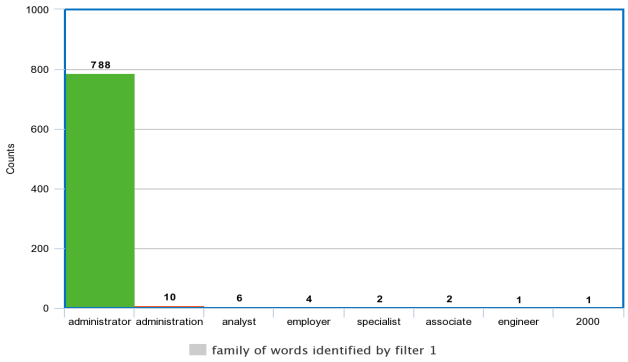
First of all, we would like to determine which words each filter is specialized in detecting. To find out those words, we ran the model on a test set of resumes of a specific competence. Then, we computed the number of times each word was selected by each filter. Figures (3.13a,3.13b,3.13c,3.13d) show the words identified by four filters using a test set of resumes labeled with the competence "database administrator".

We observe from Figure 3.13a that the filter 1 has detected the word *administration* 788 times out of 814 resumes. By observing this result, we can assume that, filter 1 has specialized in the detection of the word *administration*. The other words *(analyst, employer, specialist, ...)* were identified in resumes where the word *administration* was not found. Similarly, the filter 2 (Figure 3.13b) is specialized in the detection of the word *database* following the same reasoning. Since the test set was restricted to resumes labeled with the "database administrator" competence, it is normal that the term "database" appears in almost all the resumes. That is why that filter selected the word database 811 times out of 814. When the word database is not present in a resume, the filter still selects a term closely related to it (sql). Filter 3 (Figure 3.13c) mainly detects the word *migration*. In case this word is not in the resume, the filter selects other words such as *administration, query,...* Filter 4 (presented in Figure 3.13d) specializes in detecting the words : *tune, database, tuning, oracle*. We observe that those filters mainly select words closely related to the competency *database administrator*. The filter-level analysis, can be used to identify the sets of low-level features that explain each competence.

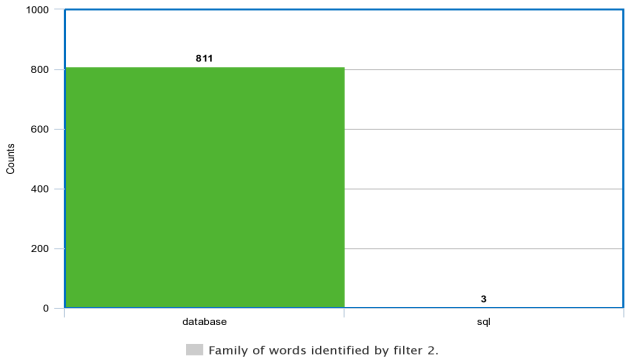
3.5.2 Resume-level analysis

After having analyzed the general behavior of filters, we now focus on the explanation of the model prediction for a given input resume (local explanation). Figure 3.14 shows the resume of a *project manager* expert where the positive contributing terms selected by the convolutional filters have been projected. The label of that resume was correctly predicted by the classifier. The *multi-highlight browser plugin*[2] was used to highlight the words selected by the filters on the original resume. The colors are related to the frequency of the highlighted words in the resume. We also observe that many words related to project management (project, program, implementation, analysis, resolution, manager, deliverable, plan, organize, ...) were selected by filters. And those words

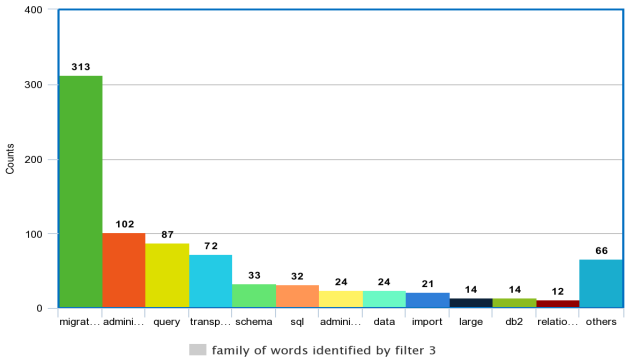
[2]https://chrome.google.com/webstore/detail/multi-highlight/pfgfgjlejbbpfmcfjhdmikihihddeeji : a browser extension used to highlight words in web pages.



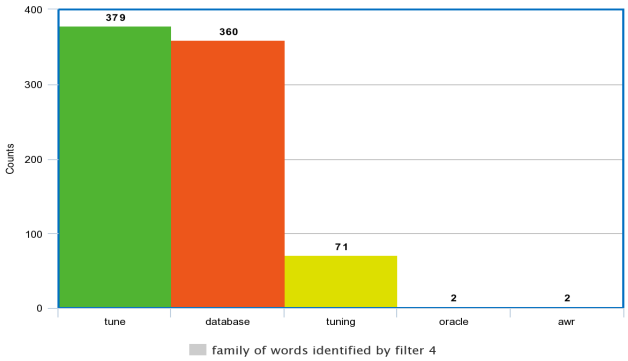
(a) Words selected by filter 1



(b) Words selected by filter 2



(c) Words selected by filter 3



(d) Words selected by filter 4

Figure 3.13: Filter analysis for the “Database administrator” specialized model

can provide the user with an explanation of the classifier decision. From this analysis, the model provides the following explanation to the user: *the expert is classified as a project manager because the terms : project, program, implementation, analysis, resolution, manager, deliverable, plan, organize* were found in his resume. Indeed, this set of terms was found sufficient (according to the model) to predict the class *project manager* for that resume.

We also consider the case of an incorrect prediction. Figure 3.15 is a resume predicted by the model as *python developer* but the title *python developer* does not appear in that resume. We recall that they are those occupation titles that were used to label the resume for training. Figure 3.15 also shows the words selected by the filters to produce the final decision. In fact, even though the resume was not labeled as a *python developer*, the resume actually expresses python developer's skills: *developer, regression, python, scikit-learn, tool, test, script, flask (a python framework)*, etc. So, the model has identified some words characteristics of a *python developer* and has produced its decision based on those words. That is why concerning the model evaluation, we said that in practice the precision might be greater than that we evaluated on the dataset due to incomplete resumes labeling. In fact, it is difficult to manually label resumes with all the competences. But when a recruiter reads the resume, he can identify certain competences that might not have been

Figure 3.14: words identified by filters in a project manager resume

explicitly mentioned by the expert but that might be deduced from his experience or other skills he has mentioned. The high recall rate that we obtained, guarantees that whenever a resume will have elements describing a certain competence, the model will almost always be able to predict that competence. Even though the prediction has been marked as incorrect, the following explanation can still be provided to the user : *this expert has python developer competence because the terms : developer, regression, python, scikit-learn, tool, test, script, flask, etc ... are present in his resume.* The above terms are in fact closely related to “python development”.

Chapter Summary

The purpose of this chapter was to present the explanation method proposed to explain deep neural network predictions especially 1D-CNN. We started by presenting the state of the art of explain-

Figure 3.15: A resume which was not labeled as “python developer” but predicted by the model as such.

able AI for text processing, highlighting existing approaches to explain, visualize and evaluate an explanation.

The approach to explain 1D-CNN consists in using LRP to compute the relevance of the components of the max-pooled vector; and thereafter recovering the n-gram associated to each of these components since the components of the max-pooled vector correspond to the n-grams which produced the maximum convolution score with the convolutional filter. And finally, sufficient and necessary sets are computed.

We tested the overall method on general purpose datasets including Question-Answering and Sentiment Analysis before testing it on the skills prediction model.

Regards the young age of the domain, there is not yet a standard evaluation scheme, and the majority of existing methods use informal evaluation and so do we.

The results obtained from the experiments were corroborated by the ones obtained from a well-known state of the art model.

The computational complexity of the overall method was also evaluated and the algorithm which computes n-gram relevances was found to perform asymptotically better than LIME.

CONCLUSION

The aim of this thesis was to build an explainable deep neural network to predict high-level skills from resumes. To achieve this goal, we first built a multi-label architecture for skills prediction from resume using One-Dimensional Convolutional Neural Network (1D-CNN) as base classifiers. The choice of CNN was due to their demonstrated performances in text classification and the facility to analyze convolutional filters for explanation purposes. Having built the skills prediction model, we proposed thereafter, a generic explanation method that we applied on the skills prediction model.

Skills prediction architecture

The first contribution of this thesis is the description of a multi-label architecture based on CNN to predict high-level competences from input resumes. We first designed the model architecture using the binary relevance principle for multi-label classification. After, we described a method to conveniently process input resumes. A self-trained word embedding model was built using a dataset of resumes collected from the Internet and tagged automatically. The experiments conducted on this dataset showed the effectiveness of the method reaching 98,79% of recall and 91,34% of precision outperforming other models based on different text encoding methods.

During the experiments, we figured out that the similarity between competences was the main cause of errors when testing the model because the model predicts competences based on terms it has identified in the resume, and the competences which have closely related terms describing them will likely be predicted at the same time, even if one or both competences are not explicitly mentioned in the resume description. After the validation of the skills prediction model, we designed a method to explain those predictions.

Explanation method

Since 1D-CNN was used as based classifiers in the multi-label architecture built for skills prediction, we elaborated a method to explain 1D-CNN. The explanation method proposed is not problem centric and it has been tested on multiple NLP tasks including Quenstion-Answering and Sentiment Analysis. The contribution in this area of research is threefold :

- First we have proposed an efficient way to compute the relevance of n-gram features in 1D-CNN using LRP;
- We have shown the limitation of the standard LRP(LRP-0, LRP-ϵ) and have proposed an adaptation;
- We finally proposed to simplify the explanations for model decisions by providing sufficient and necessary features to users.

The explanation method proposed has been applied on the skills prediction model by projecting on the resume description the terms which were responsible for the classifier decision.

Experiments carried out on various datasets have demonstrated the effectiveness of the explanation method proposed. Indeed, the distribution of relevance obtained with our method is similar to that obtained with a well-known state of the art method (LIME) and our method has been proven to perform asymptotically faster than that method.

Perspectives

The main application of skills extraction is job recommendation. The majority of existing methods extract skills from jobs and resumes, and thereafter use a similarity measure to compute the matching degree between them. Instead, we think of designing an explainable model which will learn by reinforcement how to match job postings written in natural language with raw text resumes without the need to manually define a similarity measure.

In Addition, when experimenting the explanation method proposed, we identified a problem that we think if it is correctly handled will greatly improve the accuracy of learning algorithms especially when data are imbalanced. Indeed, assume we have a binary classification problem with two classes A and B. Assume that A is 20% of instances in the dataset and B is 80%. We also assume that there is a feature x which appear in 100% of instances of class A i.e. 20% of the overall

dataset, and x appear in 50% of the instances of class B i.e. 40% of the overall dataset. In this context a learning algorithm will tend to associate the feature x as characterizing class B more than A. But intuitively, x characterizes class A more than B. In future works, we will attempt to conduct more experiment to assess the validity of the problem and propose a solution thereafter.

Another important aspect to consider during the recruitment is how to detect the false skills mentioned in a candidate's resume (how to trust the candidate?). As the work presented in this thesis makes it possible to characterize high level skills, it could be extended to false skills detection from resumes by verifying if the low level skills mentioned match with the high level skills of the candidates. This implies to build a model capable of learning the correspondence between the low level skills set and the associated high level skills.

REFERENCES

[1] ASSOCIATION FOR TALENT DEVELOPMENT. *Bridging the skills gap : workforce development is everyone's business n2*. 2015.

[2] *Skill gap the cost*. http://press.careerbuilder.com/2017-04-13-The-Skills-Gap-is-Costing-Companies-Nearly-1-Million-Annually-According-to-New-CareerBuilder-Survey. Accessed: 2020-12-02.

[3] Shobhana Sosale and Kirsten Majgaard. *Fostering skills in Cameroon: inclusive workforce development, competitiveness, and growth*. The World Bank, 2016.

[4] Shivam Bansal, Aman Srivastava, and Anuja Arora. "Topic Modeling Driven Content Based Jobs Recommendation Engine for Recruitment Industry". In: *Procedia computer science* 122 (2017), pp. 865–872.

[5] Duygu Çelik and Atilla Elçi. "An ontology-based information extraction approach for résumés". In: *Joint International Conference on Pervasive Computing and the Networked World*. Springer. 2012, pp. 165–179.

[6] Duygu Çelik et al. "Towards an information extraction system based on ontology to match resumes and jobs". In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. IEEE. 2013, pp. 333–338.

[7] Meng Zhao et al. "SKILL: A System for Skill Identification and Normalization". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2015, pp. 4012–4018.

[8] Faizan Javed et al. "Large-scale occupational skills normalization for online recruitment". In: *Twenty-Ninth IAAI Conference*. 2017.

[9] Evanthia Faliagka et al. "On-line Consistent Ranking on E-recruitment; Seeking the truth behind a well-formed CV". In: *Artificial Intelligence Review* (Oct. 2013). DOI: 10.1007/s10462-013-9414-y.

[10] Harry BG Ganzeboom and Donald J Treiman. “Internationally comparable measures of occupational status for the 1988 International Standard Classification of Occupations”. In: *Social science research* 25.3 (1996), pp. 201–239.

[11] Luiza Sayfullina, Eric Malmi, and Juho Kannala. “Learning representations for soft skill matching”. In: *International conference on analysis of images, social networks and texts*. Springer. 2018, pp. 141–152.

[12] Amina Adadi and Mohammed Berrada. “Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)”. In: *IEEE Access* 6 (2018), pp. 52138–52160.

[13] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. “Understanding Convolutional Neural Networks for Text Classification”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 56–65. URL: http://aclweb.org/anthology/W18-5408.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. San Francisco, California, USA: ACM, 2016, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778. URL: http://doi.acm.org/10.1145/2939672.2939778.

[15] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://www.aclweb.org/anthology/D14-1181.

[16] Jalaj Thanaki. *Python natural language processing*. Packt Publishing Ltd, 2017.

[17] Koushal Kumar and Gour Sundar Mitra Thakur. “Advanced applications of neural networks and artificial intelligence: A review”. In: *International journal of information technology and computer science* 4.6 (2012), p. 57.

[18] Mark Aronoff and Kirsten Fudeman. *What is morphology?* Vol. 8. John Wiley & Sons, 2011.

[19] Cambridge University Press. *Cambridge Academic Content Dictionary*. New York : Cambridge University Press, 2009.

[20] Julie Beth Lovins. “Development of a stemming algorithm”. In: *Mech. Transl. Comput. Linguistics* 11.1-2 (1968), pp. 22–31.

[21] M.F.Porter. “An algorithm for suffix stripping”. In: *Mech. Transl. Comput. Linguistics* 14.3 (1980), pp. 130–137.

[22] Chris D. Paice. “Another stemmer”. In: *SIGIR Forum*. Vol. 24. 3. 1990, pp. 56–61.

[23] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[24] Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. “A rule based approach to word lemmatization”. In: *Proceedings of IS*. Vol. 3. 2004, pp. 83–86.

[25] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. “Building a large annotated corpus of English: The Penn Treebank”. In: (1993).

[26] Christopher D Manning et al. “The Stanford CoreNLP natural language processing toolkit”. In: *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 2014, pp. 55–60.

[27] Dirk Geeraerts. *Lexical Semantics*. Jan. 2017. DOI: 10.1093/acrefore/9780199384655.013.29. URL: https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-29.

[28] David Alan and Cruse. *Lexical semantics*. Cambridge university press, 1986.

[29] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[30] Abdalghani Abujabal et al. “Quint: Interpretable question answering over knowledge bases”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2017, pp. 61–66.

[31] Weiwei Yuan et al. “Initial training data selection for active learning”. In: Jan. 2011, p. 5. DOI: 10.1145/1968613.1968619.

[32] Thiago Fraga-Silva et al. “Active Learning based data selection for limited resource STT and KWS”. In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[33] Kai Wei, Rishabh Iyer, and Jeff Bilmes. “Submodularity in data subset selection and active learning”. In: *International Conference on Machine Learning*. 2015, pp. 1954–1963.

[34] Christopher Fox. “A Stop List for General Text”. In: *SIGIR Forum* 24.1–2 (Sept. 1989), pp. 19–21. ISSN: 0163-5840. DOI: 10.1145/378881.378888. URL: https://doi.org/10.1145/378881.378888.

[35] W John Wilbur and Karl Sirotkin. “The automatic identification of stop words”. In: *Journal of information science* 18.1 (1992), pp. 45–55.

[36] Lili Hao and Lizhu Hao. “Automatic identification of stop words in Chinese text classification”. In: *2008 International conference on computer science and software engineering*. Vol. 1. IEEE. 2008, pp. 718–722.

[37] Juan Ramos et al. “Using tf-idf to determine word relevance in document queries”. In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. New Jersey, USA. 2003, pp. 133–142.

[38] Yitan Li et al. “Word embedding revisited: A new representation learning and explicit matrix factorization perspective”. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[39] Vikas Raunak, Vivek Gupta, and Florian Metze. “Effective Dimensionality Reduction for Word Embeddings”. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 235–243. DOI: 10.18653/v1/W19-4328. URL: https://www.aclweb.org/anthology/W19-4328.

[40] Sam T Roweis and Lawrence K Saul. “Nonlinear dimensionality reduction by locally linear embedding”. In: *science* 290.5500 (2000), pp. 2323–2326.

[41] Amir Globerson et al. “Euclidean embedding of co-occurrence data”. In: *Journal of Machine Learning Research* 8.Oct (2007), pp. 2265–2295.

[42] M Atif Qureshi and Derek Greene. “EVE: explainable vector based embedding technique using Wikipedia”. In: *Journal of Intelligent Information Systems* 53.1 (2019), pp. 137–165.

[43] Omer Levy and Yoav Goldberg. “Linguistic regularities in sparse and explicit word representations”. In: *Proceedings of the eighteenth conference on computational natural language learning*. 2014, pp. 171–180.

[44] Richard Socher et al. “Parsing with compositional vector grammars”. In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2013, pp. 455–465.

[45] Richard Socher et al. “Recursive deep models for semantic compositionality over a sentiment treebank”. In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[46] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[47] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. “Exploiting similarities among languages for machine translation”. In: *arXiv preprint arXiv:1309.4168* (2013).

[48] Richard Socher, Milad Mohammadi, and Rohit Mundra. *Cs 224d: Deep learning for NLP*. 2015.

[49] Chris McCormick. *Word2vec tutorial-the skip-gram model*. 2016.

[50] Palash Goyal, Sumit Pandey, and Karan Jain. “Deep learning for natural language processing”. In: *Deep Learning for Natural Language Processing: Creating Neural Networks with Python [Berkeley, CA]: Apress* (2018), pp. 138–143.

[51] Jianbo Yang et al. “Deep convolutional neural networks on multichannel time series for human activity recognition.” In: *Ijcai*. Vol. 15. Citeseer. 2015, pp. 3995–4001.

[52] Bendong Zhao et al. “Convolutional neural networks for time series classification”. In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169.

[53] Levent Eren, Turker Ince, and Serkan Kiranyaz. “A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier”. In: *Journal of Signal Processing Systems* 91.2 (2019), pp. 179–189.

[54] Palash Goyal, Sumit Pandey, and Karan Jain. “Introduction to natural language processing and deep learning”. In: *Deep Learning for Natural Language Processing*. Springer, 2018, pp. 1–74.

[55] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription”. In: *arXiv preprint arXiv:1206.6392* (2012).

[56] Kyunghyun Cho et al. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014).

[57] Alex Graves and Jürgen Schmidhuber. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[59] Ilya Sutskever et al. “On the importance of initialization and momentum in deep learning”. In: *International conference on machine learning*. 2013, pp. 1139–1147.

[60] Shiqiang Guo, Folami Alamudun, and Tracy Hammond. “ResuMatcher: A personalized resume-job matching system”. In: *Expert Systems with Applications* 60 (2016), pp. 169–182. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.04.013. URL: http://www.sciencedirect.com/science/article/pii/S0957417416301798.

[61] Ilkka Kivimäki et al. “A graph-based approach to skill extraction from text”. In: *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing*. 2013, pp. 79–87.

[62] Paul R Cohen and Rick Kjeldsen. “Information retrieval by constrained spreading activation in semantic networks”. In: *Information processing & management* 23.4 (1987), pp. 255–268.

[63] Florentin Flambeau Jiechieu Kameni and Norbert Tsopze. “Approche hiérarchique d’extraction des compétences dans des CVs en format PDF”. In: *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées* 32 (2019), pp. 37–56.

[64] Eva Gibaja and Sebastián Ventura. “A tutorial on multilabel learning”. In: *ACM Computing Surveys (CSUR)* 47.3 (2015), pp. 1–38.

[65] Raed Alazaidah and Farzana Kabir Ahmad. “Trending challenges in multi label classification”. In: *International Journal of Advanced Computer Science and Applications* 7.10 (2016), pp. 127–131.

[66] Eva Gibaja and Sebastián Ventura. “Multi-label Learning: A Review of the State of the Art and Ongoing Research”. In: *Wiley Int. Rev. Data Min. and Knowl. Disc.* 4.6 (Nov. 2014), pp. 411–444. ISSN: 1942-4787. DOI: 10.1002/widm.1139. URL: http://dx.doi.org/10.1002/widm.1139.

[67] Jiang Bian, Bin Gao, and Tie-Yan Liu. “Knowledge-powered deep learning for word embedding”. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2014, pp. 132–148.

[68] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[69] Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. “Evaluation of domain-specific word embeddings using knowledge resources”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[70] Haoyue Shi et al. “Constructing High Quality Sense-Specific Corpus and Word Embedding via Unsupervised Elimination of Pseudo Multi-sense”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[71] Lei Zhang, Shuai Wang, and Bing Liu. “Deep learning for sentiment analysis: A survey”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1253.

[72] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2016.12.035. URL: http://www.sciencedirect.com/science/article/pii/S0957417416307175.

[73] Riccardo Guidotti et al. “A Survey of Methods for Explaining Black Box Models”. In: *ACM Comput. Surv.* 51.5 (Aug. 2018), 93:1–93:42. ISSN: 0360-0300. DOI: 10.1145/3236009. URL: http://doi.acm.org/10.1145/3236009.

[74] Stella Lowry and Gordon Macpherson. “A blot on the profession”. In: *British medical journal (Clinical research ed.)* 296.6623 (1988), p. 657.

[75] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL: http://www.sciencedirect.com/science/article/pii/S0004370218305988.

[76] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI”. In: *CoRR* abs/1910.10045 (2019). arXiv: 1910.10045. URL: http://arxiv.org/abs/1910.10045.

[77] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[78] Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.

[79] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[80] Marina Danilevsky et al. “A Survey of the State of Explainable AI for Natural Language Processing”. In: *arXiv preprint arXiv:2010.00711* (2020).

[81] Erico Tjoa and Cuntai Guan. “A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI”. In: *ArXiv* abs/1907.07374 (2019).

[82] Riccardo Guidotti et al. “A survey of methods for explaining black box models”. In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[83] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[84] Amirata Ghorbani et al. “Towards automatic concept-based explanations”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9277–9286.

[85] Avanti Shrikumar et al. “Not just a black box: Learning important features through propagating activation differences”. In: *arXiv preprint arXiv:1605.01713* (2016).

[86] Luisa M Zintgraf et al. “Visualizing deep neural network decisions: Prediction difference analysis”. In: *arXiv preprint arXiv:1702.04595* (2017).

[87] Pieter-Jan Kindermans et al. “Learning how to explain neural networks: Patternnet and patternattribution”. In: *arXiv preprint arXiv:1705.05598* (2017).

[88] Plamen Angelov and Eduardo Soares. “Towards explainable deep neural networks (xDNN)”. In: *Neural Networks* 130 (July 2020). DOI: 10.1016/j.neunet.2020.07.010.

[89] Sebastian Bach et al. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140. URL: https://doi.org/10.1371/journal.pone.0130140.

[90] Dumitru Erhan et al. *Visualizing Higher-Layer Features of a Deep Network*. Tech. rep. Univeristy of Montreal, 2009.

[91] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[92] Pieter-Jan Kindermans et al. “The (un) reliability of saliency methods”. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019, pp. 267–280.

[93] Rich Caruana et al. “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1721–1730.

[94] Benjamin Letham et al. “Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model”. In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371.

[95] D Zeiler Matthew and Fergus Rob. “Visualizing and Understanding Convolutional Networks”. In: *CoRR.—-2013.—-Vol. abs/1311.2901.—-URL: http://arxiv. org/abs/1311.2901* (2013).

[96] Been Kim et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)”. In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677.

[97] Naftali Tishby and Noga Zaslavsky. “Deep learning and the information bottleneck principle”. In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE. 2015, pp. 1–5.

[98] Ravid Shwartz-Ziv and Naftali Tishby. “Opening the black box of deep neural networks via information”. In: *arXiv preprint arXiv:1703.00810* (2017).

[99] R. R. Selvaraju et al. “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[100] Pengyu Liu, Avraham A. Melkman, and Tatsuya Akutsu. “Extracting boolean and probabilistic rules from trained neural networks”. In: *Neural Networks* 126 (2020), pp. 300–311. DOI: 10.1016/j.neunet.2020.03.024. URL: https://doi.org/10.1016/j.neunet.2020.03.024.

[101] SA Dudani. “The distance-weighted k-nearest neighbor rule”. In: *IEEE trans. on systems, man and cybernetics* 8.4 (1978), pp. 311–313.

[102] Danilo Croce, Daniele Rossini, and Roberto Basili. “Auditing Deep Learning processes through Kernel-based Explanatory Models”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 4028–4037.

[103] Nazneen Fatema Rajani et al. “Explain Yourself! Leveraging Language Models for Commonsense Reasoning”. In: *arXiv* (2019), arXiv–1906.

[104] Yichen Jiang et al. “Explore, propose, and assemble: An interpretable model for multi-hop reading comprehension”. In: *arXiv preprint arXiv:1906.05210* (2019).

[105] Qizhe Xie et al. “An interpretable knowledge transfer model for knowledge base completion”. In: *arXiv preprint arXiv:1704.05908* (2017).

[106] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. “Right for the right reasons: Training differentiable models by constraining their explanations”. In: *arXiv preprint arXiv:1703.03717* (2017).

[107] Samuel Carton, Qiaozhu Mei, and Paul Resnick. “Extractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts”. In: *arXiv preprint arXiv:1809.01499* (2018).

[108] Wang Ling et al. “Program induction by rationale generation: Learning to solve and explain algebraic word problems”. In: *arXiv preprint arXiv:1705.04146* (2017).

[109] James Thorne et al. “Generating token-level explanations for natural language inference”. In: *arXiv preprint arXiv:1904.10717* (2019).

[110] Alona Sydorova, Nina Poerner, and Benjamin Roth. “Interpretable question answering on knowledge bases and text”. In: *arXiv preprint arXiv:1906.10924* (2019).

[111] Yue Dong et al. “EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing”. In: *arXiv preprint arXiv:1906.08104* (2019).

[112] James Mullenbach et al. “Explainable prediction of medical codes from clinical text”. In: *arXiv preprint arXiv:1802.05695* (2018).

[113] Sebastian Lapuschkin et al. “Unmasking clever hans predictors and assessing what machines really learn”. In: *Nature communications* 10.1 (2019), pp. 1–8.

[114] Wojciech Samek et al. “Evaluating the visualization of what a deep neural network has learned”. In: *IEEE transactions on neural networks and learning systems* 28.11 (2016), pp. 2660–2673.

[115] Leila Arras et al. “" What is relevant in a text document?": An interpretable machine learning approach”. In: *PloS one* 12.8 (2017), e0181142.

[116] Kameni Florentin Flambeau Jiechieu and Norbert Tsopze. “Skills prediction based on multi-label resume classification using CNN with model predictions explanation”. In: *Neural Computing and Applications* (2020), pp. 1–19. DOI: https://doi.org/10.1007/s00521-020-05302-x.

[117] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017), pp. 5998–6008.

[118] Andrew L. Maas et al. “Learning Word Vectors for Sentiment Analysis”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[119] Alec Go, Richa Bhayani, and Lei Huang. “Twitter sentiment classification using distant supervision”. In: *CS224N project report, Stanford* 1.12 (2009), p. 2009.

[120] Ellen Voorhees. “The TREC question answering track”. In: *Nat. Lang. Eng.* 7 (Dec. 2001), pp. 361–378. DOI: 10.1017/S1351324901002789.

[121] Dimitrios Kotzias et al. “From group to individual labels using deep features”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 597–606.

[122] Orestis Lampridis, Riccardo Guidotti, and Salvatore Ruggieri. “Explaining Sentiment Classification with Synthetic Exemplars and Counter-Exemplars”. In: *Discovery Science*. Ed. by Annalisa Appice et al. Cham: Springer International Publishing, 2020, pp. 357–373. ISBN: 978-3-030-61527-7.

[123] Luai Al Shalabi and Zyad Shaaban. “Normalization as a preprocessing engine for data mining and the approach of preference matrix”. In: *2006 International conference on dependability of computer systems*. IEEE. 2006, pp. 207–214.

[124] Mounira Harzallah, Michel Leclère, and Francky Trichet. “CommOnCV: modelling the competencies underlying a curriculum vitae”. In: *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. 2002, pp. 65–71.

[125] Leila Yahiaoui, Zizette Boufaida, and Yannick Prié. “Automatisation du e-recrutement dans le cadre du Web sémantique”. In: 2006.

APPENDICES

A Data Availability

Resumes preprocessing program: https://github.com/florex/preprocessing

Multi-label classifier: https://github.com/florex/cnn_classifiers

Resume dataset: https://github.com/florex/resume_corpus

1D-CNN explanation program: https://github.com/florex/xai-cnn-lrp

B Extended Abstract

B.1 Context and Motivation

The Association for Talent Development (ATD) defines a skill-gap as a significant gap between an organization's current human capabilities and the skills it needs to achieve its goals and meet customer demand [1]. While more and more graduated students are unemployed, employers sometimes complain not being able to find appropriate human resources to perform critical tasks.

The US Economic Center of Research reported in a survey realized in 2018 that the "skills gap" costs about 160$ billion to companies a year in US [2]. Moreover, CarrierBuilder[3] reported in a study carried out in 2017 that 60% of employers have jobs that remain vacant for more than two weeks [2].

In Cameroon, in a study realized in 2016, the World Bank reported that in order to achieve a real structural transformation, the country must identify the "skill gaps" in the areas of technology and innovation [3]. The international institution argues that to end poverty among young people, it is necessary to bridge the "skill gap" [3]. To achieve this goal, it is necessary to be able to formally define the "skill gap" by identifying the skills held by young people and comparing them with those required by companies.

Online recruitment or "freelance" platforms have emerged in the last decades and enable companies to easily find the expertise they need anywhere, regardless of geographical constraints. The resulting consequence is a multitude of candidates applying for a job posting, sending their resumes electronically. That situation makes the process of selecting the best candidates cumbersome. This is why automatic job-resume matching algorithms have been developed to recommend a short-list of candidates whose profiles best fit the competencies needed. These types of algorithm generally require the ability to automatically identify the skills contained in documents (CV, job postings, etc.).

The main challenges related to automatic skills identification from text are :

1. First, competency is an abstract notion that can be represented in texts with different terms. For example, a PMP certification may indicate a competence in "project management", "project manager" and "project leader", or "AI" and "artificial intelligence" may refer to the same com-

[3] CarrierBuilder is an online hiring plateforme accessible at https://hiring.careerbuilder.com/?_ga=2.130206863.1209475669.1606937649-429933661.1606937649

petence. A competency may also be misspelled;

2. Second, competencies can be organized into different levels of abstraction in the sense that some competencies may explain other competencies. Therefore, a competency may not be explicitly mentioned in the CV, but could be inferred from other competencies explicitly mentioned. We will call them implicit competencies or high-level competencies.

A Competence can be defined as a professional qualification or the ability to do something. The competence has three components : knowledge (knowledge), know-how (practical) and interpersonal skills (interpersonal behaviours) [124, 125].

B.2 Problem Statement

Let's suppose that a recruiter is looking for a candidate who masters structural programming. But, the skill structural programming is not explicitly mentioned in the candidate's resume but instead C, ada, Pascal which, according to the taxonomy elaborated by Faliagka et al. in [9] are characteristics of structural programming. Explicit skills extraction methods cannot identify those skills which are not explicitly referenced in the resume description.

The vast majority of existing approaches focus solely on extracting the competencies explicitly described in the document by terms representing technologies, tools, certifications, etc. The existing approaches to skills extraction from texts range from the use of dictionaries [63], to named entity recognition [7, 8], or ontological modeling to identify the competencies explicitly mentioned in the resumes [5, 6]. Only few approaches have focused on the identification of implicit competencies, but have not addressed the need for model explanability [61, 63]. In addition, the nature of certain methods makes it difficult to automatically assess their validity without knowing the expected results [61].

The problem we are solving in this thesis can therefore be stated as follows : ***Given a raw text resume written in natural language, what are the set of high level skills that the resume expresses and how to convince a human that the resume actually expresses those skills ?***

Indeed, the fact that a resume can express multiple competences at the same time makes the skills prediction from resumes a multi-label classification problem where classes are skills.

Solving this problem will consist in designing an accurate model capable of predicting a set of skills that a resume expresses, and derive a method to justify the predictions provided.

B.3 Objective and Methodology

Convolutional neural networks have demonstrated outstanding performance in several problems such as image classification and text classification, achieving state of the art results in several text classification problems [15]. In addition, the work of jacovi et al. in [13] on the explainability of CNN built for text classification has drawn new perspectives concerning the interpretation of the decisions provided by these types of model.

Considering the performances of convolutional neural networks, as well as the advances made in their interpretation, we believe that their use to automatically identify competencies in resumes would be an appropriate choice.

On the other hand, the fact that the CV can express several competencies makes the problem of identifying competencies in resumes a multi-label classification problem.

The objective of this work is therefore to design a multi-label classification model that can predict a set of competencies contained in a resume while justifying the predictions made by highlighting the low level competencies that contributed to produce these decisions.

The methodology used to solve the problem consists in two main steps:

- First, the design of a multi-label classification model using the CNN as a base classifiers. At this level, the architecture of the multi-label classification model is first described with its various components. Then, the method used to preprocess resumes is explained;
- Secondly, the description of the method used to explain the predictions of each base CNN classifier. The explanation method proposed is first tested on a set of other text classification problems including Question Answering and sentiment analysis to demonstrate their genericity before being applied to the skills prediction model.

B.4 Contributions

The contributions of this thesis can be explored following two main aspects : the skills predictions model; and the method to explain the CNN models built for the text classification.

Competencies prediction model

In this aspect, we present the architecture (Figure 16) of a multi-label classification model capable of predicting high-level skills from resumes. The architecture proposed has two main components

: the preprocessing component, which goal is to transform a CV written in natural language into a matrix of numerics; and a set of binary classifiers whose role is to predict whether an input resume expresses a particular skill or not.

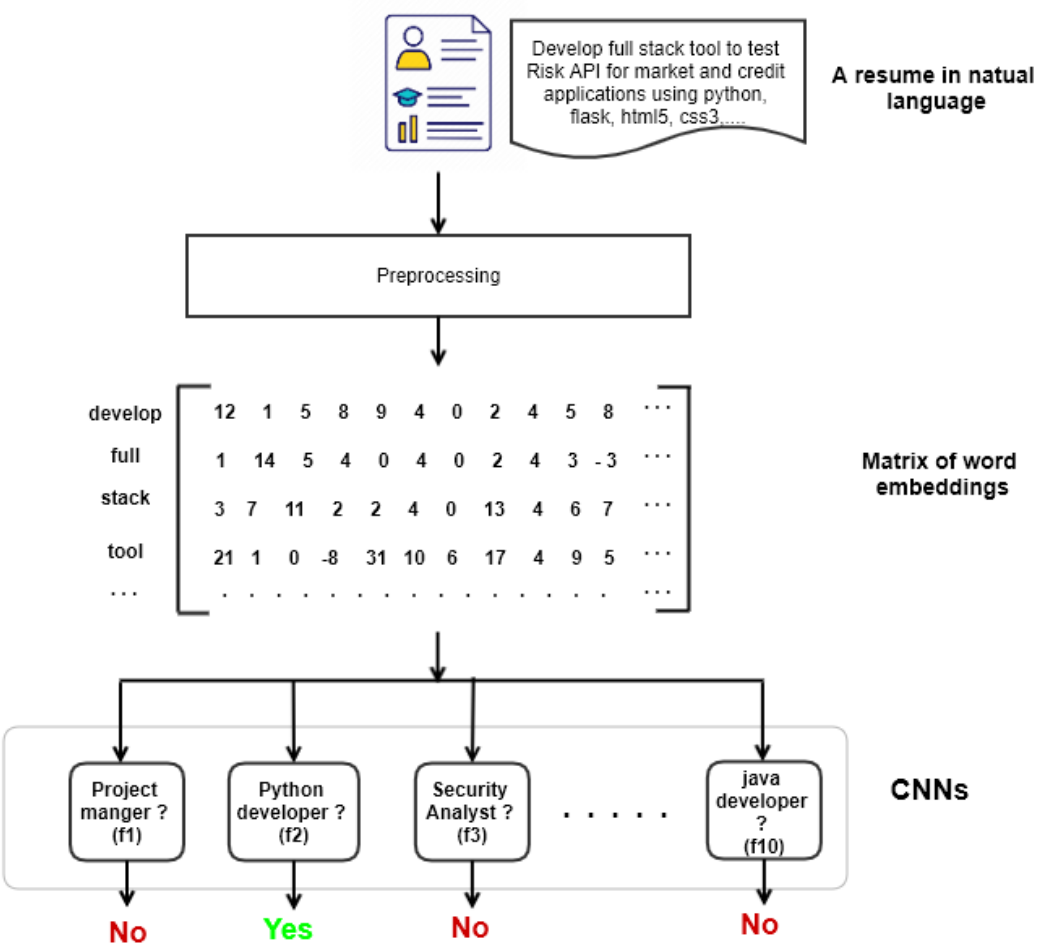


Figure 16: Multi-label skills prediction architecture model.

More precisely the contributions made in this aspect are:

- The construction of a context-specific word2vec[29] model from IT resumes. The skip-gram algorithm is used to build the word2vec model, which is then used to transform resumes into matrices;
- The definition of an efficient method for resumes transformation into matrices using the word2vec model built in the previous step. To reduce the memory space needed to store the resumes, an appropriate filtering method is applied and consists in retaining only terms relevant to the classification. This method has reduced the memory space required by half;

- We also built a dataset of 30000 multi-labeled resumes with the list of skills contained in these resumes. The resumes were automatically labeled using the professional qualifications mentioned by the candidates. A competency normalization algorithm was developed to associate different expressions referring to the same competency.

The experiments carried out have demonstrated the effectiveness of the proposed model, which attained a recall of 98.79% and an accuracy of 91.34%.

A method to explain CNN built for text classification

In this aspect, we proposed a general method for explaining CNN models designed for text classification. The developed method has been tested on several classification problems including sentiment analysis, and question classification before being applied to the skill prediction model. More precisely, the contributions highlighted in this section are :

- We propose a method using the LRP (Layer-wise Relevance Propagation) principle to calculate the contribution of each n-gram to the output value of the CNN model;
- We have highlighted a limitation of the original LRP method, particularly in the contribution ratio formula. Indeed, we have shown that when the denominator is negative, the interpretations of the contributions can be erroneous. To correct this behavior, we proposed an adaptation that preserves the real effect of each input on the output value predicted by the model;
- Finally we propose two algorithms to evaluate the sufficient and necessary n-gram features according to the model. Indeed, we show that it is not necessary to present to the user all the characteristics that could have had an influence on the results. We assume that only few of them are sufficient to understand the model decision.

B.5 Publications

The work described in this thesis has been the subject of : a communication at the “Conférence de Recherche en Informatique” (CRI 2017); an article published in the ARIMA journal indexed DBLP [63] and an article published in Neural Computing And Application (NCAA) of impact factor 4.774 (2019) [116].

- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze, *Skills prediction based on multi-label resume classification using CNN with model predictions explanation.* Neural Computing And Application, 2020;
- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze, *Approche hiérarchique d'extraction des compétences dans les CV en format PDF*. Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées, INRIA, 2019, volume 32, 2019.
- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze. *Approache hierarchique d'extraction des compétences dans les CV en format PDF*. In the proc. of the 3rd Edition of CRI (Conférence de Recherche en Informatique), Yaoundé, Cameroon, From 28th to 30th November, 2017;

B.6 Perspectives

The automatic extraction or identification of skills from text documents such as resumes of job postings is an important task which find its application mainly in job recommender systems. However, the vast majority of existing recommendation systems first proceed by extracting skills from both resumes and job offers before applying a similarity measure to evaluate the degree of matching between the candidate profile and the skills required. We believe that it would be more interesting to build an explainable model that would itself learn by reinforcement how to match job offers to profiles and justify its decisions by highlighting the skills identified in the profiles and their correspondence with the job offer.

Additionally, when applying the interpretability method, we identified a problem that we think if it is not correctly handled will greatly improve the accuracy of learning algorithm especially when data are imbalanced. Indeed, suppose we have a binary classification problem with two classes A and B. Assume that A is 20% of instances in the dataset and B is 80%. We also assume that there is a feature x which appears in 100% of instances of class A i.e. 20% of the overall dataset, and x appears in 50% of the instances of class B i.e. 40% of the overall dataset. In this context a learning algorithm will tend to associate the feature x more to class B than to class A. But intuitively, x characterizes class A more than B. In future works, we will attempt to conduct more experiments to assess the validity of the problem and propose a solution thereafter.

Another important aspect to consider during the recruitment is how to detect the false skills mentioned in a candidate's resume (how trust the candidate?). As the work presented in this thesis makes it possible to characterize high level skills, it could be extended to false skill detection from resumes by verifying if the low level skills mentioned match with the high level skills of the can-

didates. This implies to build a model capable to learn the correspondence between the low level skills set and the associated high level skills.

C Résumé étendu

C.1 Contexte et Motivation

Le « Skills Gap » qui se définit comme étant l'écart entre les compétences détenues par la ressource humaine d'une organisation et celles dont elle a besoin pour son développement est un problème extrêmement crucial [1]. En effet, alors que de plus en plus de jeunes diplômés sont sans emplois dans le monde, les employeurs se plaignent parfois de ne pas pouvoir trouver des ressources humaines possédant certaines qualifications.

Le Centre pour l'économie et la recherche basé aux Etats Unis, indique que le « skills gap » fait perdre en moyenne, près de 160 milliards de dollars chaque année aux entreprises américaines [2]. Aussi, une étude réalisée en 2017 par CarrierBuilder, établie que 60% des employeurs ont des postes qui restent vacants pendant plus de deux semaines [2].

Au Cameroun, une étude réalisée en 2016 par la banque mondiale, indique que pour réaliser une véritable transformation structurelle, le pays doit identifier les « skill gaps » dans les domaines des technologies et de l'innovation [3]. L'institution renchérit en indiquant que pour mettre fin à la pauvreté chez les jeunes, il faut combler le « skills gap » [3]. Pour ce faire, il faut pouvoir définir formellement le « skills gap » en identifiant les compétences détenues par les jeunes et en les comparant avec celles requises par les entreprises.

Des plateformes de recrutement en ligne ou de « freelance » ont vu le jour ces dernières décennies et permettent aux entreprises de trouver l'expertise recherchée n'importe où, en faisant fi de la contrainte géographique. Il en résulte généralement une multitude de candidats qui postulent pour une offre d'emploi en envoyant leur CV par voie électronique ; ce qui rend davantage complexe le processus de sélection du (des) meilleur(s) candidats. C'est la raison pour laquelle des algorithmes de « matching » automatique CV-offres d'emploi ont vu le jour et proposent de recommander une liste réduite (« short-list ») des candidats dont les profils correspondent le mieux à ceux recherchés. Ces algorithmes nécessitent généralement de pouvoir identifier automatiquement les compétences contenues dans les documents (CV, offres d'emploi, etc.).

Les principaux défis liés à l'identification automatique des compétences dans les textes sont:

1. La compétence est une notion abstraite qui peut être représentée dans les textes de plusieurs manières (avec des termes différents dans des contextes différents). Par exemple, une certification PMP peut indiquer une compétence en « gestion des projets »; les termes "conduite

des projets" et "pilotage des projets" ou encore "IA" et "intelligence artificielle" peuvent faire référence à la même compétence;

2. Les compétences peuvent être hiérarchisées en ce sens que certaines compétences peuvent expliquer d'autres compétences. Par conséquent, une compétence peut ne pas être explicitement mentionnée dans le CV, mais se déduire d'autres compétences explicitement mentionnées. Nous les appellerons des compétences de haut niveau qui peuvent être implicites.

La compétence peut se définir comme étant une qualification professionnelle et se décline généralement en : savoirs (connaissances), savoir-faire (pratique) et savoir-être (comportements relationnels) [124, 125].

C.2 Enoncé du problème

Supposons qu'un employeur recherche une personne qui maîtrise la programmation structurée, et que, la compétence "programmation structurée" ne soit pas explicitement mentionnée dans le CV d'un candidat, mais à la place, les technologies : C, ada, pascal, etc. qui selon la taxonomie élaborée par Evanthia et al. [9] peuvent être caractéristiques d'une certaine maîtrise de la "programmation structurée". Les modèles d'extraction des compétences explicites ne sont pas en mesure d'identifier ces compétences qui ne sont pas explicitement indiquées dans le CV.

L'immense majorité des approches existantes s'intéressent uniquement à l'extraction des compétences explicitement décrites dans les documents par des termes représentant des technologies, des techniques, outils, certifications, etc. Ces approches vont de l'utilisation des dictionnaires [63], à la reconnaissance des entités nommées [7, 8] en passant par la modélisation ontologique [5, 6] pour identifier les compétences explicitement mentionnées dans le CV. Seules quelques approches se sont intéressées à l'identification des compétences implicites mais ne se sont pas intéressées à l'explicabilité des modèles utilisés [61, 63]. Pour d'autres, la méthode employée rend difficile l'évaluation automatique sans connaissance du résultat attendu [61].

Le problème que nous resolvons tout au long de cette thèse s'énonce donc comme suit : ***Etant donné un CV écrit en langage naturel, quelles sont les compétences qui se dégagent de la description de ce CV et comment justifier que le CV exprime effectivement ces compétences ?***

En effet, le fait que le CV puisse exprimer plusieurs compétences fait du problème de prédiction des compétences à partir du CV un problème de classification multi-étiquette où chaque

étiquette représente une compétence.

Dans cette perspective, résoudre le problème énoncé ci-dessus consiste à construire dans un premier temps un modèle capable de prédire un ensemble de compétences à partir du CV, et dans un second temps, à décrire une méthode permettant de justifier les prédictions effectuées par le modèle.

C.3 Objectif et Méthodologie

Les réseaux de neurones convolutionnels ont démontré des performances exceptionnelles dans plusieurs problèmes tels que la classification des images et la classification des textes, atteignant les résultats de l'état de l'art dans plusieurs problèmes de classification de texte [15]. De plus, les travaux de jacovi dans [13] concernant l'explicabilité des CNN pour la classification des textes ouvrent des perspectives nouvelles dans la compréhension des décisions fournies par ce type de modèle.

Considérant les performances des réseaux de neurones convolutionnels, ainsi que des avancées réalisées dans leur interprétation, nous pensons que leur mise à contribution dans la détection des compétences dans les CV serait un choix judicieux.

L'objectif de ce travail est donc de concevoir un modèle de classification multi-étiquette capable de prédire un ensemble de compétences contenues dans les CV tout en justifiant les prédictions réalisées; ceci en mettant en exergue les compétences de niveau inférieure qui ont permis de produire ces décisions.

La méthodologie employée pour résoudre le problème consiste en deux grandes étapes :

- Premièrement, la conception du modèle de classification multi-étiquette en utilisant les CNN comme modèle de base pour la classification. A ce niveau, l'architecture du modèle de classification multi-étiquette est d'abord décrite avec ses différents composants. Par la suite, la méthode de prétraitement des CV est expliquée;
- Deuxièmement, la description d'une méthode permettant d'expliquer les prédictions de chaque prédicteur de base (CNN). La méthode d'explication est d'abord testée sur d'autres problèmes de classification des textes (classification des questions et analyse des sentiments) pour en démontrer sa généricité avant d'être par la suite appliquée au problème de classification des CV.

C.4 Contributions

Les contributions produites dans le cadre de nos travaux se déclinent suivant deux grands aspects a savoir : le modèle de prédictions des compétences ; et la méthodes permettant d'expliquer les modèles de CNN conçus pour la classification des textes.

Modèle de prédiction des compétences

Dans cet aspect, nous présentons l'architecture d'un modèle de classification multi-label (Figure 17) permettant de prédire les compétences de haut niveau à partir des CV. L'architecture du modèle proposé a deux principaux composants, un composant de prétraitrement qui permet de transformer un CV en matrice et un ensemble de modèles de classification binaire dont le rôle est de prédire si oui ou non le CV exprime une compétence particulière.

Plus spécifiquement les contributions apportées dans cet aspect sont:

- La construction d'un modèle word2vec spécifique au contexte à partir des CV du domaine informatique. L'algorithme skip-gram [29] est utilisé pour construire le modèle *word2vec* qui servira par la suite à transformer le CV en matrice;
- La définition d'une méthode efficiente de transformation des CV en matrices en utilisant le modèle *word2vec* construit à l'étape précédente. Pour réduire l'espace mémoire nécessaire pour stocker les CV, une méthode appropriée de selection de caractéristiques est appliquée et consiste à ne retenir que des termes pertinents pour la classification. Cette méthode a permis de réduire de moitié l'espace mémoire nécessaire tout en gardant un haut niveau de précision;
- Nous avons également construit un jeu de données[4] d'environ 30000 CV multi-étiqueté avec la liste des compétences contenues dans les CV. Les CV ont été étiquetés automatiquement en utilisant les qualifications professionelles mentionnées par les experts. Un algorithme de normalisation des compétences a été élaboré afin d'associer les expressions différentes renvoyant à la même compétence.

Les expériementations effectuées ont permis de démontrer l'effectivité du modèle proposé dont le rappel atteint 98,79% et la précision 91,32%.

[4] link to the multi-labeled resume dataset $https://github.com/florex/resume_corpus$

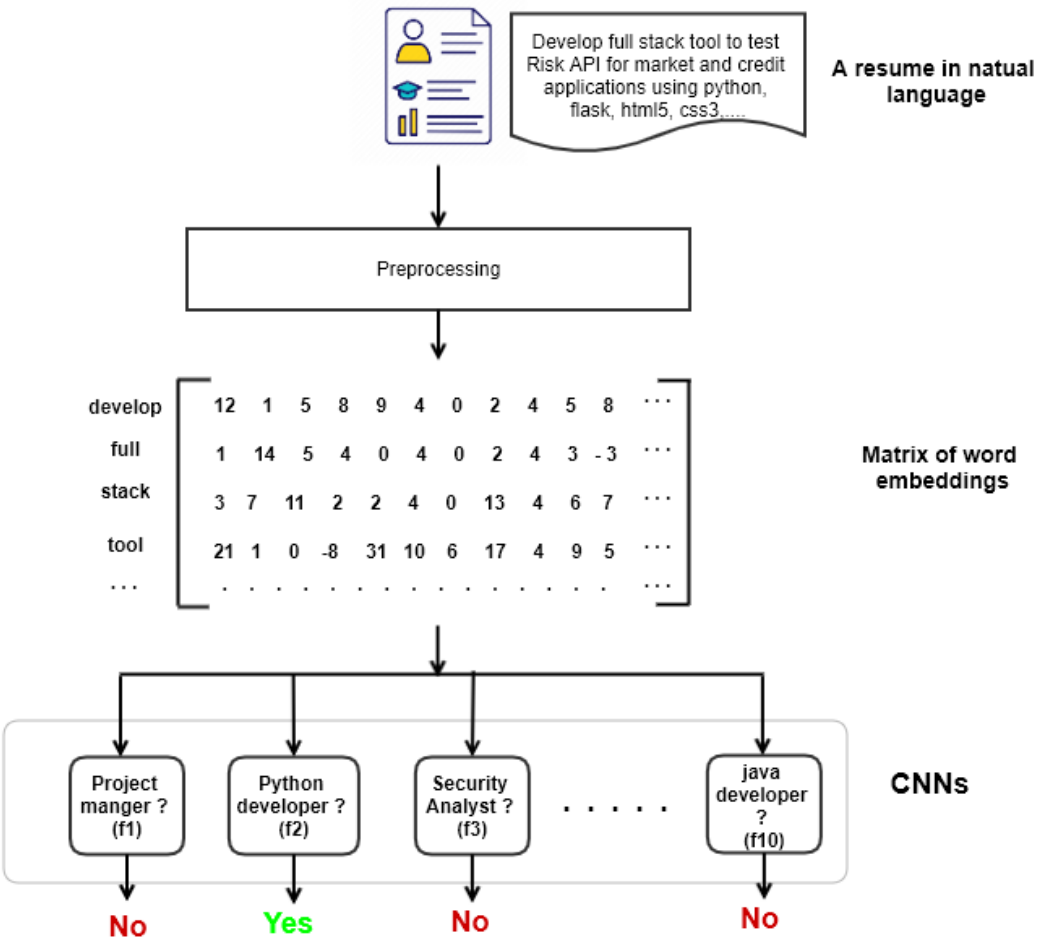


Figure 17: Architecture du modèle de prédiction des compétences.

Méthode d'explication des modèles de CNN conçus pour la classification des textes

Dans cette partie nous proposons une méthode générale d'explication des modèles de CNN conçus pour la classification des textes. La méthode élaborée a été testée sur plusieurs problèmes de classification y compris l'analyse des sentiments, et la classification des questions avant d'être appliquée au modèle de prédiction des compétences. Plus spécifiquement, les contributions mises en exergue dans cette partie sont :

- Nous proposons une méthode utilisant le principe de la méthode LRP (Layer-wise Relevance Propagation) pour calculer la contribution de chaque n-grams à la prédiction du modèle de CNN ;
- Nous avons mis en évidence une limite de la méthode LRP notamment au niveau de la for-

mule de calcul de la contribution. Nous démontrons en effet que lorsque le dénominateur est négatif, les interprétations des contributions peuvent s'avérer erronées. Nous proposons à cet effet une adaptation qui conserve l'effet réel de chaque entrée sur la sortie du modèle ;

- Enfin nous proposons deux algorithmes permettant d'évaluer les caractéristiques suffisantes et nécessaires selon le modèle. Car en effet il n'est pas nécessaire de présenter à l'utilisateur toutes les caractéristiques qui ont eu une incidence sur les résultats. Nous supposons que quelques-unes suffisent.

C.5 Publications

Les travaux contenus dans cette thèse ont fait l'objet d'une communication à la Conférence de Recherche en Informatique (CRI) en 2017 ; d'un article publié dans le journal ARIMA indexé DBLP et d'un article publié dans le journal Neural Computing And Application.

- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze, *Skills prediction based on multi-label resume classification using CNN with model predictions explanation.* Neural Computing And Application, 2020;
- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze, *Approche hiérarchique d'extraction des compétences dans les CV en format PDF*. Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées, INRIA, 2019, volume 32, 2019.
- Jiechieu Kameni Florentin Flambeau, Norbert Tsopze. *Approache hierarchique d'extraction des compétences dans les CV en format PDF*. In the proc. of the 3rd Edition of CRI (Conférence de Recherche en Informatique), Yaoundé, Cameroon, From 28th to 30th November, 2017;

C.6 Perspectives

L'identification des compétences dans les CV trouve son application principalement dans les systèmes de recommandation experts-offres d'emploi. Or la majorité des systèmes de recommandation existant procède d'abord par extraction des compétences aussi bien dans les CV que dans les offres d'emploi avant d'appliquer une mesure de similarité qui permet d'évaluer le degré de correspondance entre les profils des candidats et les compétences requises. Nous pensons qu'il serait a priori plus intéressant de construire un modèle explicable qui apprendrait lui-même par renforcement à faire la correspondance entre les offres d'emploi et les profils et qui justifierait ses

décisions en mettant en exergue les compétences identifiées dans les profils et leur correspondance avec l'offre d'emploi.

Par ailleurs, lors de l'application de la méthode d'explicabilité proposée, nous avons constaté un problème qui, si a priori est correctement solutionné, pourrait grandement améliorer la qualité des algorithmes d'apprentissage surtout dans le cas où la distribution des données entre les classes est très déséquilibrée. En effet, supposons que nous soyons dans un problème de classification binaire avec deux classes A et B. Supposons que 20% des données du jeu de données sont de la classe A et que 80% appartiennent à la classe B. Supposons également qu'il existe une caractéristique x qui apparait dans 100% des données de la classe A soit 20% du jeu de données total, et que x apparait dans 50% des données de la classe B, soit 40% du jeu de donnée. Dans ces conditions, les algorithmes d'apprentissages tels que pensés actuellement auront tendance à attribuer x comme caractérisant plus la classe B car, x apparait avec 2 fois plus de données de la classe B que celles de la classe A. Pourtant, statistiquement, x caractérise a priori A beaucoup plus que B car apparait dans 100% des données de la classe A.

Un autre aspect important à considérer dans le recrutement est la détection des fausses compétences mentionées par le candidat. En d'autres termes, comment faire confiance au candidat ?. Etant donné que le travail décrit dans cette thèse permet déjà de caractériser les compétences de haut niveau, il pourrait être étendu à la détection des fausses compétences en vérifiant si les compétences de base mentionées dans le CV correspondent effectivement aux profils (compétences de haut niveau) indiqués par le candidat.

D Rapport de Pré-Soutenance

Rapport de Pré-soutenance sur la thèse intitulée
«Explainable Deep Neural Network for Skills Prediction from Resumes»

Nom de l'étudiant : JIECHIEU KAMENI Florentin Flambeau

En vue de l'obtention du titre de
Docteur/PhD en Informatique de l'Université de Yaoundé I
Filière: Informatique
Ecole doctorale: STG

Les travaux présentés par **M. JIECHIEU KAMENI Florentin Flambeau** se sont effectués sous la direction de **Dr. Norbert TSOPZE** et la supervision de **Pr. Maurice TCHUENTE**. Dans cette thèse, le candidat s'intéresse à l'identification automatique des compétences dans les CV, en utilisant les réseaux de neurones profonds. Il propose également d'expliquer les compétences identifiées par ces modèles neuronaux.

1 –Problématique de la Thèse

Le « Skills Gap » qui se définit comme l'écart entre les compétences détenues par les ressources humaines d'une organisation et celles dont elle a besoin pour son développement, est un problème majeur, qui selon une étude récente réalisée par la banque mondiale est un facteur de frein à l'émergence du Cameroun.

Des plateformes de recrutement en ligne ou de « freelance » ont vu le jour pour pouvoir permettre aux entreprises de trouver l'expertise recherchée en faisant fi de la contrainte géographique. La conséquence qui en résulte est généralement une multitude de demandes pour une offre d'emploi, rendant complexe le processus de sélection du (des) meilleur(s) candidats. C'est la raison pour laquelle des algorithmes de *matching* automatique entre CV et offres d'emploi, ont vu le jour et proposent de recommander une *short list* de candidats dont les profils correspondent le mieux à ceux recherchés. Ces algorithmes nécessitent de pouvoir identifier automatiquement les compétences contenues dans les documents (CV, offres d'emploi).

Plusieurs chercheurs ont proposé des approches pour extraire les compétences dans les CV allant de l'utilisation des dictionnaires, à la reconnaissance des entités nommées, et en passant par la modélisation ontologique.

Le candidat note toutefois que l'immense majorité des méthodes existantes ne s'intéressent qu'à l'extraction des compétences explicitement décrites par des termes clés du domaine. Peu d'auteurs ont abordé la problématique de l'identification des compétences implicites qui sont des compétences qui ne sont pas explicitement identifiées par des termes dans les documents, mais pourraient se déduire de la description du document. Par ailleurs, en dehors du fait que les auteurs ne se soient pas intéressés à l'explicabilité des modèles proposés, leur définition de la compétence implicite n'intègre pas ce qu'on qualifie de compétence de « haut niveau » ("high level skill") ou profil professionnel qui peut être : « gestion de projet », « programmation web », etc.

Le problème abordé dans la thèse se reformule ainsi qu'il suit : *« Étant donné un CV écrit en langage naturel, quels sont les profils professionnels qui se dégagent de la description du CV ? et comment justifier les profils identifiés ?»*.

Les travaux effectués par le candidat et décrits dans cette thèse ont deux objectifs majeurs : (1) proposer un modèle capable d'identifier les compétences de « haut niveau » dans les CV ; (2) proposer une méthode pour expliquer les prédictions effectuée par le modèle.

La thèse est rédigée en Anglais et est organisée en 5 chapitres comme décrit ci-dessous.

2 –Contenu du mémoire

Chapitre 1 : General Introduction

Dans ce chapitre, le candidat présente le contexte de son travail qui est marqué par un environnement où les entreprises font de plus en plus recours aux solutions de recrutement en ligne (plateforme de recrutement, email, etc), et qui donnent la possibilité à toute personne à travers le monde, de candidater à une offre d'emploi, ce qui peut alourdir les activités de sélection des meilleurs profils lorsque celles-ci sont faites manuellement.

Le candidat commence par présenter les approches existantes pour résoudre ce problème et constate qu'elles n'abordent pas l'identification des profils professionnels qui sont considérés comme compétences de haut niveau (selon la hiérarchisation des compétences) parce qu'ils peuvent s'expliquer par un ensemble de caractéristiques de base (technologies, outils, techniques, etc).

Enfin il décline sa méthodologie de résolution du problème qui implique deux grandes étapes :

1. La proposition d'un modèle de classification multi-label basée sur les réseaux de neurones profonds en l'occurrence le CNN (Convolutional Neural Network) capable de prédire la liste des compétences contenues dans un CV ;
2. La spécification d'une méthode permettant d'expliquer les prédictions de ce modèle.

Chapitre 2: LITERATURE REVIEW.

Dans ce chapitre, le candidat présente les fondements théoriques du domaine de NLP (Natural Language Processing) qui encadre ses travaux.

En effet, le CV étant un texte écrit en langage naturel, pour pouvoir identifier les compétences, il est

nécessaire d'utiliser des techniques de Traitement Automatique du Langage Naturel (TALN). C'est la raison pour laquelle le candidat consacre ce chapitre à la présentation des principes, techniques, outils et méthodes permettant de traiter les textes écrits en langage naturel, y compris les modèles d'apprentissage profond (deep learning).

Chapitre 3: SKILLS PREDICTION MODEL.

Dans le chapitre 3, après une étude critique des méthodes d'extraction des compétences existantes, le candidat présente la méthode proposée.

Il décrit donc dans un premier temps l'architecture du modèle de classification qu'il propose avec les différents composants qui sont : (1) le composant de prétraitement ; et (2) un ensemble de modèles de classification binaire conçus à base des Réseau de Neurones Convolutionnels (CNN en Anglais pour Convolutional Neural Networks). Le composant de prétraitement permet de transformer un CV en matrice à partir d'un modèle de *word embedding* construit par lui-même à partir d'un ensemble de 30000 CVs téléchargés sur Internet. Le deuxième composant décrit par le candidat est un ensemble de modèles de CNN, chacun spécialisé dans la prédiction d'une compétence spécifique.

M. Jiechieu a expérimenté son modèle sur le jeu de données décrit plus haut et obtient de bonnes performances (98,79% de rappel et 91,34% de précision) comparées à des modèles utilisant d'autres techniques d'encodage de texte (one-hot et doc2vec).

Chapitre 4: MODEL EXPLANATION

Après avoir brièvement présenté l'état de la connaissance sur l'explicabilité des modèles de traitement automatique du langage naturel, le candidat présente dans ce chapitre, la méthode qu'il propose pour expliquer les décisions des modèles de CNN pour la classification des textes.

La méthode qu'il propose se base sur l'algorithme LRP (Layerwire Relevance Propagation) pour calculer la contribution de chaque caractéristique en entrée de la couche densément connectée du CNN. Par la suite, il détermine les termes associés à chacune des caractéristiques précédentes et détermine également leur contribution. Le candidat identifie au passage une limite dans la formule de calcul des contributions de la méthode LRP de base. En effet, il montre que lorsque le dénominateur est négatif, les interprétations des contributions peuvent être erronées. Il propose donc d'encadrer cette valeur. Enfin, l'une des contributions dans cette partie du travail effectué par le candidat, est l'élaboration des algorithmes permettant de déterminer les caractéristiques suffisantes et les caractéristiques nécessaires pour l'explication.

Les expérimentations effectuées par le candidat ont permis d'attester que la méthode proposée est efficace et ouvre des perspectives nouvelles en ce qui concerne la problématique d'explicabilité des modèles d'intelligence artificielle. Le candidat a en outre comparé les résultats de la distribution des importances attribuées aux termes contenus dans le CV avec celle obtenue à partir de LIME un modèle très connu de l'état de l'art. Les résultats se sont avérés très similaires, sauf que la méthode proposée par M. Jiechieu est plus efficiente en ce qui est de la complexité en temps.

Chapitre 5 : GENERAL CONCLUSION

Le chapitre 5 est consacré au bilan des travaux menés. Le candidat rappelle dans ce chapitre les grandes lignes de sa méthodologie ainsi que les principaux résultats obtenus. Il a non seulement proposé un modèle d'identification des compétences contenues dans un CV, en démontrant son efficacité, mais aussi proposé une méthode d'explication consistant à calculer la contribution des termes en entrée d'un modèle CNN aux valeurs prédites par ce dernier. Ensuite, il a proposé une adaptation de l'algorithme LRP ainsi qu'un algorithme permettant de déterminer les caractéristiques suffisantes et les caractéristiques nécessaires pour simplifier les explications fournies aux utilisateurs.

Ce travail ouvre plusieurs perspectives, entre autres, en expliquant les résultats d'un classifieur, on peut améliorer les prédictions de ce dernier en mettant plus d'importance sur les attributs les plus importants pour une classe. Par ailleurs, on peut détecter des fausses informations dans les CVs.

3 –Publications

Les travaux contenus dans cette thèse ont fait l'objet d'une communication à la Conférence de Recherche en Informatique (CRI) en 2017 ; d'un article publié dans le journal ARIMA indexé DBLP et d'un article publié dans la revue *Neural Computing And Application* de facteur d'impact **4.774 (2019)**.

- **Jiechieu Kameni Florentin Flambeau**, Norbert Tsopze, *Skills prediction based on multi-label resume classification using CNN with model predictions explanation*. Neural Computing And Application, 2020

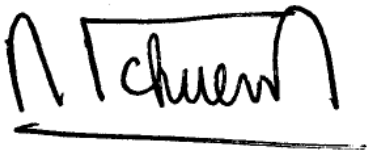
- **Jiechieu Kameni Florentin Flambeau,** Norbert Tsopze, Approche hiérarchique d'extraction des compétences dans les CV en format PDF. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, INRIA, 2019, volume 32, 2019.

- **Jiechieu Kameni Florentin Flambeau**, Norbert Tsopze. Approache hierarchique d'extraction des compétences dans les CV en format PDF. *In the proc. of the 3rd Edition of CRI (Conférence de Recherche en Informatique),* Yaoundé, Cameroon, From 28th to 30th November, 2017.

Un autre travail sur l'explicabilité des modèles de *deep learning* est en cours de révision dans le journal *Neural Networks* de Elsevier.

En conclusion, nous avons suivi l'ensemble des travaux de M. JIECHIEU et nous tenons à signifier notre entière satisfaction quant aux résultats obtenus et donnons par conséquent un **avis favorable** pour la soutenance publique de cette thèse en vue de l'obtention du grade de Docteur/PhD en informatique de l'Université de Yaoundé I.

Le Directeur de thèse **Le Superviseur de thèse**



Norbert TSOPZE
(Chargé de cours, UYI)

Maurice TCHUENTE
(Professeur, UYI)

E Publications

E.1 Skills prediction based on multi-label resume classification using CNN with model predictions explanation

Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05302-x

ORIGINAL ARTICLE

Skills prediction based on multi-label resume classification using CNN with model predictions explanation

Kameni Florentin Flambeau Jiechieu[1,2] · Norbert Tsopze[1,2]

Received: 6 February 2020 / Accepted: 18 August 2020

Abstract
Skills extraction is a critical task when creating job recommender systems. It is also useful for building skills profiles and skills knowledge bases for organizations. The aim of skills extraction is to identify the skills expressed in documents such as resumes or job postings. Several methods have been proposed to tackle this problem. These methods already perform well when it comes to extracting explicitly mentioned skills from resumes. But skills have different levels of abstraction: high-level skills can be determined by low-level ones. Instead of just extracting skill-related terms, we propose a multi-label classification architecture model based on convolutional neural networks to predict high-level skills from resumes even if they are not explicitly mentioned in these resumes. Experiments carried out on a set of anonymous IT resumes collected from the Internet have shown the effectiveness of our method reaching 98.79% of recall and 91.34% of precision. In addition, features (terms) detected by convolutional filters are projected on the input resumes in order to present to the user, the terms which contributed to the model decision.

Keywords Skill-gap · Resume · Skills extraction · Multi-label classification · Convolutional neural network · Model explanation

1 Introduction

The Association for Talent Development (ATD) defines a *skill-gap* as a significant gap between an organization's current capabilities and the skills it needs to achieve its goals and meet customer demand [7]. Organizations bridge the skills gap by hiring candidates with specific skills to perform critical tasks. Nowadays, hiring processes are often conducted through the Internet. Applications (resumes and cover letters) are sent via e-mails, web sites or job posting platforms (indeed.com, freelancer.com, upwork.com, ...). The number of applications for a particular job can be very important, making the candidates selection cumbersome. It is to solve this problem that job-resume matching algorithms have been developed in recent years in order to quickly find candidates whose skills match those required to perform the job. But, many of these algorithms require user inputs [13, 16, 27]: a user should input his skills as keywords list; as well, skills needed to perform the job are supposed to be known in advance. The matching then consists in computing a similarity score between competences of candidates with those required and ranking the candidates according to their scores. However, things become more challenging when dealing with raw text resumes. In that case, a prior work should consist in extracting topic words (skills in our context) from resumes before matching them to the ones extracted from jobs. It has been shown that using topic words in content-based recommendation is more accurate than using any other word [2].

Lots of works have been done to tackle the problem of skills extraction from resumes. Some researchers have proposed the use of ontology to identify skills from resumes [4, 5], while others have suggested to handle the skills extraction problem as a named entity recognition (NER) problem where skills are considered as named

✉ Kameni Florentin Flambeau Jiechieu
florentin.jiechieu@mintp.cm

Norbert Tsopze
norbert.tsopze@facsciences-uy1.cm

[1] Department of Computer Science, University of Yaounde I, Yaounde, Cameroon

[2] IRD, UMMISCO, F-93143 Bondy, France

Published online: 28 August 2020

entities [15, 30]. Moreover, skills can be organized into different levels of abstraction. There are low-level skills (e.g., css, html, php, ...) and high-level ones (e.g., web developer, front-end developer, ...). A huge work has already been done by some researchers to build skills taxonomies [9, 10] which represent the hierarchical relationship between skills of many domains including IT. We assume that high-level skills are characterized by a set of low-level ones. While existing methods [15, 26, 30] already perform remarkably well when extracting skill-related terms from resumes, they are limited by their inability to infer high-level skills not explicitly mentioned in resumes. In IT, for example, the observation of *html, css, javascript, etc.* could lead a recruiter to deduce that the candidate is a *front-end developer* even if this was not explicitly mentioned in his resume. On the other hand, a recruiter could doubt that a candidate has a certain competence despite the fact that it has been mentioned in his resume; just because there were no sufficient elements to convince him that the candidate truly has that expertise.

Artificial intelligence models have evolved over the years and offer excellent performances in solving varieties of problems in different fields (job recommendations, image processing, word processing, medical analysis, etc.). However, if these models are powerful in terms of the accuracy of predictions, they often suffer from the problem of opacity: in fact, they operate more like black boxes providing results without being able to explain them. However, entrusting an important decision to a system that cannot be explained presents an obvious danger [1]. This is the main reason why the construction of explainable artificial intelligence models has been gaining increasing attention in recent years.

Moreover, Deep learning models especially Convolutional Neural Networks (CNN) has shown surprising results in text processing, achieving states of the art results in many classification tasks. One advantage of CNN is that they are able to automatically discover features from inputs to perform classification. This property could be used to automatically identify terms (low-level skills) describing high-level skills. In addition, many research works have been conducted in the sense of analyzing CNN in order to explain predictions in text classification [14]

Following the above considerations, we propose in this article a method to build an explainable model based on Convolutional Neural Networks (CNN) to identify high-level skills from raw text resumes. The resulting model is capable of predicting the set of high-level skills expressed in an input resume, while highlighting the low-levels skills which have led to that prediction. We believe that the proposed method would be more trustful and transparent for recruiters. We also believe that this work could lay the foundations for the construction of transparent and explainable job recommender systems using CNN.

To achieve this goal, we first propose to handle the problem of skills extraction from resumes as a multi-label classification problem, where the output classes are high-level skills; then, we analyze the predictions in order to derive for each CNN filter the set of words that it is specialized in identifying, and to know which words contributed to the prediction of a particular competence for a given input resume (prediction explanation). The multi-label classification model is a binary relevance-based model [11] where we have as much binary classifiers as there are classes in the overall dataset. Each binary classifier is based on CNN. To explain the classifier results, we rely on a part of the work done by Jacovi et al. [14] to detect, project and visualize the words selected by filters on the original resume.

The main contributions of this article are summarized as follows:

1. The use of a multi-label classification model based on CNN to predict high-level competences from resumes. Resumes are transformed into matrices using a convenient method, and the resulting matrices are used as inputs to CNN;
2. The construction of a context-specific word embedding model for resumes of the IT domain;
3. The analysis of filters to explain predictions and to guarantee confidence in the classifier decisions;
4. We conducted several experiments to demonstrate the effectiveness of our method.

The rest of this article is structured as follows: the next section presents existing works about skills processing and model predictions explanation. The third section describes the proposed method. Then, the results of our experiments are presented before ending with a conclusion.

2 Related works

Skills extraction from resumes is a critical task when building job recommender systems. It is also useful for automatically building experts competence profiles and companies competences knowledge-bases.

2.1 Works about skills extraction

Many parsing algorithms have been developed to tackle the problem of skills extraction from resumes:

Shiqiang et al. [13] have proposed a personalized job-resume matching system, which could help job seekers to easily find appropriate jobs. They created a finite state transducer-based information extraction library to extract

models from resumes and job descriptions. Then, they defined a new statistical-based ontology similarity measure to match the resume models and the job models. The extracted model consists of degrees, majors and skills.

Zhao et al. [30] proposed a combination of Named Entity Recognition (NER) and Named Entity Normalization (NEN) to identify skills from texts, considering them as named entities. The approach consists in annotating a set of resumes with skills they contain in order to build a dataset that will be used to train a Named Entity Recognition model capable of identifying the set of skills contained in a text resume. This approach has the potential to recognize all the skills mentioned inside the resume and can be used to build a kind of resume summary containing the list of skills that the expert possesses. But generally, Named Entities Recognition needs huge amount of annotated corpora to perform well; and it is difficult to manually build such datasets. To tackle this problem, they proposed a method to automatically annotate the resumes using Wikipedia Common Categories, which they consider as a taxonomy of skills. However, the problem now resides in the precision and the completeness of the labeling. Further in [15], the same authors added a skill entity word sense disambiguation component which infers the correct meaning of an identified skill by leveraging Markov Chain Monte Carlo (MCMC) algorithms.

Sayfullina et al. [26] proposed an ontology-based model to extract skills from resumes. To be tagged as a skill, a noun phrase in a resume must be found inside the ontology. This poses the problem of the completeness of the ontology in the sense that if the ontology is not enough complete, then many skills will not be identified. To deal with that, they proposed a way to find other skills that are not present in the ontology. This is done with the help of some specific and lexicalized multi-word expression patterns (i.e., specific contexts) that could surround new and unknown skills. New skills are validated by an expert and added to the ontology.

The common point between the preceding methods is that they all parse input resumes and extract skills from them. This means that the resume must actually contain those skills. But in practice, some skills can be deduced by the recruiters when reading the resume, even if they do not appear explicitly inside the resume description. For example, an expert's resume might contain basic skills such as (css, html, javascript, ...) which normally would lead a recruiter deducing that the expert has the profile of a web developer. The approaches which consist in just parsing the resume cannot be used to find implicit high-level skills.

Kiyimaki et al. [18] proposed *Elisit*, a graph-based approach to skills extraction from resume. They use the hyperlink graph of *wikipedia* and skills extracted from *linkedIn* to associate skills with documents. They first analyze the input document with a vector space model in order to associate it with a *wikipedia* article. From this initial article, they use Spreading Activation Algorithm [6] on the hyperlink graph of *wikipedia* in order to find articles that correspond to *LinkedIn* skills and are related to the initial pages. As stated by the authors, developing a completely automatic optimization scheme for this model selection task is difficult because of the number of different parameters, the size of the Wikipedia graph and the heuristic nature of the whole methodology. That is why they evaluated their model manually.

2.2 Related works about models explanation

The majority of works about CNN models explanation have been done for image processing [8, 12, 19, 21, 24, 25]. Until now, only few research works concern the interpretation of CNN models built for text classification tasks. Jacovi et al. in [14] have recently proposed an approach to explain CNN models built for text classification tasks. They show that filters can be classified into three categories based on their contribution in the classifier prediction: (1) accidental filters which do not influence the classifier decision; (2) filters which recognize good patterns (patterns which actually describe the target class); (3) finally, filters which recognize bad patterns (patterns which do not describe the target class). The main limitation of their approach is that it applies only to a strictly limited range of CNN architectures: architectures with just one layer (the output layer) in the fully connected stage. Indeed, the output layer should come immediately after the max-pooling layer. This is why they assumed that the class of a filter is the class which maximizes the weights of the connections between the max-pooled value corresponding to that filter and the outputs neurons.

In [23], the authors proposed LIME, a model-independent approach to interpret the predictions. The idea is to consider the model to interpret as a black-box. Input features are varied to observe their impact on the outputs. Thus, the influence of input features on the outputs can be evaluated. This approach grows exponentially (in terms of time complexity) with respect to the number of features; and in text classification, we generally deal with more than dozens of thousands of features.

3 Proposed method

The method proposed in this article to deal with the problem of skills prediction as a multi-label classification task will be presented following the steps below:

1. First, the architecture of the multi-label skills prediction model is described along with its components;
2. Then, the methods used to "preprocess" the training data and build the components of the architecture are described ;
3. Finally, we describe how filters are analyzed to explain the predictions of the built model.

3.1 Multi-label classification architecture model for skills prediction

The first step in our method is to design a model capable of predicting a set of skills from an input resume. Since a resume might express multiple skills, we propose to handle the problem as a multi-label classification task. Figure 1 describes the architecture of the multi-label skills prediction model.

The architecture defined in Fig. 1 consists of two main components:

1. A preprocessing component which goal is to transform a raw text-resume, into a matrix which will serve as input for the CNN;
2. And a multi-label classifier component composed of a set of binary CNN classifiers.

3.1.1 Preprocessing component

When a raw text resume is passed as input to the model, the preprocessing component (preprocessor) transforms this resume into a matrix, before passing it as input to the CNN classifiers. The preprocessing consists in using a domain dictionary and a word embedding model to filter and embed the word vectors of a resume into a numerical matrix. We consider all the resume matrices to have the same length L which is defined empirically. The transformation of resume from text to matrix operates as follows:

1. Initially, an empty row matrix M is created;
2. Repeat the followings for each term w in the resume, until the length of the matrix is reached, or there is no other word to proceed:
   - if w is found in the domain dictionary and if there is an entry corresponding to the word w in the word embedding model then:
     (a) convert w into a numerical vector v using the embedding model;
     (b) append the corresponding vector v to the bottom of the matrix M;
3. If the length of the matrix is less than L then zero-padding is added to the bottom of the matrix to reach L.

At the end of these steps, we have the matrix of the resume which then goes as input to each base CNN classifier.

3.1.2 CNN classifiers

The second component of the architecture, consists of multiple independent binary CNN classifiers. Each CNN is used to classify resume according to a single class. In other words, a base CNN classifier f_j is designed to predict whether an input resume x belongs to the class c_j or not. The number n of base classifiers corresponds to the number of competences (classes) in the dataset (in practice, recruiters may know in advance, the sets of competences they would like to identify from resumes).

The prediction of the multi-label model then consists of the union of the competences predicted by the different base classifiers as specified in formula 1.

$$C_i = \bigcup_{j=1, f_j=1}^{j=n} c_j \tag{1}$$

We choose CNN as based classifiers for two main reasons:

1. Firstly, input of CNN generally consists of raw data which are less subject to information losses as compared to other models which generally required handcrafted features.
2. Secondly, convolutional filters can easily be analyzed for explanation purpose;

Figure 2 shows a view of the architecture of a base CNN classifier. This architecture is inspired from the architecture proposed by Kim, Yoon [17] and consists of a convolutional layer followed by a max-pooling layer, a ReLU layer and a fully connected layer. The model takes as input a matrix representation of a resume and predicts a value corresponding to the probability that the resume belongs to the target class or not.

Formally, a resume can be modeled as an n-words input text $w_1, w_2, \ldots, w_n$, and each word is embedded in a d dimension vector $w_i \in R^d$. So a resume can be represented as a matrix $M \in R^{d \times n}$. A convolution filter can be represented by a d dimension vector. For each filter f_j, the convolution among the resume matrix performs the scalar product $< w_i, f_j > (1 \leq i \leq n)$. The convolutions result in a matrix $F \in R^{n \times m}$ where m is the total number of filters. The columns of F represent the features maps. Applying max-pooling on the features maps results in a vector $p \in R^m$ which contains the maximum values of each features map. The components of the vector p are rectified by the ReLU activation unit before being passed as input to the fully connected neural network (FCNN) which computes the final decision. These steps are summarized as follows:

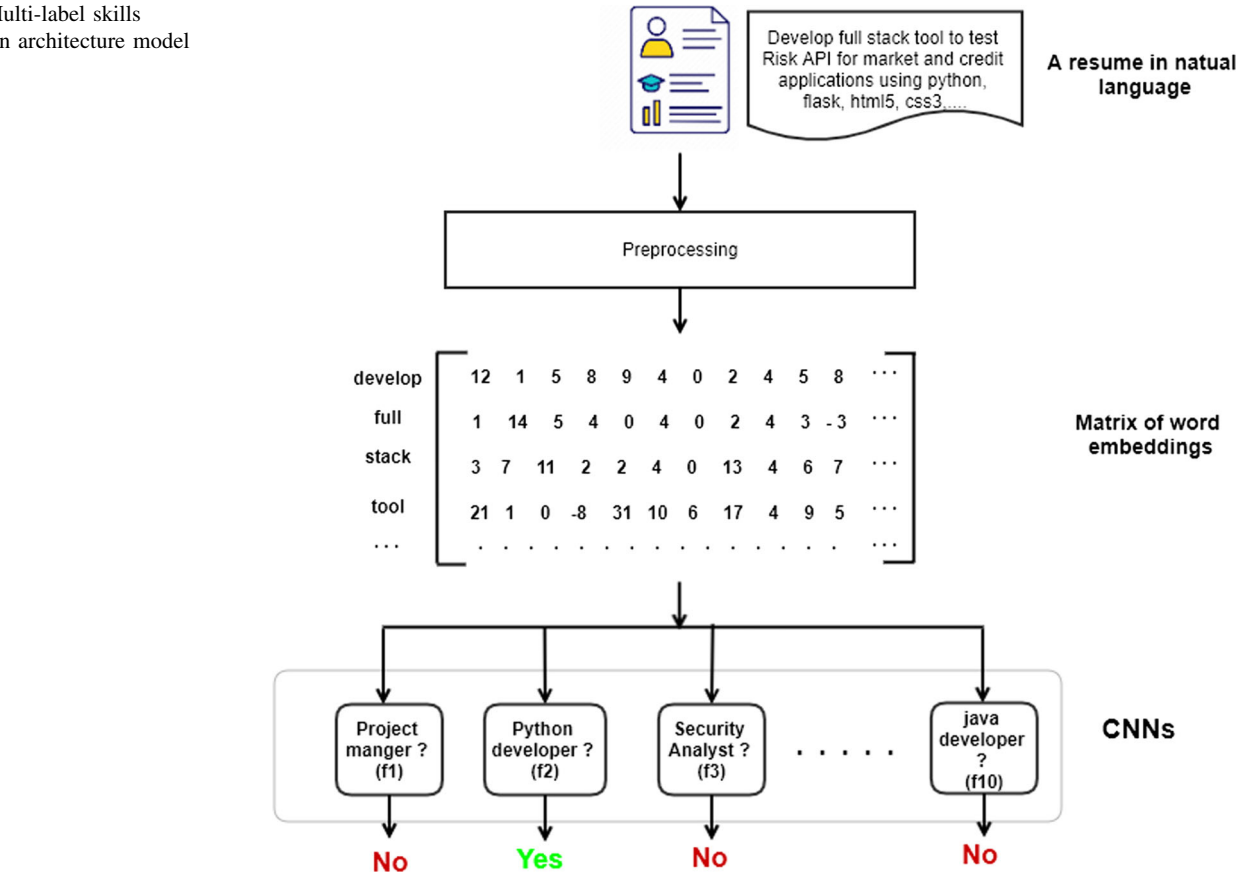


Fig. 1 Multi-label skills prediction architecture model

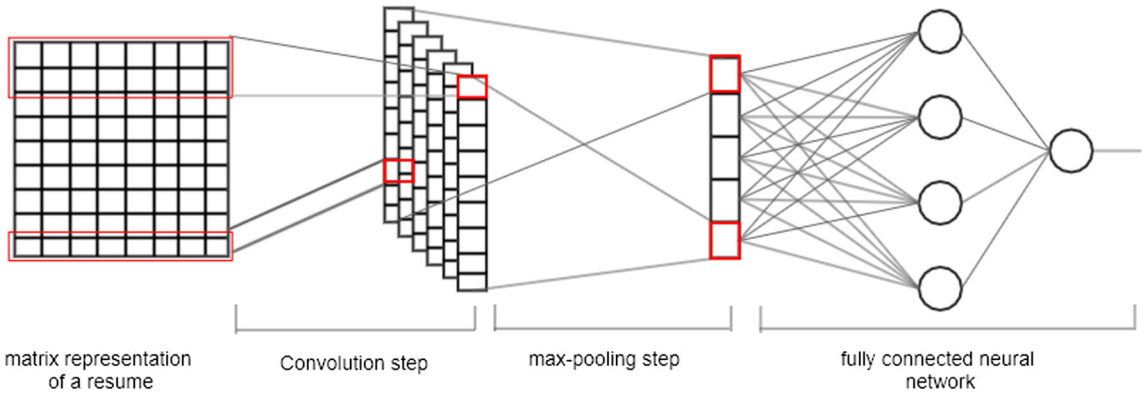


Fig. 2 CNN architecture for text classification

1. **Convolution**: this operation is used to compute a similarity score between a filter and a word vector (Eq. 2). For simplicity, we considered filters with kernel size equal to 1 (filter height). It means that the convolution filters will specialize in recognizing single words. However, this work can also be easily adapted to work with variable kernel size filters.

$$F_{ij} = < w_i, f_j > \tag{2}$$

2. **ReLU (Rectified Linear Unit)**: this operation is used to rectify the convolution output by setting any negative value to zero. The ReLU function is defined by Eq. 3, where x is a real value.

$$y = \mathrm{ReLU}(x) = \max(x, 0) \tag{3}$$

3. **Max-pooling**: the operation performed here is a global max-pooling; meaning that the max-pool filter operates over the whole features map and selects its maximum value. This value is associated with the word which

produced the highest convolution score with the convolution filter. The component p_j of the max-pooled vector is given by Eq. 4, where F is the matrix of features maps and the column $F_{.j}$ is the features map associated with the filter f_j.

$$p_j = \text{Max}_i(F_{ij}) \tag{4}$$

4. **Classification**: this is done by the fully connected neural network (FCNN). In the proposed architecture, the FCNN is composed by one hidden layer with the ReLU activation and one output unit (output layer) with a sigmoid activation. The hidden units receive as input, the max-pooled vector p and compute their activation h_i by Eq. 5 where W is the matrix of weights of the connections between hidden units and the input of FCNN (components of p); b_i are biases.

$$h_i = \text{ReLU}(\sum_j W_{ij} \times p_j + bi) \tag{5}$$

The final output is calculated by Eq. 6.

$$y = sigmoid(W_i \times h_i + b) \tag{6}$$

The sigmoid function used is the logistic function: $sigmoid(x) = \frac{1}{1+e^{-x}}$, where W is the vector of synaptic weights of the connections between the output unit and the hidden units.

The overall network is trained using the stochastic gradient descent (SGD) algorithm which consists in searching the best parameters (W, b) i.e., the parameters which minimize the gradient of the error function. During the backpropagation, the weights W are updated using the general equation.

$$W_i^{t+1} = W_i^t + \eta \frac{\partial E_i}{\partial W_i} \tag{7}$$

W is the matrix representing the weights of the units in the fully connected neural network, η the learning rate and E the classification error.

After having described the overall architecture model and its main components, we now focus on how the various models composing this architecture are built (trained) to solve the multi-label classification problem.

3.2 Building the architecture models

To build and train the models composing the architecture described in the above section, we have assembled a set of resumes collected from the Internet. Each resume is labeled with the set of professional occupations of its owner.

The following processing steps are applied in order to build and train the models used in the architecture:

1. First of all, classes (competences) are normalized to have a common representation for competences;
2. Next, a word embedding model is trained from the corpus of resumes using the skip-gram algorithm ;
3. Then, resumes are transformed into matrices using the trained word embedding model;
4. After that, the original multi-label dataset is splitted into n single-label dataset; each used to train a single base classifier;
5. Finally, the base CNN classifiers are trained using the appropriate sub-datasets.

3.2.1 Classes normalization

The first step before building the models is to normalize the competences. In fact, input resumes consist of raw text resumes from the IT domain and are written in HTML format. We consider occupation titles as high-level skills (java developer, project manager, ...): the resumes have been labeled with these titles. Since HTML is a structured format, it is easy to extract those occupation titles to automatically label resumes. But occupation titles are not normalized: experts might use different expressions to represent the same occupation. For example: experts might use *DBA, database admin, database administrator or database administration to represent the title "Database administrator"* or *java programmer, java coder, java developer, java engineer, python/java developer to represent the title "java developer"*. Since we have dozens of thousands of resumes, it would be fastidious to normalize all those titles manually. To deal with that problem, we wrote an automated "normalizer" algorithm which operates with a hand written dictionary. The dictionary contains for each normalized competence, a list of expressions which are generally used to identify that competence. Table 1 presents an overview of the dictionary.

The algorithm operates as follows:

1. Iterate over all the expressions in the dictionary and compare each of them to the occupation titles defined in resumes:
2. If an expression matches with one of the occupation titles of the resume then, the corresponding resume is labeled with the normalized competence associated with that expression.

Instead of comparing the whole expression, we compare words because expressions in the dictionary and those in the resume might not match exactly. For example: An expert with a title "developer (java, python)" will be labeled with the competences "java developer" and "python developer" since his occupation title contains all the words of the expression "java developer" and those of

Table 1 Different expressions of occupation titles

Normalized class	Expressions
Python_Developer	python django, python flask, python developer, python programmer, python engineer, python software developer, python software engineer, django developer, django programmer, django engineer, django software developer, django software engineer, python web developer, python application developer, python application engineer
Systems_Administrator	sys admin, sysadmin, systems engineer, system administrator, system admin, systems admin, systems administrator, systems specialist, sys administrator, system engineer, systems engineer
Database_Administrator	database administrator, database engineer, database programmer, database developer, database admin, dba

"python developer." Comparing words instead of the whole expressions also prevents us from putting inside the dictionary all the different expressions that experts might use to express the same occupation; what would be practically impossible.

In this step, we also consider the hierarchical relationship among competences [9]. Figure 3 shows an example of the hierarchical relationship between the high-level competences: *java developer, python developer, front-end developer, web developer and software developer*. It also shows examples of low-level features which describe each competence. We recall that if an expert is a java developer, then he is also a software developer. To handle the hierarchical relationship among competences, we use a simple taxonomy derived from [9] and restricted to the set of normalized competences that we have considered as the output classes. Using the skills taxonomy, the initial set of competences of a resumes is extended by adding their parent competences. For example, based on taxonomy presented in Fig. 3 "software developer" generalizes "python developer" and "java developer." Therefore, "Software Developer" skill will be added to the set of competences of resumes labeled with "java developer" or "python developer."

3.2.2 Training word embedding

Word embeddings are used to build the matrix representation of resumes. The interest of using word embedding mainly resides in the fact that it captures the semantic of words and preserves the semantic similarity between them. It also prevents the problem of data sparseness when it comes to representing text documents. The effectiveness of word vectors has been proven in natural language processing [3]. Embedding the semantic of words into vectors is particularly useful when working with CNN for text classification, because we want each filter to identify closely related words. Given a word vector, the cosine measure can be used to compute the similarity with other word vectors.

We use the skip-gram [20] algorithm to train our own word embedding model from the resume corpus. It has been proved that context-specific word embedding models better capture the relatedness between words than general models [22]. In fact, context-specific corpora are less subject to the problem of polysemous words [28, 29]: a word can have different meanings depending on the context. For example, the term java can mean a coffee or a programming language. Generally, when training word embedding models, two main parameters are considered: the dimension of the word vector and the windows size which refers to the number of words to consider before and after the target word to represent its context. The word embedding model used in this work was built using a dimension of 100 and a windows size of 5. The size of the dimension and the windows size were defined empirically. We observed satisfactory performances using these values. Example 1 shows the most similar words to the term

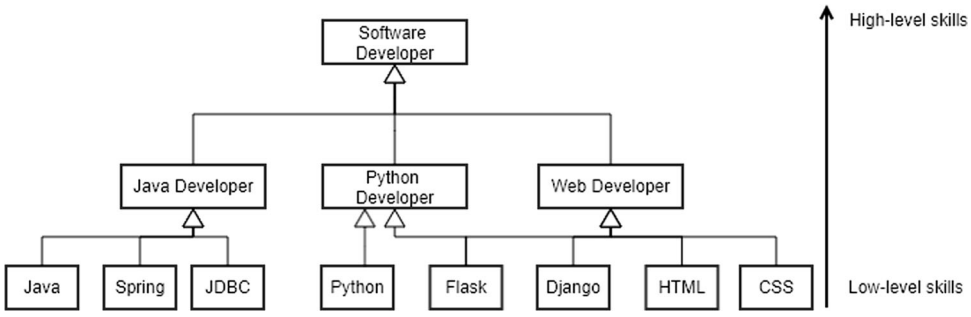


Fig. 3 Example of hierarchical relationship among competences

angular along with their similarity values. From this example, we observe that the most similar terms to angular (angularjs, zepto, vue, angular4, backbone, react, ...) out of hundreds of thousands of words in the vocabulary are either other written forms of angular (angularjs, angular2, angular4) or other javascript frameworks (zepto, backbone, react, ...). This illustrates the effectiveness of our self-trained word embedding model.

Example 1 Most similar words to the word angular in the resume corpus:

[('angularjs', 0.720), ('angular2', 0.718, ('zepto', 0.7039), ('vue', 0.686), ('angular4', 0.6773), ('backbone', 0.666), ('react', 0.663), ('skrollr', 0.658), ('angualr', 0.656), ('colorbox', 0.654)].

3.2.3 Words filtering and resumes transformation into matrices

CNNs were initially designed for images which are represented as 2-dimension matrices. If we want to use CNN to classify texts, then we will have to convert input texts to matrices. The common method to convert a text to a matrix is to convert each of its words in a vector and assemble them into a matrix. The detail method is described in Sect. 3.1.1.

In many text classification tasks, the length of the resulting matrix is generally fixed to the length of the resume which has the maximum number of words. If that method is practical when dealing with small documents like tweets, it can be problematic when dealing with huge documents in terms of number of words. Indeed, a resume could have up to 12 000 words. If we consider that the real numbers in the word vectors are represented in simple precision, and that the dimension of each word vector is set to 100, then we will need about 4.8Mo of memory to represent a resume matrix and 144Go to represent a dataset of about 30 000 resumes. To reduce the total amount of memory used, common methods suggest to fix the length of the resume to a smaller value than the maximum length. Therefore, if a resume has a length greater than that value, then extra words will be removed from it to reach the fixed size. If instead, its length is less than the fixed value, then 0-padding will be added at the bottom of the resume to reach the fixed length. However, the problem with just setting the length is that many relevant terms can be ignored. Instead, in this article, we empirically fixed the size of the resume matrices and considered a domain dictionary where relevant words are selected. The dictionary was established using IT skills gathered from *dice.com*[1] and other collected from the corpus of resumes.

[1] https://www.dice.com/skills.

3.2.4 Building sub-datasets to train base classifiers

To form the training set for each base classifier, the original multi-labeled dataset is divided into n single-labeled datasets (n being the total number of classes in the original dataset). Each resulting sub-dataset corresponds to a binary classification problem focused on a unique competence. To build the single-label dataset D_i, for a target class c_i, the class of each resume example x_j is set to *positive* (1) if in the original dataset the input x_j belongs to the class c_i and to *negative* (encoded by 0) else. This procedure is illustrated in Fig. 4.

Moreover, assuming that we have n classes, and that resumes are equally distributed among them, the dataset used to train a base classifier, will have in average $1/n$ of examples belonging to the target class and the rest belonging to the other classes. For example: if $n = 10$, then 10% of the examples will belong to the target class, and 90% will be of the other classes. That unequal distribution of examples among positive class (target class) and negative class (other classes except target class) can lead the model to a kind of overfitting: the model specializes in learning the features of the negative class in detriment of those of the positive class. To deal with that problem, we equilibrate the dataset accordingly: all the examples belonging to the target competence are retained, and an equal number of examples are randomly selected among the examples of the other classes. The sub-dataset formed is then shuffled before training the corresponding base classifier.

3.3 Model predictions explanation

The main goal of model predictions explanation is to go over the black box predictions by providing a human understandable justification to the model predictions [21]. To justify the model result, we recover words detected by convolution filters and project them on the original resume. We assume that each filter recognizes a single word or an n-gram depending on the filter kernel size. The goal is then to recognize which words each filter identifies, or which terms have led the model to predict a particular class. At the end of this step, the algorithm provides the different terms that might have influenced the classifier output decision. In the case of images classification, this set of features, mainly responsible for the classifier decision is called saliency map [12]. The sentences (explanations) in the form "*the model predicts these classes because the following terms (basic skills) are present in the input resume*" are provided to the user.

Knowing that filters are used to recognize features from the input resumes, and that features can be assimilated to

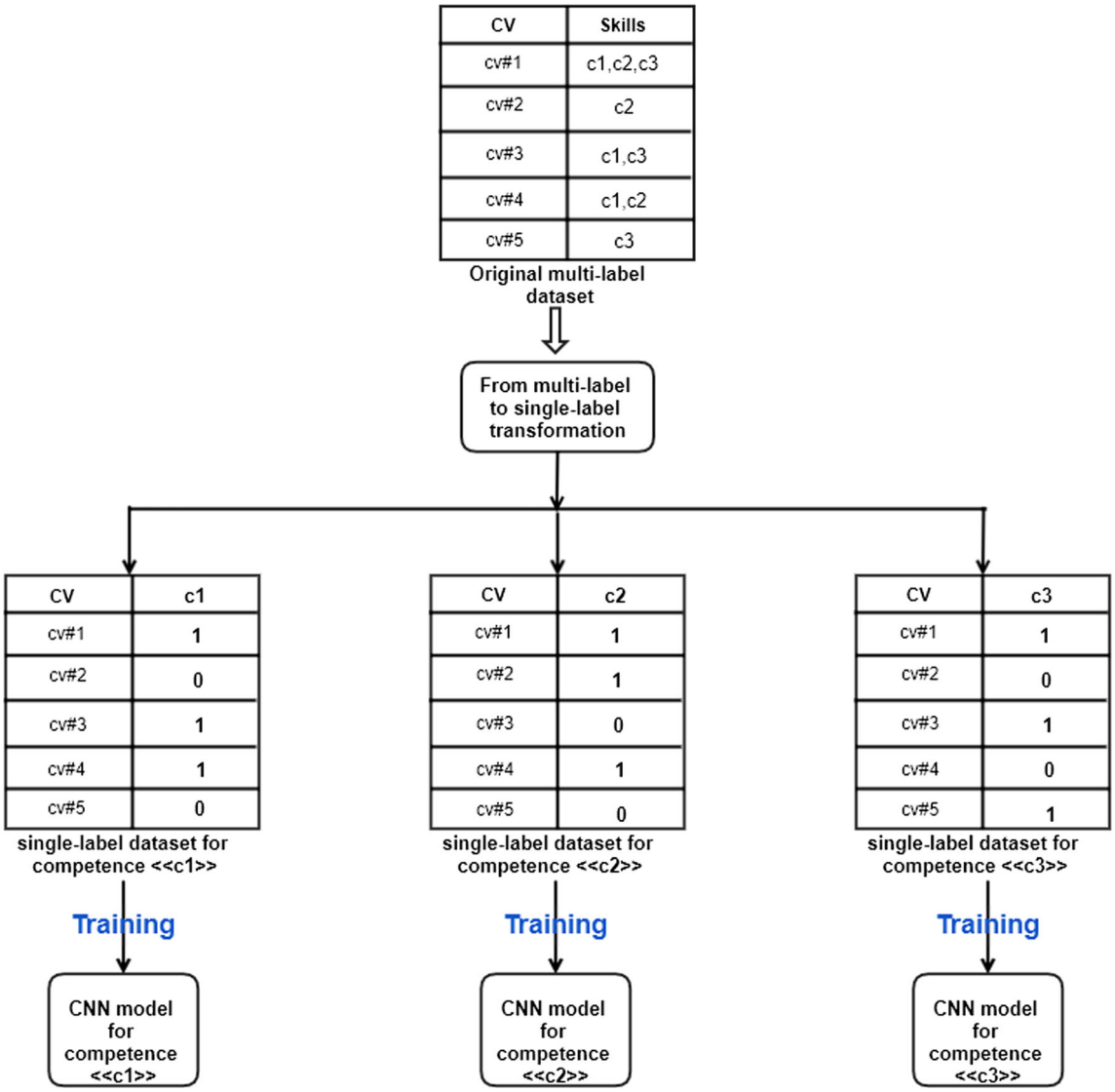


Fig. 4 From multi-label dataset to single-label datasets

words or n-grams regarding their shape, it is easy to find which words or n-grams were selected by a given filter. In fact, the words or n-grams which were not selected in the max-pooling layer do not influence the classifier decision since the values of their convolution with the filter will not be considered in the next stage (convolution and max-pooling are discussed in detail in Sect. 3.1.2).

Let f_j be a filter of kernel size 1, and $W = w_1, w_2, \ldots, w_n$ the set of words contained in the resume. The word detected by f_j is the word w_j^* which maximizes the scalar product $< w, f_j >$. w_j^* is determined by the formula 8.

$$w_j^* = \arg\max_w (< w, f_j >), w \in W \tag{8}$$

Only those words will influence the classifier decision since they are associated with the max-pooled values of the features maps, and they are those values which are used as input to the fully connected layer. The formula (8) is used to recover the words detected by filters which are then projected on the original resume. From these words, user might recognize the ones which have led the model to predict a particular competence.

4 Results

In this section, we first present the analysis and description of data before discussing the results of the experiments.

4.1 Descriptive data analysis

The experiments were carried out on a collection of 28 707 anonymous resumes available publicly on

indeed.com[2] website and distributed in 10 classes with the proportions in Fig. 5. The number of pages of resumes ranges from 1 to 6. The extracted resumes are all from the IT domain and are related to the competences: *Software Developer, Front-End Developer, Network Administrator, Web Developer, Project Manager, Database Administrator, Security Analyst, Systems Administrator, Python Developer, Java Developer*. As shown in Fig. 5, the skill *Software developer* is the most represented, while *Python developer* is the least represented. We recall that *Software Developer* includes (*Java Developers, Python Developers, Web Developers* and *Front-End Developer*).

A descriptive text analysis has also been done on resumes of each class to find out the most relevant terms describing each class. To achieve this goal, we evaluated the importance of each term with respect to a competence. TF-IDF (formula 9) was used to evaluate the importance of a term with respect to a competence. The importance of a term w with respect to a competence c, is calculated by Eq. 9. The intuition behind this method of evaluating the importance of terms is that an important term for a competence is a term which appears frequently in resumes labeled with that competence and less frequently in resumes labeled with other competences. In fact, terms which appear frequently in all resumes, are not discriminant enough; that is why the standard frequency is not appropriate since it will highlight several non-discriminant terms.

Let D be the corpus of resumes. Let D_i be the corpus of documents which belongs to the class i. We consider the following definitions:

Definition 4.1 The importance of a term $w \in D$ with respect to a class i is the product of the frequency of that term with respect to the class i, and the inverted document frequency of the term with respect to the class i.

$$\mathrm{TF-IDF}_i[w] = \mathrm{TF}_i[w] * \mathrm{IDF}_i[w]. \tag{9}$$

Definition 4.2 The frequency of the term w with respect to the class i ($\mathrm{TF}_i[w]$) is the frequency of w in the restricted corpus of resumes labeled with the class i. $\mathrm{TF}_i[w]$ is calculated by the formula 10.

$$\mathrm{TF}_i[w] = \frac{\text{Number of times } w \text{ appears in the corpus } D_i}{\text{Total number of terms in } D_i}. \tag{10}$$

Definition 4.3 The inverted document frequency of the term w with respect to the class i is calculated by the logarithm of the inverse of the document frequency of the term w in the counter corpus (corpus of documents which are not labeled with the class i).

IDF_i is calculated with respect to the counter corpus of D_i in order to penalize terms which appear frequently in resumes which are not of the target class i.

$$\mathrm{IDF}_i[w] = \log\left(\frac{|D|}{|d, d \in D - D_i|}\right). \tag{11}$$

Table 2 shows an overview of the 10 most important terms per class listed among 56000 terms. From this table, we observe that *System_Administrator* and *Network_Administrator* are closely related competences because several of their most important terms (*switch, cisco, firewall, directory, vmware*) are similar. On the other side, we observe that the top-words describing competences *python developer, java developer, security analyst* and *database administrator* do not overlap much with those of other competences. But, the competences *front-end* and *web_developer* have similar top-terms *(css3, responsive, html5, bootstrap*). These observations are confirmed by Table 3 which shows the degree of co-occurrences of the 100 most important terms of each competence. From this table, we can observe that 65 out of the 100 most important terms describing *Front-end Developer* and *Web developer* are the same; and 81 out of the 100 most important terms describing *Network administrator* and those describing *Systems administrator* are the same. *Software developer* has common descriptive terms with other developer-related competences because it generalizes them.

4.2 Evaluation of each CNN model

First, the accuracies of base classifiers are evaluated separately. After, the overall multi-label model is evaluated using appropriate measures. Table 4 shows the accuracy of each individual base classifier. These results were obtained with the following parameters: the number of filters is equal to 100; the filters kernel size set to 1; the stochastic gradient descent used as the optimizer; the batch size equal to 64 with 100 epochs, 0.01 as the learning rate and a momentum of 0.1. Those parameters were found through experiments, and they produced satisfactory results.

These results show that the model performs very well with an accuracy of about 99% for the competences: *Java Developer, Python Developer, Security analyst, Database administrator* and *Project Manager*. In the same time, the model predicts the competences: *Web developer, Front-end Developer, Network administrator* and *System administrator* with an accuracy of around 95%. The high accuracy of the first group can be explained by the fact that those competences can easily be distinguished from others because their low-level features do not overlap much with

[2] https://www.indeed.com (consulted on the 21th *Aug* 2019).

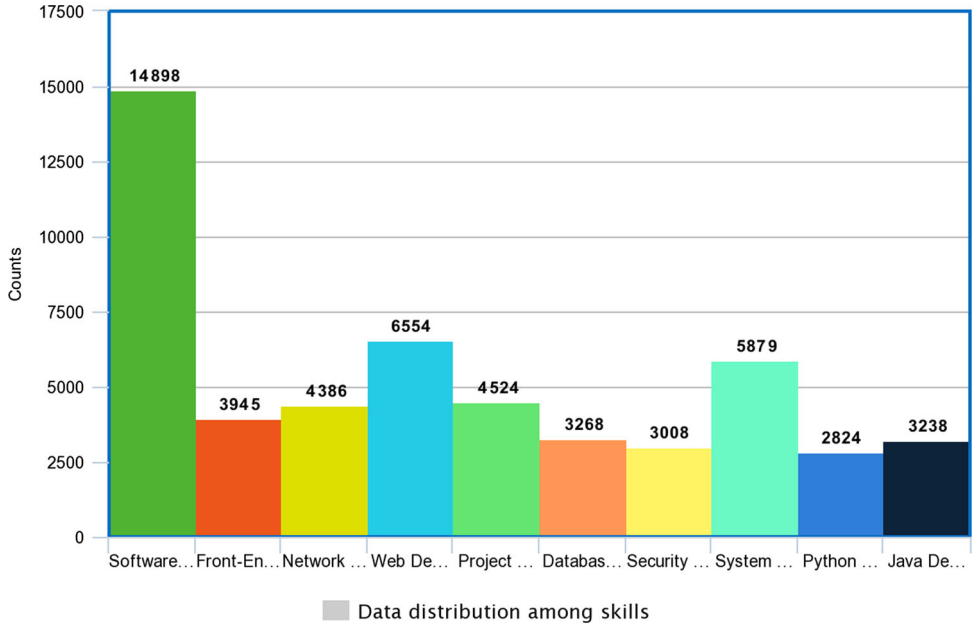


Fig. 5 Classes distribution in the dataset

Table 2 10 most important terms per classes

Classes	10 Most important terms
Software_Developer	('jquery', 0.709), ('git', 0.684), ('api', 0.679), ('ui', 0.679), ('css', 0.674), ('bootstrap', 0.673), ('html5', 0.672), ('front', 0.666), ('javascript', 0.663), ('ajax', 0.657)
Front_End_Developer	('responsive', 1.111), ('front', 1.047), ('sass', 1.031), ('css3', 1.026), ('photoshop', 0.976), ('browser', 0.945), ('html5', 0 .941), ('bootstrap', 0.932), ('react', 0.920), ('ui', 0.917)
Network_Admin	('switch', 1.050), ('cisco', 1.036), ('firewall', 0.953), ('lan', 0.937), ('router', 0.891), ('directory', 0.888), ('wan', 0.888), ('vpn', 0.848), ('vmware', 0.842), ('equipment', 0.805)
Web_Developer	('jquery', 0.813), ('html5', 0.789), ('bootstrap', 0.784), ('css3', 0.765), ('front', 0.754), ('responsive', 0.745), ('css', 0.744), ('website', 0.740), ('page', 0.736), ('wordpress', 0.724)
Database_Administrator	('dba', 1.303), ('tune', 1.015), ('rman', 0.967), ('tuning', 0.942), ('replication', 0.940), ('recovery', 0.908), ('index', 0.880), ('backup', 0.859), ('rac', 0.854), ('administrator', 0.799)
Security_Analyst	('vulnerability', 1.339), ('cyber', 1.141), ('threat', 1.119), ('nist', 1.077), ('analyst', 1.074), ('incident', 1.070), ('remediation', 1.021), ('assessment', 0.987), ('risk', 0.971), ('compliance', 0.970)
Systems_Administrator	('vmware', 0.935), ('directory', 0.918), ('active', 0.801), ('hardware', 0.791), ('administrator', 0.787), ('cisco', 0.784), ('switch', 0.761), ('dns', 0.751), ('backup', 0.738), ('firewall', 0.721)
Python_Developer	('django', 1.815), ('flask', 1.618), ('panda', 1.581), ('numpy', 1.546), ('python', 1.412), ('pycharm', 1.409), ('matplotlib', 1 .395), ('scipy', 1.258), ('2.7', 1.229), ('postgresql', 1.227)
Java_Developer	('hibernate', 1.647), ('j2ee', 1.629), ('jdbc', 1.566), ('servlet', 1.560), ('junit', 1.556), ('jsp', 1.522), ('strut', 1.495), ('maven', 1.490), ('log4j', 1.456), ('spring', 1.450)
Project_Manager	('budget', 1.025), ('leadership', 0.738), ('manager', 0.732), ('vendor', 0.730), ('strategic', 0.717), ('stakeholder', 0.708), ('cost', 0.705), ('risk', 0.693), ('scope', 0.684), ('executive', 0.683)

that of other competences (as discussed in Sect. 4.1). Indeed, many of the terms describing java developers are not the same as those describing python developers and others. But many of the terms (81 out of the 100 most important terms) describing network administrators overlap with those describing System administrators because those competences are closely related (Table 3). As well, terms describing web developers are much the same as those describing front-end developers. Following these observations, some *network administrators* will likely be classified as *systems administrators* and vice versa because both competences involve almost the same basic skills. The same reasoning applies for *web developers* and *front-end developers*. That situation will inevitably lead to ambiguities in both the learning and the evaluation processes because some examples will actually have the

Table 3 Overlapping of most important terms describing competences

	Software Developer	Front-End Developer	Network Admin	Web Developer	Project Manager	Database Admin	Security Analyst	Systems Administrator	Python Developer	Java Developer
Software Developer	100	45	0	76	0	1	0	0	37	37
Front-End Developer	45	100	0	65	1	0	0	0	13	10
Network Admin	0	0	100	0	14	13	21	81	0	0
Web Developer	76	65	0	100	1	0	0	0	30	28
Project Manager	0	1	14	1	100	3	17	15	0	0
Database Admin	1	0	13	0	3	100	3	16	1	3
Security Analyst	0	0	21	0	17	3	100	22	0	0
Systems Admin	0	0	81	0	15	16	22	100	0	0
Python Developer	37	13	0	30	0	1	0	0	100	17
Java Developer	37	10	0	28	0	3	0	0	17	100

Table 4 Accuracy of each base classifier

Classifiers	Accuracy (%)
Java Developer	99.18
Python Developer	99.20
Web Developer	96.30
Front-End Developer	95.18
Network Administrator	94.88
Systems Administrator	95.65
Software Developer	99.08
Security Analyst	99.30
Database administrator	99.11
Project Manager	99.29

characteristics of a competence but not labeled as such: the prediction will be evaluated as incorrect even if in fact it could be correct.

4.3 Evaluation of the overall multi-label model

Let n be the number of instances (resumes) in the test-set, A_i (resp. Z_i) the set of actual (resp. predicted) competences for the instance i. To evaluate the multi-label model, we use three measures generally used in the literature [11]:

4.4 The accuracy

The accuracy for an instance i measures the proportion of competences which were predicted correctly to the total number (predicted and actual) of competences of that instance. Then, the accuracy of the multi-label architecture is the average of accuracies calculated over all the instances. The global accuracy is calculated by the formula 12.

$$A = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i \cap Z_i|}{|A_i \cup Z_i|}. \tag{12}$$

4.5 The precision

The precision for an instance i measures the proportion of competences which were predicted correctly by the model to the total number of predicted competences of that instance. The overall precision is the average of the precisions evaluated over all the instances. The global precision is calculated by the formula 13.

$$A = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i \cap Z_i|}{|Z_i|}. \tag{13}$$

4.6 The recall

The recall for an instance measures the proportion of competences which were predicted correctly by the model to the total number of actual competences of that instance. The overall recall is the average of recalls evaluated over all the instances. The global recall is calculated by the formula 14.

$$A = \frac{1}{n}\sum_{i=1}^{n}\frac{|A_i \cap Z_i|}{|A_i|}. \quad (14)$$

Table 5 shows a comparison of the results of our multi-label classification model with those obtained from other models which are based on different resumes' encoding methods. The recall of 98,79% indicates that our model almost always predicts all the expected competences. On the other hand, the value of the precision shows that 91,34% of competences are well predicted by the model. However, if we can rely on the fact that when a resume is labeled by a competence, it means its owner really has that competence, the opposite is not always true. More precisely, the fact that a resume has not been labeled by a particular competence does not necessarily mean that it does not express that competence. An expert can be labeled as a *System administrator* but actually has skills of a *network administrator* or *project manager*. This could explain the gap between the recall and the precision. Indeed, in many cases, the model predicts a competence based on words it has identified in the resume. But just because the resume was not labeled with that competence, the evaluation will mark it as incorrect. So, in practice, the precision of the model would be greater than that we evaluated on the test set due to incomplete resumes labeling by experts. In addition, even though a high recall and a high precision are preferable in all cases, in the context of job-resume matching, the recall seems to be more important than the precision because job-matching algorithms are not meant to recruit, but to produce a shortlist of resumes in order to ease the recruiter work. Thus, the fact that a job recommender system fails to select good candidates is more a major concern than the fact that it accidentally selects bad ones. In the latter case, we can still rely on human to filter and eliminate unfit candidates.

Table 5 Comparison of our method with other text encoding methods applied on resumes

	Recall (%)	Precision (%)	Accuracy (%)
word2vec+CNN	98.79	91.34	90.22
doc2vec+LR	94.68	81.45	78.63
one-hot+LR	92.28	80.11	78.07

In Table 5, our model (word2vec+CNN) is compared to some models built using other resumes' encoding methods and logistic regression as classifier: (1) one-hot encoding + logistic regression(LR) as base classifiers and (2) doc2vec + logistic regression (LR) as base classifiers. We observe that our model outperforms the model based on "doc2vec encoding" where resumes are transformed into vectors using a self-trained doc2vec model. Our model also outperforms the one based on "one-hot encoding." This is likely because the matrices of words' vectors better preserve the semantic of words and reduce information loss as compared to "doc2vec encoding" where the whole semantic of the resume is embedded into a single vector or to the "one-hot encoding" where the semantic relationship between words is not taken into account. The classifier used (CNN vs LR) also plays a significant role in the model performance.

These results demonstrate the effectiveness of embedding resumes into matrices and using CNN as base classifiers in the multi-label architecture.

4.7 Contribution of word filtering

Here, we analyze the contribution of word filtering in the model performance. When describing the proposed method, we said that instead of selecting any word from the resume, we selected domain related words because they might be more informative than common language words. We will call the method which consist in selecting domain words **DWS** which stands for domain words selection in opposition to the standard method where only common stop words are removed. Figures 6 and 7 show, respectively, the comparisons of the evolution of the overall recall and the precision of both methods (DWS and standard) with respect to the length of the resume. From these figures, we observe that when using the standard filtering method the recall is high while the precision is very low. This can be explained by the fact that many common words were selected to encode the resume into a matrix. This assumption is confirmed by Table 6 which shows the number of words per resumes when no filtering is applied, after stopwords filtering is applied, and after only domain words are selected. Those common words a priori cannot suitably discriminate resumes according to their competences; instead, they might introduce noises and classification errors when training the base classifiers. We also observe that the performances of both method increases with the length of the resume: that means a resume representation with much words increases the number of features used for classification thus improving the overall model performance. In conclusion, we can say that the DWS filtering method presents a better trade-off between

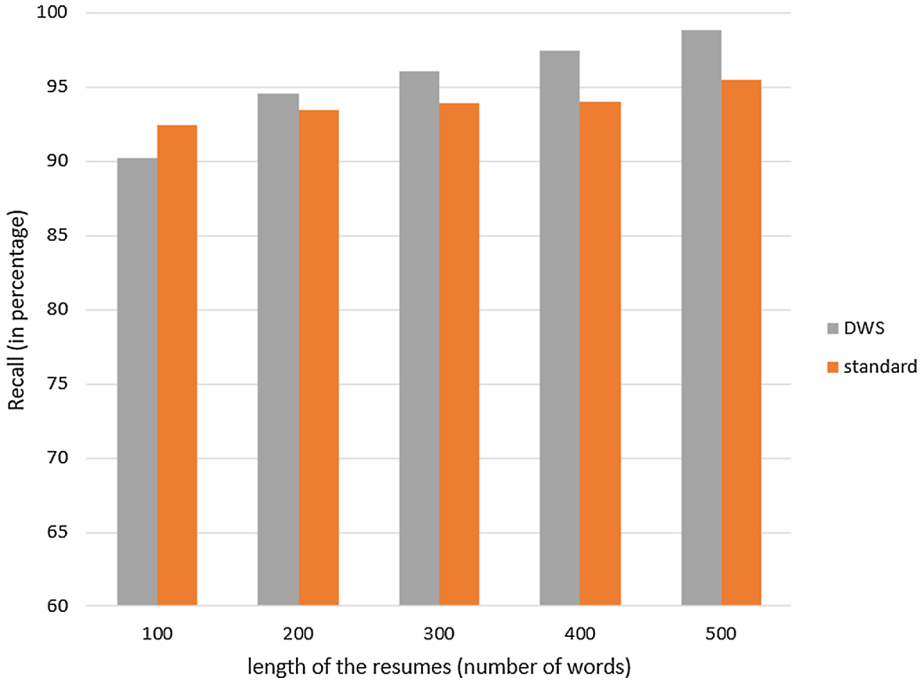


Fig. 6 Recall of DWS filtering versus that of standard filtering

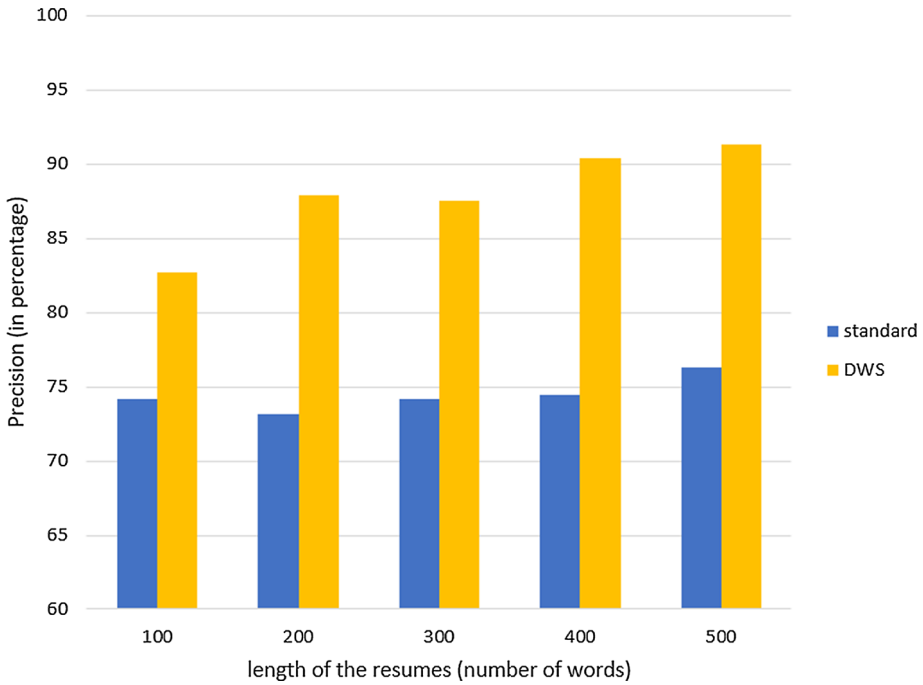


Fig. 7 Precision of DWS filtering versus that of standard filtering

Table 6 Word counts (wc) per resume after filtering

Filtering method	min wc	max wc	avg wc
No filtering applied	17	12479	948
Stopwords filtering	0	9809	724
Domain words selection	0	5526	450

the precision and the recall than the standard filtering method.

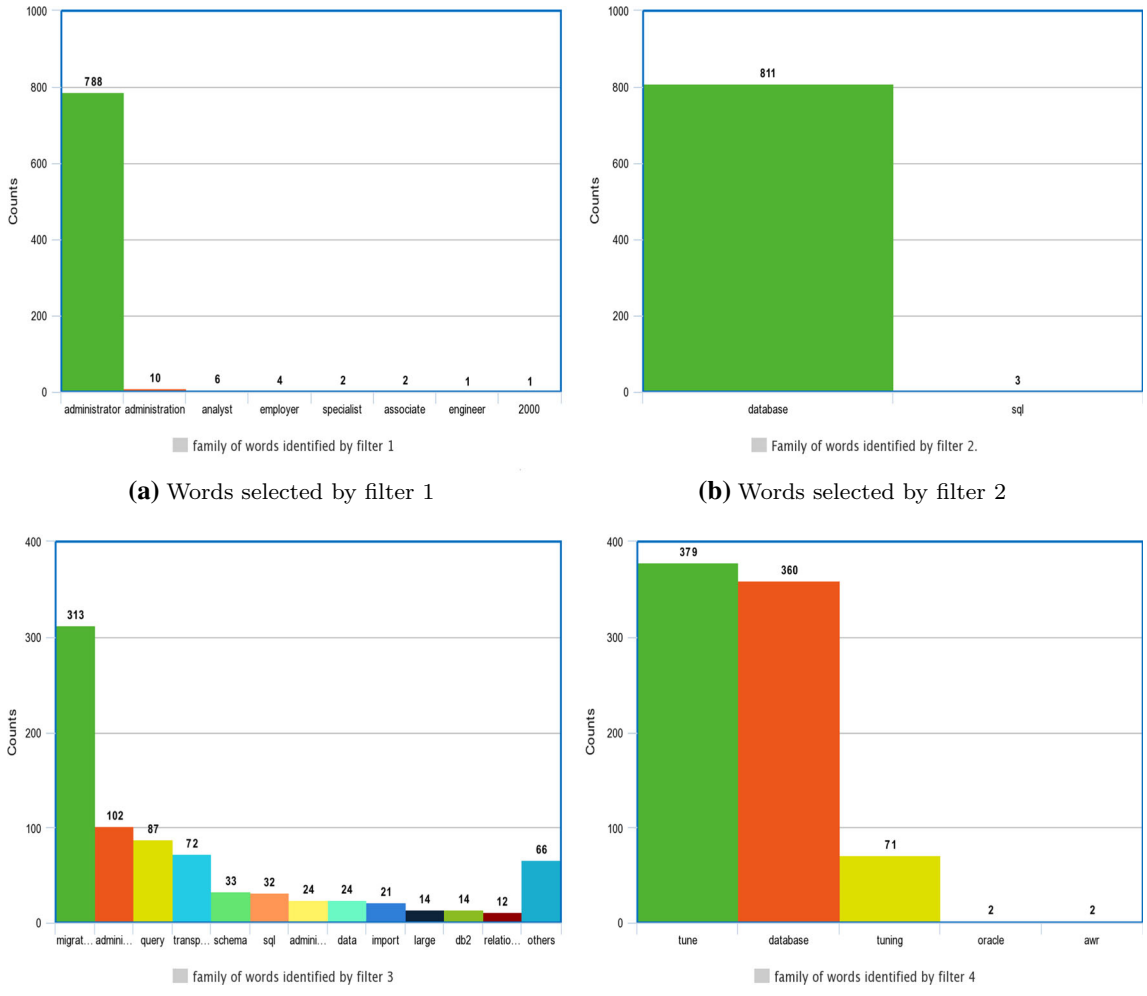
4.8 Understanding the model predictions

In this section, we analyze filters to understand the predictions of each single base CNN classifier. Two different levels of analysis are considered:

1. First, a filter-level analysis is done to understand which words each filter is specialized in identifying;
2. Secondly, an analysis at the resume level is performed to understand which words were responsible for the classifier decision for an input resume.

4.8.1 Filter-level analysis

First of all, we would like to determine which words each filter is specialized in detecting. To find out those words, we ran the model on a test set of resumes of a specific competence. Then, we computed the number of times each word was selected by each filter. Figure 8a–d shows the words identified by four filters using a test set of resumes labeled with the competence "database administrator." We observe in Fig. 8a that the filter 1 has detected the word *administration* 788 times over 814 resumes. By observing this result, we can assume that filter 1 is specialized in the detection of the word *administration*. The other words *(analyst, employer, specialist, ...)* were identified in resumes where the word *administration* was not found. Similarly, the filter 2 (Fig. 8b) is specialized in the detection of the word *database* following the same reasoning. Since the test set was restricted to resumes labeled with the "database administrator" competence, it is normal that the term "database" appears in almost all the resumes. That is why that filter selected the word database 811 times over 814. When the word database is not present in a resume, the filter still selects a term closely related to it (sql). Filter 3 (Fig. 8c) mainly detects the word *migration*. In case, this word is not in the resume, the filter selects other words such as *administration, query,...* Filter 4 (presented in Fig. 8d) specializes in detecting the words: *tune, database, tuning, oracle*. We observe that those filters mainly select words closely related to the competence *database administrator*. The filter-level analysis can be used to identify the sets of low-level features that explain each competence.



(a) Words selected by filter 1

(b) Words selected by filter 2

(c) Words selected by filter 3

(d) Words selected by filter 4

Fig. 8 Filter analysis for the "Database administrator" specialized model

Fig. 9 Words identified by filters in a project manager resume

4.8.2 Resume-level analysis

After having analyzed the general behavior of filters, we now focus on the explanation of the model prediction for a given input resume. Figure 9 shows the resume of a *project manager* expert where the words selected by the convolution filters have been projected. The label of that resume has been correctly predicted by the classifier. The *multi-highlight browser plugin*[3] was used to highlight the words selected by the filters on the original resume. The colors are related to the frequency of the highlighted words in the resume. We also observe that many words related to project management (project, program, implementation, analysis, resolution, manager, deliverable, plan, organize, ...) were selected by filters. And those words can give to the user an explanation of the classifier decision. From the analysis, the model provides the following explanation to the user: *the expert is classified as a project manger because the terms: project, program, implementation, analysis, resolution, manager, deliverable, plan, organize ...were found in his resume*.

We also consider the case of an incorrect prediction. Figure 10 is a resume predicted by the model as *python developer*, but the title *python developer* does not appear in that resume. We recall that they are those occupation titles that were used to label the resume for training. Figure 10 also shows the words selected by the filters to produce the final decision. In fact, even though the resume was not labeled as a *python developer*, the resume actually

[3] https://chrome.google.com/webstore/detail/multi-highlight/pfgfgjlejbbpfmcfjhdmikihihddeeji: a browser extension used to highlight words in web pages.

Fig. 10 A resume which was not labeled as "python developer" but predicted by the model as such

expresses python developers skills: *developer, regression, python, scikit-learn, tool, test, script, flask (a python framework)*, etc. So, the model has identified some words characteristic of a *python developer* and has produced his decision based on those words. That is why concerning the model evaluation, we said that, in practice, the precision might be greater than that we evaluated on the dataset due to incomplete resumes labeling. In fact, it is difficult to manually label resumes with all the competences. But when a recruiter reads the resume, he can identify certain competences that might not have been explicitly mentioned by the expert but might be deduced from his experience or other skills he has mentioned. The high recall rate that we obtained, guarantees that whenever a resume will have elements describing a certain competence, the model will almost always be able to predict that competence. Even though the prediction has been marked as incorrect, the following explanation can still be provided to the user: *this expert has python developer competence because the terms: developer, regression, python, scikit-learn, tool, test, script, flask, etc ...are present in his resume*. These terms are in fact closely related to "python development".

5 Conclusion and future work

We have shown the effectiveness of using a multi-label architecture based on CNN to predict high-level competences from resumes: the overall model performs very well reaching 98,79% of recall and 91,34% of precision, achieving more than 99% of accuracy for certain competences. As we have demonstrated in our experiments, in practice, the precision of the overall multi-label architecture might be greater than the evaluated value due to incomplete resume labeling. We also showed that convolutional filters are helpful to conveniently extract terms (low-level features) which explain target skills (classes). Finally, the visualization of words selected by the filters allows human to understand why the model produced its decision.

As for our future work, in addition to the filter's visualization, we will analyze and compute the importance or the contribution of each selected term in the classifier decision. The evaluation of contributions will enable us to sort out selected features with respect to their contribution

to the value predicted by the classifier, thus identifying salient features among others.

Data availability statement Resumes repository: https://github.com/florex/resume_corpus.

Code availability The preprocessing codes: https://github.com/florex/preprocessing. The CNN classifiers codes: https://github.com/florex/cnn_classifiers.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access 6:52138–52160
2. Bansal S, Srivastava A, Arora A (2017) Topic modeling driven content based jobs recommendation engine for recruitment industry. Proc Comput Sci 122:865–872
3. Bian J, Gao B, Liu TY (2014) Knowledge-powered deep learning for word embedding. In: Joint European conference on machine learning and knowledge discovery in databases, pp 132–148. Springer
4. Çelik D, Elçi A (2012) An ontology-based information extraction approach for résumés. In: Joint international conference on pervasive computing and the networked world, pp 165–179. Springer
5. Çelik D, Karakas A, Bal G, Gültunca C, Elçi A, Buluz B, Alevli MC (2013) Towards an information extraction system based on ontology to match resumes and jobs. In: 2013 IEEE 37th annual computer software and applications conference workshops, pp 333–338. IEEE
6. Cohen PR, Kjeldsen R (1987) Information retrieval by constrained spreading activation in semantic networks. Inform Process Manag 23(4):255–268
7. Development AFT (2015) Bridging the skills gap: workforce development is everyone's business n2
8. de Jesús Rubio J (2017) Usnfis: uniform stable neuro fuzzy inference system. Neurocomputing 262:57–66
9. Faliagka E, Liadis L, Karydis I, Rigou M, Sioutas S, Tsakalidis A, Tzimas G (2013) On-line consistent ranking on e-recruitment; seeking the truth behind a well-formed cv. Artif Intell Rev. https://doi.org/10.1007/s10462-013-9414-y
10. Ganzeboom HB, Treiman DJ (1996) Internationally comparable measures of occupational status for the 1988 international standard classification of occupations. Soc Sci Res 25(3):201–239
11. Gibaja E, Ventura S (2014) Multi-label learning: a review of the state of the art and ongoing research. Wiley Int Rev Data Min Knowl Disc 4(6):411–444. https://doi.org/10.1002/widm.1139
12. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A survey of methods for explaining black box models. ACM Comput Surv 51(5):93:1–93:42. https://doi.org/10.1145/3236009
13. Guo S, Alamudun F, Hammond T (2016) Resumatcher: a personalized resume-job matching system. Expert Syst Appl 60:169–182. https://doi.org/10.1016/j.eswa.2016.04.013
14. Jacovi A, Sar Shalom O, Goldberg Y (2018) Understanding convolutional neural networks for text classification. In: Proceedings of the 2018 EMNLP workshop blackbox NLP: analyzing and interpreting neural networks for NLP, pp 56–65. Association for Computational Linguistics. http://aclweb.org/anthology/W18-5408
15. Javed F, Hoang P, Mahoney T, McNair M (2017) Large-scale occupational skills normalization for online recruitment. In: Twenty-ninth IAAI conference
16. Kenthapadi K, Le B, Venkataraman G (2017) Personalized job recommendation system at linkedin: practical challenges and lessons learned. In: Proceedings of the eleventh ACM conference on recommender systems, pp 346–347. ACM
17. Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1746–1751. Association for computational linguistics, Doha, Qatar. https://doi.org/10.3115/v1/D14-1181
18. Kivimäki I, Panchenko A, Dessy A, Verdegem D, Francq P, Bersini H, Saerens M (2013) A graph-based approach to skill extraction from text. In: Proceedings of TextGraphs-8 graph-based methods for natural language processing, pp 79–87
19. Meda-Campaña JA (2018) On the estimation and control of nonlinear systems with parametric uncertainties and noisy outputs. IEEE Access 6:31968–31973
20. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in neural information processing systems, vol 26, pp 3111–3119. Curran Associates, Inc. http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf
21. Miller T (2019) Explanation in artificial intelligence: insights from the social sciences. Artif Intell 267:1–38. https://doi.org/10.1016/j.artint.2018.07.007
22. Nooralahzadeh F, Øvrelid L, Lønning JT (2018) Evaluation of domain-specific word embeddings using knowledge resources. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)
23. Ribeiro MT, Singh S, Guestrin C (2016) "Why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD '16, pp 1135–1144. ACM, New York, NY, USA. https://doi.org/10.1145/2939672.2939778
24. Rubio DJJ (2009) Sofmls: online self-organizing fuzzy modified least-squares network. IEEE Trans Fuzzy Syst 17(6):1296–1309. https://doi.org/10.1109/TFUZZ.2009.2029569
25. Rubio J (2017) Usnfis: uniform stable neuro fuzzy inference system. Neurocomputing. https://doi.org/10.1016/j.neucom.2016.08.150
26. Sayfullina L, Malmi E, Kannala J (2018) Learning representations for soft skill matching. In: International conference on analysis of images, social networks and texts, pp 141–152. Springer
27. Shalaby W, AlAila B, Korayem M, Pournajaf L, AlJadda K, Quinn S, Zadrozny W (2017) Help me find a job: a graph-based approach for job recommendation at scale. In: 2017 IEEE international conference on big data (big data), pp 1544–1553. IEEE
28. Shi H, Wang X, Sun Y, Hu J (2018) Constructing high quality sense-specific corpus and word embedding via unsupervised elimination of pseudo multi-sense. In: Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)

29. Zhang L, Wang S, Liu B (2018) Deep learning for sentiment analysis: a survey. Wiley Interdiscip Rev Data Min Knowl Discov 8(4):e1253
30. Zhao M, Javed F, Jacob F, McNair M (2015) SKILL: a system for skill identification and normalization. In: Proceedings of the twenty-ninth AAAI conference on artificial intelligence, January 25–30, 2015, Austin, Texas, USA, pp 4012–4018

E.2 Approche hiérarchique d'extraction des compétences dans des CVs en format PDF

Approche hiérarchique d'extraction des compétences dans des CVs en format PDF

Florentin Flambeau Jiechieu Kameni, Norbert Tsopze

Approche hiérarchique d'extraction des compétences dans les CV en format PDF

Florentin Flambeau Jiechieu Kameni[1,2]
Norbert Tsopze[1,2]

[1]Département d'Informatique - Université de Yaoundé I, Yaoundé, Cameroun
[2]Sorbonne Université, IRD, UMMISCO, F-93143, Bondy, France
jkflorex@gmail.com
tsopze.norbert@gmail.com

RÉSUMÉ. L'objectif de ce travail est d'utiliser l'approche héirarchique d'extraction des informations dans le CV pour en extraire les compétences. L'approche d'extraction des compétences proposée s'effectue en deux grandes phases : une phase de segmentation du CV en sections, classées suivant leurs contenus, et à partir desquelles les termes représentant les compétences (compétences de base) sont extraits; et une phase de prédiction qui consiste à partir des caractéristiques extraites précédemment, à prédire un ensemble de compétences qu'un expert aurait déduites, et qui ne seraient pas nécessairement mentionnées dans le CV (compétences implicites). Les principales contributions de ce travail sont : l'utilisation de l'approche hiérarchique de segmentation du CV en sections pour extraire les compétences dans le CV; l'amélioration de l'approche de segmentation des CV; enfin, l'utilisation de l'approche binary relevance de classification *multi-label* pour prédire les compétences implicites du CV. Les expérimentations effectuées sur un jeu de CV collectés sur Internet ont montré une amélioration de la précision de l'identification des blocs de plus de 10%, comparée à un modèle de l'état de l'art. Aussi, le modèle de prédiction *multi-label* des compétences, permet de retrouver la liste des compétences avec une précision et un rappel respectivement de l'ordre de 90,5% et 92,3%.

ABSTRACT. The aim of this work is to use the hierarchical approach of information extraction from resume to extract competences. The extraction approach envoles two phases : a segmentation phase in which the resume is segmented into sections : these sections are classified according to their contents and relevant features are extracted; and a prediction phase that consists in predicting a set of competences which are not necessarily mentioned in the resume based on the features extracted previously. The main contributions of this work are : the use of the hierarchical approach of information extraction from resume to find relevant skills; the improvement of the segmentation algorithm; and finally, the use of a multi-label learning model based on the Support Vector Machine (SVM) to predict implicit skills. Experiments carried out on a set of resumes collected from the Internet have shown more than 10% of improvement in the identification of blocs compared to a model from the start of the art; and, the multi-label prediction model permits to find skills from resume, with a precision and a recall respectively of 90.5 % and 92.3 %.

MOTS-CLÉS : Skill Gap, CV, Extraction des Compétences, Classification Multi-label

KEYWORDS : Skill Gap, Resume, Skills Extraction, Multi-label classification

1. Introduction

L'écart entre les compétences détenues par la ressource humaine d'une entreprise et celles qu'elle a besoin pour son développement est connu sous le nom de *skill-gap* [2, 21, 12]. Pour combler cet écart, les entreprises organisent généralement des recrutements en sélectionnant les candidats parmi ceux qui ont déposé un dossier physique, ou électronique à travers le site internet de l'entreprise, la messagerie électronique, ou encore un site d'offres d'emploi. Ces dossiers sont composés entre autres des demandes d'emploi, des CV ainsi que des lettres de motivation. Une des tâches de la direction des ressources humaines est alors d'indentifier les compétences détenues par les candidats à la lecture de leur CV. L'extraction des compétences dans les CV est un problème bien connu dans la littérature [12, 4, 14]. Il s'agit d'une activité qui trouve son application dans les processus de recrutement automatique et dont la finalité est d'effectuer un *matching* automatique entre les compétences extraites du CV et celles requises par l'organisation pour effectuer certaines tâches [16]. L'extraction automatique des compétences à partir des CV permet aussi de réduire l'erreur liée à la subjectivité humaine dans le processus de recrutement.

Plusieurs travaux dans la littérature ont abordé le problème de *skill-gap* allant de la modélisation ontologique [14] au matching entre les profils des candidats et les offres d'emplois [16] en passant par les modèles d'évolution et d'extraction des compétences [4, 8]. La compétence est une notion abstraite et est généralement représentée dans les textes par des phrases, des mots, ou groupes de mots. (Ex : *Java, computer programming, big data, troubleshoot network infrastructure*, ...). L'extraction automatique des compétences est une étape très importante dans le matching automatique entre CV et offres d'emploi. L'approche hiérarchique d'extraction des informations dans le CV[4] considère un CV comme un ensemble de blocs ou sections (*Education, Publications, Expérience professionnelle*,...) facilitant ainsi l'opération de spécialisation au cas où l'on s'intéresse au contenu des sections.

Cet article est une extension de [8]. En plus de la segmentation du CV en sections, ainsi que de l'extraction des compétences de base particulières à chaque section (comme dans [8]), le présent article propose un modèle pour prédire des compétences implicites (compétences qui peuvent être déduites de la lecture du CV sans pour autant être mentionnées dans le CV) à partir des caractéristiques de base extraites dans les différentes sections. L'objectif global de ce travail est donc d'adapter le modèle proposé par Zhen Chen et al. [4] qui se base sur l'approche hiérarchique consistant à segmenter le CV en sections pour extraire les caractéristiques appropriées dans chaque section. En fait, l'algorithme de segmentation hiérarchique utilisé dans l'approche de Zhen Chen et al. [4] se base sur la densité des mots et considère les titres de section comme délimitateurs des sections. Les auteurs estiment que cet algorithme peut être amélioré, car les performances n'étant pas très satisfaisantes ; ceci peut s'expliquer entre autres par le fait que les caractéristiques utilisées par les auteurs pour identifier les titres de section soient définies de manière empirique ; et les mêmes caractéristiques sont utilisées pour tous les CV. Or en pratique ces caractéristiques peuvent varier d'un CV à l'autre. Pour améliorer la détection des titres de section et par ricochet la segmentation du CV en sections, nous avons proposé une approche basée sur l'identification des titres connues *(titres-exemples)* à l'aide d'un dictionnaire, et l'utilisation de leurs caractéristiques morphologiques dans le CV pour retrouver les autres titres de section. Après l'extraction des compétences de base de chaque section, un modèle de classification *multi-label* [6] est proposé pour prédire les compé-

tences des candidats : ce modèle a l'avantage de pouvoir prédire des compétences qui ne sont pas explicitement mentionnées dans le CV. Le format de CV traité est le format *PDF* car est encore très utilisé dans les processus de recrutement en ligne. De plus, les autres formats comme le *HMTL* ne posent pas de réel problème de segmentation car, en général bien structurés.

Le reste de cet article est organisé de la manière suivante : la prochaine section présente un ensemble de travaux traitant différents aspects du problème de *skill-gap* y compris l'extraction des compétences. La section 3 sera consacrée à notre proposition. Le modèle général et les détails sur les différentes étapes y seront présentés. Les expérimentations et les résultats obtenus feront l'objet de la quatrième section. Nous terminerons par la conclusion.

2. Etat de l'art

Le *skill-gap* est un problème traité par plusieurs communautés de chercheurs. Plusieurs aspects du problème et plusieurs approches de résolution sont présentés dans la littérature parmi lesquels : la correspondance ou *matching* entre les CV et les offres d'emploi ; la mise en place des systèmes de recommandation d'offres d'emploi aux candidats ; la construction des modèles de compétence en utilisant les ontologies ; la recherche des compétences dans les CV, etc.

Benson et al. présentent dans [3] une discussion sur un ensemble de compétences permettant de profiter des opportunités présentées par les médias sociaux. Il s'agit pour eux de comprendre comment les professionnels exploitent les réseaux sociaux et de proposer les extensions de la formation dans l'exploitation des opportunités offertes par ces réseaux. Ils recommandent au gouvernement britanique de se pencher sur cette question afin de créer des compétences se basant sur la manière avec laquelle les professionnels exploitent les réseaux sociaux. Partant du constat de la disparition des certaines compétences aux Etats-Unis et de la difficulté à avoir les travailleurs qualifiés pour certaines tâches, Bednarek propose dans [21] un modèle d'évolution des compténces qui tient compte du caractère (du travail) variant avec le temps et de la liaison avec les caractéristiques de la population.

Miranda et al. dans [14] proposent une ontologie pour représenter aussi bien les compétences que les offres d'emplois dans le but d'améliorer la stratégie d'employabilité. L'ontologie est construite à partir de la littérature spécialisée sur les compétences et les offres d'emploi ; et aussi sur le recrutement dans un projet de R&D. Cette ontologie est donc utile dans l'interopérabilité entre les outils de gestion des ressources humaines et la détection des influences entre les compétences. A travers cette ontologie, il est alors possible de faire des inférences et déduire des compétences nouvelles. D'autres aspects importants de l'ontologie concernent la planification et la gestion des changements dans l'entreprise, l'évaluation des performances et la programmation des formations. D'autres recherches comme celles présentées par Shaha dans [1] ont proposé de traiter le *skill-gap* comme un problème de recommandation bidirectionnelle : recommander un emploi à un candidat ou recommander un candidat à un recruteur. Dans la même lancée, Guo et al. dans [16] proposent *RésuMatcher*, un système capable d'extraire les qualifications et l'expérience professionnelle des candidats, d'extraire également les caractéristiques sur les offres d'emploi et d'utiliser ces informations (caractéristiques du candidat et caractéristiques de l'offre d'emploi) pour calculer un score permettant de déterminer si l'offre correspond au candidat. A la différence de cet outil (RésumerMatcher) qui fait le mat-

ching entre les CV et les offres d'emploi, le travail présenté dans cet article s'intéresse à l'extraction des compétences contenues dans les CV.

Au sujet de l'extraction des compétences dans les textes, des approches basées sur la reconnaissance des entités nommées [12, 17] et les approches basées sur l'utilisation des graphes [9] ont aussi été proposées. Les approches du premier groupe construisent par apprentissage un modèle d'étiquettage des entités à partir d'un corpus, et utilisent ce modèle pour détecter les compétences. Leurs performances sont fortement liées à la qualité de l'algorithme d'apprentissage ainsi que celle du jeu d'apprentissage.

La plupart des modèles d'extraction des compétences dans les CV traitent ces derniers comme des textes bruts ne possédant aucune structure particulière [12, 10]. Ces modèles ne font pas d'hypothèse concernant la *semi-structuration* (organisation en sections) du CV. Or Jun Yu et al. [20] ont montré que la prise en compte de la propriété *semi-structurée* du CV permet de faire une extraction d'information de meilleure qualité si une technique d'extraction appropriée était appliquée à chacune de ses sections. Maheshwari et al.[10] ont proposé un modèle hiérarchique de traitement des CV dans lequel ils supposent que chaque section du CV (pour un candidat donné) détient une information spéciale permettant de distinguer ce CV. Le modèle recherche d'abord cette information spéciale et l'utilise pour classer les candidats dans le processus de sélection.

L'un des modèles hiérarchiques d'extraction d'information dans les CV parmi les plus récents a été proposé en 2016 par Chen et al. [4] ; il s'agit d'une amélioration du modèle hybride en cascade proposé par Jun Yu [20] pour l'extraction des informations dans les CV au format PDF. La principale limite de ce modèle est qu'il n'extrait pas les compétences ; mais plutôt les informations d'état civil (*nom, prénom, âge,...*) ainsi que les informations sur l'éducation (*diplômes, années d'obtention, etc.*). De plus, pendant la phase de segmentation, les caractéristiques des titres de section sont définies de manière empirique pour tous les CV [4]. Pourtant, en pratique ces caractéristiques varient d'un CV à un autre.

Le modèle développé dans ce travail apporte une amélioration à celui présenté dans [4] en s'intéressant au contenu des sections et en utilisant une autre approche de segmentation. Ce travail aborde aussi la recherche des compétences dites de haut niveau des candidats. Les compétences de haut niveau sont celles que le candidat utilise pour résumer son CV. Les exemples peuvent être *Software engineer*, *Data scientist*,... Certaines de ces compétences sont explicitement mentionnées dans le CV (compétences explicites) et d'autres pas (compétences implicites).

3. Méthodologie

Le modèle d'identification des compétences dans les CV développé dans cet article est un modèle basé sur l'approche hiérarchique d'extraction des informations dans les CV. L'approche procède en deux grandes étapes :

1) la division du CV en sections et extraction dans chaque section, des termes traduisant la compétence que l'on appellera aussi compétences de base ;

2) l'utilisation des caractéristiques extraites à l'étape précédente pour prédire des compétences de haut niveau que possède le candidat.

3.1. Extraction des caractéristiques par l'approche hiérarchique

Les approches hiérarchiques d'extraction des informations dans les CV procèdent généralement en trois principales étapes telles que schématisées dans la figure 1 :

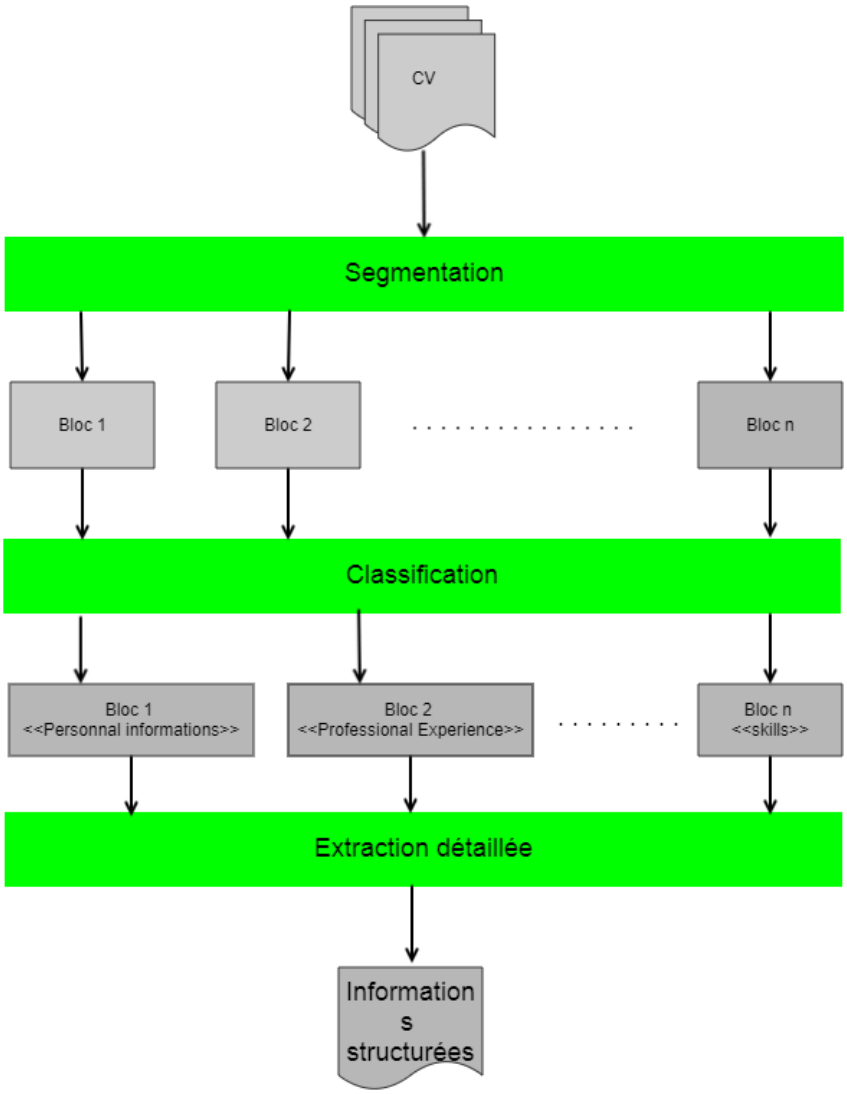


Figure 1. *Schéma d'extraction hiérarchique des informations dans un CV*

1) Segmentation du CV en blocs de texte représentant des sections potentielles ;

2) Classification des blocs ;

3) Extraction des informations détaillées dans chaque bloc.

La première étape consiste à décomposer le CV en des blocs de texte, chacun représentant à priori une section ou la partie d'une section dans un CV. La deuxième étape quant à elle, a pour but d'affecter à chaque bloc identifié à la première étape une étiquette représentant une classe sémantique d'information généralement contenue dans le CV. La classe sémantique peut être *EDUCATION*, *SKILLS*, etc,... Enfin la dernière étape va consister à extraire les informations détaillées dans chaque section en fonction de la nature des informations qui y sont contenues.

Une section dans le CV est un groupe d'informations appartenant à une même classe sémantique. Elle est généralement composée d'un titre et d'un contenu qui détaille les informations du candidat relatives à cette section. Par exemple la section "*SKILLS*" contient généralement les compétences de l'expert ; alors que la section "*EDUCATION*" contient les informations relatives à son éducation.

3.1.1. Segmentation des blocs

Comme des paragraphes dans un texte qui se distinguent visuellement par le retour à la ligne à la fin du paragraphe, et l'indentation au début, les blocs dans les CV sont des ensembles consécutifs de textes relativement proches les uns des autres qui au niveau visuel se distinguent par la largeur de l'espacement vertical ou horizontal (CV organisé en colonnes) entre ces blocs ou par un titre de section. Autrement dit, un bloc dans un CV est graphiquement une région dense en terme de distribution des mots dans l'espace [4]. Et, l'espacement entre deux blocs est une zone peu dense. L'espacement entre le contenu d'une section et le titre de la section suivante est plus grand que celui entre les lignes constituant la section.

Définition 1 *Deux mots m_1 et m_2 sont proches si la distance entre les deux mots est inférieure à un seuil s.*

La figure 2 illustre de manière schématique comment la proximité entre deux mots est évaluée. Cette distance est évaluée comme étant la largeur de l'espacement horizontal (dh) (si ces deux mots sont sur la même ligne) ou verticale (dv) (si les deux mots appartiennent à des lignes consécutives) qui existe entre ces deux mots. Si les deux mots m_1 et m_2 sont sur la même ligne, alors, la distance entre eux est définie comme étant le nombre de pixels entre la position x du dernier caractère du mot m_1 et la position x du premier caractère du mot m_2. Dans l'autre cas où les deux mots sont sur des lignes consécutives, la distance est évaluée comme étant l'écart vertical en pixels entre les deux mots (dh sur la figure 2). Dans la pratique, le seuil de proximité horizontal (s_h) peut être différent du seuil de proximité vertical (s_v).

Définition 2 *Le s-voisinage d'un mot m noté $V(m)$ est défini comme étant l'ensemble des mots qui sont proches de m au regard des seuils de proximité s_h et s_v.*

Définition 3 *La fermeture transitive d'un mot m est l'ensemble des mots qu'on peut atteindre verticalement ou horizontalement à partir de m par pas de longueur inférieure à s_h si le déplacement est fait horizontalement ou à s_v s'il est fait verticalement.*

Plus formellement, la fermeture transitive $F(m)$ d'un mot m dans le CV peut être éfini comme suit :

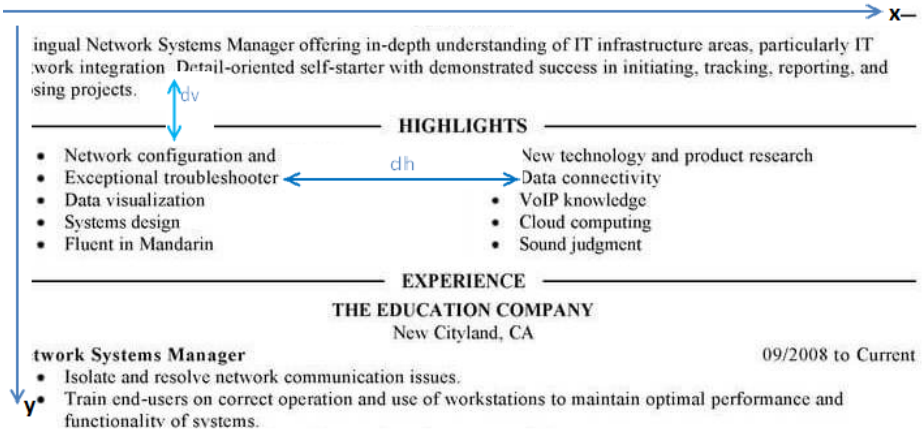


Figure 2. *Evaluation de la distance entre mots.*

– $m \in F(m)$

– $V(m) \subset F(m)$

– $m_{k,(k\geq 2)}$ appartient à $F(m)$ si et seulement si il existe une suite de mots $m_1, m_2, ..., m_{k-1}$, telle que $\forall_{i=1,2,...k-1}\ d(m_i, m_{i+1}) \leq s$.

Chaque mot dans un bloc a donc pour fermeture transitive le bloc lui-même. Le principe de l'algorithme de segmentation des blocs consiste donc à identifier toutes les fermetures transitives des mots du bloc. La rencontre d'un titre de section est considérée comme étant la séparation entre deux blocs.

L'algorithme de segmentation est semblable à celui décrit dans l'article [4] et procède comme suit :

1) Définir un critère d'indentification des titres de section ;

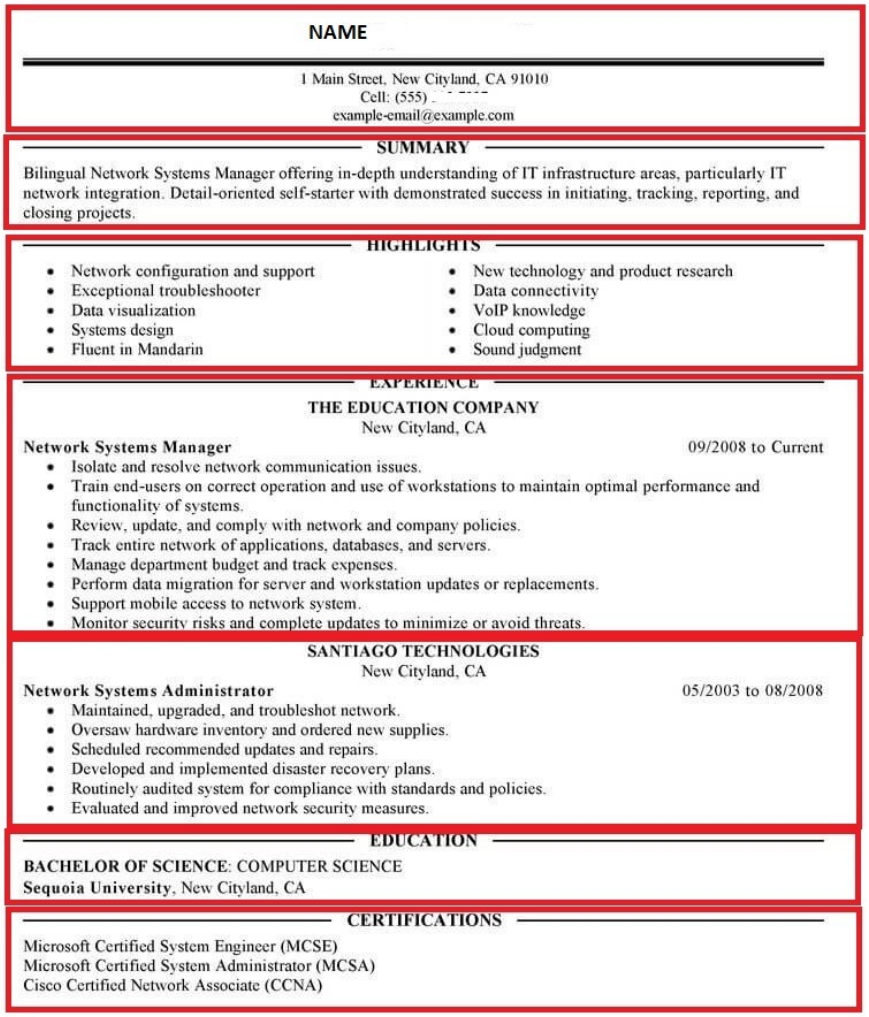
2) Initialement, on considère que chaque mot est un bloc. Les fermetures de ces mots sont construites en rajoutant progressivement les mots voisins, les voisins des voisins et ainsi de suite jusqu'à ce qu'on ne puisse plus rajouter un mot qui respecte le critère de proximité ou alors lorsqu'on recontre un titre de section.

3) S'il y a chevauchement entre des blocs alors ces blocs sont fusionnés.

Un exemple de résultat la segmentation appliquée à un CV anonyme est présenté en figure 3. Les différents blocs identifiés par l'algorithme sont représentés par des rectangles.

3.1.2. Identification des titres des différents blocs.

La rencontre d'un titre de section est un indicateur de la fin d'un bloc et du début d'un autre bloc. Identifier les titres de section dans un CV permet de délimiter les sections. A la différence de la méthode utilisée dans l'article [4] qui définit les caractéristiques des titres de section de manière empirique (taille de la police de caractère, épaisseur de la police de caractère, couleur) avec les mêmes valeurs peu importe le CV, l'approche proposée dans cet article pour identifier les titres se base sur l'hypothèse selon laquelle, les titres sont généralement rédigés avec le même style dans un CV et leurs caractéristiques se distinguent visuellement du reste du texte. Cette considération peut trouver une illustration dans le CV présenté en exemple à la figure 3 où les titres de section (*SUMMARY, HIGHLIGHTS, EXPERIENCE, THE EDUCATION COMPANY, SANTIAGO TECHNOLOGY, EDUCATION, CERTIFICATIONS*) ont des caractéristiques morphologiques (*majuscule, taille de*



NAME

1 Main Street, New Cityland, CA 91010
Cell: (555)
example-email@example.com

SUMMARY

Bilingual Network Systems Manager offering in-depth understanding of IT infrastructure areas, particularly IT network integration. Detail-oriented self-starter with demonstrated success in initiating, tracking, reporting, and closing projects.

HIGHLIGHTS

- Network configuration and support
- Exceptional troubleshooter
- Data visualization
- Systems design
- Fluent in Mandarin
- New technology and product research
- Data connectivity
- VoIP knowledge
- Cloud computing
- Sound judgment

EXPERIENCE

THE EDUCATION COMPANY
New Cityland, CA

Network Systems Manager 09/2008 to Current

- Isolate and resolve network communication issues.
- Train end-users on correct operation and use of workstations to maintain optimal performance and functionality of systems.
- Review, update, and comply with network and company policies.
- Track entire network of applications, databases, and servers.
- Manage department budget and track expenses.
- Perform data migration for server and workstation updates or replacements.
- Support mobile access to network system.
- Monitor security risks and complete updates to minimize or avoid threats.

SANTIAGO TECHNOLOGIES
New Cityland, CA

Network Systems Administrator 05/2003 to 08/2008

- Maintained, upgraded, and troubleshot network.
- Oversaw hardware inventory and ordered new supplies.
- Scheduled recommended updates and repairs.
- Developed and implemented disaster recovery plans.
- Routinely audited system for compliance with standards and policies.
- Evaluated and improved network security measures.

EDUCATION

BACHELOR OF SCIENCE: COMPUTER SCIENCE
Sequoia University, New Cityland, CA

CERTIFICATIONS

Microsoft Certified System Engineer (MCSE)
Microsoft Certified System Administrator (MCSA)
Cisco Certified Network Associate (CCNA)

Figure 3. *Example de CV segmenté en blocs*

la police de caractère, epaisseur de la police de caractère) qui se distinguent clairement du reste du texte. Mais les caractéristiques qui font différer les titres du reste du texte sont propres à chaque CV et ne sauraient être généralisées à tous les CV comme dans [4]. Car les choix de ces caractéristiques sont propres à chaque candidat. Pour certains, l'épaisseur de la police de caractère fait la distinction, alors que pour d'autres la couleur pourrait aussi faire la différence, etc. C'est pour cette raison que l'approche proposée se base sur l'identification des caractéristiques distinctives propres au CV en procédant par la reconnaissance des titres bien connus.

Il existe des formulations de titres qui apparaissent très fréquemment dans les CV. Ces formulations bien connues sont par exemple : Langues (*Languages*), Compétences (*Skills*), Formation académique (*Education*), Expériences professionnelles (*Work Experience*), ... Un dictionnaire de ces différentes formulations connues des titres de CV rédigé en langue anglaise a été ainsi constitué et contient 284 manières (pour nos expérimentations) de formuler les différents titres de sections du CV. Le dictionnaire va servir à stocker des titres fréquemment utilisés. Les caractéristiques des titres reconnus vont permettre de retrouver les autres titres du CV qui ne se retrouvent pas dans le dictionnaire. Les étapes d'identification des titres de sections du CV sont donc les suivantes :

1) Identifier dans le CV des lignes de textes qui correspondent aux titres renseignés dans le dictionnaire ;

2) Extraire les caractéristiques morphologiques communes de ces lignes de texte ;

3) Enfin, comparer les caractéristiques des titres-exemples identifiés à la première étape avec celles des autres lignes de texte. En cas de similarité des caractéristiques d'une ligne avec un titre-exemple (titre inclus dans le dictionnaire), la ligne de texte est considérée comme étant un titre.
Sur une echelle de 0 à 1, le seuil de similarité doit être suffisamment grand (>0.95 utilisé pendant les expérimentations) pour éviter de détecter un titre qui n'en est pas un.

Les caractéristiques utilisées pour évaluer la similarité entre les titres sont :

– La taille de la police de caractère ;

– Un indicateur positionné à 1 si la première lettre du titre est majuscule et le reste en minuscule et 0 sinon ;

– Un indicateur positionné à 1 si tout le titre est en majuscule et 0 sinon ;

– La couleur du titre (niveau de rouge de gris et de bleu) ;

– L'épaisseur de la police de caractère ;

La mesure de similarité utilisée est la similarité cosinus [7]. A chaque caractéristique est associé un poids qui traduit son degré de discrimination entre les titres et les portions de textes qui ne sont pas des titres. Il est possible dans de rares cas qu'une ligne ne représentant pas un titre dans le CV, mais apparaîssant dans le dictionnaire soit identifié comme titre. Etant donné que les titres de section ont des caractéristiques similaires qui se distinguent des autres lignes de texte du CV, un filtrage de ces lignes est fait en comparant entre eux, les titres indentifiés et en ne retenant que ceux ayant des caractéristiques similaires telles que la taille de la police de caractère, l'épaisseur de la police, la couleur, ou la casse.

3.1.3. Classification des blocs

Une fois les blocs délimités, il faut les classer en utilisant un modèle de classification construit à partir d'un algorithme d'apprentissage. Le modèle de classification utilisé est le même que celui décrit dans l'article [4]. Les différentes classes que nous avons

considérées pour les expérimentations sont les suivantes : *Work Experience*, *Education*, *Skills*, *Languages*, *Certification*, *Personnal Information*, *Hobbies*. Le modèle d'apprentissage utilisé est le SVM [19]. L'apprentissage est fait avec les exemples constitués des caractéristiques extraites des blocs. Ces caractéristiques extraites dans des blocs pour la classification sont celles listées dans [4] à savoir :

– La présence des séparateurs (",", " ;" ou " :") ;

– La présence des dates ;

– La présence de nom de personnes ;

– La présence d'adresse mail ;

– Les mots-clés apparaissant dans les titres de chaque bloc ;

A ces caractéristiques sont ajoutées la présence des noms des organisations et verbes d'action (*design, implement, troubleshoot*, etc.). Ces verbes sont très souvent utilisés dans les sections relatives à l'*expérience professionnelle* dans un CV. Une bonne identification des titres de section a également un grand impact dans la classification des blocs car, le titre est un élément très informatif sur la nature du contenu du bloc. Les termes extraits aussi bien dans les titres que dans les blocs sont lemmatizés en utilisant le *lemmatizer* de *wordnet* (WordnetLemmatizer) [5], à l'effet d'associer les termes qui partagent en commun un même lemme et qui traduisent à la base une information similaire. Ces informations sont utilisées pour représenter chaque bloc sous forme d'un vecteur de numériques en y associant sa classe qui n'est rien d'autre que le titre de la section (*EDUCATION, SKILLS*,...). Le jeu de données ainsi consitué est utilisé pour entrainer un SVM à l'effet de prédire pour un nouveau bloc son étiquette étant données les caractéristiques associées au bloc.

Après avoir étiqueté les différents blocs, la prochaine étape consiste à extraire les informations détaillées dans chacun des blocs étiquettés en fonction de la nature et de la structure des informations qui y sont contenues.

3.2. Extraction des informations détaillées dans chaque bloc

L'article [4] ne s'intéresse qu'à l'extraction des informations détaillées dans les blocs informations personnelles et éducation. Ce travail l'étend aux autres blocs qui renferment les caractéristiques permettant d'apprécier la compétence du candidat à savoir :

– Les blocs étiquetés « Compétences » et « Certifications »

– Les blocs étiquetés « Expérience professionnelle »

3.2.1. Extraction des compétences dans les sections *compétences* et Certifications

Généralement, dans les CV, les éléments de la section « Compétence » *SKILLS, COMPETENCIES, TECHNOLOGIES, etc.* sont constitués de liste des compétences délimitées par un séparateur qui peut être (la virgule, le point-virgule, ou le caractère de retour à la ligne). Ainsi, pour cette section, les compétences sont extraites en scindant les éléments du texte avec comme séparateur la virgule, le point-virgule, l'espace ou le caractère de retour à la ligne. Pour chaque terme, une taxonomie du domaine est utilisée pour valider s'il s'agit d'une compétence du domaine ou pas. La taxonomie utilisée a été construite sur la base des compétences extraites de dice.com [1] qui contient plus de 5000 termes du domaine de l'informatique.

. consulté le 21 septembre 2018

3.2.2. Extraction des compétences dans les blocs *Expérience professionnelle*

Plusieurs titres de section dans un CV correspondent à l'expérience professionnelle. Il s'agit des titres tels que : (*Work experience, Project experience, Publications, etc.*) Dans ces sections, les compétences sont généralement exprimées sous forme de phrases commençant par un verbe d'action (par exemple *identify cyber threat signature*) ou sous sa forme nominale du verbe suivie par la préposition « of » (par exemple *Indentification of cyber threat signatures*). Un dictionnaire de tels verbes d'action et de leur forme nominale a été constitué. Une règle est appliquée pour identifier ce style de proposition dans les textes. La règle qui a été définie pour extraire ces clauses est la suivante :

<VRB|NN of><complément>[<IN><complément>]. (R)

avec la nomemclature proposée par [11] où *VRB* est mis pour *verbe* (Verb), *IN* pour préposition (in), *NN* pour *nom* (Noun), etc. Avant d'appliquer cette règle, la simplification de phrase est utilisée pour décomposer toutes les phrases composées (celles ayant plusieurs propositions reliées par les conjonctions) en plusieurs sous phrases ayant un sens unique. Il a été montré dans [15] que la simplification des phrases est un prétraitement qui permet aux outils de Traitement Automatique de la Langue Naturelle (TALN) d'avoir de meilleurs résultats.

Exemple 1 *Le résultat de la simplification syntaxique de l'expression "detection and identification of cyber-attack signatures" est constitué des propositions simples suivantes :*

– *detection of cyber-attack signatures.*

– *identification of cyber-attack signatures.*

Après cette étape de simplification, l'algorithme d'extraction peut facilement extraire les savoir-faire en appliquant la règle précédemment définie.

L'outil *stanford dependency parser* [11] est utilisé pour étiquetter le texte c'est-à-dire, attribuer une étiquette syntaxique à chaque mot. Cet outil identifie aussi les dépendances entre les mots en déterminant les relations entre les mots dans l'expression (par exemple *subj* : sujet du verbe, *cobj* : complément d'objet, *comp* : mot composé à, etc.). A partir de ces étiquettes et des relations entre les mots, un parcours du graphe de dépendance est effectué à l'effet d'extraire les expressions caractérisant des compétences : celles qui respectent la règle (R).

L'algorithme d'extraction utilisant la règle (R) appliquée à l'exemple 1 va générer les savoir-faire ci-après :

– *detection of cyber-attack signatures*.

– *identification of cyber-attack signatures*.

Les compétences exprimées sous forme de savoirs (technologies, outils, sujets maîtrisés) sont également extraites dans cette section en recherchant les *n-grams* (de 1 à 5) et en se servant d'une taxonomie du domaine pour déterminer si chaque *gram* extrait de la section *Experience professionnelle* correspond à une compétence. Chaque *gram* est lemmatisé avant d'être comparé au lemme du nom de l'entité dans la taxonomie. Les mots vides (*stop words*) sont également éliminés.

Exemple 2 *La procédure d'extraction des compétences appliquées à l'exemple 1, permet d'obtenir les compétences suivants :*

– *cyber threat signatures*

– *cyber attack*

3.3. Prédiction des compétences de haut niveau à partir des caractéristiques extraites

Une compétence est dite de haut niveau si elle peut être déduite de la connaissance d'un certain nombre de compétences de base (*skills*). Elle peut être explicite ou implicite selon qu'elle est mentionnée ou pas dans le CV. Dans cette partie, il s'agit d'utiliser les mots-clés constituant les compétences de base extraites dans les étapes précédentes, pour déduire des compétences de haut niveau.

Exemple 3 *Il s'agira par exemple de déduire la compétence développeur web à partir des compétences de base : html, css, javascript, php. Ou encore de déduire la compétence developpeur mobile à partir des compétences de base : android, ionic, ios, java, cordova.*

Pour cette tâche, une approche suppervisée où la liste des compétences est connue à l'avance pour chaque CV est utilisée. Ainsi, Il est question de prédire la liste des compétences de haut niveau, détenues par un expert à partir de l'ensemble des caractéristiques extraites de son CV. Généralement dans les processus de recrutement, l'ensemble des compétences attendues des candidats est connu à l'avance. En suivant ce principe, le modèle dévelopé dans cette section se base sur une liste de compétences mentionnées utilisées comme classes. Ainsi étant donné un CV, il est question de savoir lesquelles parmi ces compétences il possède. Pour nos expérimentations la liste des compétences retenues comme classes est la suivante :

project manager	oracle database administrator	business analyst
technical lead	java developer	software ingeneer
consultant	analyst	software developer
network administrator	web developer	IT analyst
IT consultant	manager	

Il s'agit des compétences qui apparaissent plus fréquemment dans le jeu de CV utilisé.

Le problème de prédiction des compétences d'un expert peut être vu comme un problème de classification *multi-label* [6] où chaque CV peut être étiqueté par plusieurs compétences simulatanément. Par exemple, un expert peut avoir en même temps les compétences *systems administrator* et *network administrator*. Formellement un problème de classification *multi-label* se définit comme suit [6] :

Définition 4 *Etant donné un ensemble de données X et un ensemble de classes Y , la classification multi-label consiste à associer, à chaque élément de X, un sous ensemble de Y.*

Pour les expérimentations, l'ensemble Y est constitué des quatorze compétences citées plus haut.

La figure 4 illustre l'architecture globale de construction et de fonctionnement du modèle de prédiction des compétences de haut niveau des CV. Partant d'un jeu de données constitués à partir d'un ensemble de CV, chacun étiqueté par un ensemble de compétences, le processus de construction du modèle de prédiction se fait en plusieurs étapes comme suit :

– représentation des données ;

– transformation du jeux de données en n jeux chacun centré sur une compétence ;

– apprentissage d'un modèle par jeu pour prédire la compétence associée au jeu ;

– aggregation des résultats des prédicteurs de base.

3.3.1. Représentation des CV

Chaque CV est initialement représenté par un sac de mots, puis transformé en une représentation vectorielle en utilisant la technique *word2vec* [13]. Les mots dans le sac sont ceux qui renseignent sur les compétences. Le modèle de représentation utilisé est le modèle *skip gram* [18] avec comme taille de la fenêtre 5 et comme dimension 300. La représentation *skip gram* permet de prendre en considération le lien sémantique entre les termes dans la construction du modèle d'apprentissage. Deux termes (*programmer* et *developer*) par exemple auront des représentations vectorielles similaires car, ils sont très souvent employés dans des contextes similaires. Ainsi, la représentation vectorielle du CV est obtenue en agrégeant (moyenne sur chaque composante) les représentations vectorielles des termes-clés qui le constitue.

3.3.2. Construction du jeu de données

La méthode *Binary Relevance* est utilisée pour faire la classification *multi-label*[6] des CV. Elle consiste à transformer le problème de classification *multi-label* en autant de problèmes de classification binaire qu'il y a de classes. Le jeu de données initial est utilisé n fois (où n représente le nombre de classes/compétences), et chaque fois pour une compétence particulière. Pour chaque compétence, le vecteur des caractéristiques du CV obtenu à l'étape de représentation reste inchangé. Seule la classe change. Si le jeu i est fait pour la compétence C_i, alors la classe de chaque exemple représentera la présence (1) ou l'absence (0) de la compétence C_i dans l'exemple considéré. Ainsi donc, le modèle construit à partir du jeu de données i permettra de prédire si un exemple possède (1) ou non (0) la compétence C_i.

3.3.3. Apprentissage du modèle de classification

Une fois le jeu de données initial transformé en n jeux centrés chacun sur une compétence unique, chaque jeu de données est ensuite utilisé pour entrainer un prédicteur binaire dont le rôle est de prédire si l'exemple placé en entrée possède la compétence. Le modèle de classification utilisé pour se faire est le SVM qui est bien adapté pour les problèmes de classification binaire.

3.3.4. Aggregation des résultats

A la fin du processus, les prédictions de chaque modèle de base sont aggrégées en faisant l'union des classes prédites par chacun. Ainsi, nous obtenons la liste des compétences prédites de l'expert parmi les compétences d'intérêt.

4. Expérimentations

4.1. Données

Un "*web scraper*" a été développé pour collecter automatiquement les CV sur les sites Internet. Ces CV sont stockés au format PDF. Un jeu de 800 CV rédigés en anglais a été constitué pour les expérimentations. Le jeu données a été divisé en 4 sous-ensembles. L'expérimentation s'effectue en quatre étapes et à chaque étape, un sous-ensemble distinct (1/4 du jeu de donnée) est utilisé pour le test et les autres sous-ensembles (3/4 du jeu de données) sont utilisés pour la construction du modèle. Les résultats finaux consignés dans des tableaux sont obtenus en faisant la moyenne des résultats sur les quatres exépriences.

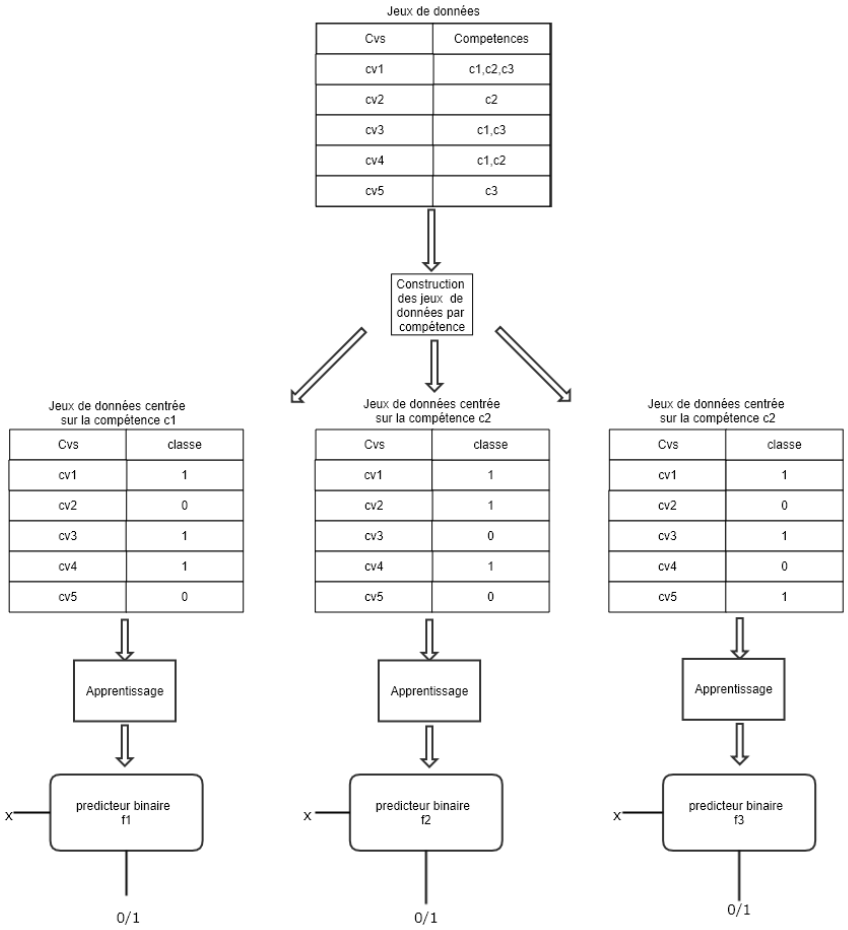


Figure 4. *Modèle de prédiction multi-label.*
Chaque aspect important du modèle a été évalué séparemment. Une mesure d'évaluation appropriée a été utilisée pour chacun de ces aspects.

4.2. Evaluation de l'algorithme de division du CV en blocs

Mesure d'évaluation

La mesure d'évaluation utilisée pour mesurer la qualité de l'algorithme de division du CV en blocs est la *précision* que nous nous proposons de la calculer avec la formule 1.

$$Precision = \frac{NBC}{NT}. \quad [1]$$

Tableau 1. *Résultat de la division du CV en bloc.*

Format du CV	Tabulaire	Linéaire
Précision	84,32%	89,86%

Dans la formule 1, NBC représente le nombre de blocs correctement délimités et NT le nombre total de blocs. Un bloc est considéré comme correctement délimité s'il s'agit d'une section de premier niveau (bloc non imbriqué dans un autre) ou d'une sous section (bloc situé au premier niveau d'imbrication). Cette considération évite de constituer des blocs de très petites tailles. Car les blocs de très petites tailles ne permettent pas dans bien des cas, d'avoir suffisamment d'informations pour déduire la classe du bloc. L'idéal aurait été que les blocs correspondent exactement aux sections du CV. Néanmoins, après la phase de classification, les blocs ayant la même étiquette sont fusionnés pour reconstituer la section initiale correspondante à l'étiquette en question. Le tableau 1 présente la précision obtenue dans la tâche d'extraction des blocs. La précision varie en fonction de la structure du CV. La détection des blocs opère assez bien avec une précision de 89,86% sur les CV rédigés dans une structure linéaire-verticale (où les sections sont empilées les unes sur les autres 3). En effet, dans les CV structurés de cette façon, les erreurs constatées dans la division des blocs sont principalement dues aux erreurs de détection des titres de section (le style d'écriture du titre de la section se distingue très peu du style d'écriture du reste du texte). Par contre, lorsque le CV présente une structure complexe (à l'exemple des CV disposés sous forme de tableau), l'algorithme de détection des blocs ne produit pas toujours de très bons résultats comme dans le cas linéaire.

4.3. Evaluation de la classification des blocs

Cette sous section présente les résultats du modèle de classification des blocs.

Mesures d'évaluation

Pour une évaluation détaillée de l'algorithme de classification des blocs, une matrice de confusion (C) (tableau 2) est calculée. Dans cette matrice, les lignes représentent les classes réelles et les colonnes représentent les classes prédites. Chaque élément C_{ij} de la matrice représente le nombre d'éléments dont la classe réelle est i et qui ont été prédits par le modèle comme appartenant à la classe j. Par la suite, quatre mesures synthétiques sont utilisées pour évaluer la qualité du modèle pour chaque classe d'une part, et la qualité générale du modèle en faisant la moyenne des valeurs obtenues pour les différentes classes d'autre part. Ces mesures sont :

– Le Rappel (R_i) pour une classe i, mesure la proportion d'exemples qui ont été correctement assignés à la classe i parmi tous les exemples dont la classe réelle est i. La formule est donnée par :

$$R_i = \frac{C_{ii}}{\sum_j C_{ij}}. \quad [2]$$

– La précision (P_i) qui mesure la proportion des bonnes prédictions sur le nombre total de prédictions, et donc la formule est donnée par l'équation 3 :

$$P_i = \frac{C_{ii}}{\sum_j C_{ji}}. \quad [3]$$

– La précision moyenne peut être obtenue en faisant simplement la moyenne des précisions des différentes classes :

$$Precision = \frac{\sum_i P_i}{N}. \quad [4]$$

– Le rappel moyen est obtenu en calculant la moyenne des rappels des différentes classes :

$$Rappel = \frac{\sum_i R_i}{N}. \quad [5]$$

Tableau 2. *Matrice de confusion de la classification des blocs*

Classes réelles \ Classes prédites	Education	Work Experience	Skills	Certifications	Personal Infos	Languages	Hobbies	Rappel
Education	180	12	0	4	4	0	0	90%
Work Experience	7	178	13	2	0	0	0	89%
Skills	2	3	178	10	10	6	0	89%
Certifications	16	0	7	171	6	0	0	85,5%
Personal infos	0	7	0	0	193	0	0	96,5%
Languages	2	0	7	0	0	191	0	95,5%
Hobbies	0	0	5	0	0	0	195	97,5%
Précision	86,95%	90,35%	84,76%	91,44%	92,34%	96,95%	100%	

La matrice de confusion représentée dans le tableau 2, nous montre que le modèle prédit certaines classes avec une précision plus importante que d'autres. Par exemple, le modèle prédit un bloc d'étiquette réelle ***Languages*** avec une précision de 96,95% alors qu'un bloc d'étiquette réelle *Education* est prédite avec une précision de 90%. La valeur élevée de la précision pour la section ***Languages*** est probablement dûe au fait que cette section se distingue avec peu d'ambiguité des autres sections par la présence des *language*, des termes clés d'appréciation du niveau de la langue, et par la taille relativement courte du bloc. Les erreurs dans les autres blocs sont dûes à la ressemblance des contenus de ces blocs. Par exemple, comme on peut le constater dans la matrice de confusion, sur un total de 178 exemples appartenant à la classe "*Work Experience*", 7 ont été prédits comme étant de la classe *Education*, et 13 ont été prédits comme étant de la classe *Skills*. En effet, les mêmes caractéristiques qu'on peut retrouver dans la section "*Work Expérience*" peuvent se retrouver dans la section "*Education*"; ce qui peut justifier les confusions effectuées par le modèle. Le modèle de classification des blocs opère globalement avec une précision moyenne évaluée à 91.83% et un rappel global évalué à 91,85% (tableau 3).

Tableau 3. *Classification des blocs : Evaluation globale*

	Précision moyenne	Rappel moyen	F-mesure
Modèle	91,82%	91,85%	91,39%

4.4. Evaluation du modèle d'extraction des compétences explicites

Mesure d'évaluation

L'idée de l'évaluation est de mesurer l'aptitude du système à extraire tous les termes caractérisants la compétence et uniquement ceux là. Pour un CV i donné, notons Y_i l'ensemble des termes clés extraits par le modèle, Z_i l'ensemble des termes clés extraits par l'expert du domaine, et n le nombre de CV soumis à l'évaluation. Les mesures utilisées pour évaluer la classification des compétences par le modèle sont les suivantes :

– L'exactitude qui mesure pour un CV i donné, le pourcentage des termes correctement identifiés par l'algorithme sur l'ensemble des termes (ceux prédits par l'algorithme et ceux identifiés par l'expert). L'exactitude moyenne est obtenue en faisant la moyenne des exactitudes calculées sur tous les CV. Elle se calcule par la formule 6 :

$$Exactitude = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i \cap Z_i}{Y_i \cup Z_i}\right|. \quad [6]$$

– Le Rappel qui mesure pour un CV i donné, la proportion des termes identifiés par l'expert qui ont été correctement identifiés par l'algorithme. Le rappel moyen est calculé en faisant la moyenne des rappels obtenus sur tous les CV. Il est donné par la formule 7 :

$$Rappel = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i \cap Z_i}{Z_i}\right|. \quad [7]$$

– La précision pour un CV i mesure la proportion des classes qui ont été correctement prédites parmi celles que le modèle a prédite. La précision moyenne (formule 8) est déterminée en faisant la moyenne des précisions calculées sur tous les exemples de CV utilisés pour le test.

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_i \cap Z_i}{Y_i}\right|. \quad [8]$$

– La *F-mesure* qui est une mesure qui agrège les deux mesures précédentes, permet de trouver un compromis entre la précision et le rappel.

$$F - mesure = \frac{2 * Rappel * Precision}{Rappel + Precision}. \quad [9]$$

Résultats

Les résultats d'expérimentations sont répertoriés dans le tableau 4. La précision de 96.2% signifie 96.2% des termes exprimant les compétences qui ont été indentifiés par le modèle sont corrects. Les erreurs sont principalement dues au fait que le modèle identifie des termes homonymes des termes du domaine mais dont le contexte d'emploi n'en fait pas une caractéristique du domaine ; l'exactitude de 81,4% traduit en moyenne, le pourcentage de correspondance entre les compétences prédites et celles attendues . D'autre part, le rappel de 84.32% signifie que le modèle réussit à retrouver 84.32% des caractéristiques qu'on souhaitait extraire du modèle et qui ont été identifiées au préalable par un expert du domaine informatique. Le gap au niveau du rappel est dû au fait que la règle (R) n'est pas une règle exhaustive et ne permet pas d'extraitre tous les compétences de type

Tableau 4. *Résultat de l'extraction des compétences explicites.*

	Exactitude	Précision	Rappel	F-mesure
Modèle	81,4%	96,2%	84,32%	89,86%

savoir-faire. Par ailleurs, les erreurs liées à la lemmatisation induisent le modèle à ne pas reconnaitre des termes qui ont une même racine lexicale.

4.5. Evaluation du modèle de prédiction des compétences implicites

Les CV ont été étiquettés en y introduisant les compétences que les candidats ont explicitement mentionnés.

Mesures d'évaluation

Les mêmes mesures d'évaluation utilisées au niveau de l'extraction des compétences explicites sont considérées dans cette partie. La semantique de ces mesures dans le contexte de la prédiction des compétences implicites est la suivante :

– L'exactitude (formule 6) qui mesure la proportion des compétences prédites correctement parmi l'ensemble des compétences (prédites et celles qu'on devait prédire).

– La précision (formule 8) qui mesure la proportion des compétences correctement prédites parmi les compétences qui ont été prédites par la modèle.

– Le rappel (formule 7) qui mesure la proportion des compétences correctement prédites parmi les compétences qui devaient être prédites.

Le résultat de la prédiction des compétences implicites est représenté dans le tableau 5. Dans ce tableau, l'exactitude traduit à priori qu'en moyenne, la probabilité que le résultat prédit par le modèle corresponde au résultat correct est de 88.6%. La précision de 92,3% signifie qu'en moyenne, 92,3% des compétences prédites sont correctes. Le rappel de 90,5% signifie que le modèle prédit 90,5% des compétences consituant les étiquettes de chaque exemple. Ces résultats montrent la capacité du modèle à proposer aux récruteurs les autres compétences non explicitement mentionnées dans le CV par les candidats.

Tableau 5. *Résultat de la prédiction des compétences implicites.*

	Exactitude	Précision	Rappel	F-mesure
Modèle	88,6%	92,3%	90,5%	91,39%

4.6. Contribution de l'approche de détection des titres sur la division des blocs

Le tableau 6 présente les résultats des approches de segmentation.

Tableau 6. *Comparaison des approches de segmentation.*

Approches	Précision
Approche basée sur les titres-exemples	82.00%
Approche caractérisant les titres de façon empirique	68.82%

Comme le montre le tableau 6 l'approche modifiée pour effectuer la segmentation est plus précise que celle décrite dans l'article [4]. Ceci peut s'expliquer par le fait que dans un cas, les caractéristiques des titres sont définies de manière empirique ; or dans l'autre cas, ces caractéristiques sont extraites des titres identifiés dans le CV et donc sont spécifiques au CV en question.

5. Conclusion et Perspectives

Cet article décrit la construction d'un modèle basé sur l'approche hiérarchique d'extraction des informations dans le CV telle que illustrée dans[4] pour extraire les compétences. Le CV est segmenté en sections en utilisant une version modifiée de l'algorithme proposé dans [4]. La classification des sections a aussi été adaptée en intégrant le titre de la section ainsi que les verbes d'action communément utilisés dans la rédaction des CV comme caractéristiques pour la classification. Après l'étiquettage des blocs, des caractéristiques ont été extraites et utilisées pour construire un modèle de claissification *multi-label* permettant de classer les CV suivant les compétences. L'algorithme de segmentation modifié offre une meilleure précision dans la segmentation que la méthode de base. Aussi, le modèle de classification *multi-label* a donné des résultats satisfaisants (de l'ordre de 90%).

En perspective, il sera question de voir dans quelle mesure utiliser un réseau de neurones profond ; en l'occurence le réseau de neurones convolutionnel pour extraire automatiquement les caractéristiques pertinentes dans le but de faire de la prédiction. En effet, les filtres utilisés dans les réseaux de neurones convolutionnels peuvent permettre d'identifier automatiquement les termes ou compétences de base qui caractérisent chaque compétence.

6. Bibliographie

[1] ALOTAIBI, SHAHA, « A survey of job recommender systems », *International Journal of the Physical Sciences*, vol. 7, 2012.

[2] ASSOCIATION FOR TALENT DEVELOPMENT, « Bridging the skills gap : workforce development is everyone's business », n° 2, 2015.

[3] BENSON, VLADLENA, FILIPPAIOS, FRAGKISKOS, MORGAN, STEPHANIE, « Social Career Management : Social Media and Employability Skills Gap », *Comput. Hum. Behav.*, vol. 30, 519-525, 2014.

[4] CHEN, JIAZE, GAO, LIANGCAI, TANG, ZHI, « Information Extraction from Resume Documents in PDF Format », *Electronic Imaging*, n° 17, 1-8, 2016.

[5] FELLBAUM, CHRISTIANE, , « WordNet : an electronic lexical database », 1998.

[6] GIBAJA, EVA, SEBASTIÁN, , « Multi-label Learning : A Review of the State of the Art and Ongoing Research », *Wiley Int. Rev. Data Min. and Knowl. Disc.*, n° 6, 411-444, 2014.

[7] HUANG, ANNA, « Similarity measures for text document clustering », *Proceedings of the 6th New Zealand Computer Science Research Student Conference*. 2008.

[8] JIECHIEU KAMENI FLORENTIN FLAMBEAU, NORBET TSOPZE, « Approche hiérarchique pour extraire les compétences dans les CV en format pdf », *Proceedings de la conférence sur la recherche en Informatique CRI'17, ENSP Yaoundé*, 2017.

[9] KIVIMÄKI, ILKKA, PANCHENKO, ALEXANDER, DESSY, ADRIEN, VERDEGEM, DRIES, FRANCQ, PASCAL, BERSINI, HUGUES, SAERENS, MARCO, « A Graph-Based Approach to Skill Extraction from Text », *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing*, 79-87, 2013.

[10] MAHESHWARI, SUMIT, SAINANI, ABHISHEK, REDDY, P. KRISHNA, « An Approach to Extract Special Skills to Improve the Performance of Resume Selection », *Databases in Networked Information Systems*, 256-273, 2010.

[11] MARIE-CATHERINE DE MARNEFFE, CHRISTOPHER D. MANNING, « Stanford typed dependencies manual », *Comput. Hum. Behav.*, 2008.

[12] MENG ZHAO, FAIZAN JAVED, FEROSH JACOB, MATT MCNAI, « SKILL : A System for Skill Identification and Normalization », *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,January 25-30, 2015, Austin, Texas, USA.*, 4012-4018, 2015.

[13] MIKOLOV, TOMAS, SUTSKEVER, ILYA, CHEN, KAI, CORRADO, GREG S, DEAN, JEFF, « Distributed Representations of Words and Phrases and their Compositionality », *Advances in Neural Information Processing Systems 26*, 3111-3119, 2013.

[14] MIRANDA, SERGIO, ORCIUOLI, FRANCESCO, LOIA, VINCENZO, SAMPSON, DEMETRIOS, « An Ontology-based Model for Competence Management », *Data Knowl. Eng.*, vol. 107, n° C, 51-66, January 2017.

[15] PENG, YIFAN, CATALINA O, TORII, MANABU, WU, CATHY H, VIJAY-SHANKER, K, « iSimp : A sentence simplification system for biomedicail text », *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, 1-6, 2012.

[16] SHIQIANG GUO, FOLAMI ALAMUDUN, TRACY HAMMOND, « RésuMatcher : A personalized résumé-job matching system », *Wiley Int. Rev. Data Min. and Knowl. Disc.*, vol. 60, 169-182, 2016.

[17] THIMMA REDDY KALVA, « All Graduate Plan B and other Reports », *Skill Finder : Automated Job-Resume Matching System*, 2013.

[18] TOMAS MIKOLOV, KAI CHEN, GREG CORRADO, JEFFREY DEAN, « Efficient Estimation of Word Representations in Vector Space », *Databases in Networked Information Systems*, 256-273, 2013.

[19] VAPNIK, VLADIMIR N., « Statistical Learning Theory », n° 1998.

[20] YU, KUN, GANG GUAN, MING ZHOU, « Resume information extraction with cascaded hybrid model », *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics*, 499-506, 2005.

[21] ZIEMOWIT BEDNAREK, , « Skills gap : The timing of technical changes », *Journal of Economics and Business*, vol. 74, 54-64, 2014.

F Liste protocolaire

UNIVERSITÉ DE YAOUNDÉ I **Faculté des Sciences** Division de la Programmation et du Suivi des Activités Académiques		THE UNIVERSITY OF YAOUNDE I **Faculty of Science** Division of Programming and Follow-up of Academic Affaires
LISTE DES ENSEIGNANTS PERMANENTS		**LIST OF PERMANENT TEACHING STAFF**

ANNÉE ACADEMIQUE 2019/2020
(Par Département et par Grade)
DATE D'ACTUALISATION 03 Mars 2020

ADMINISTRATION

DOYEN : TCHOUANKEU Jean- Claude, *Maitre de Conférences*
VICE-DOYEN / DPSAA : ATCHADE Alex de Théodore, *Maitre de Conférences*
VICE-DOYEN / DSSE : AJEAGAH Gideon AGHAINDUM, *Professeur*
VICE-DOYEN / DRC : ABOSSOLO Monique, *Maitre de Conférences*
Chef Division Administrative et Financière : NDOYE FOE Marie C. F., *Maitre de Conférences*
Chef Division des Affaires Académiques, de la Scolarité et de la Recherche DAASR : MBAZE MEVA'A Luc Léonard, *Professeur*

1- DÉPARTEMENT DE BIOCHIMIE (BC) (38)

N°	NOMS ET PRÉNOMS	GRADE	OBSERVATIONS
1	BIGOGA DIAGA Jude	Professeur	En poste
2	FEKAM BOYOM Fabrice	Professeur	En poste
3	FOKOU Elie	Professeur	En poste
4	KANSCI Germain	Professeur	En poste
5	MBACHAM FON Wilfried	Professeur	En poste
6	MOUNDIPA FEWOU Paul	Professeur	Chef de Département
7	NINTCHOM PENLAP V. épse BENG	Professeur	En poste
8	OBEN Julius ENYONG	Professeur	En poste
9	ACHU Merci BIH	Maître de Conférences	En poste
10	ATOGHO Barbara Mma	Maître de Conférences	En poste
11	AZANTSA KINGUE GABIN BORIS	Maître de Conférences	En poste
12	BELINGA née NDOYE FOE M. C. F.	Maître de Conférences	Chef DAF / FS
13	BOUDJEKO Thaddée	Maître de Conférences	En poste
14	DJUIDJE NGOUNOUE Marcelline	Maître de Conférences	En poste
15	EFFA NNOMO Pierre	Maître de Conférences	En poste

16	NANA Louise épouse WAKAM	Maître de Conférences	En poste
17	NGONDI Judith Laure	Maître de Conférences	En poste
18	NGUEFACK Julienne	Maître de Conférences	En poste
19	NJAYOU Frédéric Nico	Maître de Conférences	En poste
20	MOFOR née TEUGWA Clotilde	Maître de Conférences	Inspecteur de Service MINESUP
21	TCHANA KOUATCHOUA Angèle	Maître de Conférences	En poste

22	AKINDEH MBUH NJI	Chargé de Cours	En poste
23	BEBOY EDZENGUELE Sara Nathalie	Chargée de Cours	En poste
24	DAKOLE DABOY Charles	Chargé de Cours	En poste
25	DJUIKWO NKONGA Ruth Viviane	Chargée de Cours	En poste
26	DONGMO LEKAGNE Joseph Blaise	Chargé de Cours	En poste
27	EWANE Cécile Anne	Chargée de Cours	En poste
28	FONKOUA Martin	Chargé de Cours	En poste
29	BEBEE Fadimatou	Chargée de Cours	En poste
30	KOTUE KAPTUE Charles	Chargé de Cours	En poste
31	LUNGA Paul KEILAH	Chargé de Cours	En poste
32	MANANGA Marlyse Joséphine	Chargée de Cours	En poste
33	MBONG ANGIE M. Mary Anne	Chargée de Cours	En poste
34	PECHANGOU NSANGOU Sylvain	Chargé de Cours	En poste
35	Palmer MASUMBE NETONGO	Chargé de Cours	En poste

36	MBOUCHE FANMOE Marceline Joëlle	Assistante	En poste
37	OWONA AYISSI Vincent Brice	Assistant	En poste
38	WILFRIED ANGIE Abia	Assistante	En poste

2- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE ANIMALES (BPA) (48)

1	AJEAGAH Gideon AGHAINDUM	Professeur	*VICE-DOYEN / DSSE*
2	BILONG BILONG Charles-Félix	Professeur	Chef de Département
3	DIMO Théophile	Professeur	En Poste
4	DJIETO LORDON Champlain	Professeur	En Poste
5	ESSOMBA née NTSAMA MBALA	Professeur	*Vice Doyen/FMSB/UYI*
6	FOMENA Abraham	Professeur	En Poste
7	KAMTCHOUING Pierre	Professeur	En poste
8	NJAMEN Dieudonné	Professeur	En poste

9	NJIOKOU Flobert	Professeur	En Poste
10	NOLA Moïse	Professeur	En poste
11	TAN Paul VERNYUY	Professeur	En poste
12	TCHUEM TCHUENTE Louis Albert	Professeur	*Inspecteur de service* *Coord.Progr./MINSANTE*
13	ZEBAZE TOGOUET Serge Hubert	Professeur	*En poste*

14	BILANDA Danielle Claude	Maître de Conférences	En poste
15	DJIOGUE Séfirin	Maître de Conférences	En poste
16	DZEUFIET DJOMENI Paul Désiré	Maître de Conférences	En poste
17	JATSA BOUKENG Hermine épse MEGAPTCHE	Maître de Conférences	En Poste
18	KEKEUNOU Sévilor	Maître de Conférences	En poste
19	MEGNEKOU Rosette	Maître de Conférences	En poste
20	MONY Ruth épse NTONE	Maître de Conférences	En Poste
21	NGUEGUIM TSOFACK Florence	Maître de Conférences	En poste
22	TOMBI Jeannette	Maître de Conférences	En poste

23	ALENE Désirée Chantal	Chargée de Cours	En poste
26	ATSAMO Albert Donatien	Chargé de Cours	En poste
27	BELLET EDIMO Oscar Roger	Chargé de Cours	En poste
28	DONFACK Mireille	Chargée de Cours	En poste
29	ETEME ENAMA Serge	Chargé de Cours	En poste
30	GOUNOUE KAMKUMO Raceline	Chargée de Cours	En poste
31	KANDEDA KAVAYE Antoine	Chargé de Cours	En poste
32	LEKEUFACK FOLEFACK Guy B.	Chargé de Cours	En poste
33	MAHOB Raymond Joseph	Chargé de Cours	En poste
34	MBENOUN MASSE Paul Serge	Chargé de Cours	En poste
35	MOUNGANG LucianeMarlyse	Chargée de Cours	En poste
36	MVEYO NDANKEU Yves Patrick	Chargé de Cours	En poste
37	NGOUATEU KENFACK Omer Bébé	Chargé de Cours	En poste
38	NGUEMBOK	Chargé de Cours	En poste
39	NJUA Clarisse Yafi	Chargée de Cours	Chef Div. UBA
40	NOAH EWOTI Olive Vivien	Chargée de Cours	En poste
41	TADU Zephyrin	Chargé de Cours	En poste
42	TAMSA ARFAO Antoine	Chargé de Cours	En poste
43	YEDE	Chargé de Cours	En poste

44	BASSOCK BAYIHA Etienne Didier	Assistant	En poste
45	ESSAMA MBIDA Désirée Sandrine	Assistante	En poste
46	KOGA MANG DOBARA	Assistant	En poste
47	LEME BANOCK Lucie	Assistante	En poste
48	YOUNOUSSA LAME	Assistant	En poste

3- DÉPARTEMENT DE BIOLOGIE ET PHYSIOLOGIE VÉGÉTALES (BPV) (33)

1	AMBANG Zachée	Professeur	Chef Division/UYII
2	BELL Joseph Martin	Professeur	En poste
3	DJOCGOUE Pierre François	Professeur	En poste
4	MOSSEBO Dominique Claude	Professeur	En poste
5	YOUMBI Emmanuel	Professeur	Chef de Département
6	ZAPFACK Louis	Professeur	En poste

7	ANGONI Hyacinthe	Maître de Conférences	En poste
8	BIYE Elvire Hortense	Maître de Conférences	En poste
9	KENGNE NOUMSI Ives Magloire	Maître de Conférences	En poste
10	MALA Armand William	Maître de Conférences	En poste
11	MBARGA BINDZI Marie Alain	Maître de Conférences	CT/ MINESUP
12	MBOLO Marie	Maître de Conférences	En poste
13	NDONGO BEKOLO	Maître de Conférences	*CE / MINRESI*
14	NGODO MELINGUI Jean Baptiste	Maître de Conférences	En poste
15	NGONKEU MAGAPTCHE Eddy L.	Maître de Conférences	En poste
16	TSOATA Esaïe	Maître de Conférences	En poste
17	TONFACK Libert Brice	Maître de Conférences	En poste

18	DJEUANI Astride Carole	Chargé de Cours	En poste
19	GOMANDJE Christelle	Chargée de Cours	En poste
20	MAFFO MAFFO Nicole Liliane	Chargé de Cours	En poste
21	MAHBOU SOMO TOUKAM. Gabriel	Chargé de Cours	En poste
22	NGALLE Hermine BILLE	Chargée de Cours	En poste
23	NGOUO Lucas Vincent	Chargé de Cours	En poste
24	NNANGA MEBENGA Ruth Laure	Chargé de Cours	En poste
25	NOUKEU KOUAKAM Armelle	Chargé de Cours	En poste

26	ONANA JEAN MICHEL	Chargé de Cours	En poste
27	GODSWILL NTSOMBAH NTSEFONG	Assistant	En poste
28	KABELONG BANAHO Louis-Paul-Roger	Assistant	En poste
29	KONO Léon Dieudonné	Assistant	En poste
30	LIBALAH Moses BAKONCK	Assistant	En poste
31	LIKENG-LI-NGUE Benoit C	Assistant	En poste
32	TAEDOUNG Evariste Hermann	Assistant	En poste
33	TEMEGNE NONO Carine	Assistant	En poste

4- DÉPARTEMENT DE CHIMIE INORGANIQUE (CI) (34)

1	AGWARA ONDOH Moïse	Professeur	*Chef de Département*
2	ELIMBI Antoine	Professeur	En poste
3	Florence UFI CHINJE épouse MELO	Professeur	*Recteur Univ.Ngaoundere*
4	GHOGOMU Paul MINGO	Professeur	*Ministre Chargé deMiss.PR*
5	NANSEU Njiki Charles Péguy	Professeur	En poste
6	NDIFON Peter TEKE	Professeur	*CT MINRESI*
7	NGOMO Horace MANGA	Professeur	*Vice Chancelor/UB*
8	NDIKONTAR Maurice KOR	Professeur	*Vice-Doyen Univ. Bamenda*
9	NENWA Justin	Professeur	En poste
10	NGAMENI Emmanuel	Professeur	*DOYEN FS UDs*

11	BABALE née DJAM DOUDOU	Maître de Conférences	*Chargée Mission P.R.*
12	DJOUFAC WOUMFO Emmanuel	Maître de Conférences	En poste
13	EMADACK Alphonse	Maître de Conférences	En poste
14	KAMGANG YOUBI Georges	Maître de Conférences	En poste
15	KEMMEGNE MBOUGUEM Jean C.	Maître de Conférences	En poste
16	KONG SAKEO	Maître de Conférences	En poste
17	NDI NSAMI Julius	Maître de Conférences	En poste
18	NJIOMOU C. épse DJANGANG	Maître de Conférences	En poste
19	NJOYA Dayirou	Maître de	En poste

		Conférences	

20	ACAYANKA Elie	Chargé de Cours	En poste
21	BELIBI BELIBI Placide Désiré	Chargé de Cours	CS/ ENS Bertoua
22	CHEUMANI YONA Arnaud M.	Chargé de Cours	En poste
23	KENNE DEDZO GUSTAVE	Chargé de Cours	En poste
24	KOUOTOU DAOUDA	Chargé de Cours	En poste
25	MAKON Thomas Beauregard	Chargé de Cours	En poste
26	MBEY Jean Aime	Chargé de Cours	En poste
27	NCHIMI NONO KATIA	Chargé de Cours	En poste
28	NEBA nee NDOSIRI Bridget NDOYE	Chargée de Cours	CT/ MINFEM
29	NYAMEN Linda Dyorisse	Chargée de Cours	En poste
30	PABOUDAM GBAMBIE A.	Chargée de Cours	En poste
31	TCHAKOUTE KOUAMO Hervé	Chargé de Cours	En poste
32	NJANKWA NJABONG N. Eric	Assistant	En poste
33	PATOUOSSA ISSOFA	Assistant	En poste
34	SIEWE Jean Mermoz	Assistant	En Poste

5- DÉPARTEMENT DE CHIMIE ORGANIQUE (CO) (35)			
1	DONGO Etienne	Professeur	Vice-Doyen
2	GHOGOMU TIH Robert Ralph	Professeur	Dir. IBAF/UDA
3	NGOUELA Silvère Augustin	Professeur	Chef de Departement UDS
4	NKENGFACK Augustin Ephrem	Professeur	Chef de Département
5	NYASSE Barthélemy	Professeur	En poste
6	PEGNYEMB Dieudonné Emmanuel	Professeur	*Directeur/ MINESUP*
7	WANDJI Jean	Professeur	En poste

8	Alex de Théodore ATCHADE	Maître de Conférences	Vice-Doyen / DPSAA
9	EYONG Kenneth OBEN	Maître de Conférences	En poste
10	FOLEFOC Gabriel NGOSONG	Maître de Conférences	En poste
11	FOTSO WABO Ghislain	Maître de Conférences	En poste
12	KEUMEDJIO Félix	Maître de Conférences	En poste
13	KEUMOGNE Marguerite	Maître de Conférences	En poste
14	KOUAM Jacques	Maître de Conférences	En poste
15	MBAZOA née DJAMA Céline	Maître de	En poste

		Conférences	
16	MKOUNGA Pierre	Maître de Conférences	En poste
17	NOTE LOUGBOT Olivier Placide	Maître de Conférences	Chef Service/MINESUP
18	NGO MBING Joséphine	Maître de Conférences	Sous/Direct. MINERESI
19	NGONO BIKOBO Dominique Serge	Maître de Conférences	En poste
20	NOUNGOUE TCHAMO Diderot	Maître de Conférences	En poste
21	TABOPDA KUATE Turibio	Maître de Conférences	En poste
22	TCHOUANKEU Jean-Claude	Maître de Conférences	*Doyen /FS/ UYI*
23	TIH née NGO BILONG E. Anastasie	Maître de Conférences	En poste
24	YANKEP Emmanuel	Maître de Conférences	En poste

25	AMBASSA Pantaléon	Chargé de Cours	En poste
26	KAMTO Eutrophe Le Doux	Chargé de Cours	En poste
27	MVOT AKAK CARINE	Chargé de Cours	En poste
28	NGNINTEDO Dominique	Chargé de Cours	En poste
29	NGOMO Orléans	Chargée de Cours	En poste
30	OUAHOUO WACHE Blandine M.	Chargée de Cours	En poste
31	SIELINOU TEDJON Valérie	Chargé de Cours	En poste
32	TAGATSING FOTSING Maurice	Chargé de Cours	En poste
33	ZONDENDEGOUMBA Ernestine	Chargée de Cours	En poste

34	MESSI Angélique Nicolas	Assistant	En poste
35	TSEMEUGNE Joseph	Assistant	En poste

6- DÉPARTEMENT D'INFORMATIQUE (IN) (27)

1	ATSA ETOUNDI Roger	Professeur	*Chef Div.MINESUP*
2	FOUDA NDJODO Marcel Laurent	Professeur	*Chef Dpt ENS/Chef IGA.MINESUP*

3	NDOUNDAM Réné	Maître de Conférences	En poste

4	AMINOU Halidou	Chargé de Cours	*Chef de Département*
5	DJAM Xaviera YOUH - KIMBI	Chargé de Cours	En Poste
6	EBELE Serge Alain	Chargé de Cours	En poste
7	KOUOKAM KOUOKAM E. A.	Chargé de Cours	En poste

8	MELATAGIA YONTA Paulin	Chargé de Cours	En poste
9	MOTO MPONG Serge Alain	Chargé de Cours	En poste
10	TAPAMO Hyppolite	Chargé de Cours	En poste
11	ABESSOLO ALO'O Gislain	Chargé de Cours	En poste
12	MONTHE DJIADEU Valery M.	Chargé de Cours	En poste
13	OLLE OLLE Daniel Claude Delort	Chargé de Cours	C/D Enset. Ebolowa
14	TINDO Gilbert	Chargé de Cours	En poste
15	TSOPZE Norbert	Chargé de Cours	En poste
16	WAKU KOUAMOU Jules	Chargé de Cours	En poste

17	BAYEM Jacques Narcisse	Assistant	En poste
18	DOMGA KOMGUEM Rodrigue	Assistant	En poste
19	EKODECK Stéphane Gaël Raymond	Assistant	En poste
20	HAMZA Adamou	Assistant	En poste
21	JIOMEKONG AZANZI Fidel	Assistant	En poste
22	MAKEMBE. S . Oswald	Assistant	En poste
23	MESSI NGUELE Thomas	Assistant	En poste
24	MEYEMDOU Nadège Sylvianne	Assistante	En poste
25	NKONDOCK. MI. BAHANACK.N.	Assistant	En poste

7- DÉPARTEMENT DE MATHÉMATIQUES (MA) (31)

1	EMVUDU WONO Yves S.	Professeur	*Inspecteur MINESUP*

2	AYISSI Raoult Domingo	Maître de Conférences	Chef de Département
3	NKUIMI JUGNIA Célestin	Maître de Conférences	En poste
4	NOUNDJEU Pierre	Maître de Conférences	*Chef service des programmes & Diplômes*
5	MBEHOU Mohamed	Maître de Conférences	En poste
6	TCHAPNDA NJABO Sophonie B.	Maître de	Directeur/AIMS

		Conférences	Rwanda

7	AGHOUKENG JIOFACK Jean Gérard	Chargé de Cours	Chef Cellule MINPLAMAT
8	CHENDJOU Gilbert	Chargé de Cours	En poste
9	DJIADEU NGAHA Michel	Chargé de Cours	En poste
10	DOUANLA YONTA Herman	Chargé de Cours	En poste
11	FOMEKONG Christophe	Chargé de Cours	En poste
12	KIANPI Maurice	Chargé de Cours	En poste
13	KIKI Maxime Armand	Chargé de Cours	En poste
14	MBAKOP Guy Merlin	Chargé de Cours	En poste
15	MBANG Joseph	Chargé de Cours	En poste
16	MBELE BIDIMA Martin Ledoux	Chargé de Cours	En poste
17	MENGUE MENGUE David Joe	Chargé de Cours	En poste
18	NGUEFACK Bernard	Chargé de Cours	En poste
19	NIMPA PEFOUKEU Romain	Chargée de Cours	En poste
20	POLA DOUNDOU Emmanuel	Chargé de Cours	En poste
21	TAKAM SOH Patrice	Chargé de Cours	En poste
22	TCHANGANG Roger Duclos	Chargé de Cours	En poste
23	TCHOUNDJA Edgar Landry	Chargé de Cours	En poste
24	TETSADJIO TCHILEPECK M. E.	Chargée de Cours	En poste
25	TIAYA TSAGUE N. Anne-Marie	Chargée de Cours	En poste
26	MBIAKOP Hilaire George	Assistant	En poste
27	BITYE MVONDO Esther Claudine	Assistante	En poste
28	MBATAKOU Salomon Joseph	Assistant	En poste
29	MEFENZA NOUNTU Thiery	Assistant	En poste
30	TCHEUTIA Daniel Duviol	Assistant	En poste

8- DÉPARTEMENT DE MICROBIOLOGIE (MIB) (18)			
1	ESSIA NGANG Jean Justin	Professeur	*Chef de Département*

2	BOYOMO ONANA	Maître de Conférences	En poste
3	NWAGA Dieudonné M.	Maître de Conférences	En poste
4	NYEGUE Maximilienne Ascension	Maître de	En poste

		Conférences	
5	RIWOM Sara Honorine	Maître de Conférences	En poste
6	SADO KAMDEM Sylvain Leroy	Maître de Conférences	En poste

7	ASSAM ASSAM Jean Paul	Chargé de Cours	En poste
8	BODA Maurice	Chargé de Cours	En poste
9	BOUGNOM Blaise Pascal	Chargé de Cours	En poste
10	ESSONO OBOUGOU Germain G.	Chargé de Cours	En poste
11	NJIKI BIKOÏ Jacky	Chargée de Cours	En poste
12	TCHIKOUA Roger	Chargé de Cours	En poste
13	ESSONO Damien Marie	Assistant	En poste
14	LAMYE Glory MOH	Assistant	En poste
15	MEYIN A EBONG Solange	Assistante	En poste
16	NKOUDOU ZE Nardis	Assistant	En poste
17	SAKE NGANE Carole Stéphanie	Assistante	En poste
18	TOBOLBAÏ Richard	Assistant	En poste

9. DEPARTEMENT DE PYSIQUE(PHY) (40)

1	BEN- BOLIE Germain Hubert	Professeur	En poste
2	EKOBENA FOUDA Henri Paul	Professeur	*Chef Division. UN*
3	ESSIMBI ZOBO Bernard	Professeur	En poste
4	KOFANE Timoléon Crépin	Professeur	En poste
5	NANA ENGO Serge Guy	Professeur	En poste
6	NDJAKA Jean Marie Bienvenu	Professeur	Chef de Département
7	NOUAYOU Robert	Professeur	En poste
8	NJANDJOCK NOUCK Philippe	Professeur	*Sous Directeur/ MINRESI*
9	PEMHA Elkana	Professeur	En poste
10	TABOD Charles TABOD	Professeur	Doyen Univ/Bda
11	TCHAWOUA Clément	Professeur	En poste
12	WOAFO Paul	Professeur	En poste

13	BIYA MOTTO Frédéric	Maître de Conférences	DG/HYDRO Mekin
14	BODO Bertrand	Maître de Conférences	En poste
15	DJUIDJE KENMOE épouse ALOYEM	Maître de Conférences	En poste
16	EYEBE FOUDA Jean sire	Maître de	En poste

		Conférences	
17	FEWO Serge Ibraïd	Maître de Conférences	En poste
18	HONA Jacques	Maître de Conférences	En poste
19	MBANE BIOUELE César	Maître de Conférences	En poste
20	NANA NBENDJO Blaise	Maître de Conférences	En poste
21	NDOP Joseph	Maître de Conférences	En poste
22	SAIDOU	Maître de Conférences	MINERESI
23	SIEWE SIEWE Martin	Maître de Conférences	En poste
24	SIMO Elie	Maître de Conférences	En poste
25	VONDOU Derbetini Appolinaire	Maître de Conférences	En poste
26	WAKATA née BEYA Annie	Maître de Conférences	*Sous Directeur/ MINESUP*
27	ZEKENG Serge Sylvain	Maître de Conférences	En poste

28	ABDOURAHIMI	Chargé de Cours	En poste
29	EDONGUE HERVAIS	Chargé de Cours	En poste
30	ENYEGUE A NYAM épse BELINGA	Chargée de Cours	En poste
31	FOUEDJIO David	Chargé de Cours	Chef Cell. MINADER
32	MBINACK Clément	Chargé de Cours	En poste
33	MBONO SAMBA Yves Christian U.	Chargé de Cours	En poste
34	MELI'I Joelle Larissa	Chargée de Cours	En poste
35	MVOGO ALAIN	Chargé de Cours	En poste
36	OBOUNOU Marcel	Chargé de Cours	DA/Univ Inter Etat/Sangmalima
37	WOULACHE Rosalie Laure	Chargée de Cours	En poste

38	AYISSI EYEBE Guy François Valérie	Assistant	En poste
39	CHAMANI Roméo	Assistant	En poste
40	TEYOU NGOUPOU Ariel	Assistant	En poste

10- DÉPARTEMENT DE SCIENCES DE LA TERRE (ST) (43)

1	BITOM Dieudonné	Professeur	*Doyen / FASA / UDs*
2	FOUATEU Rose épse YONGUE	Professeur	En poste
3	KAMGANG Pierre	Professeur	En poste
4	NDJIGUI Paul Désiré	Professeur	Chef de Département
5	NDAM NGOUPAYOU Jules-Remy	Professeur	En poste

6	NGOS III Simon	Professeur	DAAC/Uma
7	NKOUMBOU Charles	Professeur	En poste
8	NZENTI Jean-Paul	Professeur	En poste

9	ABOSSOLO née ANGUE Monique	Maître de Conférences	*Vice-Doyen / DRC*
10	GHOGOMU Richard TANWI	Maître de Conférences	CD/Uma
11	MOUNDI Amidou	Maître de Conférences	*CT/ MINIMDT*
12	NGUEUTCHOUA Gabriel	Maître de Conférences	CEA/MINRESI
13	NJILAH Isaac KONFOR	Maître de Conférences	En poste
14	ONANA Vincent Laurent	Maître de Conférences	*Chef service Maintenance & du Matériel*
15	BISSO Dieudonné	Maître de Conférences	*Directeur/Projet Barrage Memve'ele*
16	EKOMANE Emile	Maître de Conférences	En poste
17	GANNO Sylvestre	Maître de Conférences	En poste
18	NYECK Bruno	Maître de Conférences	En poste
19	TCHOUANKOUE Jean-Pierre	Maître de Conférences	En poste
20	TEMDJIM Robert	Maître de Conférences	En poste
21	YENE ATANGANA Joseph Q.	Maître de Conférences	*Chef Div. /MINTP*
22	ZO'O ZAME Philémon	Maître de Conférences	*DG/ART*

23	ANABA ONANA Achille Basile	Chargé de Cours	En poste
24	BEKOA Etienne	Chargé de Cours	En poste
25	ELISE SABABA	Chargé de Cours	En poste
26	ESSONO Jean	Chargé de Cours	En poste
27	EYONG JOHN TAKEM	Chargé de Cours	En poste
28	FUH Calistus Gentry	Chargé de Cours	*Sec. D'Etat/MINMIDT*
29	LAMILEN BILLA Daniel	Chargé de Cours	En poste
30	MBESSE CECILE OLIVE	Chargée de Cours	En poste
31	MBIDA YEM	Chargé de Cours	En poste
32	METANG Victor	Chargé de Cours	En poste
33	MINYEM Dieudonné-Lucien	Chargé de Cours	*CD/Uma*
34	NGO BELNOUN Rose Noël	Chargée de Cours	En poste
35	NGO BIDJECK Louise Marie	Chargée de Cours	En poste
36	NOMO NEGUE Emmanuel	Chargé de Cours	En poste
37	NTSAMA ATANGANA	Chargé de Cours	En poste

	Jacqueline		
38	TCHAKOUNTE J. épse NOUMBEM	Chargée de Cours	*Chef.cell / MINRESI*
39	TCHAPTCHET TCHATO De P.	Chargé de Cours	En poste
40	TEHNA Nathanaël	Chargé de Cours	En poste
41	TEMGA Jean Pierre	Chargé de Cours	En poste
42	FEUMBA Roger	Assistant	En poste
43	MBANGA NYOBE Jules	Assistant	En poste

Répartition chiffrée des Enseignants de la Faculté des Sciences de l'Université de Yaoundé I

NOMBRE D'ENSEIGNANTS					
DÉPARTEMENT	**Professeurs**	**Maîtres de Conférences**	**Chargés de Cours**	**Assistants**	**Total**
BCH	9 (1)	13 (09)	14 (06)	3 (2)	**39 (18)**
BPA	13 (1)	09 (06)	19 (05)	05 (2)	**46 (14)**
BPV	06 (0)	11 (02)	9 (06)	07 (01)	**33 (9)**
CI	10 (1)	9 (02)	12 (02)	03 (0)	**34 (5)**
CO	7 (0)	17 (04)	09 (03)	02 (0)	**35(7)**
IN	2 (0)	1 (0)	13 (01)	09 (01)	**25 (2)**
MAT	1 (0)	5 (0)	19 (01)	06 (02)	**31 (3)**
MIB	1 (0)	5 (02)	06 (01)	06 (02)	**18 (5)**
PHY	12 (0)	15 (02)	10 (03)	03 (0)	**40 (5)**
ST	8 (1)	14 (01)	19 (05)	02 (0)	**43(7)**
Total	**69 (4)**	**99 (28)**	**130 (33)**	**46 (10)**	**344 (75)**

Soit un total de **344 (**75**)** dont :

- Professeurs **68 (4)**
- Maîtres de Conférences **99 (**28**)**
- Chargés de Cours **130 (33)**
- Assistants **46** (10**)**

( ) = Nombre de Femmes **75**