2022-1

Measuring Semantic Similarity of Documents by Using Named Entity Recognition Methods

David Efraín Muñoz Morales
Technological University Dublin, x00151532@mytudublin.ie

Recommended Citation

Muñoz Morales, D. E. (2022). Measuring Semantic Similarity of Documents by Using Named Entity Recognition Methods. Technological University Dublin. DOI: 10.21427/F1S5-2C94

Measuring Semantic Similarity of Documents by Using Named Entity Recognition Methods

David Efraín Muñoz Morales

Supervisor: Dr. Fernando Pérez Téllez
Dr. David Eduardo Pinto Avendaño

Department of Computing
TU Dublin, Tallaght Campus

This dissertation is submitted for the degree of
Masters by research in Computing

January 2022

Declaration

I certify that this thesis which I now submit for examination for the award of Masters by Research, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for graduate study by research of the Technological University Dublin (TU Dublin) and has not been submitted in whole or in part for another award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the TU Dublin's guidelines for ethics in research.

TU Dublin has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

[illegible]

David Efraín Muñoz Morales
January 2022

Acknowledgements

First of all, I want to mention that I am enormously grateful to God for allowing me to get here. Although there were many difficulties and obstacles on my way to finishing my master's degree on time, it is also true that I had the support of my beloved wife, my precious family, and my invaluable friends. In addition, the university was flexible and understood my declining health situation in recent months that made it impossible for me to continue with the same workload and even totally pause me for a few months. All the academics and staff at the university were very empathetic with me, and that is something that I will always appreciate.

Sometimes we can find ourselves in situations that make us take advantage of others, but it is always good to remember that we have the opportunity to CHOOSE THE RIGHT.

Abstract

The work presented in this thesis was born from the desire to map documents with similar semantic concepts between them. We decided to address this problem as a named entity recognition task, where we have identified key concepts in the texts we use, and we have categorized them. So, we can apply named entity recognition techniques and automatically recognize these key concepts inside other documents. However, we propose the use of a classification method based on the recognition of named entities or key phrases, where the method can detect similarities between key concepts of the texts to be analyzed, and through the use of Poincaré embeddings, the model can associate the existing relationship between these concepts. Thanks to the Poincaré Embeddings' ability to capture relationships between words, we were able to implement this feature in our classifier. Consequently for each word in a text we check if there are words close to it that are also close to the words that make up the key phrases that we use as Gold Standard. Therefore when detecting potential close words that make up a named entity, the classifier then applies a series of characteristics to classify it.

The methodology used performed better than when we only considered the POS structure of the named entities and their n-grams. However, determining the POS structure and the n-grams were important to improve the recognition of named entities in our research. By improving time to recognize similar key phrases between documents, some common tasks in large companies can have a notorious benefit. An important example is the evaluation of resumes, to determine the best professional for a specific position. This task is characterized by consuming a lot of time to find the best profiles for a position, but our contribution in this research work considerably reduces that time, finding the best profiles for a job. Here the experiments are shown considering job descriptions and real resumes, and the methodology used to determine the representation of each of these documents through their key phrases is explained.

Publications during this research work

- Muñoz, D., Perez F., Pinto D., **Poincaré Embeddings in the Task of Named Entity Recognition**. In Mexican International Conference on Artificial Intelligence,Springer, Cham, 2020.

- Muñoz, D., Perez, F., Pinto, D., **Named Entity Recognition Based on a Graph Structure**. Computación y Sistemas, ISSN: 2395-8618, 2020.

- Muñoz, D., Perez, F., Pinto, D., **A Web-based Tagger for Named Entities Detection**. Research in Computer Science, ISSN:1870-4069 (Indexed by DBLP, LatIndex, Periodica), 2019.

- Muñoz, D., Perez, F., Pinto, D., **Collaborative Web-based tagger for Named Entities in the Task of Information Extraction**. Journal: Pistas Educativas, Vol 40, ISSN:2448-847X. 2018.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Today's world is very demanding with the use of the latest technologies, so practically everyone should know how to use a computer. This same thing has caused the number of users on the internet to grow enormously every day, and consequently the amount of textual information in digital form. This is because it is easier to digitize the textual information and make it accessible to everyone. So that practically now any document is found electronically, from books, news, articles, job descriptions, to personal documents such as birth certificates or resumes. All this causes the need to look for methodologies for easy information management, such as information extraction (IE) and text classification.

In this sense, information extraction plays a very important role when extracting structured information from unstructured electronic texts. And since IE considers that in a set of documents these can be represented by entities or subjects, it is also possible that these documents could be classified by subject. Thus, the classification of texts becomes extremely useful when what is intended is to handle large amounts of information.

The classification of texts allows to find texts similar to a certain text, and in this way to discard texts that do not keep a relationship with the initial text. This has allowed Internet users to find information that is relevant to them when searching for a certain topic on the Internet, thanks to the fact that search engines use ranking and classification algorithms, and this allows them to improve their results. Besides, many online store platforms, as well as music and streaming platforms, have incorporated classification algorithms to recommend to their users some movies, songs, games, books, or articles related to their search.

For this reason, it is of great interest for the development of this research to analyze the methods of classification of texts based on the extraction of key concepts.

1.1 Background

Key phrase extraction is the process through which phrases that represent the most relevant topics of a text are selected, and this may be done through named entities, n-grams, noun phrases that meet certain lexical patterns, or label sequences using Conditional Random Fields. In this way, the extracted key phrases provide the necessary information so that the documents can be categorized according to common themes (Sahrawat *et al.*, 2020). From this perspective, it is possible to perform key phrase extraction employing methods such as named entity classifiers. Well, in essence, a key phrase can be seen as a named entity that can be defined depending on the context in question. For example, the entity named *"Skill"* could be defined in the context of job offers, where it is important to highlight the skills required by the company for that job.

However, some key phrases or relevant concepts from the text can be ignored if you try to extract them through methods such as named entity classifiers. For this reason, it would be possible to take advantage of measures for semantic similarity between words, sentences, or whole texts. This is known as Semantic Textual Similarity (STS) (Chandrasekaran and Mago, 2020).

Even taking into consideration the textual semantic similarity, it might not be enough to effectively classify similar concepts. Fortunately, several techniques can encode the relationships between words, just as vector models do. These vector models represent the semantic relationships between the words in a text (Wang *et al.*, 2020), but it is important to know which model to use to get the most out of these representations. Being able to find similarities between key concepts in large sets of texts represents a challenge, and for that, it is necessary to address different techniques for extracting key phrases, as well as implementing methods to capture textual semantic relationships. It is necessary to have a methodology that can efficiently solve this task.

1.2 Research questions

Being able to find reliable similarities between key phrases (semantically speaking) can lead to an improvement of text classification, and thus improve the processing of information in the industry. Although there are different techniques to extract and classify information, it is necessary to conduct a study that focuses on the analysis and comparison of these techniques to know their efficiency.

This research aims to explore the different current methodologies to find similarities between texts and then determine the best of them. Knowing how these methodologies work, it might be possible to implement some of its techniques and generate a competitive proposal. Because of all the above mentioned, the research questions that guide this work are formulated:

- *Does semantic similarity between key phrases have an improvement when classifying documents?*
- *Do word embeddings have importance when capturing semantic relationships between words?*
- *Does a graph structure have the ability to effectively represent relations between words?*
- *Can key phrases be represented as named entities by using n-grams?*
- *Can Part-of-Speech be used to represent the relationship between words?*

To answer these research questions, there are some tasks that we must fulfill. These tasks are listed below:

- Do a study on the scientific investigations related to the classification of texts and the extraction of key phrases.
- Investigate the measures used to treat semantic similarity.
- Investigate the different models proposed to encode the relationships between words in a text.
- Collect text documents and manually extract the key phrases to consider them as Gold Standard.
- Carry out comparative experiments between the different methods studied and determine their level of effectiveness.
- Analyze the set of key phrases that make up the Gold Standard, in order to determine possible characteristics of each of the classes.
- Design a technique that allows the generation of key phrases similar to the original ones of the Gold Standard.
- Define rules about the Gold Standard classes that allow the identification of new key phrases.

- Develop a classifier model based on a graph structure, which includes the Part-of-Speech as well as the use of synonyms.
- Compare the proposal with the studied methods, and show the results obtained.

1.3 Motivation

The current demand for professionals in the technological sector has grown disproportionately due to the huge technological advances of the last decade. This has led to the emergence of multiple technology-oriented companies around the world. From small companies to multinationals, they all offer job vacancies almost daily.

On the other hand, there are more and more people specializing in the use and development of technologies. This causes HR departments to spend a lot of time reading applicants' resumes. Besides, many of the applicants can be arbitrarily discarded because of not having enough capacity in the HR departments to read all the resumes in such a short period of time.

This is a very time-consuming and inaccurate task as it does not consider all the profiles available for the job offered. However, with the help of computers, thousands of documents could be processed and analyzed in a matter of minutes. For this reason, it is necessary to clean the documents, eliminating the information overload that does not provide relevant aspects of the content. This is achieved by applying some machine learning techniques to extract the key concepts from each text, through which a document can be represented.

By applying some machine learning methods it may be possible to find similarities between the key concepts of a resume and the key concepts of a job offer. In this way, the best profiles of professionals for a given job offer you can found, and thus start by interviewing them without wasting a lot of time reading resumes that fail to satisfy the skills necessary for the job.

For this dissertation, Job Descriptions of the website *www.jobs.ie* in the *Information Technology* sector have been considered. These Job Descriptions have been manually labeled, highlighting their key concepts based on categories defined by the type of document.

1.4 Contribution

We have developed a web-based tagger system, which allowed the key phrases of the Job Descriptions in the IT sector to be manually labeled. This system also allow the possibility of labeling any type of text document, as well as defining its classes (labels) as needed. This system have the ability to export a labeled document to a structured format, ready to be used for machine learning algorithms.

We are going to design a series of rules based on the features of manually labeled key phrases. These rules will be incorporated into the classifier model that we will propose.

The key phrase classification model that we are going to present will be based on a graph structure, where valuable information about each key phrase will be stored, such as the Part-of-Speech or the synonyms of each word that makes it up. This model will consider some characteristics of the grammatical structures of manually labeled key phrases. In this way it is intended to offer an alternative to the classification of texts, detecting the key phrases of a text document.

1.5 Thesis outline

The rest of the thesis is structured in 6 chapters as follows:

Chapter 2: Literature Review

In this chapter, we show a critical analysis of the different sources that we have consulted regarding the area of text classification where the different existing approaches are studied from the simplest to the most elaborate, the use of semantic textual relations in the improvement of this classification task. It also critically analyzes how the extraction of key phrases and named entities can contribute to the classification of texts. And it is right there in that Named Entity Recognition task, which shows a critical analysis on supervised, semi-supervised and unsupervised methods.

Chapter 3: Named Entities

Chapter 3 addresses the topic of Named Entities as an important IE subtask since it seeks to locate and classify named entities of a text, which represent important and remarkable data, and with these named entities a text could be represented. Depending on the type of texts to be studied (scientific articles, sports news, job offers, etc.) it is how these named entities can also be studied as n-grams and for the same reason be extracted as n-grams. So an n-gram is a subsequence of n words of a sequence of words in some text. However, it is necessary to consider that each of the words in this subsequence of words belongs to a grammatical category known as Part-of-Speech (POS). In this way, the category of a word can be known or predicted based on the words that precede or follow it.

Chapter 4: Semantic Similarity

This chapter addresses a very important aspect of text classification that is known as semantic similarity. When it comes to comparing whether a document is similar to another different techniques can be useful to determine the degree of similarity, even when they do not use the same words. For this, different mathematical mechanisms calculate the level of semantic relationship between concepts, known as semantic measures. Specifically, a similarity measure known as Wu&Palmer is explained, which is a Knowledge-based measure.

Also, this chapter explains the selection process developed to select new key phrases created from the set of original key phrases, and that with the help of synonyms it was possible to create them. These new key phrases that were created and selected were added to the original set, thus expanding the training set.

Chapter 5: Enriched Graph Structure for Key Phrases Recognition

This chapter presents a proposal for a key phrase classifier, based on a data structure represented by a graph. The key phrase storage and enrichment technique proposed in this section are based on a graph structure. Here each key phrase is tokenized and the tokens are stored as nodes of the structure, and the edges that connect to each token have a weight that changes depending on whether a key phrase similar to an existing one is added.

On the other hand, alternate nodes are added to the original ones, based on their synonyms. The alternate routes are created from the synonyms of the tokens that make up each key phrase, in this way the classifier could recognize key phrases where some of its words have some relationship with those of the original key phrases. This proposed classifier is compared with other classifiers and the results are discussed.

Chapter 6: Models for Word Representations

This chapter explains the importance of representing textual information with numeric data, for this reason, it is necessary to use models with the ability to represent words optimally. This section deals with the topic of word embeddings as models for the representation of words. For this, some models that stand out for the representation of words are explained. First, the implementation of Word2Vec for computing vector representations of words is addressed, where for each word a vector of words close to it that have a semantic relationship is generated. Then an unsupervised learning algorithm called GloVe is explained, which also generates vectors for the representation of words, calculating the relationship between words. Finally, it is explained how Poincaré Embeddings can improve the representation of words based on hyperbolic space, allowing the learning of parsimonious representations of symbolic data by simultaneously capturing hierarchy and similarity. This has an advantage over vector models based on Euclidean space.

Chapter 7: Poincaré Embeddings for Named Entities Recognition

This chapter explains how we can take advantage of Poincaré embeddings to capture the syntactic and semantic relationships between the words that make up the named entities that we have used. Thanks to the fact that this implementation allows mapping the words used in hyperbolic space, we can easily and reliably determine which are the closest words to a given word. Following this logic, we can repeat the process of obtaining the closest words for each word found, so that we can build sentences. And thus determine if these phrases have logic and resemble any of the named entities of the training set.

Chapter 8: Conclusions

Finally, the findings of this research are shown based on the results obtained with our proposals and experiments, discussing the limitations as well as the contributions made throughout the work. The possible applications that this work could take in the future are also explained.

Chapter 2

Literature Review

Different automatic methods for classifying textual information are studied separated depending on the different approaches implemented in the last years. The first approaches refers to the task of classifying texts into classes or groups, where traditional and the most recent methods are covered. The importance of this task is highlighted, and some of the ways to easy represent a text document are included such as key phrases. Then, state-of-the-art approaches related to the task of key phrase extraction are discussed since high quality key phrases can represent the content of text documents such as scientific publications, books, news, etc., and then improve the information retrieval and classification of them (Papagiannopoulou and Tsoumakas, 2020). Named Entity Recognition (NER) seen as a similar task of key phrase extraction is then studied and the state of the art is presented.

When conducting an in-depth study on the state of the art for text classification and the different techniques or methods used to achieve it, we wish to propose a competent alternative that considers the main points found in this study.

2.1 Text Classification

Automatic text classification is the task of grouping textual documents into predefined classes. Text classification systems are based on 4 levels: document level, paragraph level, sentence level and sub-sentence level. This task problem is widely studied since nowadays with the daily amount of textual information generated thanks to the Internet, it has become necessary to automatic classify gigantic amounts of information when searching an specific topic. However, the performance of the correct text classification is affected by the type of algorithm that will be used. In this sense, there are algorithms that are based on probabilistic and statistical models which require some characteristics to be extracted from the data set. And on the other hand, there are the algorithms that make use of deep neural networks.

Probabilistic classification techniques require textual information to be processed previously to obtain good results, this previous stage is known as feature extraction, which consists of extracting and labeling through artificial methods characteristics that can serve as an example for that textual data set. In this way, a selection of the most relevant information is made, avoiding information overload. Some feature extraction methods that can be applied are weighted word techniques (TF-IDF and TF) and word embeddings such as Word2Vec (Mikolov *et al.*, 2013b) and GloVe (Jeffrey Pennington, 2014). Then statistic-based models can be applied for text classification, such as Naïve Bayes, K-nearest neighbor, Support Vector Machine, Decision Trees and Random Forest. However, this is an expensive task as we need to extract first these features in order to train a classifier model.

On the other hand, deep learning techniques (deep neural networks) automatically find high-level features from data, providing valuable semantic representations for text. The first two implementations using deep learning in the task of text classification are feed-forward neural network and recursive neural network having an improvement over the probabilistic methods mentioned before. These deep neural networks can be classified by structure in: Recursive Neural Network methods, Multilayer Perceptron methods, Recurrent Neural Networks, Convolutional Neural Network methods, Attention-based methods, Transformer-based methods, and Graph Neural Network methods.

2.1.1 Shallow Text Classification

Shallow learning is the term given to neural network approaches that mainly use one hidden layer. These approaches also make use of handcrafted features,representing a time consuming activity.

Naïve Bayes approach is one of the simplest shallow classifiers where it is assumed that all features are independent of each other. This technique requires a small training data. Naïve Bayes approach is one of the simplest shallow classifiers where it is assumed that the value of a determined feature is dependent on the value of another feature. This technique requires a small training data. In this sense (Ruan *et al.*, 2020), proposed a new class-specific deep feature weighting technique for Multinomial Naïve Bayes text classifiers by using a statistical feature weighting technique. This approach calculates the conditional probabilities of text classifiers by computing the frequencies of features from training data. This approach considers the statistical metric R_{a_ic} that can measure the dependency between a term and a class. However, it introduces two new factors, considering the class distribution of the documents containing a term a_i and the class frequency factor. Where a term a_i denotes a class c_i feature, and the class frequency factor refers to the number of classes of the documents containing a term a_i. The experiments were carried out over 19 text classification benchmark datasets and using the V-fold cross validation technique. The proposed method notably outperforms other feature weighting implementations in terms of accuracy and execution time.

However, before any machine learning method can be applied to classify texts, is necessary to transform this unstructured data into a structured format to be processed for learning algorithms. In traditional text classification, documents are represented by bags-of-words (BOW) where every word in a document is included only if it appears 3 or more times and if it is not an stop-word (Joachims, 1996), (Scott and Matwin, 1998).

2.1.2 Deep Text Classification

Approaches based on multiple hidden layers are considered as Deep Learning techniques because they involve multiple levels of representations and features that can be learned automatically from the input dataset. However, the training of these kinds of models is more difficult than training shallow models (Li *et al.*, 2020).

These approaches have a huge disadvantage since they only consider the term frequency in the document, ignoring the semantic relationships between terms. And the problem is that two documents can be treating the same information topic even when they do not use the same words (synonyms can be used to refer to the same concept).

2.2 Textual Semantic Relations

Feature selection methods are important when classifying text documents, however classical methods only consider the correlation between features and categories. Zong *et al.* (2015) propose a novel discriminative and semantic feature selection (DFS + Similarity) method consisting of two stages, first by selecting features with strong discriminative power in documents and then by computing the semantic similarity between features and documents, without relying on any other external information source. Using the selected features, every document is transformed into feature vectors, and then by employing an SVM model with these vectors as input data the effectiveness of the proposed method is evaluated in terms of macro-F measure and micro-F measure. The performance of the proposed method (DFS + Similarity) is compared with the traditional feature selection methods: Chi-Square statistic, information gain (IG), and mutual information (MI). Two different datasets often used in text categorization tasks are used to make comparisons of methods: Reuter-21578 and 20-Newsgroups. The results obtained by the proposed method outperforms other methods over a full range of different number of features, achieving its highest macro-F (0.9220) and micro-F (0.9682) when the number of features is 5000. The number of features is a factor in the performance of text categorization.

2.3 Key phrase Extraction

In the task of automatically extracting key phrases from text, Witten *et al.* (2005) proposed an algorithm called KEA which consists of two phases training and extraction. In this approach, a prediction model is constructed by using a set of documents with key phrases labeled, and then the model is used to find key phrases in new documents. In this approach the Naïve Bayes technique is used due to its simplicity and its capacity to learn two sets of numeric weights, positive (*is a key phrase*) and negative (*is not a key phrase*) instances, selecting the best set of key phrases. However, best key phrases are not always statistically significant (frequent) or they even do not appear in the document. Liu *et al.* (2011) called this phenomenon as *vocabulary gap* between documents and key phrases, and that is why they suggested a more flexible and reliable method.

Liu *et al.* provided a new perspective to key phrase extraction, considering a document and its key phrases as two different languages, even when both describe the same object. Liu *et al.* used word alignment models (WAM) in statistical machine translation (SMT) to learn translation probabilities between the words in documents and the words in key phrases. The result is a unified framework for key phrase extraction, where the appropriate suggested key phrases are not necessarily frequent in their corresponding text documents. This approach has the advantage of being language-independent (with the possibility of be performed in different language documents). However, as textual information daily increases on the Internet more precise methods are necessary to correctly extract the most representative key phrases.

In 2017 was introduced a task for extracting from scientific publications key phrases and the relations between them, the task was called: SemEval 2017 Task 10 (Augenstein *et al.*, 2017), and it has a big interest for publishers who can recommend similar publications to readers if they have the correct most representative key phrases for each article. In this task Buscaldi *et al.* (2017) presented an approach for key phrases classification based on Support Vector Machines (SVM), using WordNet and word embeddings as features. External knowledge was integrated from an existing resource through WordNet and some other learned from a large corpus (Google News Dataset) using word embeddings. The obtained results in Buscaldi *et al.* method outperform the results obtained by the best system presented at SemEval 2017.

On the other hand, the research study made in (Mothe *et al.*, 2018) showed that there are not differences when using word embeddings and when not using it. Mothe *et al.* proposed an automatic key phrase extraction method using graph-based methods, where the principle is to construct a graph of words and/or phrases. In this way, nodes represents the candidate key phrases and edges connects two nodes if they are related, and the weighting for edges is computed using co-occurrences or semantic relatedness. Authors showed how to integrate word embeddings into key phrase extraction models considering the graph-based method. The obtained results showed that integrating word embeddings does not improve enough the results when compared with to the same methods without using word embeddings.

However, the results obtained in (Mothe *et al.*, 2018) are poor Precision, Recall and F-score since unsupervised methods do not generalize well. And for this reason, in (Bennani-Smires *et al.*, 2018) address the extraction of key phrases with EmbedRank (an unsupervised method to automatically extract key phrases from a document taking advantage of sentence embeddings), achieving higher F-scores than graph-based systems. Also the diversity and coverage among the selected key phrases is increased by the method, and evaluated by a user study conducted by the authors. This research work was the first to present an unsupervised method based on phrase embeddings based for key phrase extraction. EmbedRank approach was compared with the corpus-independent methods: TextRank, SingleRank, WordAttractionRank, TopicRank and Multipartite, outperforming them.

Some recent research works (Gollapalli *et al.*, 2017), (Alzaidy *et al.*, 2019) stated that employing linear-chain Conditional Random Fields (CRF) can improve the performance over baseline models. Alzaidy *et al.* address key phrases extraction problem as a sequence labeling problem, by exploring a neural learning model by combining a bi-directional Long Short-Term Memory (Bi-LSTM) layer to capture the semantics of the sequential input text data with a CRF layer to model dependencies in the output producing a probability distribution over the tag sequence. This approach works better when constructing document level models than sentence level models, as the first captures more context and resulting more accurate. The best sequence of labels is found thanks to the Viterbi algorithm. This approach outperforms the baseline models.

2.4 Named Entity Recognition (NER)

The Named Entity (NE) task is a subtask of Information Extraction (IE) and was developed by the committee in the Sixth Message Understanding Conference in 1995. It became an important task in IE field since it aims to locate and classify named entities in a text into categories previously defined (e.g. People, Organizations, Geographic Locations) (Grishman and Sundheim, 1995). In this way, named entities can be used to represent text documents by highlighting the most relevant information and thus reducing information overload.

2.4.1 Supervised Approaches

Supervised methods use annotated data, it means input and output variables are given. However, create this annotated data is very time consuming, but the results of the models trained can be good enough in the task of NER.

In the task of biomedical NER, (Dingare *et al.*, 2005) present a Maximum Entropy Markov Model based system for identifying protein, DNA, RNA, cell line, cell type and newgene. By using a logistic regression model to classify each word and thanks to its ability to incorporate a variety of overlapping features. The system makes use of exhaustive local context using different features to represent the context of each word, considering the word itself, previous and next words, bi-grams, character n-grams up to a length of 6, and word shapes referring to mappings of words to describe attributes (if it contains capitalization, numerals, greek letter). POS tags from the GENIA gold standard are also integrated and the use of abbreviation matching to ensure consistency of labels is also considered. The system performance is evaluated in the BioCreative NER task (Ando, 2007) and the BioNLP NER task (Nigel Collier, 2004) to identify NEs in biomedical abstracts. The results obtained are high with f-score = 83.2% for BioCreative task and f-score = 70.1% for BioNLP task, achieving the state-of-the-art performance.

As seen in (Alzaidy *et al.*, 2019) approach, key phrases extraction problem can be treated as a sequence labeling problem by using conditional random fields same as used in (Finkel *et al.*, 2005) for the Stanford Named Entity Recognizer (NER). Named Entity Recognition (NER) labels sequences of words in a text that are names of things (persons, organizations, places, time, etc.) and where these are called classes. As Stanford NER implements the conditional random fields it is also known as a CRF Classifier. CRF are discriminative models suitable for prediction tasks where contextual information is considered to make a prediction as it captures sequential relations.

2.4.2 Semi-Supervised Approaches

Having manually tagged data is very expensive since it takes a long time to generate them. On the other hand, the amount of unlabeled texts is abundant and that is why it is necessary to use them in NER task. Semi-Supervised methods attempt to use some of the data labeled.

(Liao and Veeramachaneni, 2009) propose a simple semi-supervised approach in the task of NER by implementing Conditional Random Fields (CRF) since NER task is seen as a sequence classification problem. This algorithm uses independent evidence to produce labels of high precision. First, the algorithm uses a small amount of manually annotated data to obtain after a high-confidence data, and then it is used to discover low-confidence data by using independent evidence. And this process is repeated until no impressive performance is achieved. The final model obtained outperforms the initial supervised model with gold standard provided. In addition, this method can use an initial gold standard (CoNLL 2003 English NER) being from a different domain than the testing dataset (TF financial news corpus). When using data from different domains the accuracy achieved in this semi-supervised method is better than in supervised methods. Another advantage is that this approach achieves better results than supervised methods that requires large amounts of manually annotated data.

In the NER task in Tweets, Ritter *et al.* (2011) proposed a distantly supervised approach which applies Labeled LDA to leverage large amounts of unlabeled data in addition to large dictionaries of entities gathered from Freebase, and combining information about an entity's context across its mentions. This because classifying named entities in tweets is a difficult task since tweets contain a plethora of distinctive named entity types (Companies, Products, Bands, Movies and more), and almost all these types are relatively infrequent. On the other hand, tweets often lack sufficient context to determine an entity's type without the aid of background knowledge.

Mohamed and Oussalah (2014a) presented an approach by using the Wikipedia article info boxes where it has significantly reduced the classifier's processing time since the information inside the info box is structured. The proposed approach achieved a classification accuracy of above 97% with 3600 named entities and CoNLL-2003 shared task NER dataset used to validate the classifier's performance.

Certainly, large manually tagged data sets improve the performance of named entity classifiers, even though this involves a lot of time and human effort. Despite all this, there are labeled data sets for specific domains, which can strengthen the confidence of a classifier. However, it would be ideal if this knowledge could be transferred to other classifiers without having to create a specific corpus for each domain. In this way Hao *et al.* (2021) they propose a semi-supervised disentangled framework for transferable NER, capable of extracting and using domain-invariant and domain-specific information to predict named entities. To achieve disentanglement the authors employ three mutual information regularization terms and domain tagging supervision. The method they propose consists of 3 parts: input embedding with word-level and character-level information, disentanglement, and label prediction. And it shows optimal performance compared to other base methods such as: In domain, Init tuning, Multi-task learning, Layer Adaptation, Cross-Lingual Transfer

Learning, Multi-Task Cross-Lingual. It also proves to be superior in all transfer directions, managing to be generalizable to various domains. Several experiments were done considering only 20% of the data set, and gradually increasing until reaching 100%, always offering the best results in F-1 Score compared to the other methods.

2.4.3 Unsupervised Approaches

Unsupervised methods use no manually-annotated data and use ranking techniques to classify data. However, they are computationally complex models in comparison with supervised methods and less accurate, but no limited to specific domains.

In this sense, Nadeau *et al.* (2006) present a NER system that does not require manually tagged training data. This system consists of two stages or modules, where it first automatically generates a long list of named entities using the named entity extraction algorithm proposed by Etzioni *et al.* (2005) and then by resolving three problems (entity-noun ambiguity, entity boundary detection and entity-entity ambiguity) through heuristics addressed in (Mikheev, 1999), (Palmer and Day, 1997) and (Petasis *et al.*, 2001) respectively, achieving disambiguation of named entities. The system is evaluated with the standard corpus *MUC-7 Enamex* (Chinchor and Robinson, 1997) containing three classical named entity types (person, location and organization), in addition to one new corpus including a new class (car brands). Heuristics applied to address the problem of named-entity ambiguity improves the performance considerably in comparison with the first stage. The system achieves good enough results to be no supervised, however it is not competitive with the most complex supervised systems.

Munro and Manning (2012) present an unsupervised approach that learns to recognize named-entities in parallel corpora that does not need to be easily alignible. The proposed system first produces seed candidates by calculating the edit likelihood deviation (selecting only the word/phrase pairs with the highest and lowest values). Then, using the seeds the system learn context, word-structure and alignment models. And finally Stanford NER predictions were added as features in the final model to allow optimal weights. The system is evaluated with the parallel corpus of English and Haitian Krèyol text messages used in the 2010 Shared Task for the Workshop on Machine Translation. The obtained results in this unsupervised method are strong enough with F-measure= 0.861 for English language and F-measure = 0.846 for Krèyol, however the linguistic nature of the language pairs could affect the performance of the method.

Certainly exists the necessity to design and build models capable of transferring NER knowledge from one language to another, since not all languages have labeled data sets, and building these data sets is very time-consuming. The most recent research in this area has led to proposing really complex and very effective methods, given the lack of manually labeled resources for many languages. Bari *et al.* (2020) present an unsupervised cross-lingual NER model capable of transferring NER knowledge from one language to another. This means that there is no need to resort to external resources such as dictionaries or domain-specific tagged data sets. This model is based on the CO theory, about how it is that humans learn a second language, in which there is a transfer of knowledge from the base language to the new language, associating the meanings of the original language with the new words. The way this model works is by promoting knowledge sharing through cross-lingual mapping, making use of a common embedding matrix of the two languages. It also uses adversarial training which provides an initial word-level mapping, and after that, a fine-tuning is applied iteratively to improve the mapping. In addition to this, an augmented fine-tuning is applied, to find a relationship between the words as a whole and avoid erroneous labels that can generate noise in each iteration, since larger named entities can be affected by noise than named entities with few words. This model was tested with 5 different languages and proved to have excellent results over previous models from the scientific community. The authors emphasize that their model is the first to provide a completely unsupervised solution with real results, which is why it is considered the state of the art in unsupervised methods for the recognition of named entities.

2.4.4 Neural Network Approaches

In Dernoncourt *et al.* (2017) a tool based on ANN to easy recognizing named entities is proposed, it is called NeuroNER. It implements the LSTM model, that is a variant of RNN. The NeuroNER engine contains three layers: *Character-enhanced token-embedding layer*, *Label prediction layer* and *Label sequence optimization layer*. This tool allows to train the ANN model and evaluate the performance obtained by the created model, facilitating the annotation, training and prediction to anyone, resulting a free source to be used online [1].

However, rule-based systems are time consuming to be developed and generally works well only for specific domain named entities. Artificial Neural Networks (ANN) models have improved the performance on NER task, thanks to their ability to learn effective features directly from the training dataset, instead of handcrafted features developed for an specific domain. Kuru *et al.* (2016) considered a sentence as a sequence of characters instead of a sequence of words and presented a character-level tagger for language-independent NER. With this idea and using a stacked bidirectional LSTM model for encoding patterns on the sequential data, the character level tag probabilities produced by the LSTM are passed to

[1]http://neuroner.com/

a Viterbi decoder to convert them into word level tags. This model has the advantage of considerably reducing the vocabulary size compared with word-level models, at the same moment that computational cost is reduced. The results obtained in this research work perform similar to the state of the art NER models.

Some recent approaches that implements deep learning models have achieved the state of the art performance in NER task. Yadav *et al.* (2018) presented a NER model combining character and word-level information with specific learned representations of prefixes and suffixes of the words, allowing for a better semantic representation of words improving the meaning understanding of words. This method is language-independent and outperforms any other RNN models using character and words levels, this is how the model achieved the state of the art performance.

There exists surveys focused on NER systems, some of them covering a variety of supervised, semi-supervised, unsupervised and neural network methods (Sharnagat, 2014), others covering domain specific NER systems (Shaalan, 2014), (Etaiwi *et al.*, 2017) and (Patil *et al.*, 2016), and the most recent of them a comprehensive survey of deep neural network architectures for NER highlighting the improvements achieved by neural networks (Yadav and Bethard, 2019).

Deep Learning Approaches

(Akbik *et al.*, 2019) have moved from the original "one word, one embedding" paradigm to use contextualized embedding models, where different embeddings are produced for the same Word depending on its context. In this work the authors try to address the issue of underspecified embeddings that causes incorrect classification of named entities. So the approach dynamically aggregates contextualized embeddings of each unique string, and then a pooling operation is implemented to represent a global Word representation from all contextualized instances that authors use in combination with the current contextualized representation as new Word embedding. Therefore, this approach produces representations of words that change as new instances of the same word are found in the training data. First, the algorithm embed a Word in a sentence context and aggregate the produced embedding to the memory for this Word. Then, the pooling operation is applied overall contextualized embeddings for that Word in the memory to create the pooled contextualized embedding. Finally, this last embedding is concatenated with the original contextual embedding. So, the final pooled contextualized embedding contains local and global interpretations. The proposed method is evaluated in the CoNLL-03 shared task, and the WNUT-17 task on emerging entity detection, and is compared with BERT approach (Devlin *et al.*, 2018), the semi-supervised multitask learning approach (Clark *et al.*, 2018), the ELMo word-level language modeling approach (Peters *et al.*, 2018), and the best published numbers for WNUT-17 (Aguilar *et al.*, 2019). The proposed method present better results than the other implementations, indicating that the method is perfect for NER task.

In the field of Biomedical research the task of Named Entity Recognition has become so important, however traditional NER methods use pre-defined features to capture specific properties of named entities, which is expensive. In this sense, (Habibi *et al.*, 2017) present a generic method based on deep learning and word embeddings which outperforms state-of-the-art entity specific NER tools. The architecture of this classifier system is made up of three main layers. The first layer is the embedding layer, where the input is a raw sentence and produces an embedding for each word of the sentence. Then the resulting sequence of embeddings is used in the second layer, which is a bi-directional LSTM-layer, generating a more detailed representation of the input. Finally, the third layer receives that refined representation in a CRF-layer where a final output will be obtained by using the Viterbi algorithm. All these layers conform a neural network that can be trained by propagation. In this work authors are interested in recognizing five types of named entities: *genes/proteins*, *chemicals*, *diseases*, *cell lines* and *species*. The performance of their classifier systems is compared with five baseline NER tools and a CRF classifier, and results are evaluated in terms of *precision*, *recall* and *F-1 score*. The results obtained by the authors approach stands out considerably above the others in F-1 score. Their method manages to tag precisely single-token mentions more than on multi-token entities, but it also can tag precisely multi-token entities, without relying on any post-processing rules. For this reason, the costly development of domain-specific NER tools could be minimized.

After the study about the classification of texts, we realized that this can be carried out through key concepts. These concepts or key phrases can be extracted through the use of methods such as Naive Bayes, Support Vector Machines, Conditional Random Fields, among others. These same methods or techniques are used for the recognition of named entities, for which we will use these principles to develop a classifier model with the ability to identify the most relevant concepts in texts of our interest.

Chapter 3

Named Entities

Since our case study needs to identify key concepts with specific characteristics of our set of texts, we have proposed and developed a web-based tagger system. Thanks to our tagger system, it is possible to identify the most relevant key concepts or phrases inside Job Descriptions. At the same time, the system facilitates tagging through suggestions learned by previously tagged key phrases. This allows key phrases manually tagged by people to be highly reliable since as humans we can naturally distinguish between one category and another. Another great advantage of our system is that it allows us to create or define our classes or categories. So it lets us treat practically any type of text. Finally, it permits us to export the tagged texts to a structured format, which can be processed by classifying models such as Stanford's. In this way, we achieve a high-quality training set and save time by automatically converting to a structured format. Without a doubt, this is a significant contribution that we make, as we facilitate the labeling of texts.

3.1 Introduction

The Named Entity Recognition (NER) task is a subtask of information extraction (IE) and was developed by the committee in the Sixth Message Understanding Conference in 1995. It became an important task in IE field since it aims to locate and classify named entities in a text into categories previously defined (e.g. People, Organizations, Geographic Locations). In this way, texts can be represented by their named entities (R. Grishman, 1996). Although, sometimes many of the NEs can be ambiguous to be classified in more than one class, e. g. the automotive company created by Henry Ford in 1903, where "Ford" can be referred to many entities (Name, Company, etc.) (Mohamed and Oussalah, 2014b).

On the other hand, NER systems require a large amount of highly accurate training data to perform well at the task named entities recognition (S. Tardif, 2009). In this way, excellent training data can be achieved by human feedback, since humans can easily differentiate from one context and another, assigning the correct tag to each named entity in the texts.

That is why our set of texts (Job Descriptions) has been manually tagged highlighting the most representative named entities of each text. These named entities are classified into 6 categories: *Role, Knowledge, Skill, Character, Responsibility* and *Talent*. However, these key phrases that represent our texts range from 1 to 14 words in length and are very specific to the field of Information Technology (IT), so we will not refer to them as named entities but as "key phrases", since our objective it is to identify these specific key phrases that represent our documents and not to identify named entities in general as described above.

3.2 N-grams

An n-gram is a subsequence of n elements of a sequence of words given in a text and is widely used in the study of natural language. The way in which the grams are extracted depends on the area that is being studied as well as the objective in mind. In the case of studying natural language, grams are a technique used that allows machine learning algorithms to extract data from text strings.

3.2.1 N-gram language model

An n-gram model is a type of probabilistic model that allows a statistical prediction to be made of the next element of a certain sequence of elements that has occurred so far. An n-gram model can be defined by a Markov chain of order $n-1$, due to its relative simplicity to increase the study context by increasing the size of n.

The Markov model is a special type of discrete stochastic process in which the probability of an event occurring depends only on the immediately preceding event. A Markov chain is a sequence $X_1, X_2, X_3, ...$, of random variables. The domain of these variables is called the state space; the value of X_n is the state of the process at time n. If the conditional probability distribution of $X_n + 1$ in past states is a function of X_n by itself, then:

$$P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, ..., X_2 = x_2, X_1 = x_1) = P(X_{n+1} = x_{n+1} | X_n = x_n) \tag{3.1}$$

where x_i is the state of the process at time i (David. C. Lay and McDonald, 2016).

3.2.2 N-grams used

In our case, we have a corpus composed of plain text Job Descriptions in the field of IT. And these kind of documents can be represented by the classes: *Role, Knowledge, Skill, Character, Responsibility* and *Talent*. However, we want to know what n-grams are used most in each class. To achieve this we have measured the length of the key phrases (n-grams) that make up each class, immediately we grouped them by n size, and then we counted their frequencies in the corpus. Figure 3.1 shows the distribution of the existing n-grams in each class.

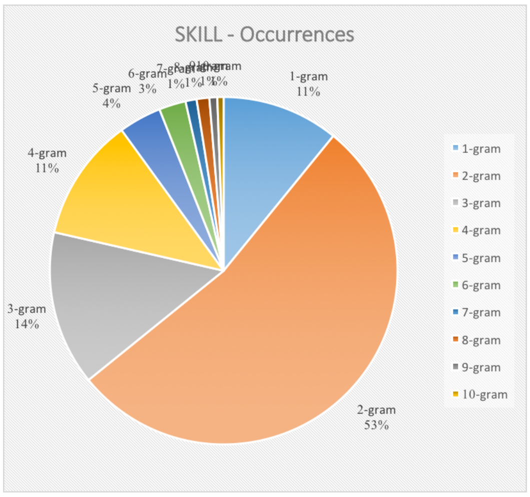
With the graphic representations of the n-grams can be seen that most of the classes are mainly made up of bigrams, with the *Role, Skill* and *Knowledge* classes being the most common. However, the *Knowledge* and *Character* classes are mainly represented by unigrams, with a large percentage difference from the rest of the classes. That means that these two classes are characterized by being unigrams and bigramas.

On the other hand, it is possible to observe that the trigrams occupy the third position in representing each of the classes, being the only one more balanced group based on the number of occurrences. Four-grams represent from 6% to 12% of the composition of each class, ranking fourth in the main named entity sizes. As the rest of the n-grams have such a low presence in the classes, with the exception of the *Responsibility* class, being the only class that represents a less disproportionate distribution of key phrases of short and long length. Furthermore, the *Responsibility* class has the longest n-grams existing in the corpus, being something characteristic of this class.

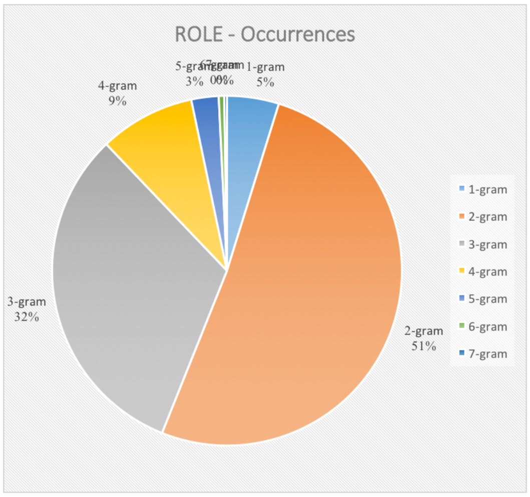
3.3 Part-of-Speech (POS)

In grammar, a part of speech is a category for words that have similar grammatical properties, in this way those words that belong to the same category show a very similar syntactic behavior. So based on their different uses, the words are classified into different types of Part of Speech. The Parts of speech found in the English language are:

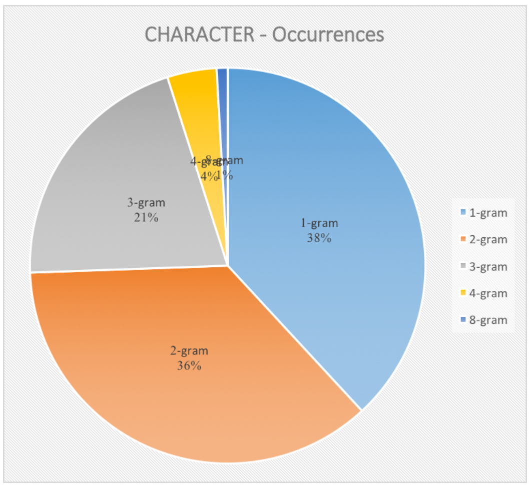
- **Noun**: is a word that refers to a thing *(book)*, a person *(Betty Crocker)*, an animal *(cat)*, a place *(Omaha)*, a quality *(softness)*, an idea *(justice)*, or an action *(yodeling)*. It's usually a single word, but not always: *cake, shoes, school bus,* and time and a half are all nouns.
- **Pronoun**: is a word that is used instead of a noun or noun phrase. Pronouns refer to either a noun that has already been mentioned or to a noun that does not need to be named specifically. The most common pronouns are the personal pronouns, which



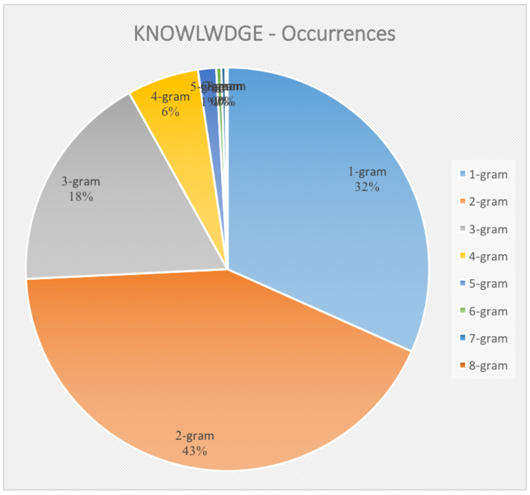
(a) Skill



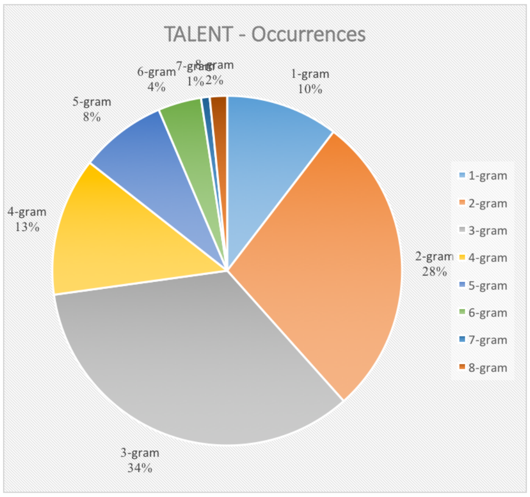
(b) Role



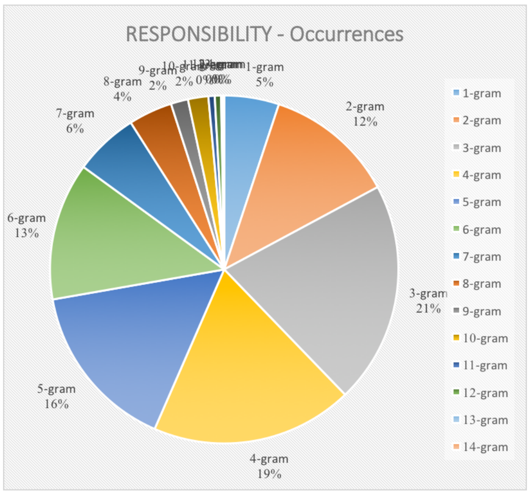
(c) Character



(d) Knowledge



(e) Talent



(f) Responsibility

Fig. 3.1 Distribution of n-grams in each class

refer to the person or people speaking or writing (first person), the person or people being spoken to (second person), or other people or things (third person).

- **Verb**: verbs are words that show an action *(sing)*, occurrence *(develop)*, or state of being *(exist)*. The basic form of a verb is known as its infinitive. The forms call, love, break, and go are all infinitives.
- **Adverb**: adverbs are words that usually modify verbs. They may also modify adjectives, other adverbs, phrases, or even entire sentences.
- **Adjective**: describe or modify —that is, they limit or restrict the meaning of— nouns and pronouns. They may name qualities of all kinds: *huge, red, angry, tremendous, unique, rare, etc*.
- **Preposition**: A preposition is a word —and almost always a very small, very common word— that shows direction (*to* in "a letter to you"), location (*at* in "at the door"), or time (*by* in "by noon"), or that introduces an object (*of* in "a basket of apples"). Prepositions are typically followed by an object, which can be a noun *(noon)*, a noun phrase *(the door)*, or a pronoun *(you*.
- **Conjunction**: is an uninflected linguistic form that joins together sentences, clauses, phrases, or words. Some common conjunctions are *and*, *or*, *but*, and *although*.
- **Interjection**: is a word or phrase that is grammatically independent from the words around it, and mainly expresses feeling rather than meaning. Some examples are: *Alas! ouch! phooey! ugh! Uh-oh! um! wow2!*
- **Numeral**: is a conventional symbol that represents a number.
- **Article**: is used with nouns to specify grammatical definiteness of the noun (such as *"a", "an",* and *"the"*). Articles are frequently considered part of a broader category called **determiners**, which contains articles, demonstratives (such as *"this"* and *"that"*), possessive determiners (such as *"my"* and *"his"*), and quantifiers (such as *"all"* and *"few"*) (Merriam Webster, 2020).

3.3.1 POS Tagging

Part-of-speech tagging is the process of marking a word in a text with a tag carrying the corresponding information to a particular part of speech, based both on its definition and its context (its relation to adjacent and related words in a phrase, sentence or paragraph) (Sketch

Engine, 2020). This tagging is done using computational algorithms that associate discrete terms, as well as hidden parts of the speech, through a set of descriptive labels. These POS tagging algorithms are divided into two types: rule-based and stochastic.

However, doing a part-of-speech tagging is not so easy since some of the words to be tagged can represent more than one POS depending on their use and context. So it is not uncommon for some words to be ambiguous.

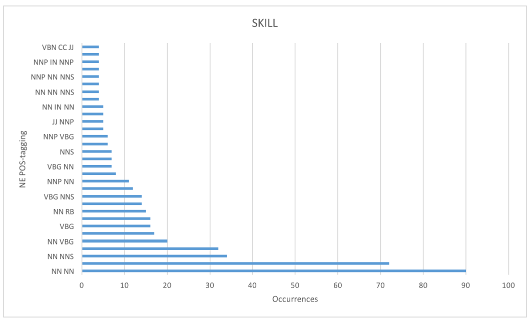
POS Tagging of the corpus

POS tagging was applied to the key phrases contained in the corpus of Job Descriptions concerning the IT field used in this investigation. To achieve the task, the tagger provided by the *Natural Language Toolkit (NLTK)* library was used through the python programming language. This tagger receives and processes a sequence of words (previously tokenized), adding to each of them its POS tag. So for each of the 3336 key phrases in the corpus, the pos_tag() method was called and then all the sequences found in the key phrases were grouped (and the occurrences of each one were counted).

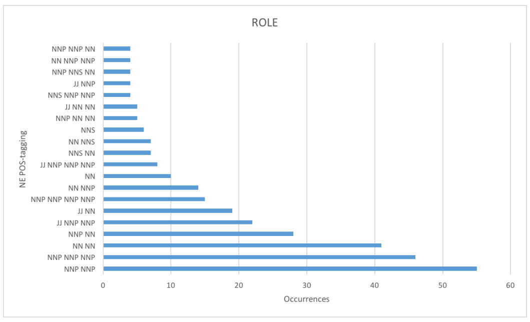
The reason for using the default function POS tagset and not the universal POS tagset is that the first one allows more detailed classification. While the universal POS tagset only has 12 categories, the default POS tagset offers the possibility of tagging with 35 different classes, making the structure of a named entity more precise and thus recognizing its similarity with another. The 35 classes that make up the default tagset are as follows (Bird *et al.*, 2009):

- **CC** coordinating conjunction
- **CD** cardinal digit
- **DT** determiner
- **EX** existential there
- **FW** foreign word
- **IN** preposition/subordinating conjunction
- **JJ** adjective ‘big’
- **JJR** adjective, comparative ‘bigger’
- **JJS** adjective, superlative ‘biggest’
- **LS** list marker 1)

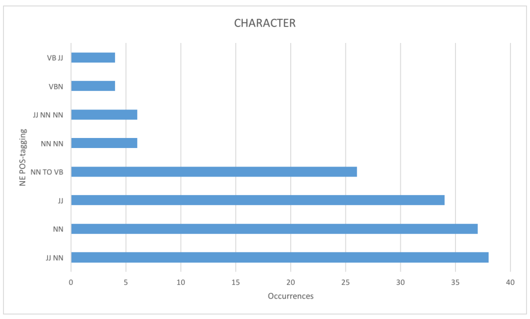
- **MD** modal could, will
- **NN** noun, singular ‘desk’
- **NNS** noun plural ‘desks’
- **NNP** proper noun, singular ‘Harrison’
- **NNPS** proper noun, plural ‘Americans’
- **PDT** predeterminer ‘all the kids’
- **POS** possessive ending parent’s
- **PRP** personal pronoun I, he, she
- **PRP$** possessive pronoun my, his, hers
- **RB** adverb very, silently
- **RBR** adverb, comparative better
- **RBS** adverb, superlative best
- **RP** particle give up
- **TO** to go ‘to’ the store
- **UH** interjection
- **VB** verb, base form take
- **VBD** verb, past tense, took
- **VBG** verb, gerund/present participle taking
- **VBN** verb, past participle is taken
- **VBP** verb, sing. present, known-3d take
- **VBZ** verb, 3rd person sing. present takes
- **WDT** wh-determiner which
- **WP** wh-pronoun who, what
- **WP$** possessive wh-pronoun whose



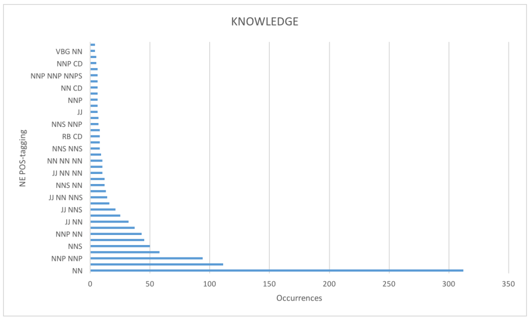
(a) Skill



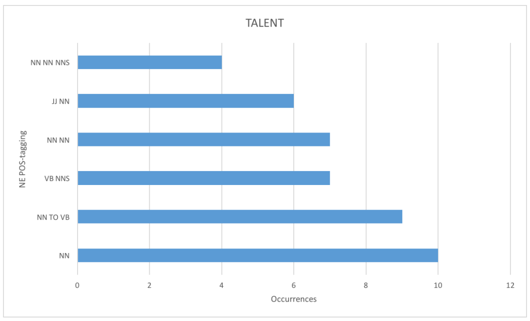
(b) Role



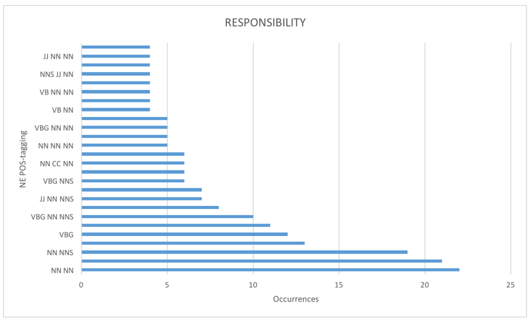
(c) Character



(d) Knowledge



(e) Talent



(f) Responsibility

Fig. 3.2 Most representative POS-tag for each class

- **WRB** wh-adverb where, when

In Figure 3.2 it can be seen which are the POS-tags of the most representative key phrases for each class. In the class ***Character*** mainly unigrams are used (*NN* and *JJ*), while the most present bigrams have the structure *JJ NN*, and most of the trigrams have the structure *NN TO VB*.

While the class ***Knowledge*** is strongly represented by unigrams of the *NN* form, followed by the *NNS* form, and bigrams with the structures *NN NNS*, *NNP NNP*, *NN NN*, *NNP NNS* and *NNP NN*. So it would be rare to find a trigram in this class, or a higher-order n-gram.

On the other hand, the class ***Role*** is made up mostly of bigrams and trigrams, so it would be very rare for an named entity in this class to be just one word. The main structures of bigrams in this class are: *NNP NNP, NN NN, JJ NN* and *NN NNP*, while the main structures of trigrams are: *NNP NNP NNP* and *JJ NNP NNP*, and for first time four-grams play a representative role in a class, having the structure: *NNP NNP NNP NNP*. Something important to mention in this class is that its key phrases are mainly made up of proper nouns in the singular form, being something characteristic of this class.

The class ***Skill*** is mainly made up of bigrams, which is something that characterizes the class and it is worth looking at its component structures: *NN NN, JJ NN, NN NNS, NN VBG, JJ JJ, NN RB, NNP NNP, VBG NNS, JJ NNS* and *NNP NN*, where nouns and adjectives are the most used POS tags and resulting characteristic of their key phrases.

The class ***Talent*** is made up of unigrams, bigrams and trigrams, with the bigrams being the most representative and with the following structures: *VB NNS, NN NN* and *JJ NN*. The main trigrams have the structures: *NN TO VB* and *NN NN NNS*, and unigrams are represented with the form *NN*. This class strongly uses nouns, which are part of any of its key phrases.

Finally, the class ***Responsibility*** is mainly made up of trigrams that have the following structures: *NNP NN NNS, NN NN NNS, VBG NN NNS, VBG JJ NN, JJ NN NNS, NN CC NN, VBG JJ NNS, NN NN NN, VB JJ NNS, VBG NN NN* and *NNP JJ NNS*. And those trigrams are formed with singular and plural nouns, adjectives and verbs in gerund (present participle). The main bigrams that conform the class have the structures: *NN NN, NN NNS, JJ NN, VBG NNS* and *NNP NN*, where it is possible to realize that these bigramas are formed in the same way as the tigrams (with nouns, adjectives and verbs in gerund). And as for the unigrams they are characterized for being *NN* and *VBG*, concluding that the use of verbs in present participle is something characteristic of the class *responsibility*.

In this chapter, we managed to map all the key phrases as n-grams, so that we could analyze how each of the classes that we defined for the Job Descriptions behaves. Another interesting finding is to know the POS structures that characterize each of our 6 previously defined categories or classes. The study of these features is important because it will help us to define rules later. Thanks to this we could propose a competitive classifier model.

Chapter 4

Semantic Similarity

One of the main contributions made in this chapter is to propose a way to generate new key phrases from an original set. This through synonyms for each of the words that make up each key phrase. Of course, not all the newly generated key phrases could refer to the same meaning as the original ones, so we propose a selection process to only preserve those key phrases that express the same concept as the original key phrases. Our selection process considers the results returned by the Google search engine for a new key phrase as the first filter, and it is important to mention that this stage is crucial since through it we know if the newly generated key phrase is used by people around the world. As we know, the Google search engine has access to practically all electronic texts on the Internet, so it is easier to discern if a key phrase is used frequently in texts or if its use is sporadic or even null. In addition to this filter, a stage is used where a semantic similarity measure is made between the original key phrase and the new key phrases. Finally, a filter is made where those key phrases that do not comply with the POS structures of those observed in the previous chapter are discarded.

4.1 Introduction

When comparing one text with another it is possible to measure its similarity through different techniques, for example, one text can be similar to another if both have high occurrences of the same words (that is, the word that is in text A is also found in text B). However, two texts could be talking about the same topic without using the same words and then it would not be useful to compare them syntactically since syntactically speaking they are not similar, for this it is necessary to find a semantic similarity between both texts (which can use different words to refer to the same topic).

Semantic similarity is a metric determined on a set of terms, where the distance between them is based on the similarity of their meanings or semantic contents. In this way it is defined that two entities are similar if: (i) both belong to the same class, (ii) both belong to classes that have a common primary class, or (iii) one entity belongs to a class that is a primary class to which the other entity belongs. Furthermore, two relationships are similar if: (i) they both belong to the same class, (ii) they both belong to classes that have a common principal class, or (iii) a relation belongs to a class that is a principal class to which the another relationship belongs (Paolozzi Stefano and Grifoni, 2009). There exists different different semantic measures which are mathematical mechanisms used to calculate the strength of the semantic relationship between concepts, and this relationship is normally represented with numerical values in a range from 0 to 1, where 0 is not at all similar and 1 means completely similar (Harispe *et al.*, 2017).

It is important to note that semantic similarity differs from semantic relatedness since the latter refers to any relationship between two concepts such as the words neighboring the concepts and that are part of their context, while the semantic similarity is limited to concepts that mean the same to the originals (Feng *et al.*, 2017). For example, *"engineer"* is a concept similar to *"architect"*, but is related to *"software"* and *"methodologies"*.

4.2 Semantic Measures

The objective of semantic measures is to evaluate the similarity or relationship between two entities taking into account their meanings. For example, *"serve"* and *"help"* are semantically more similar than *"serve* and *"server"* even though these two share a similar high syntactic structure. A semantic measure can be defined as

$$\sigma_k : E_k \times E_k \to \mathbb{R}^+ \tag{4.1}$$

with E_k representing the set of elements of type $k \in \mathbb{K}$, and $\mathbb{K}$ representing the different types of elements that can be compared with respect to their semantics, for example, $\mathbb{K} = \{words, concepts, sentences, texts, etc...\}$ (Harispe *et al.*, 2017).

Semantic measures play an important role in a wide variety of NLP applications, and therefore there is a variety of them that is mainly divided into two categories according to Harispe *et al.*:

- **Corpus-based measures** allow the comparison of language units (words, concepts, sentences or texts) from the analysis of unstructured or semi-structured texts, and are mainly only based on statistical analyzes of the use of words in texts (based in the analysis of co-occurrences of words and the linguistic contexts in which they appear). These measures are based on a strategy that will be used to capture the meaning of a word (this meaning is seen as a function of its use in a semantic space built from a set of texts). And depending on the strategy adopted to characterize the meaning of a word, and to represent the semantic space in which this meaning is defined, it is how a specific canonical form will be selected to represent a word (this canonical form corresponds to a data structure that is expected to encompass evidence of the meaning of the word).

- **Knowledge-based measures** are designed to compare entities defined in ontologies which are formal definitions of types, properties, and relationships between entities that fundamentally exist for a particular domain of discourse (e.g., the field of computing). These measures can also be used to compare units of language (sentences or texts) but they are not limited to just that, so they can be used to make the comparison of any piece of knowledge formally described and covering a wide variety of elements (concepts , genes, people, music, etc...).

4.2.1 Wu&Palmer similarity measure

Wu&Palmer similarity is a Knowledge-based measure where we have used the semantic dictionary WordNet (Miller, 1995). This measure returns a value denoting how similar are two concepts, considering the depth of the two concepts in the taxonomy and that of their Least Common Subsumer (most specific ancestor node) (Bird *et al.*, 2009). Wu&Palmer similarity measure can be defined as:

$$Sim_{w\&p}(C1,C2) = \frac{2*N}{N1+N2+2*N} \tag{4.2}$$

Where C1 and C2 are two concepts in the taxonomy, N1 and N2 are the distance that separates C1 and C2, respectively from the specific common concept, and N is the distance that separates the most specific common ancestor of C1 and C2 from the root node (Slimani, 2013).

Several parents can be shared by C1 and C2 considering multiple paths, however, the LCS does not necessarily appear on the shortest path connecting C1 and C2, as its name says it is the deepest common ancestor in taxonomy. When there are multiple candidates for the LCS, then the candidate whose shortest path to the root node is the longest will be selected. And when the LCS has multiple routes to the root, the longest route is used for the purpose of the calculation. So, can be said that two concepts tend to be more similar as depth increases.

4.3 Selection process for new key phrases

Starting from the key phrases that were manually tagged in the set of job descriptions, new phrases were generated which were variants with synonyms of the first ones. For example, from the original key phrase “focus on multiple issues” seven new phrases are generated through synonyms “focusing on multiple issues”, “focussing on multiple issues”, “concentrate on multiple issues”, “focus on multiple issue”, “focus on multiple topic”, “focus on multiple subject”, “focus on multiple matter”. However, we can see that not all of these new phrases are optimal to refer to the original, and for that reason a selection process will be necessary to discard phrases that do not refer to the same concept. This selection process consists of 3 phases or selection tasks in order to discard the new phrases generated through synonyms, and these selection tasks help us to keep only the phrases that refer to the same idea as the original ones. The selection tasks are Google query results, Wu& Palmer similarity measure, and part-of-speech structure.

- **Google search results.** The first selection task used is based on the results returned by the Google search engine when a query is made for the new phrase with the exact terms (it means that Google searches for exactly that phrase on the web and not the words that compose it separately). If the results returned by Google are 0 then it means that this phrase does not exist on the web and then from there it is discarded, and if the results it returns are minimal (1 - 100) it means that this phrase, although it exists, is very little used (what which can mean that some user made a mistake when writing or that the phrase contains a new term) and then it must be analyzed by the following selection task.

- **Wu&Palmer Similarity.** The second selection task uses the similarity of Wu&Palmer, through which the similarity of each original key phrase has been compared with their respective new phrases through synonyms. If the original phrase is 20% to 100% similar to the new phrase, then the phrase is retained for analysis by the last selection task, otherwise, it is discarded.

- **Part-of-speech.** The third and final selection task refers to the part-of-speech, which is based on the expressions or structures found in the set of original key phrases of each class. So the POS structure of the new phrase is checked against the POS structure of the set of original phrases that belong to the same class and if it corresponds to any then it is preserved, otherwise, it is discarded.

In the end, a set of coherent and similar phrases to the original ones is obtained, and thus the variants in the graph structure can be expanded, now being a rich graph through synonyms.

4.3.1 Google results threshold selection

The thresholds of selection tasks one and two were determined through visual observation, for example for selection task number one, it is clear that if Google does not return a result, then the phrase does not exist, but through observation, it was possible to determine that those phrases that only occur less than 100 times on the web are meaningless, as can be seen in Table 4.1.

It can be seen that in the new key phrase *"provide production supporting"* the word *"supporting"* is not being used in the correct grammatical tense since it begins by speaking in the present tense *"provide"* and therefore *"support"* should also be present. Regarding the phrase *"furnish production support"* it is possible to observe that the word "furnish" is not in common use in this context since the closest synonym to *"provide"* according to the context is *"supply"*.

Then for the key phrase *"data entry skills"* it is possible to see that *"information entry skills"* yields very few results since when people speak in a computational context of entering information they do so referring to the specific term of the computer field *"data"* because *"information"* is a more general concept.

For the key phrase *"learn things quickly"* it is easy to see that there is a plural sense of learning things and not just one thing because otherwise, it would specify what is the thing required to learn, that is why *"learn thing quickly"* is discarded since it should speak in the plural. On the other hand, although *"speedily"* and *"promptly"* are synonyms of *"quickly"* they are not the best to be used within this context of the sentence, so there are not many appearances on the web.

Regarding the key phrase *"focus on multiple issues"* when using the word *"focussing"* with double *ss* it yields few results since it is more common to use it with an *s* in the United States. On the other hand, when the words *“topic”* and *“matter”* are used instead of *"issues"* the grammar is incorrect since these singular words go after the *“multiple”* quantifier, which requires a plural noun.

It can be seen that for the original key phrase *“understanding of typography”* the only phrase similar is *“understand of typography”* since *“intellect”* and *“reason”* lack a semantic meaning similar in context to *“understanding”*.

Table 4.1 Google results for new phrases

Original key phrase	New phrases	Google results
provide production support	supply production support	2130
	furnish production support	71
	provide production supporting	3
data entry skills	information entry skills	4
	data entering skills	86
	data entry skill	19600
learn things quickly	memorize things quickly	5850
	learn thing quickly	68
	learn things speedily	59
	learn things quick	1100
	learn things promptly	6
focus on multiple issues	focusing on multiple issues	25600
	focussing on multiple issues	7
	concentrate on multiple issues	5410
	focus on multiple issue	3210
	focus on multiple topic	3
	focus on multiple subject	13600
	focus on multiple matter	0
understanding of typography	reason of typography	3
	intellect of typography	0
	understand of typography	2230
work well under pressure	go well under pressure	8
	work good under pressure	2800
	process well under pressure	1
	make well under pressure	4
	work considerably under pressure	0
software development life cycle	software developing life cycle	674
	software development living cycle	0
	software development lifetime cycle	43

Table 4.2 Wu&Palmer similarity between original "key phrases" and new phrases

Original key phrase	New phrases	Wu&Palmer Similarity
graphic design	graphical design	0.5
	graphic designing	1.0
	graphic project	0.852
verifying	confirming	0.4
	validating	0.2
identify	key	0.166
	distinguish	0.8
	describe	0.153
captivate	beguile	0.181
	fascinate	0.285
Mac	macintosh	0.526
CPU products	processor products	0.56
	cpu merchandise	0.94
	cpu product	0.94
professional service	pro service	0.815
	professional servicing	0.5625
video editing	picture editing	0.576
	video edit	0.5
solution focussed	solution concentrate	0.583
	solution focus	0.576
C# services	C# service	0.852
	C# servicing	0.558

In summary, it can be said that for a new phrase to be considered, it must have high occurrences on the web, which means that its extensive use in existing textual information on the web supports its existence and common use in texts.

4.3.2 Wu&Palmer threshold selection

As for the Wu&Palmer measure, it is used by comparing each original key phrase with their respective phrases generated from synonyms, so that the similarity of each new phrase with the original is calculated. But we managed to see that only those similarities greater than 0.2 (20%) are important, and the rest are discarded as they do not present a strong semantic equivalence to the original one. Table 4.2 shows what is described before.

In the first example of the table, when you talk about the key phrase “graphic design” you can also refer to it as “graphical design”, then you can say that any similarity equal to or greater than 50% is trustworthy.

Table 4.3 Part-of-speech when using lowercase and uppercase for key phrases

Key phrase	Original POS	Lowercase POS
Digital Graphic Designer	NNP NNP NNP	JJ JJ NN
Adobe Creative Suite	NNP NNP NNP	RB JJ NN
IT Support	NNP NNP	PRP NN
degree in Informations Systems	NN IN NNP NNPS	NN IN NNS NNS
MS Sharepoint 2013	NNP NNP CD	NN NN CD

In the second example of the table, when the term "verifying" is used in a sentence, it can be replaced by any of the terms "confirming" or "validating" and although here the degree of similarity is less than 50%, these terms correctly replace the original so a degree of similarity greater than 20% is acceptable.

Now, in the third example of the table, the term "identify" can also be understood as "distinguish" having a high degree of similarity, but "key" and "describe" do not bear much relation to the original term representing a similarity below 20%. And the same is observed in the fourth example, where "captivate" and "fascinate" can be understood as similar with a percentage of 28.5% but the word "beguile" is no longer closely related to "captivate", being 18.1% similar.

From the above, it can be concluded that any similarity between two sentences of less than 20% represents a very weak relationship between the two and therefore it is not worth considering them for the set of new key phrases, because otherwise, they would be including phrases that they do not have a great semantic relationship to the original ones and the quality of the corpus would be affected.

4.3.3 POS threshold selection

This filter consists of only preserving new key phrases with a part-of-speech that is included in the POS list of the original key phrases.

Key phrases showed in Table 4.3 were discarded when applying the POS selection task, and this happens because these key phrases have different part of speech when using uppercase letter and when using lowercase. For that reason, both part-of-speech have to be considered when using this selection task (the part-of-speech of the original key phrase and the part-of-speech of the lowercase key phrase). That way, relevant POS structures will not be discarded.

4.4 Preserved key phrases

Once the selection tasks have been defined to select the new phrases to be contemplated within the corpus, there are the different combinations in which these selection tasks can be used to obtain the best phrase options. However, the Google selection task is the most important filter of them, since from that moment it will be validating that those new key phrases really exist on the Internet and are widely used by people. In this way, key phrases that are rarely used will be discarded, mainly due to typing errors. Sometimes users could have omitted or added extra letters at the time of writing them, or perhaps by non-native users of the language who may have mistakenly constructed a phrase where the words were not the best used. And even if there were several occurrences of those key phrases wrongly used by internet users, then the second most important filter or selection task should be the semantic similarity between the original key phrase and the new key phrase.

Algorithm 1: Algorithm employing the three selection tasks

```
Result: Selected_phrases = list
Result: Discarded_phrases = list
for each original key phrase as o_key_phrase do
    for each new phrase as p do
        if google_results(p) > 100 then
            if wu_palmer similarity between p and o_key_phrase > 0.2 then
                p_class = get_class(p);
                if part_of_speech(p) in pos_structures(p_class) then
                    Selected_phrases.append(p);
                else
                    Discarded_phrases(p);
                end
            else
                Discarded_phrases(p);
            end
        else
            Discarded_phrases(p);
        end
    end
end
```

However, the POS selection task is optional since the most important selection tasks are the first two mentioned above. And in the case of using the POS selection task, all key

Table 4.4 Key phrases manually discarded from the set of key phrases preserved by the selection process

Original key phrase	Preserved key phrase	Manually discarded
customer facing	client facing customer confront	customer face
handle various issues	handle diverse issues handle various subject	handle various subject
learn new technologies	learn new engineering hear new technologies	hear new technologies
use AutoCAD	utilize autocad apply autocad employ autocad purpose autocad	purpose autocad
learn things quickly	discover things quickly hear things quickly learn things rapidly	hear things quickly

phrases need to be converted to lowercase and consider both POS structures. The Algorithm 1 describes the combination of selection tasks used with specific conditions.

Although the use of synonyms allowed creating many phrases semantically similar to the original ones, not all of them were of high quality (since the grammatical form was incorrect in many cases). But it is possible to observe that through the selection process these grammatical errors were discarded.

4.4.1 Manual selection

This selection process help us to keep only the most important key phrases, however a last filter is necessary to guarantee the maximum quality of new key phrases. This filter must be evaluated manually by a human, discarding those key phrases that lack a logical sense of communication.

A total of 9,689 new key phrases were generated through the use of synonyms and after using the first selection task of this process, only 4,161 key phrases were preserved. Which means that 57.05% of new key phrases were discarded. Of the 4,161 key phrases preserved, only 74 key phrases were discarded in the second selection task of the process. In the last filter, 612 key phrases were discarded, which leaves a total of 3,475 key phrases preserved. This means that only 35.86% of the new key phrases are preserved after the selection process, as seen in Figure 4.1.

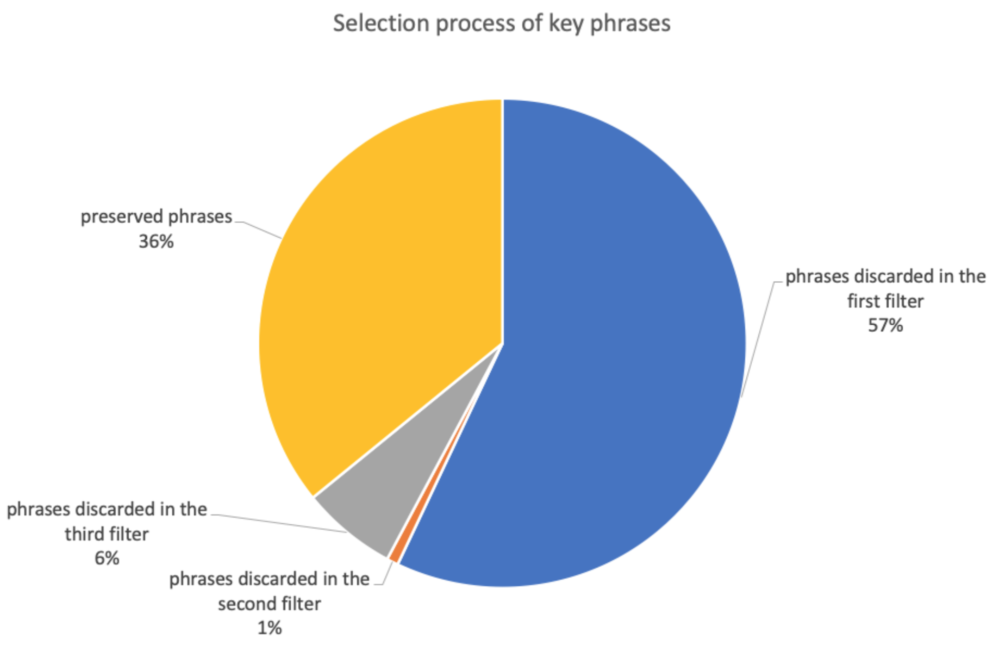


Fig. 4.1 Discarded key phrases in each selection task

However, of the key phrases preserved after this process, there are some that lack a meaning similar to the original, as it can be seen in Table 4.4. The created key phrase *"customer face"* is different from *"customer facing"*. The first phrase refers to a person's face, while the second phrase refers to dealing directly with people who buy products or services. Another example is with the phrase *"learn things quickly"* which is different in meaning from the phrase *"hear things quickly"*, where the first refers to the ability to learn and the second refers to the action of listening.

After the manual selection a total of 503 key phrases were discarded, leaving only 2,972 key phrases.

4.5 Experiment

To measure the quality of the new set of key phrases created from the originals, two classifiers will be trained with the Stanford classifier (Finkel *et al.*, 2005) . The first classifier will only contain the job descriptions tagged with their original key phrases, while the new key phrases kept after the selection process will be added to the second classifier as illustrated in Figure 4.2. This expansion of key phrases is done for each job description used for classifier training. At the end of each original job description, new key phrases that are semantically similar to those originally tagged will be added.

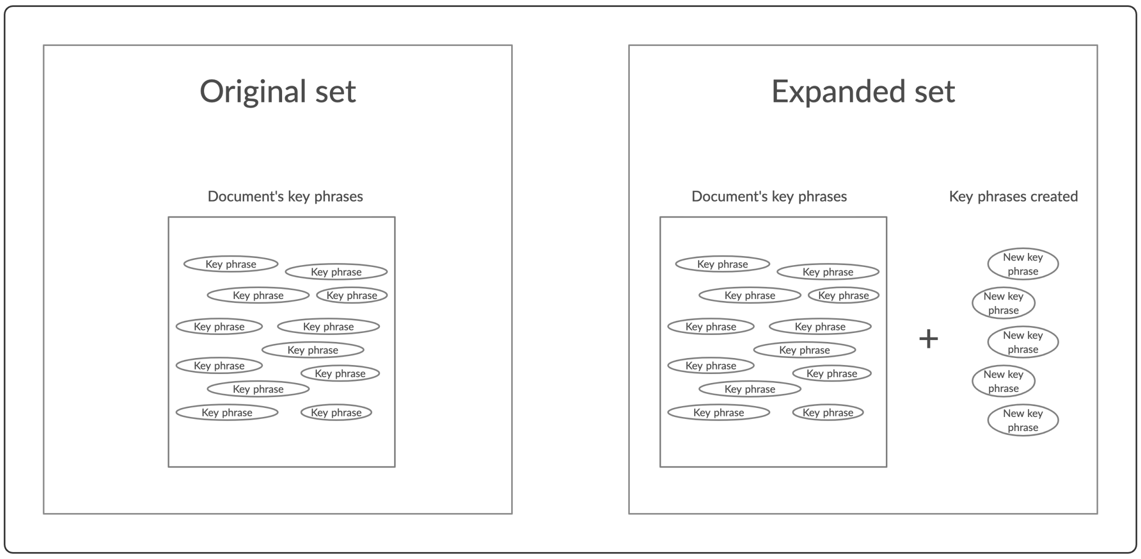


Fig. 4.2 The original set is expanded by adding new created key phrases

To know how accurately the original model versus the expanded model will be, the V-fold cross-validation technique will be used. For this work, the 70% of original job descriptions were taken to train the first classifier and the remaining 30% to evaluate its performance. These steps were executed 10 times by randomly taking 70% each time and using the remaining 30% in each case for its evaluation. For the second classifier the same process was used but using the expanded set. The results for the experiments are measured in terms of Precision, Recall and F-1 score. Although, it is important to comprehend four main metrics before explaining how these measures are defined. The metrics needed to calculate the 3 measures, are explained as follows:

- **True Positive (TP):** It means that the class predicted by the classifier was correct.
- **False Negative (FN):** It means that the classifier could not assign or predict a class, however, the word originally does have a class or label.
- **False Positive (FP):** It means that the classifier assigned a label or class, but the word did not originally have a label.
- **True Negative: (TN):** It means that the classifier could not predict a class or label for the word, but the word did not originally have a class or label either.

Once we distinguish these four metrics, it is easier to understand the 3 measures which are expressed as follows:

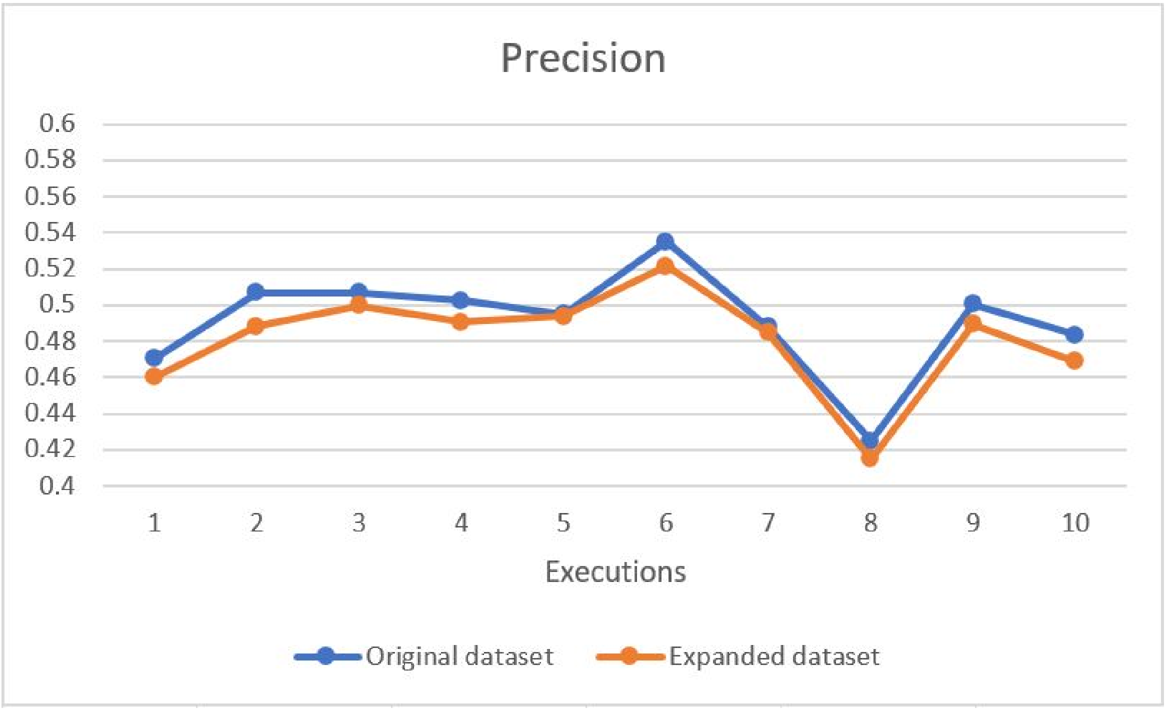


Fig. 4.3 Results obtained for both models, in 10 executions for the Precision measure

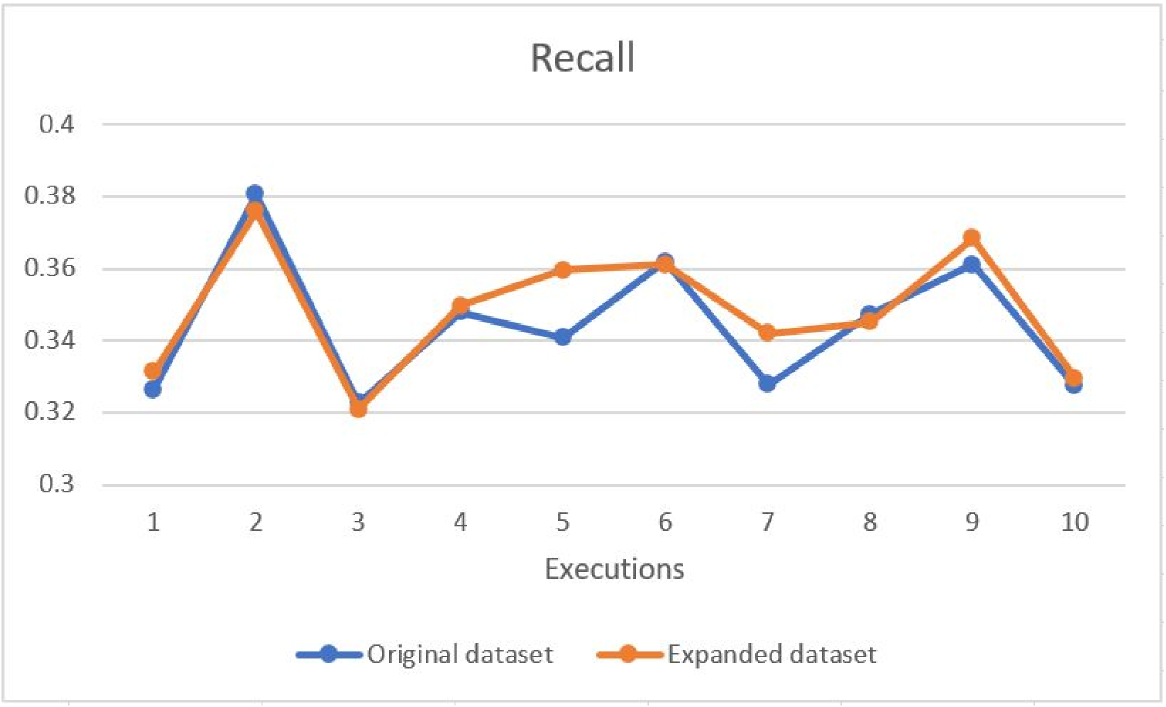


Fig. 4.4 Results obtained for both models, in 10 executions for the Recall measure

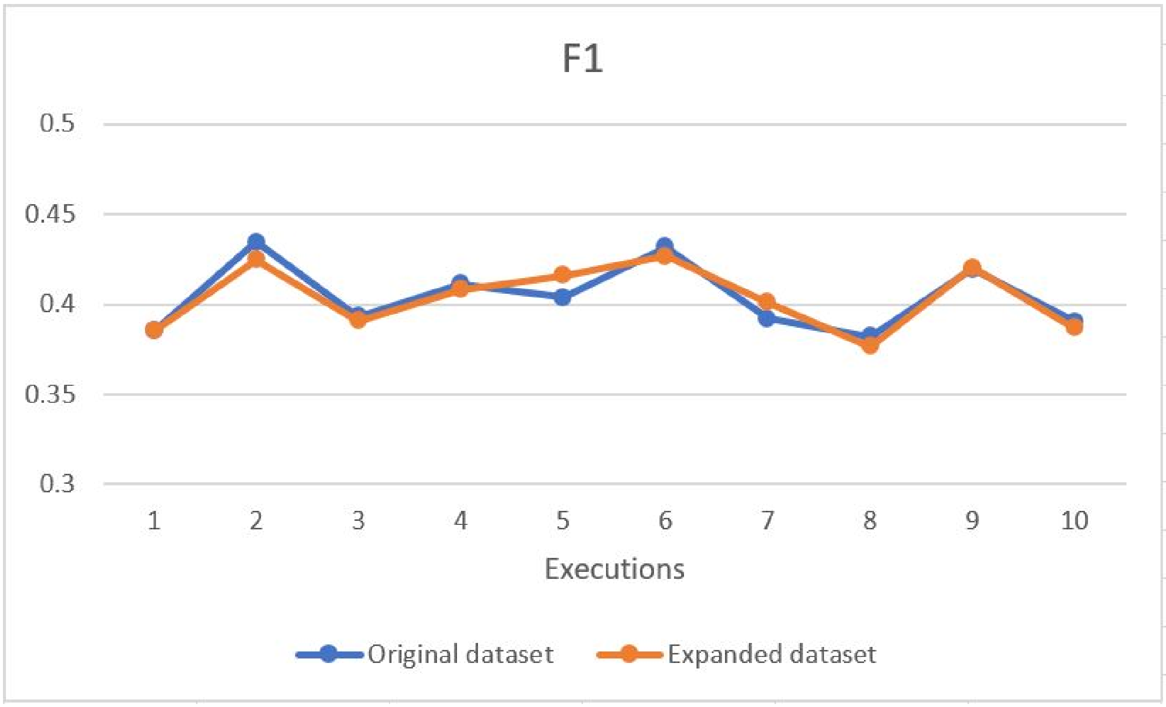


Fig. 4.5 Results obtained for both models, in 10 executions for the F1 measure

- **Precision** $\frac{TP}{TP+FP}$ **:** Represents the correct predicted positives over the total of predicted positives.
- **Recall** $\frac{TP}{TP+FN}$ **:** Represents how many of the actual positives the model predicted as positives.
- **F1** $2 * \frac{Precision*Recall}{Precision+Recall}$ **:** Represents the harmonic average of Precision and Recall.

Figure 4.3 shows the results of the classifiers expressed in terms of Precision and Figure 4.4 shows the results in terms of Recall. Figure 4.5 shows the F1-score obtained for each classifier. To analyze the results obtained in detail, consult Table B.1 in Appendix B.

As can be seen in Figure 4.3, the precision is slightly higher for the original data set. Although in Figure Figure 4.4 we observe that the Recall is slightly higher for the model with the enriched set. This leads us to recognize that performance is similar for both data sets, as seen in Figure 4.5, where it can easily be seen that there is no significant improvement to the trained classifier model with the expanded set. This may be mainly due to the fact that the new key phrases that were added to the training set do not actually appear in the test set. Therefore there is no improvement in the recognition of named entities or key phrases. To support this hypothesis, we will use the *Jaccard similarity coefficient* with the intention of knowing the degree of similarity between the set of extended entities and the test set of each execution.

4.5.1 Jaccard Similarity Coefficient

The *Jaccard Index*, also known as the *Jaccard similarity coefficient*, is a statistic used in understanding the similarities between sample sets. The measurement emphasizes similarity between finite sample sets, and is formally defined as the size of the intersection divided by the size of the union of the sample sets. The mathematical representation of the index is written as:

$$J(A,B) = \frac{A \cap B}{A \cup B}$$

Breaking down the formula, the Jaccard Index is essentially the number in both sets, divided by the number in either set, multiplied by 100. This will produce a percentage measurement of similarity between the two sample sets. That is, it is equal to zero if there are no elements that intercept and equal to one if all elements intercept.

By using this formula, we measure the degree of match between the *expanded Named Entity set* with the *test set*. Thus with the sets of the first execution, the second, the third and up to the tenth execution. The degrees of similarity between these data sets are shown in Table 4.5.

In all runs of the experiment it is notable that the degree of similarity decreases when the extended training set is used for the classifier. This means that the new named entities that are created from synonyms increase the training set (Vocabulary size) but these new named entities do not appear in the test set, so they do not cause an improvement in the performance of the classifier. To see why there are no significant improvements when the training set is expanded, we have analyzed the named entities from the original set that appear in the test set against those from the expanded set that appear in the test set, as shown at column 4. In this column can be observed that only a little more named entities are recognized compared to the original set, only from 3 to 10 more entities which does not generate a significant improvement and which explains the *F-1* results obtained in Table B.1.

In this chapter, we managed to find a way to generate new key phrases similar to those originally tagged with our web system. This was achieved through the use of synonyms and a selection process that allowed us to keep only those key phrases that were consistent and referred to the same concept as their original key phrases. We also demonstrate how to take advantage of the Google results API which returns the number of results for a specific search, thus reliably validating the generated key phrases. Later, we experimented to find

Table 4.5 Jaccard index obtained between expended sets and testing sets

Sets	**Jaccard Index**	**Vocabulary Size**	**Vocabulary Included in Test Set**
Normal dataset 1	0.07	2,202	201
Expanded dataset 1	0.04	4,533	208
Normal dataset 2	0.09	2,133	237
Expanded dataset 2	0.05	4,384	246
Normal dataset 3	0.08	2,078	228
Expanded dataset 3	0.05	4,196	236
Normal dataset 4	0.08	2,168	223
Expanded dataset 4	0.05	4,444	233
Normal dataset 5	0.08	2,229	225
Expanded dataset 5	0.05	4,444	228
Normal dataset 6	0.09	2,190	241
Expanded dataset 6	0.05	4,351	246
Normal dataset 7	0.07	2,141	204
Expanded dataset 7	0.04	4,325	216
Normal dataset 8	0.07	2,261	190
Expanded dataset 8	0.04	4,481	197
Normal dataset 9	0.08	2,175	214
Expanded dataset 9	0.04	4,427	217
Normal dataset 10	0.09	2,101	246
Expanded dataset 10	0.05	4,249	258

out if adding the new key phrases to the original set could lead to better recognition results. However, we found that by doing this we did not get a significant improvement, because these new key phrases did not appear in the test set. So in this particular case, we did not achieve an improvement in the recognition of key phrases, when we added the newly created key phrases.

Chapter 5

Enriched Graph Structure for Key Phrase Recognition

The main contribution made in this section is the capacity to map key phrases in a graph structure, at the same time that new key phrases similar to the original ones are added. This graph structure allows capturing the syntactic and semantic relationships of the key phrases, saving a lot on their storage. Thanks to the fact that this proposed structure manages to map the semantic and syntactic relationships of the key phrases, it saves a lot of space by simplifying their storage.

5.1 Introduction

The current chapter presents a proposed classifier in the task of Named Entity Recognition (NER) based on a data structure represented by a graph with enriched language functions. These rich language functions are intended to make named entity or key phrase recognition more accurate. The idea is based on the experiments in the previous chapter where new key phrases were created from the original set with the help of semantic similarity. So in this graph structure, we have nodes and weights in the links that connect these nodes, but with the difference that new nodes are added to the original nodes. These new nodes (or alternative routes) are obtained from the synonyms of each one of the words that conform to the original key phrase. In this way it is intended that the classifier can recognize key phrases that are semantically similar to those of the original set, thus expanding the possibilities of recognizing a key phrase. The performance of this proposed structure is compared against the performance of some NER classifiers.

5.2 Proposed Graph Structure

The graph structures allow mapping the syntactic and semantic relationships that exist between words and even between concepts, so they can result in powerful representations of natural language. For that, we will explain the scheme of the graph with enriched functions of the language that we propose in this work.

5.2.1 Definition

The formal definition of the proposed graph to represent the key phrases, as well as the relationships between them, is as follows:

$$G = (V, E, f_E, \alpha)$$

Where, V represents a set of vertices or nodes, E are the set of edges that connect to the set of vertices, f_E is the weighting that the edges receive, and α is the function that calculates the weight that edges receive.

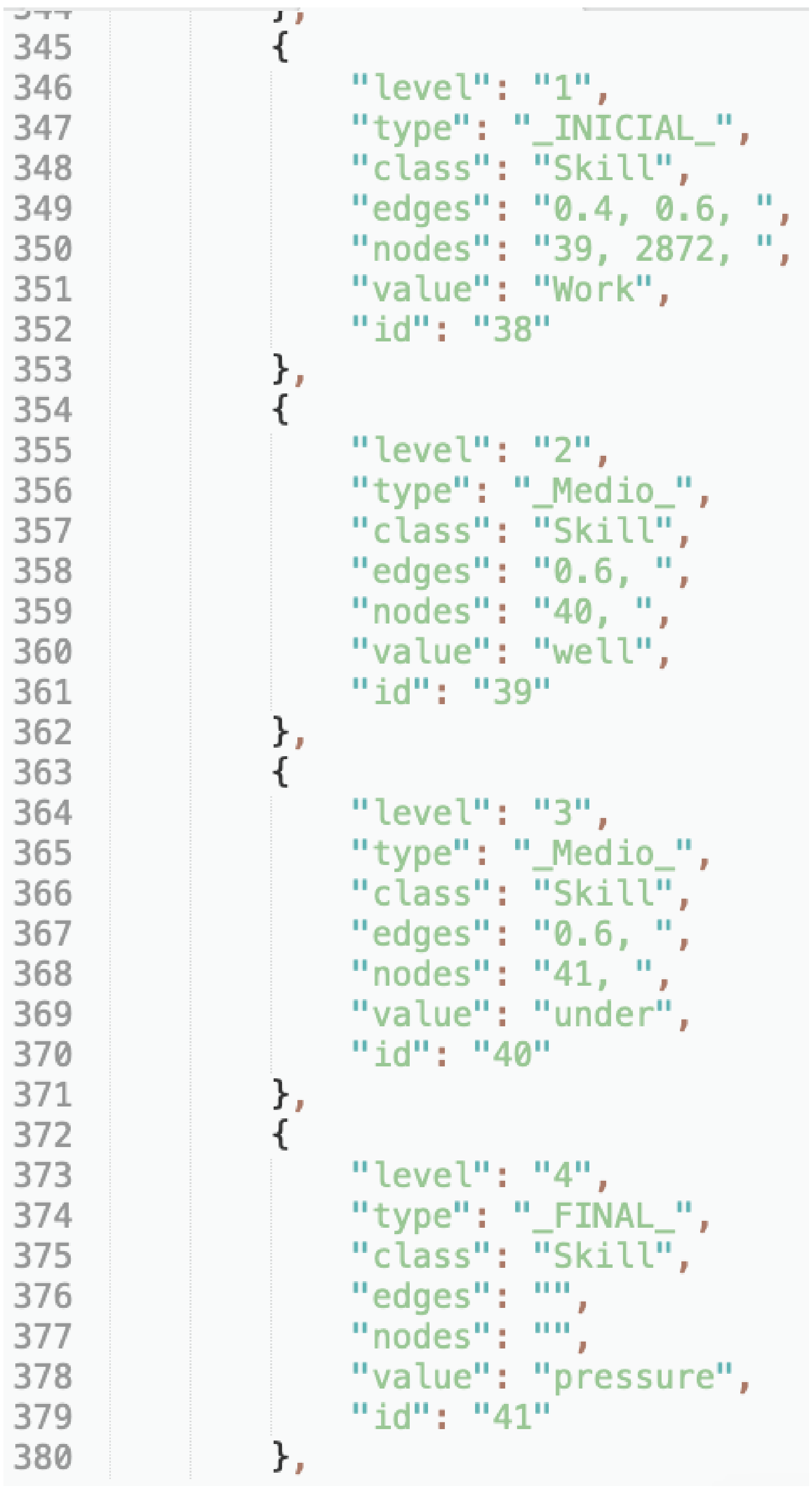
In this case, a vertex or node represents a word of a key phrase, and the edges are those that connect the words of a key phrase. In turn, these edges have a weight that represents how frequent a word is for a set of similar key phrases. This weight is recalculated every time a new key phrase is added to the graph. So, when a word of a key phrase already exists in the structure in the same branch where it is being added, then the weight of this edge increases in one unit.

5.2.2 Syntax

Figure 5.1 illustrates an example in JSON format of real data that has been stored in this graph-based structure. In this figure, it is possible to observe the features that are considered to represent the key phrases.

It is also possible to observe that the nodes of the graph are stored and structured between two curly brackets which encapsulate the properties of each node in key-value pairs. Each of these properties is explained in detail below:

- **Level:** The level refers to the depth that the node has in the graph. For example, in figure 5.1, there are 4 nodes and the level is sequential. Here the key phrase called: *"Work well under pressure"* consists of 4 words. When decomposing and storing this key phrase, the word *"Work"* is the one that starts this key phrase, so it is first stored and its assigned level is 1. Later the word *"well"* is stored and the assigned level is 2. After



```
    },
    {
        "level": "1",
        "type": "_INICIAL_",
        "class": "Skill",
        "edges": "0.4, 0.6, ",
        "nodes": "39, 2872, ",
        "value": "Work",
        "id": "38"
    },
    {
        "level": "2",
        "type": "_Medio_",
        "class": "Skill",
        "edges": "0.6, ",
        "nodes": "40, ",
        "value": "well",
        "id": "39"
    },
    {
        "level": "3",
        "type": "_Medio_",
        "class": "Skill",
        "edges": "0.6, ",
        "nodes": "41, ",
        "value": "under",
        "id": "40"
    },
    {
        "level": "4",
        "type": "_FINAL_",
        "class": "Skill",
        "edges": "",
        "nodes": "",
        "value": "pressure",
        "id": "41"
    },
```

Fig. 5.1 Storage format for key phrases on the graph structure.

the word *"under"* is stored and level 3 is assigned to it. Finally, the word *"pressure"* is stored with level 4, and this is how the storage of this key phrase is finished. So each time a different key phrase is stored, it will have its respective levels according to the number of words that make it up.

- **Type:** For each node or word that makes up a key phrase, it is assigned a state or type (INITIAL, MIDDLE, or FINAL). So in figure 5.1, the word *"Work"* is the one that starts the key phrase, so it will be of the *INITIAL* type. The subsequent words *"well"* and *"under"* will be of type *MIDDLE*. And the word *"pressure"* which is the one with which the key phrase ends will be of the *FINAL* type.

- **Class:** The class refers to the type of tag (*Role, Knowledge, Skill, Responsibility, Character, or Talent*) that was assigned to the key phrase. So in the example in figure 5.1, all the words that make up the key phrase *"Work well under pressure"* will have the same class *"Skill"*.

- **Edges:** This property is a list of weights corresponding to each of the edges that connect this node with other nodes. For example, in the key phrase in figure 5.1, the node *"38"* connects with the node *"39"* and the weight of this edge is equal to *"0.4"*. Similarly, this node *"38"* connects with node *"2872"* with a weight of *"0.6"*. Then node *"39"* connects with node *"40"* and its edge weight is *"0.6"*. Similarly, node *"40"* connects with node *"41"* with a weight of *"0.6"*. And it is observed that node *"41"* no longer connects with any other node because this node is of type *FINAL*. This means that every node of the *FINAL* type will not have information on this property.

- **Nodes:** This property allows us to save a reference to the nodes that connect to this node. So it can be easy to determine which word is next, given a starting word. In figure 5.1, it is observed that node *"38"* is followed by two more nodes which are identified by their id *"39"* and *"2872"*. Then node *"39* connects node *"40"*, and in turn node *"40* connects node *"41"*. But it is observed that node *"41"* is no longer connected to another node. So this represents the end of the key phrase and, as in the previous property, no information will be stored.

- **Value:** This property represents a unit of the key phrase to which it belongs. In other words, this property stores a word that is part of a key phrase. For example, in figure 5.1 the key phrase *"Work well under pressure"* was broken down into four words, so each one is stored in a node. In consequence, the initial node will have the value of *"Work"*, then the next node will store the value *"well"*, then the next node will store the value of *"under"*, and finally the last node will have the value of *"pressure"*.

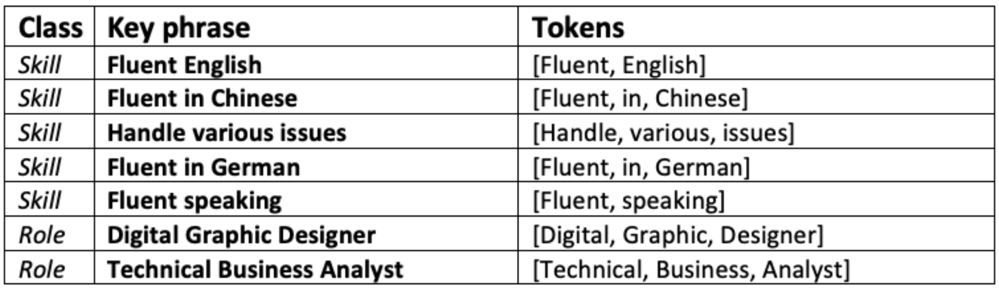
- **Id:** This property is through which the node is recognized, so it is a unique identifier. This identifier is numerical and progressive, which means that every time a node is added to the graph, this node will take the identifier of the previous node + 1. In such a way that in figure 5.1, the key phrase *"Work well under pressure"* starts its insertion into the graph at position *"38"*. Then the word *"well"* is inserted at position *"39"*, then the word *"under"* is added at position *"40"* and ends with the word "pressure" at position *"41"*.

5.2.3 Construction Process

To proceed with the construction of the graph, first, we take the set of key phrases that make up our Gold Standard, which consists of 3,336. Each of these key phrases has been previously tagged with a class (*Role, Knowledge, Skill, Responsibility, Character, or Talent*). While each of these key phrases is being added to the graph, the key phrases that were generated in the previous chapter are also added using semantic similarity. As mentioned in the previous chapter, the total number of key phrases generated using synonyms is 2,972. So at the end, the graph is made up of 6,308 key phrases. This graph starts from a root node from which several branches are born, where the classes used are those that determine these branches.

The construction process for this graph structure consists of 2 stages: Tokenization and Insertion of nodes. In the first stage, the Tokenization of the named entities is carried out for easy handling and storage. The second stage concerns the Insertion of the nodes, and for that reason we have classified each token as an initial node, a middle node, or a final node, to distinguish its position in the branch. This process is further detailed below, to explain why we intend to do so.

- **Tokenization:** For each key phrase a decomposition is performed, in consequence, each key phrase now becomes a list of tokens or words, as shown in figure 5.2. This task will facilitate the treatment of named entities at word-level.



Class	**Key phrase**	**Tokens**
Skill	**Fluent English**	[Fluent, English]
Skill	**Fluent in Chinese**	[Fluent, in, Chinese]
Skill	**Handle various issues**	[Handle, various, issues]
Skill	**Fluent in German**	[Fluent, in, German]
Skill	**Fluent speaking**	[Fluent, speaking]
Role	**Digital Graphic Designer**	[Digital, Graphic, Designer]
Role	**Technical Business Analyst**	[Technical, Business, Analyst]

Fig. 5.2 Representation of the tokenization performed in some key phrases.

- **Insertion of the first token:** Before inserting the first token of a key phrase, it is verified if that token with that value and that class already exist in the graph. If the value of that token already exists in the graph but its class is different, then another branch is created from the root node and added as a new node. But if the value of that token and its class already exist as is in the graph, then the value of its edge is simply increased, as seen in figure 5.3, where the new key phrase *"Fluent in Chinese"* has the same initial node that the key phrase *"Fluent English"*. And in the opposite case, a new node is added to the root node, as shown in figure 5.4, where the new key phrase *"Handle various issues"* belongs to the same class or category that *"Fluent in Chinese"* and *"Fluent English"* but its initial node does not exists on the graph.

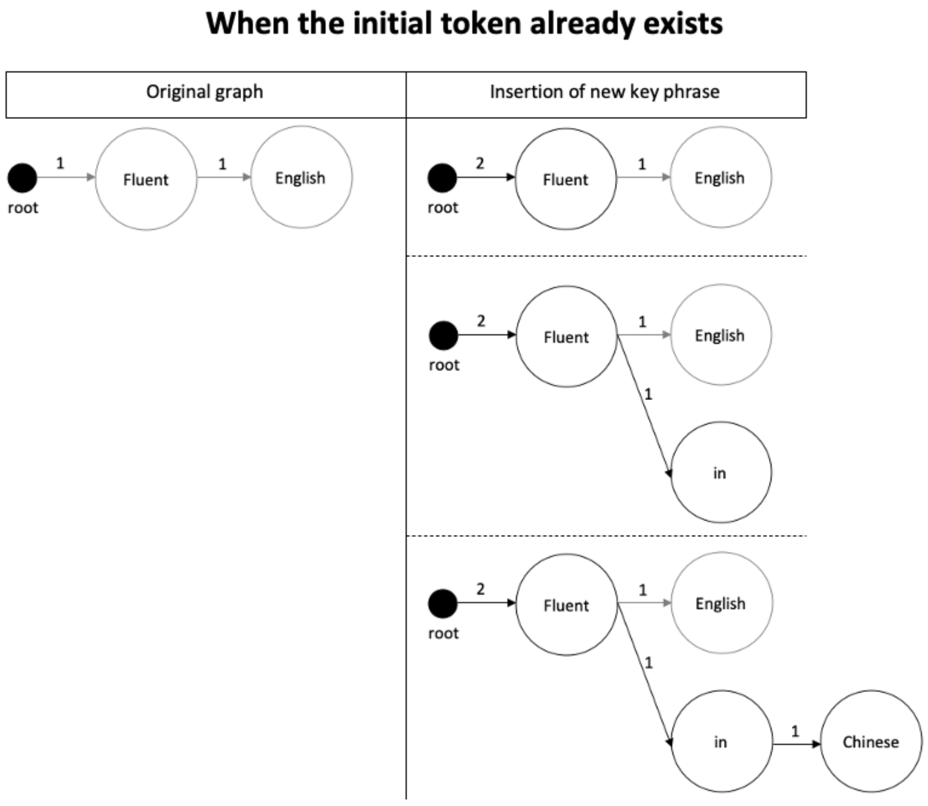

When the initial token already exists
Original graph
Insertion of new key phrase
root
Fluent
English
in
Chinese
1
2

Fig. 5.3 Insertion of a key phrase when its initial token already exists in the graph.

- **Insertion of intermediate tokens:** Before inserting an intermediate token, it is verified if that token already exists after the previous node is inserted. If the token already exists, then the value of its edge is increased. Otherwise, a new branch is generated and a new node is inserted. And this phase is applied for each intermediate token of each key phrase. This step is illustrated in figure 5.5, where the new key phrase *"Fluent in German"* belongs to the same class or category that *"Fluent in Chinese"* but also it

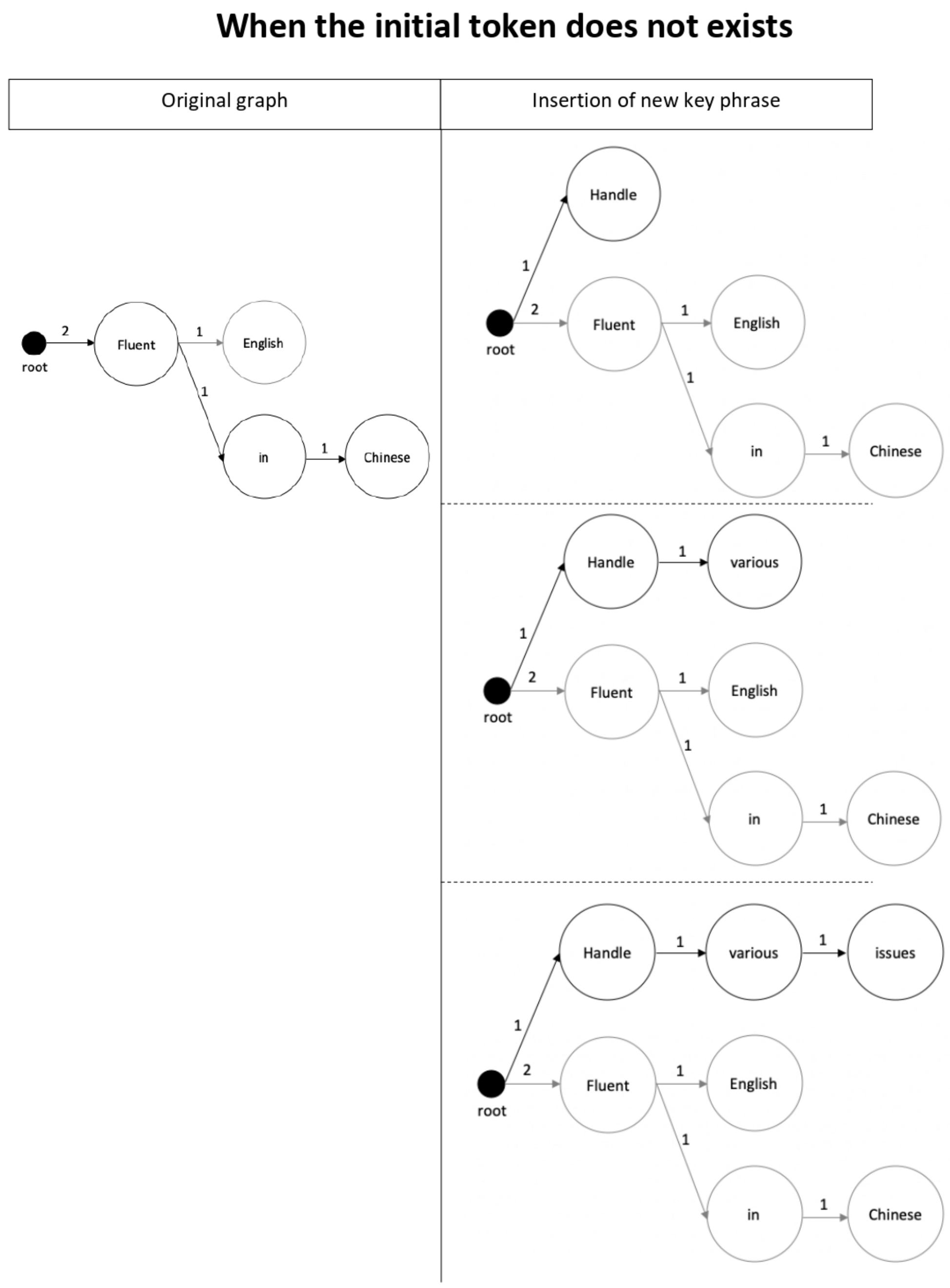

When the initial token does not exists
Original graph
Insertion of new key phrase
root
Fluent
English
in
Chinese
Handle
various
issues
1
2

Fig. 5.4 Insertion of a key phrase when its initial token does not exists in the graph.

contains the first and the second word that the other key phrase. For that reason the value of the edge *"Fluent"* increases one unit, as well as the edge *"in"*.

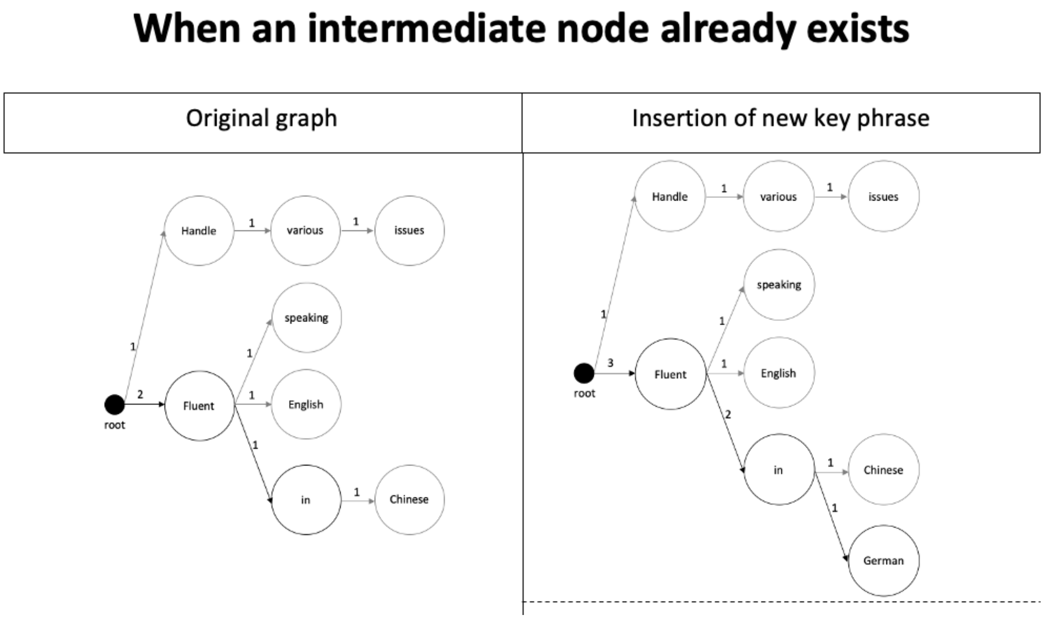


Fig. 5.5 Insertion of a key phrase when its initial and intermediate token already exists in the graph.

- **Insertion of the last token:** Finally, for the insertion of the last token of a key phrase, it is verified if that token already exists, and if so, the weight of its edge is simply increased. Otherwise, a branch is generated and a new node is inserted, which does not point to more nodes. Suppose that the key phrase *"Fluent in Chinese"* already exists in the graph, and we want to add the same key phrase again. Therefore, only the values of its edges will increase without the need to insert a new node. But otherwise, as shown in figure 5.5, the key phrase *"Fluent in Chinese"* needs to be added, however, there is already a very similar key phrase on the graph, which is *"Fluent in English"*, in this case, there is only need to insert a new node, which would be *"Chinese"*. So we comprehend the importance of storing the key phrases in a graph structure that saves us storage space and makes it easier for us to know which key phrases are most common by class.

Following this process is how we manage to obtain the complete graph that stores all the named entities that we mentioned. And then we are ready to make use of it.

5.3 Experiments

To measure the performance of this proposed graph structure as a named entity classifier, a comparison has been carried out against three other NER classifiers called Stanford NER (Finkel *et al.*, 2005), CRF++ (Lafferty *et al.*, 2001) and MITIE (King, 2009).

Making use of the Web-based Tagger System described in Appendix A of this thesis is how we have obtained the datasets that have been used to train the classifiers as well as the datasets to test them. In such a way that the Web-based Tagger System allowed us to use 75% of our total set of documents randomly as a training set for classifiers. And the remaining 25% was used as a test set. This only represented one run or execution, so to have a higher level of confidence we have repeated this process 9 more times. It means that in each of these executions, a different random 75% is taken any time. And the classifiers are tested with the remaining 25% corresponding to each execution. This is known as V-fold Cross-Validation, and it is the method we use to measure the performance of our classifier compared to the other three mentioned above. Table B.2 shows the complete results from the experiments carried out to determine the performance of different classifier models including our proposed method. However, as it is possible to observe in the table, the results are difficult to read as there are many values displayed. Consequently, they have been averaged per *class* and a new table has been generated where only the averages per *class* are displayed, as shown in Table 5.1. Also for the reader's convenience, the results obtained by each classifier in each class have been plotted. These graphs are shown in figure 5.6.

Table 5.1 Average of the results by classifier and by class of the entire experiment.

Model	Class	Measures		
		Precision	Recall	F-1
Stanford	Character	0.8772	**0.5221**	**0.6513**
	Knowledge	0.7132	**0.5832**	**0.6490**
	Responsibility	0.5038	**0.3359**	**0.3882**
	Role	0.8072	**0.6232**	**0.6883**
	Skill	0.7254	**0.4334**	**0.5400**
	Talent	0.6672	**0.2357**	**0.3494**
CRF++	Character	**0.9343**	0.2962	0.4447
	Knowledge	**0.7603**	0.3349	0.4631
	Responsibility	**0.6055**	0.1340	0.2185
	Role	**0.8717**	0.3530	0.4953

Continued on next page

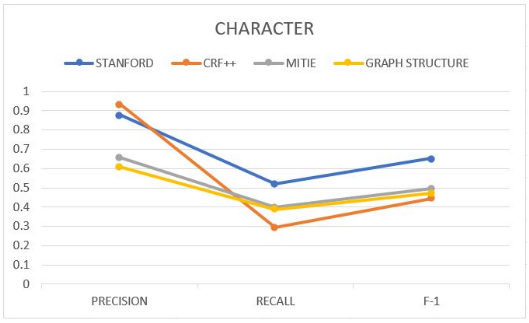
Table 5.1 – *Continued from previous page*

Model	Class	Precision	Recall	F-1
	Skill	**0.8438**	0.2633	0.3992
	Talent	**0.9127**	0.0992	0.1763
MITIE	Character	0.6585	0.4002	0.4962
	Knowledge	0.6298	0.4850	0.5469
	Responsibility	0.4506	0.2171	0.2911
	Role	0.7484	0.5825	0.6511
	Skill	0.5784	0.3636	0.4430
	Talent	0.4801	0.1343	0.2081
Graph Structure	Character	0.6094	0.3889	0.4724
	Knowledge	0.7311	0.3458	0.4686
	Responsibility	0.3881	0.0482	0.0855
	Role	0.5695	0.3835	0.4553
	Skill	0.6620	0.2388	0.3497
	Talent	0.5363	0.1900	0.2772

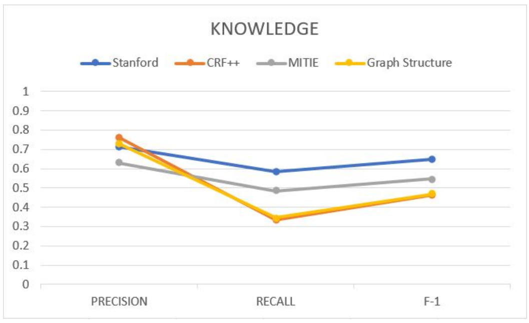
Table 5.1 shows the performance of each classifier for every class, highlighting the best results obtained for each class. When analyzing the values from the table, the CRF++ classifier presents the highest *Precision*, proving to be superior to the others for each class. It happens because CRF++ returns very few results, but the vast majority of its predicted labels are correct. For this metric, our classifier based on a graph structure proved to have lower performance than the others, which means that of the named entities recognized with a class, few of them actually belong to the assigned classes.

As for the *Recall* metric, the Stanford classifier shows better results than the other methods. It happens because the classifier returns many results or predictions, but most of the labels it predicts are incorrect. On this occasion, our method proves to be superior to CRF++ but less than the other two methods, ranking third in performance. That implies that of all the named entities tested that originally belonged to a class, not all of them managed to be identified or labeled by our method, with very few of them being able to be labeled correctly.

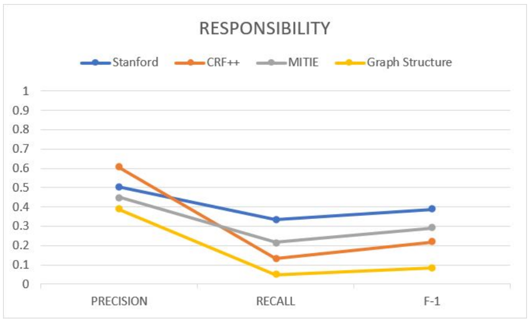
The results show that the Stanford classifier has better performance since its harmonic Precision and Recall measure denoted by *F-1* is superior to the *F-1* measure of the other methods by up to 19%. This means that the Stanford classifier returns more results where the vast majority of them are correct.



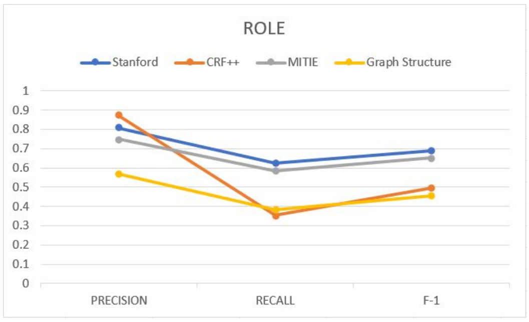
(a) Skill



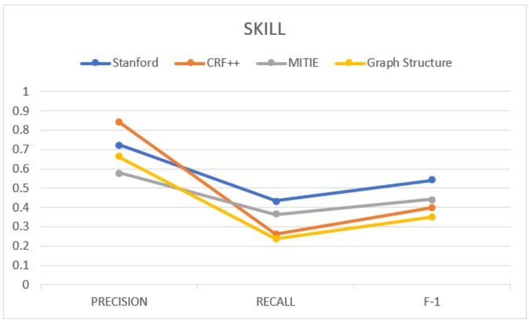
(b) Role



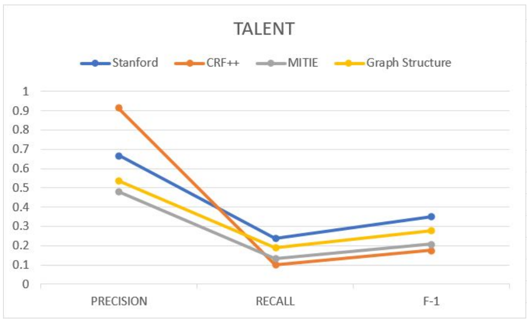
(c) Character



(d) Knowledge



(e) Talent



(f) Responsibility

Fig. 5.6 Graphic representation of Table 5.1.

The MITIE classifier takes second place in performance, showing better results in the *Role* and *Knowledge* classes. But we observed that these two classes outperform others in every case because of their nature since they are formed of specific words from a field study such as computing field, besides having an extensive variety of samples. Also, they are formed of few words compared to the *Responsibility* class, which is characterized by having named entities that are composed of many words and with a considerable variety. That is why the *Responsibility* class always returns poor results in all the classifiers. On the other hand, the *Talent* class has very few examples in the training set since they are named entities that rarely appear as much in the documents as those that belong to the *Role* or *Knowledge* class. For this reason, regardless of the classifier we use, the *Talent* class always returns the lowest results. As for the *Skill* class, it usually has an average performance due to two things, firstly because it has a good number of samples, but also, these samples have very varied sizes (some named entities formed by 2, 3, 4, 5, 6, 7, 8, 9, or 10 words). It increases the complexity of fully recognizing a named entity and causes only segments of them to be recognized. The *Character* class is the third-best performing class after the *Role* and *Knowledge* class, and this is because its named entities are frequently unigrams, bigrams, and trigrams, which makes them easy to recognize. But it does not have as many samples as the first two classes, and it does not allow to achieve better results.

After analyzing the results obtained in terms of the *F-1* measure, it was easy to determine that the graph structure we proposed to store and enrich named entities achieves similar results to those of the CRF++ classifier. Consequently, these two classifiers rank last in terms of the *F-1* measure. Although our graph structure did not produce the best results in recognizing named entities, it proves proficient against one of the tested classifiers. It leads to consider our proposed method as a starting point for improvements through incorporating new methods such as word embeddings. We will consider Poincaré embeddings in the next chapter to improve our proposal in recognizing named entities, since word embeddings can capture valuable information on the relationships between words.

In this section, we proposed a graph structure to store key phrases by capturing their semantic relationships and thus obtaining an easy way to recognize key phrases. However, the results obtained were not the best for the classification of key phrases due to the small data set. But it serves as a starting point to improve our classification of texts in the next chapter.

Chapter 6

Models for Word Representations

In this section, we want to study how the representation of words can be carried out through word embeddings and, thus, can determine the best model for our case study. We want to know the advantages of using each of the existing classifier models, and how we can take advantage of their implementations to efficiently capture the semantic relationships between the words in our set of key phrases.

6.1 Introduction

Humans have a natural ability to communicate between them through a natural language, the same that is developed and improved when interaction is continuous. In this way, humans know what to answer when another person asks something without having complications. So, languages are important when communicating, but for a machine is difficult to understand a conversation like a human can do.

Languages are extremely flexible, it means that a person can refer to the same thing or idea in multiple ways. It makes it more difficult for a machine to understand and give the best answer to a specific question. In this sense, the task of Natural Language Processing (NLP) aims to make computers understand human language in a similar way as humans do. However, this is a very complex problem to be solved as computers can only understand numbers. To provide solutions it is necessary to represent textual information with numeric values, and then be processed by computers in a relatively easy way. In this sense, the word embeddings refers to numeric representations of words.

6.2 Word Embeddings

There exist several approaches to word embeddings representing some advantages and disadvantages of each one. One of the simplest word embedding approaches is called Bag of Words (BOW), where the main advantage is that there is no need for a large corpus to get the word embedding and get good results. However, one of the biggest problems is the huge amount of memory used by this approach, since huge vectors are created including empty spaces. Another problem of the *BOW* approach consist in the lost of any semantic relationship between words, since it does not considers any context information (words that precede and proceed to a given word).

6.2.1 Word2Vec

Word2Vec is a tool that provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words, where semantically similar vectors are close to each other in N-dimensional space, where N refers to the dimension of the vector. It represents a great advantage since the size of vector embedding is very small in comparison with other word embeddings approaches (Mikolov *et al.*, 2013a). Both architectures use neural networks to learn the word representations underlying each word.

A text corpus is entered to the Word2Vec tool, producing the word vectors as output. What Word2Vec does is to construct a vocabulary from the training text corpus and learns vector representations of words. With the resulting vectors, it is possible to find the closest words for a specific given word. It is possible thanks to the context information considered by the approach, capturing some semantic relationships between the word and the words around.

One of the main advantages of Word2Vec is its capacity to work with huge data sets, allowing to train models with up to hundreds of billions of words (something impossible for the normal Bag of Words approach).

Windows Size in Embedding's Literature

It is a local and symmetric window surrounding a word, it means R number of words before and R number of words after the given word. For example, in the next sentence:

- *"To be responsible for successfully implementing IT projects to ensure delivery in line with business and customer expectation."*

Table 6.1 Comparative table between Bag of Words and Word2Vec

Bag of Words	**Word2Vec**
No need for a large corpus to get good results.	Uses deep neural networks to learn relationships between words.
Computationally, this model is not very complex.	Converts words into corresponding vectors in N-dimensional space.
It creates a dictionary of unique words from the corpus.	Considers context information (retains semantic information).
It uses zeros and ones to represent each document in vector embeddings.	Creates very small word embedding vectors.
It wastes disk space since it creates huge vectors (containing empty spaces).	Continuous Bag of Words (CBOW) implementation.
The model does not consider any context information for words. So, semantic information is discarded.	Skip-gram model implementation.

a five-word symmetric window around *implementing* would be (*"To", "be", "responsible", "for", "successfully"*) and (*"IT", "projects", "to", "ensure", "delivery"*). A three-word symmetric window around *implementing* would be (*"responsible", "for", "successfully"*) and (*"IT", "projects", "to"*). There exist models that use asymmetric windows, but they are not as common.

Is known that larger windows capture more topical relations (e.g. "Obama-President") while small windows capture syntactic relations (e.g. "jumps-jumping"). Also is known that bigger vectors capture more senses of the word. But it may not be optimal to have them too large, since this leads to redundancy (i.e. there are lots of dimensions that aren't doing much, and perhaps just capturing noise). But too small is bad too: in the limit, one may not be able to distinguish words at all (Spirling and Rodríguez, 2019).

Continuous Bag-of-Words Model

Unlike the Bag-of-words model, it employs continuous distributed representations of the context. This basically means that given user-specified context words, the continuous bag-of-words (CBOW) model tries to predict the word in the middle by combining the distributed representations of context. And where the order of context words does not have influence on prediction (Mikolov *et al.*, 2013a). Table 6.1 shows a comparison between the standard BOW model and the CBOW model. The training complexity of CBOW architecture is represented as

$$Q = N \times D + D \times \log_2(V) \tag{6.1}$$

Continuous Skip-gram Model

Given a user-specified word it is used as input to a log-linear classifier with continuous projection layer, predicting the surrounding words to the provided word. The training complexity of this architecture is proportional to

$$Q = C \times (D + D \times \log_2(V)) \tag{6.2}$$

with C representing the maximum distance of the words. This means, if we choose C=10 then for each training word a randomly number R between 1 and C will be selected, and then use R words that proceed and R words that precede the provided word as correct labels (Mikolov *et al.*, 2013a).

The Word2Vec tool was tested using the official python implementation [1]. Since Word2Vec approach considers context information, it is possible to find the most similar words given a specific word. Through the method *most_similar()* a list of the most similar words with their similarity index is returned in a list. In our case we decided to get the 10 most similar words, and to carried out the experiment was needed to provide a corpus for training which is described in next section.

Corpus and Training

The original dataset consist of 160 Job Descriptions of IT field collected from the website www.jobs.ie and tagged with 6 classes: *Role, Knowledge, Skill, Character, Responsibility* and *Talent*. All The tagged Job Descriptions were joined in a single file text containing a total of 121,292 tokens and including 3,347 key phrases (named entities). First, the dataset was converted to the *Stanford format* (structured in double column) being the first column for tokens or words and the second column for the class assigned to each token (one of the 6 classes previously mentioned) or *0* if the token does not have a class assigned. Then, a second conversion was applied to the corpus, passing everything to one line, and each token separated for just a blank space. I lowercased, tokenized and removed the stopwords from the corpus before constructing the model. Then I trained the model for creating vector representations for words of size 100 as stated by (Mikolov *et al.*, 2013a) and a context window size of 10 words as suggested for authors (it means that 10 words before and 10

[1]The official Word2Vec Implementation in Python was taken from the following link: https://pypi.org/project/word2vec/

words after the node are considered as context information). However, in order to know if 10 is the best value for window size, we have used the values of 5 and 15 as well.

Experiment

The experiments were performed with three models: a model trained using a window size of 5, a model trained using a window size of 10, and the model trained using a window size of 15. So, I looked for the 10 most similar words for each one of the classes used to tag the key phrases on the corpus, as shown in Table 6.2.

Table 6.2 Word2Vec implementation with different context window sizes, showing the most similar words for user-specified words

User-specified word	Most similar words		
	window size = 5	window size = 10	window size = 15
role	analyst	engineer	engineer
	engineer	analyst	helpdesk
	manager	administrator	years
	designer	engineering	analyst
	science	science	administrator
	trainer	field	field
	senior	developer	degree
	advisor	degree	junior
	field	senior	minimum
	degree	computer	level
knowledge	ms	azure	azure
	microsoft	operating	sql
	exchange	sql	frameworks
	git	git	operating
	jquery	ms	javascript
	oracle	microsoft	web
	web	web	app
	server	crm	techniques
	sql	understanding	git
	machine	jenkins	restful
skill	english	verbal	communication
	verbal	both	interpersonal

Continued on next page

Table 6.2 – *Continued from previous page*

User-specified word	window size = 5	window size = 10	window size = 15
	solving	communication	verbal
	communication	written	solving
	german	oral	facing
	french	solving	written
	problem	handle	excellent
	written	problem	ability
	troubleshoot	interpret	spoken
	interpersonal	spoken	deadlines
character	detail	creative	creative
	initiative	motivated	passion
	creative	passion	problem-solving
	problem-solving	prioritise	motivated
	motivated	detail	enthusiastic
	prioritise	initiative	willingness
	eager	eager	enough
	flexibility	attitude	prioritise
	attention	willingness	detail
	oriented	problem-solving	eager
responsibility	provide	determine	effectiveness
	ensure	end	promotion
	projects	smooth	assist
	end	ensure	liaise
	manage	delivery	provide
	internal	maintenance	end
	maintains	all	internal
	versions	stakeholders	ensure
	availability	assist	co-ordinate
	provision	internal	maintenance
talent	accuracy	mentoring	natural
	quickly	learn	quickly
	willingness	quickly	willingness
	mentoring	learner	learner
	learn	acumen	learn

Continued on next page

Table 6.2 – *Continued from previous page*

User-specified word	window size = 5	window size = 10	window size = 15
	lead	willingness	become
	acumen	natural	leadership
	happen	happen	acumen
	innovation	leadership	excellence
	quantitative	complex	prioritise

As can be seen in Table 6.2, the 10 most representative words for each class are listed according to the degree of similarity. And surprisingly for each of the window sizes, the results are very good since each of the words returned for each class makes sense. It is possible to observe that many of the words that appear in the model with window size 5 also appears in the other two models, only that they are in different positions. So it does not affect if we use any of the three, but for convenience we will use a window size of 10 for the model as suggested by the authors (Mikolov *et al.*, 2013a).

6.2.2 GloVe: Global Vectors for Word Representation

The Euclidean distance can be used to get nearest words to one word which is an effective method for measuring the linguistic or semantic similarity of the corresponding words. The similarity metrics used for nearest neighbour evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number. For example, man may be regarded as similar to woman in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another. In order to capture in a quantitative way, the nuance necessary to distinguish man from woman, it is necessary for a model to associate more than a single number to the word pair. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words.

GloVe is an unsupervised learning algorithm for obtaining vector representations for words (Stanford University, 2014). When talking about unsupervised methods for learning word representations, the statistics of word occurrences in a corpus is an essential source of information to achieve the task. GloVe is a model with a weighted least-squares objective.

The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. That model is represented by the equation

$$J = \sum_{i,j=1}^{V} f(x_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2 \tag{6.3}$$

where V represents the size of the vocabulary, and $f(x_{ij})$ is a weighting function that should adhere to the following properties:

1. f(0) = 0
2. f(x) should be non-decreasing so that rare co-occurrences are not overweighted.
3. f(x) should be relatively small for large values of x, so that frequent co-occurrences are not overweighted.

Although so many functions can satisfy these properties, the authors suggest to work with one class of functions parameterized as, $f(x) = (\frac{x}{x_{max}})^{\alpha}$ if $x < x_{max}$ otherwise $f(x) = 1$. The authors discovered that a good value for x_{max} is 100, and using $\alpha = \frac{3}{4}$ offers a better performance in comparison with a linear version using $\alpha = 1$ (Jeffrey Pennington, 2014).

Corpus and Training

The original dataset consist of 160 Job Descriptions of IT field collected from the website www.jobs.ie and labeled with 6 classes: *Role, Knowledge, Skill, Character, Responsibility* and *Talent*. However the vocabulary extension is very short, and in order to enrich that vocabulary a second dataset taken from http://mattmahoney.net/dc/text8.zip was aggregated to the original. The reason to merge both corpus is that in my corpus I have recent terms related to IT field, however they are not enough in order to find high quality variations of relations. In other words, the dataset taken from text8 has a large vocabulary but it does not have all the new terms used in the job descriptions dataset. The new extended corpus created from combining both datasets contains 24515391 tokens and is composed of 255001 unique words, with a total vocabulary of 72646 words. I tokenized and lowercased the combined corpus, and built the vocabulary, and then constructed a matrix of co-occurrence counts X. In constructing X, I choose a context window of 15 in order to distinguish left context from right context.

Following the explanations of authors, for the experiments carried out I set $x_{max} = 100$, = 0.75, and trained the model using AdaGrad (Duchi et al., 2011), stochastically sampling nonzero elements from X, with initial learning rate of 0.05. I ran 15 iterations for vectors

Table 6.3 Experiment performed with window sizes 10, 15 and 20

	Hybrid Model		
Tested NE	Window size 10	Window size 15	Window size 20
Integration developer	No class predicted	Knowledge	**Role**
Providing support	No class predicted	Role	No class predicted
Customer facing	No class predicted	Knowledge	No class predicted
Excellent communication	Knowledge	K / **Skill**	R / K / **Skill**
Problem solver	Knowledge	**Skill**	K / **Skill**
Software development	No class predicted	**Knowledge**	Role

of size 50, and a context of 15 words to the left and 15 words to the right since it has the ability to capture more topical relations in comparison with a low value limited to capture just syntactic relations (Spirling and Rodriguez, 2019). The model generates two sets of word vectors, w and $\tilde{w}$. When X is symmetric, b and $\tilde{b}$ are equivalent and differ only as a result of their random initialization; the two sets of vectors should perform equivalently. On the other hand, there is evidence that for certain types of neural networks, training multiple instances of the network and then combining the results can help reduce overfitting and noise and generally improve results (Ciresan et al., 2012). With this in mind, I choose to use the sum $w + \tilde{w}$ as my word vectors. Doing so typically gives a small boost in performance, with the biggest increase in the semantic analogy task.

Windows Size Selection

As mentioned before a small window size is not good enough to capture topical relations. However, when using a window size too much larger it will generate noise. Then, a small experiment was performed using different window sizes (10, 15 and 20) to know the information returned by the model in every execution. In Table 6.3 shows the experiments carried out to determine the window size to be used.

It is possible to observe that when the used size window is 10, it is almost impossible to obtain a label belonging to that key phrase. But when the window size is 20, multiple classes are suggested, making it imprecise which class is correct. For this reason the value 10 for window size is the most appropriate to use to predict a class for a key phrase.

Experiment

The experiments were performed with three models: model trained with text8 dataset, model trained only with the labeled job descriptions, and the model trained with a corpus resulting from combining text8 and the labeled job descriptions. So, I looked for the 10 closest words to a specific key phrase (the same) in each of the vector models, as shown in Table 6.4.

As can be seen in the first experiment the term scrum does not exist in the vocabulary of Text8 model as it is a recent term. In the Job Descriptions model we got the 10 closest words to the key phrase, but in the hybrid model we also got the possible label to the key phrase. That last characteristic can be observed in the other three experiments as well, where the possible label is included in the closest neighbors provided and where it has a high priority in the hybrid model.

Given a word the hybrid model provides the closest neighbors including the class or label to which it belongs and then with these closest words it is possible to create sentences or key phrases, and is in the last one of these that is helpful the use of the class or label, in this way by concatenating one closest word each time that belongs to the same class than the word before a key phrase can be created or recognized.

6.2.3 Poincaré Embeddings

Learning representation of words has become a central question in natural language processing. The ability to capture information from the data is the foundation for the learning and generation of the downstream natural language processing tasks.

Word embeddings are effective for many natural language processing tasks because they are flexible and encode valuable syntactic and semantic information. Word embeddings are motivated by the concept that semantic similarities between words are based on their distributional properties in the large amount of text. The idea of distributional properties is called distributional hypothesis (Harris, 1954), meaning that linguistic items with similar distributions have similar meanings. Popular word embeddings such as GloVe, Word2Vec, and FastText are widely used in various tasks and have shown great success. Although these embedding methods have proven successful, very few methods exist that are able to encode tree-like or graphlike hierarchical relationships of the data.

Poincaré embeddings are better at capturing latent hierarchical information than traditional Euclidean embeddings.

Latent hierarchical structures

Hyperbolic geometry is a non-Euclidean geometry which studies spaces of constant negative curvature. It is, for instance, associated with Minkowski spacetime in special relativity. In network science, hyperbolic spaces have started to receive attention as they are well-suited to model hierarchical data. For instance, consider the task of embedding a tree into a metric

Table 6.4 Experiments with three models

	Vector Model		
Tested NE	Text8	Job Descriptions	Hybrid
scrum process	Out of dictionary!	integrate	processes
		backlog	data
		to-be	governance
		insight	existing
		readiness	reporting
		risk	create
		wider	appropriate
		disaster	creating
		sales	**responsibility**
		recommendation	management
digital graphic designer	computer	accelerate	based
	designers	admin	design
	design	inductions	android
	animation	commerce	electronic
	graphics	2003	architect
	fashion	cad	layout
	display	factory	display
	processing	appointments	rendering
	art	merchandiser	designers
	designed	servicenow	**role**
microsoft azure	macintosh	architecting	**knowledge**
	mac	cloud	cloud
	xbox	**knowledge**	windows
	windows	sex	understanding
	apg	mcp	ms
	portable	discrimination	platform
	workgroups	bi	protocols
	zaku	store	mcp
	din	retail	outlook
	pc	virtual	server
Own initiative	its	merit	personal
	provided	output	their
	s	**character**	**character**
	their	flexibility	ability
	his	likes	work
	support	self-manage	individual
	for	call	desire
	which	earth–that's	s
	providing	prioritise	need
	creating	delivered	full

space such that its structure is reflected in the embedding (Nickel and Kiela, 2017). Due to the underlying hyperbolic geometry, this allows us to learn parsimonious representations of symbolic data by simultaneously capturing hierarchy and similarity.

Corpus and training

The corpus is composed of a set with 160 Job Descriptions in the field of IT collected from the website www.jobs.ie and labeled with 6 classes: *Role, Knowledge, Skill, Character, Responsibility* and *Talent*. The corpus consist of 121293 tokens, and a vocabulary of 5297 unique words. The file format in which the information is stored is double column, saving the relation of a token in the first column to its assigned class in the second column.

By using the gensim library was possible to train the Poincaré model with the corpus file. The number of dimensions of the trained model is set to 50 by default and it also needs a number of iterations which is set to 50 by default. As suggested in (Nickel and Kiela, 2017) all embeddings were randomly initialized from the uniform distribution *U(-0.001, 0.001)*, causing embeddings to be initialized close to the origin.

Experiment

The main focus for Poincare embeddings is its capability to embed data that exhibits latent hierarchical structures. Thus, we conduct the experiment using WordNet dataset. WordNet is a large lexical database of the English language and it groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms (synsets) (Miller, 1995). Once the model is trained it is possible to find the closest words to a user-specified word through the most_similar method implementation where the top-N most similar nodes to the given node (word) is returned in increasing order of distance.

This characteristic is used to get the most similar words for each one of the words conforming to a key phrase. Between the most similar words are also included classes belonging to each word. Then the most frequent class in every key phrase is selected, discarding others that occur just a few times. In Table 6.5 are showed the experiments for key phrases used in GloVe implementation.

So, for the key phrase *Excellent communication* the classes *Skill* and *Responsibility* are predicted, however the class *Skill* occurs more times. That is the reason of selecting the class *Skill* as the assigned class to the key phrase which is correct. And the same happens for every one of the key phrases, showing excellent results as the predicted class is the original assigned class in the corpus.

Table 6.5 Poincaré embeddings implementation

Tested Key phrase	Original class	Predicted classes	Most frequent class
Integration developer	Role	Role Knowledge Skill Role Role	**Role**
Providing support	Responsibility	Responsibility Responsibility Knowledge Skill Role	**Responsibility**
Customer facing	Skill	Skill Skill Knowledge Role Talent Skill	**Skill**
Excellent communication	Skill	Skill Responsibility Skill	**Skill**
Problem solver	Skill	Skill Skill	**Skill**
Software development	Role	Role Responsibility Skill Knowledge Talent Role	**Role**

6.3 Conclusions

Poincaré embeddings are better at capturing latent hierarchical information than traditional Euclidean embeddings. The results obtained when trying to predict the correct class for a key phrase outperform the GloVe implementation. As a brief analysis of the methods studied, Table 6.6 presents the main characteristics of each of the methods for obtaining word representations.

Thanks to these experiments carried out to know the performance of each model for the representation of words, we were able to identify the advantages of each one. In general, we observed that the Poincaré embeddings outperform other models because it works on hyperbolic space, allowing us to capture the relationships between words efficiently. For that reason, in the next chapter we will use Poincaré Embeddings to map our set of key phrases and later use it for key phrase recognition.

However, there are some drawbacks when using Poincaré embeddings. The first drawback is the time it takes to build a model since to achieve that the computer had to spend two days on average training the model even though we used eight cores. It represents a high cost computationally speaking since in our case we generated ten models for each type of word embeddings. It means that to have the complete experiments of the Poincaré embeddings it was necessary to wait at least 20 days to have the models ready. On the other hand, training the models with Word2Vec was easier, as it was only a matter of minutes to obtain a complete model. Similarly for GloVe, where the models were trained in just a couple of hours. For that reason, when using Poincaré embeddings needs to be considered that it will take more time than when using the other word embeddings.

In addition, it must be considered that a data source is required to train a Poincaré embeddings model, this means that more time must be spent building that data resource. In our case, we decided to treat a specific type of text (Job Descriptions), for which we had to first collect Job Descriptions and then manually identify the most representative key concepts of each document. This whole process to create the training dataset took around two months. So that represents a disadvantage, because if we want to work with another type of text, then we would have to look for an existing data resource. But if the resource does not exist then it would need to be created, and that would take a lot of time. Making this represent another great drawback to opt for these options.

But in our case we have already created that data resource, there is time available and we want to obtain better results, so this does not represent a big problem. So we will try to take advantage of the benefits of Poincaré embeddings and see what their impact is on our experiments in the next chapter.

Table 6.6 Comparative table of models used to capture relationships between words.

Word2Vec	**GloVe**	**Poincaré**
Uses deep neural networks to learn relationships between words.	Implements aggregated global word-word co-occurrence statistics.	Allows learning hierarchical representations of symbolic data by embedding them into hyperbolic space.
Converts words into corresponding vectors in N-dimensional space.	Is a new global log-bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods.	Poincaré embeddings outperform Euclidean embeddings significantly on data with latent hierarchies, both in terms of representation capacity and in terms of generalization ability.
Considers context information (retains semantic information).	The model efficiently leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix.	Is an efficient algorithm for computing the embeddings based on Riemannian optimization, which is easily parallelizable and scales to large datasets.
Creates very small word embedding vectors.	The computational complexity of the model depends on the number of nonzero elements in the matrix X.	Poincaré embeddings can simultaneously learn the similarity and the hierarchy of objects.
Continuous Bag of Words (CBOW) implementation.	Nearest neighbors calculated through the Euclidean distance between two word vectors.	Poincaré embeddings are successful in predicting links in graphs where they outperform Euclidean embeddings, especially in low dimensions.
Skip-gram model implementation.	Outperforms other models on word analogy, word similarity, and named entity recognition tasks.	Embeddings trained on WORDNET provide state-of-the-art performance for lexical entailment.

Chapter 7

Poincaré Embeddings for Key Phrases Recognition

By employing the findings obtained in previous chapters, in this section, we seek to contribute with a classifier model that performs better than others for recognizing key phrases. This model considers the POS structures studied in chapter 3, where for each class or category we obtained the most representative POS structures. So firstly, our classifier looks for these POS structures in a Job Description, assuming that the possible key phrases belong to the class of the POS structure with which it is identified. Later, to verify if it really is a key phrase and if it really belongs to that class, the Poincaré embeddings are used, through which we obtain the closest words given a starting word. So, for the first word of the possible key phrase, our model determines if the next word has a close relationship (if together they can refer to some key concept of the document). If so, the next word is analyzed, and so on successively until one of them no longer has much meaning with the previous ones.

7.1 Introduction

Recently, the amount of textual information available electronically is incredibly large, making it difficult to obtain the most relevant information on a specific topic without having to spend a lot of time reading the texts, and this is due to an overload of information since every day thousands of new textual contents are generated on the Internet. To deal with this problem there are information retrieval (IR) systems, which search for information in a collection of documents and retrieve the most relevant resources based on a specific information need. To achieve this some techniques are required such as the recognition of named entities through which a text document can be represented, reducing the information overload. In this sense NER is a sub-task of Information Extraction (IE) where the information we are looking for is known beforehand. Is through these tasks that ”the discovery of new, previously unknown information, by automatically extracting information from different resources” can be achieved. In this chapter, we describe the NER classifier developed in this research work to find the most valuable key phrases for Job Descriptions in IT field by using Poincaré Embeddings.

7.2 Proposed Classifier Workflow

By making use of Poincaré Embeddings our classifier intends to improve the recognition of key phrases or named entities which represent Job Description documents with the most relevant information and thus facilitate some tasks such as the classification of texts and retrieval information. We have manually tagged a Gold Standard from Job Description documents in the IT field and stored it into a graph structure. The stored key phrases are grouped into 6 categories: *Role, Knowledge, Skill, Character, Responsibility* and *Talent.*

The classifier works considering some features found in chapter 4 when n-grams and part-of-speech were studied over the Gold Standard.

Selected Features

In chapter 4 we found that most of the classes are conformed by unigrams, bigrams and trigrams, being cuatrigrams also present in good percentage. N-grams with five or more elements are unusual, that is why when finding key phrases we consider a window size of 4 words in order to find a relation between the current word and the next 4 words.

When studying the representative part of speech structures for each class, some patterns were found. However the most interesting structure was found for the *ROLE* class where key phrases are normally proper nouns in singular form (NNS), and that is something very characteristic of that class. Figure 7.1 shows a diagram of the workflow operation implemented on the classifier developed. The process consists of 3 stages for the recognition of key phrases.

Classifier Design

- **Match Gold Standard Key Phrases in test dataset (Stage 1).** From the original set of key phrases in training dataset, if they exist in the test dataset then they are tagged. In other words, it consists in searching the entire list of training key phrases in the test set and assign them the corresponding tags.

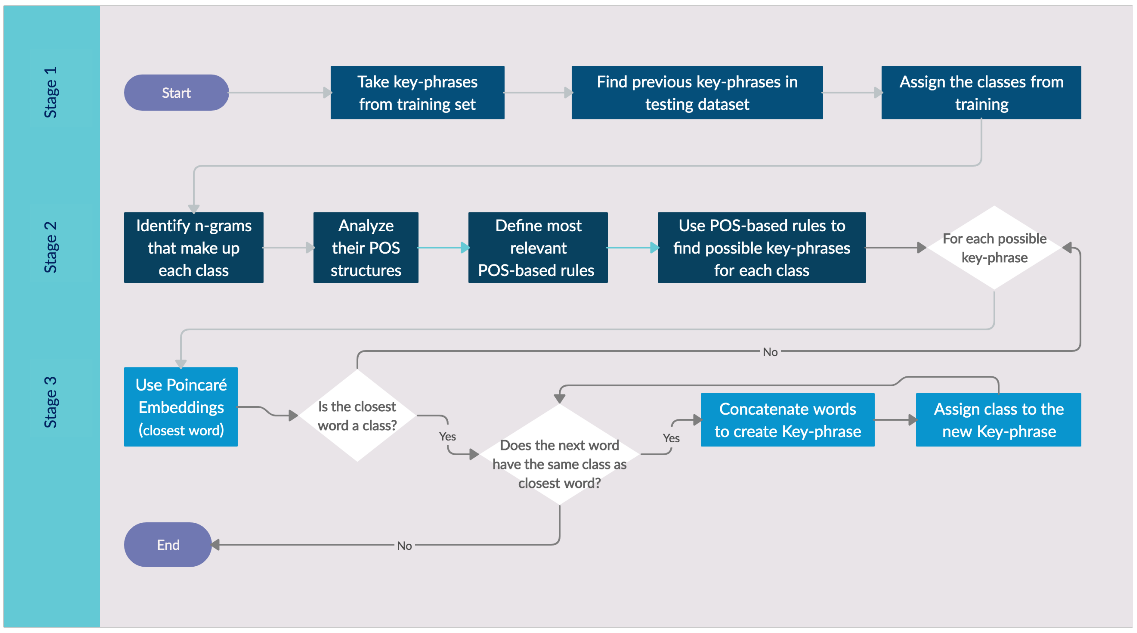
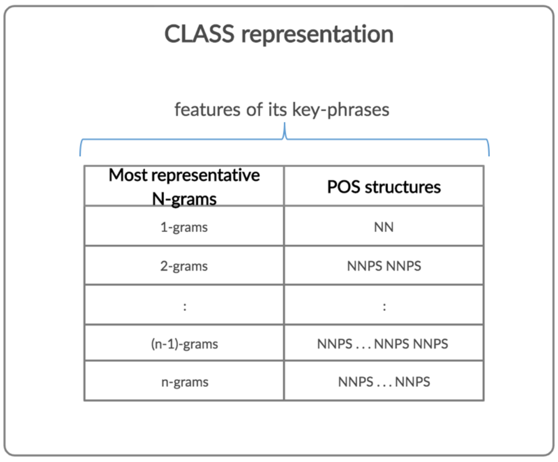

Stage 1
Stage 2
Stage 3
Start
Take key-phrases from training set
Find previous key-phrases in testing dataset
Assign the classes from training
Identify n-grams that make up each class
Analyze their POS structures
Define most relevant POS-based rules
Use POS-based rules to find possible key-phrases for each class
For each possible key-phrase
No
Use Poincaré Embeddings (closest word)
Is the closest word a class?
Yes
Does the next word have the same class as closest word?
Yes
Concatenate words to create Key-phrase
Assign class to the new Key-phrase
End
No

Fig. 7.1 Workflow diagram implemented on the classifier

- **Search POS structures (Stage 2).** In stage 2, first an analysis of the key phrases of each class is performed, thus determining the number of most used n-grams in each class, as well as their POS structures. Thus, only those structures with the most occurrences for each class will be considered, as seen in Figure 7.2. Once these most representative features of each class have been selected, they will be searched in the test dataset, and then there will be possible key phrases to be tagged.
- **Use Poincaré Embeddings and most frequent n-gram sizes (Stage 3).** By employing Poincaré Embedding models, with the most frequent n-gram sizes it is verified if the words in that windows size have the same tag as one of the 5 closest neighbors. If all of the words on the window size have the same tag, then they are tagged and considered as a key phrase. A more detailed explanation of how this stage works is shown in Figure 7.3.

In second stage possible key phrases are found considering features of each class, but is in stage 3 that these key phrases are tagged or discarded.

In stage 3, for each of the words that make up a possible key phrase, its closest neighbor is obtained, and if the closest neighbor for any of these words is one of the classes used in the corpus, then this word is tagged with that class and becomes a seed word or root word. Then a window of 4 words on the left and 4 on the right is used, and the 5 closest neighbors are obtained for each of those words and if the same class as the root word is found within the


CLASS representation
features of its key-phrases

Most representative N-grams	POS structures
1-grams	NN
2-grams	NNPS NNPS
:	:
(n-1)-grams	NNPS . . . NNPS NNPS
n-grams	NNPS . . . NNPS

Fig. 7.2 The most representative POS structures for each class considering its key phrase length (n-grams).

closest neighbors, then this word is added to the root word, and so on until the words inside the window are finished. At the end, a key phrase is obtained with the words that share the same tag within the range of the window.

7.2.1 Algorithm based on Poincaré embeddings

For our research case, we want to learn key phrase embeddings using hyperbolic space. In hyperbolic space, the circumference of the circle and disc area grows exponentially with the radius, while in Euclidean space they only grow quadratic and linearly. This makes the hyperbolic space-efficient for embedding hierarchical structures such as trees, where the number of nodes grows exponentially with depth. Thus simultaneously capturing the similarity between words and their hierarchy.

The Poincaré ball model has been used to represent this hyperbolic space because multiple hierarchies can co-exist in datasets such as text corpora, which is not always possible to model in two dimensions. Another reason is that it allows more degrees of freedom during an optimization process in order to find a better embedding. The Poincaré ball model (β^d) consists of points within the unit ball β^d, in which the distance between two points $u, v \in \beta^d$ is:

$$d(u,v) = \cosh^{-1}(1 + 2\frac{\|u-v\|^2}{(1-\|u\|^2)(1-\|v\|^2)}) \tag{7.1}$$

So while $\|u\|$ approaches 1, its distance to almost all other points will increase exponentially, so the other points (leaf nodes) will be placed on the edges of the Poincaré ball.

However, an efficient representation of our data set will place the root nodes near the origin and the leaf nodes near the edges of the Poincaré ball, to ensure that the root nodes are relatively close to all points, while the leaf nodes are relatively distant from most other leaf nodes.

Based on this definition, below we explain how we use Poincaré embeddings to obtain the closest words in the embedding space for a given word. For this, we will take a piece of text and apply the algorithm that our classifier model executes to label key phrases. As mentioned in chapter 6, before training the Poincaré embeddings, the dataset was converted to the Stanford format (structured in double column) being the first column for tokens or words and the second column for the class assigned to each token (one of the 6 classes previously mentioned) or 0 if the token does not have a class assigned. Then, a second conversion was applied to the corpus, passing everything to one line, and each token separated for just a blank space. It allows to know if a for a given word there is a class or category near.

Suppose the following text fragment:

"our client is looking for a Junior .NET developer to work as part of their cork based software team"

and its corresponding POS tagging:

"PRP$ NN VBZ VBG IN DT JJ ***NNP NN*** *TO VB IN NN IN PRP$ NN VBN NN NN"*

In stage 1, the classifier detects *".NET"* as *Knowledge"*, since it was previously labeled as *knowledge* that companies request from people in order to take a position within the company. On the other hand, the classifier also labels *"developer"* as *Role* since this term was previously labeled as a *Role* within the training set. So these words are momentarily labeled with those assigned values.

In stage 2, the classifier searches for the most representative POS structures of each class. In this example, our classifier identifies the pattern *"NNP NN"* as a possible key phrase of the class *Role*. This means that *".NET developer"* is a possible key phrase, but we do not know if the words that precede or come from it can also be part of the key phrase to be identified. For this, if we remember the distribution of N-grams observed in chapter 3, we can visualize it in Figure 3.1 that the class *Role* is composed mainly of bigrams and trigrams, so it would be very likely that *".NET developer"* is complete or missing a word. To discover this, it is time to make use of the Poincaré embeddings to obtain the closest words in the embedding space to each of them.

So in stage 3, for the word *".NET"* we get its 5 closest words in the embedding space, as well as for the 4 words that precede it. And for the word "developer" we get its 5 closest words in the embedding space, as well as for the 4 words that come from it. This experiment is shown in Table 7.1, where we can clearly see that *"Junior"*, *".NET"* and *"developer"* have the tag *Role* in common.

Table 7.1 Poincaré embeddings for closest words

Root word	Closest words				
	closest w1	closest w2	closest w3	closest w4	closest w5
looking	**0**	Useful	institution	look	personable
for	**0**	Talent	fair	sheets	introducing
a	**0**	sole	maintains	career	targeted
Junior	**Role**	Front-end	full-stack	consultant	mid-level
.NET	Knowledge	**Role**	JDBI	architecture	Jboss
developer	**Role**	mid-level	Studies	Background	Front-end
to	**0**	sites	Ballymount	detail	thus
work	**0**	Responsibility	seeking	multi-site	timescales
as	**0**	viewed	Transformation	estimating	smarter
part	**0**	negotiate	hour	failures	colleague

It is possible to observe that for the word *".NET"* its closest neighbor is *Knowledge* because .NET is a framework, and this is categorized as the knowledge that a person acquires to develop code. So *".NET"* by itself is knowledge. However, when it is part of this context we see that ***"Junior .NET developer"*** collectively refers to a Role or position within a company. So the *Role* tag has more impact than the *Knowledge* tag in this case. It is also observed that *"Junior"* and *"developer"* are closely related since they share 3 words out of 5 and both have *Role* in the first place. So *".NET"* could not be referring to something else by itself but is part of *"Junior"* and *"developer"*. So to *".NET"*, the label is reassigned, and now it becomes *Role*, like *"Junior"* and *"developer"*, obtaining their final value together. With the rest of the words, it is possible to observe that none is related to this key phrase, since they all have the label *"0"* as their closest neighbor, and they do not share words in common. So these words are discarded, and so we get a final key phrase.

7.3 Classifier Performance Using Different Features

The use of features improves the performance of the proposed classifier thanks to the fact that these characteristics allow key phrases to be identified more accurately. To demonstrate the

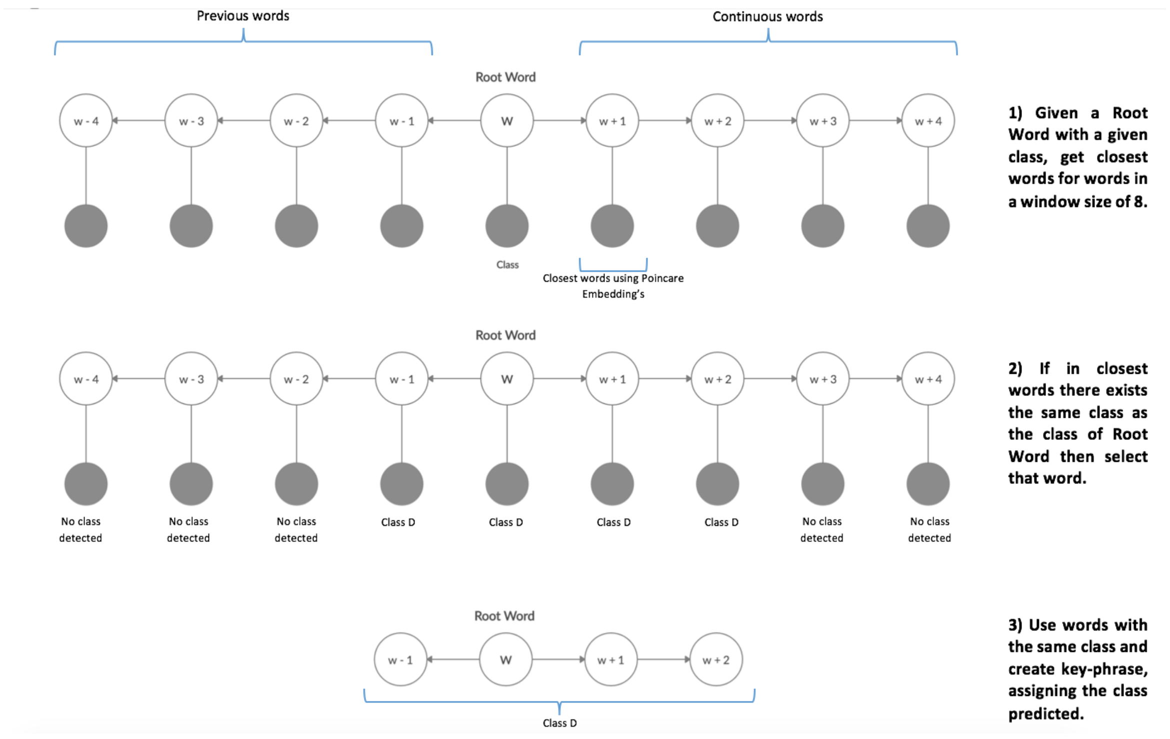

Previous words
Continuous words
Root Word
w - 4
w - 3
w - 2
w - 1
W
w + 1
w + 2
w + 3
w + 4
Class
Closest words using Poincare Embedding's
1) Given a Root Word with a given class, get closest words for words in a window size of 8.
No class detected
Class D
2) If in closest words there exists the same class as the class of Root Word then select that word.
3) Use words with the same class and create key-phrase, assigning the class predicted.

Fig. 7.3 Representation of how Poincaré Models are used to predict key phrases.

usefulness of these features, we present a series of experiments, where the first experiment only considers the first stage of the diagram, the second experiment considers the first and second stages of the diagram, and the third experiment considers all the stages of the diagram. V-fold cross validation method was used to measure the classifier's performance considering different features presented on the diagram, and the obtained results are presented in Table 7.2 as Precision, Recall and F-1.

7.3.1 Experiments and Results

Table 7.2 Performance of classifier when using different stages of the workflow process.

Execution	Stage	Class	Measures		
			Precision	Recall	F-1
1	S1	Character	0.6627	0.4710	**0.5507**
		Knowledge	0.6243	0.4371	0.5142
		Responsibility	0.3958	0.0810	0.1345
		Role	0.4743	0.3333	0.3915
		Skill	0.5894	**0.3477**	0.4374

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Talent	**0.7575**	0.3048	0.4347
	S1 & S2	Character	0.625	0.5462	0.5829
		Knowledge	0.6051	0.6164	**0.6107**
		Responsibility	0.3614	0.2842	0.3182
		Role	0.3851	**0.7572**	0.5106
		Skill	0.6159	0.4461	0.5174
		Talent	**0.7631**	0.3766	0.5043
	S1, S2, & S3	Character	0.8907	0.6794	0.7709
		Knowledge	0.8822	0.7132	**0.7887**
		Responsibility	0.5600	0.3686	0.4446
		Role	0.4927	**0.8055**	0.6114
		Skill	0.8806	0.5711	0.6928
		Talent	**0.9230**	0.4285	0.5853
2	S1	Character	0.7153	**0.6078**	**0.6572**
		Knowledge	0.7321	0.4327	0.5439
		Responsibility	0.5285	0.1332	0.2128
		Role	0.5734	0.3445	0.4304
		Skill	0.7275	0.3555	0.4776
		Talent	**0.7555**	0.2635	0.3908
	S1 & S2	Character	0.6666	0.6622	**0.6644**
		Knowledge	0.6136	0.6253	0.6194
		Responsibility	0.4341	0.3004	0.3551
		Role	0.4910	**0.7454**	0.5920
		Skill	0.7189	0.4916	0.5839
		Talent	**0.7454**	0.3306	0.4581
	S1, S2 & S3	Character	0.8378	0.7948	**0.8157**
		Knowledge	0.7968	0.6751	0.7309
		Responsibility	0.6870	0.4035	0.5084
		Role	0.6904	**0.8079**	0.7445
		Skill	0.9060	0.6113	0.7300
		Talent	**0.9180**	0.4179	0.5743
3	S1	Character	0.7481	**0.5260**	**0.6177**
		Knowledge	0.6111	0.4247	0.5011

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Responsibility	0.5869	0.0898	0.1557
		Role	0.6371	0.3272	0.4324
		Skill	**0.7973**	0.3848	0.5191
		Talent	0.66	0.2682	0.3815
	S1 & S2	Character	**0.7666**	**0.6084**	**0.6784**
		Knowledge	0.5649	0.5878	0.5761
		Responsibility	0.4222	0.2149	0.2848
		Role	0.4666	0.6596	0.5466
		Skill	0.7172	0.4551	0.5569
		Talent	0.6129	0.3333	0.4318
	S1, S2 & S3	Character	**0.8944**	**0.7705**	**0.8279**
		Knowledge	0.8569	0.7209	0.7830
		Responsibility	0.6551	0.3336	0.4421
		Role	0.5584	0.7381	0.6358
		Skill	0.8854	0.5578	0.6845
		Talent	0.8	0.4341	0.5628
4	S1	Character	0.6808	**0.6713**	**0.6760**
		Knowledge	0.6448	0.3232	0.4306
		Responsibility	0.6462	0.1096	0.1875
		Role	0.6991	0.3910	0.5015
		Skill	0.6845	0.4909	0.5718
		Talent	**0.7727**	0.4303	0.5528
	S1 & S2	Character	0.6329	0.6993	**0.6644**
		Knowledge	0.6028	0.4780	0.5332
		Responsibility	0.4300	0.3066	0.3580
		Role	0.4107	**0.7323**	0.5263
		Skill	0.6214	0.5586	0.5883
		Talent	**0.6909**	0.4810	0.5671
	S1, S2 & S3	Character	0.8555	0.77	**0.8105**
		Knowledge	**0.8632**	0.6467	0.7394
		Responsibility	0.5775	0.3787	0.4574
		Role	0.4859	**0.7723**	0.5965
		Skill	0.8560	0.6497	0.7387

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Talent	0.7636	0.5060	0.6086
5	S1	Character	0.6846	**0.5894**	**0.6334**
		Knowledge	0.5810	0.4319	0.4955
		Responsibility	0.6330	0.0975	0.1691
		Role	0.7115	0.3378	0.4582
		Skill	0.7759	0.5132	0.6178
		Talent	**0.8125**	0.3577	0.4968
	S1 & S2	Character	0.6474	**0.6778**	0.6622
		Knowledge	0.5316	0.6045	0.5657
		Responsibility	0.4090	0.2873	0.3375
		Role	0.4607	0.6459	0.5378
		Skill	0.7044	0.5944	**0.6447**
		Talent	**0.7924**	0.4038	0.5350
	S1, S2 & S3	Character	0.8172	**0.7676**	**0.7916**
		Knowledge	0.8251	0.7454	0.7832
		Responsibility	0.5575	0.375	0.4484
		Role	0.5287	0.7352	0.6151
		Skill	0.8579	0.6570	0.7441
		Talent	**0.9344**	0.4830	0.6368
6	S1	Character	0.7573	**0.7202**	**0.7383**
		Knowledge	**0.7586**	0.4649	0.5765
		Responsibility	0.5687	0.1243	0.2040
		Role	0.6347	0.3333	0.4371
		Skill	0.7457	0.3689	0.4936
		Talent	0.7333	0.2619	0.3859
	S1 & S2	Character	0.7133	**0.7887**	**0.7491**
		Knowledge	0.6793	0.6112	0.6435
		Responsibility	0.4781	0.3067	0.3737
		Role	0.4242	0.6146	0.5019
		Skill	0.6358	0.4681	0.5392
		Talent	**0.7241**	0.3387	0.4615
	S1, S2 & S3	Character	0.8333	**0.8474**	**0.8403**
		Knowledge	**0.8794**	0.7344	0.8004

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Responsibility	0.6673	0.4002	0.5004
		Role	0.5833	0.7085	0.6398
		Skill	0.8248	0.5499	0.6598
		Talent	0.8125	0.4	0.5360
7	S1	Character	**0.7**	**0.5796**	**0.6341**
		Knowledge	0.5877	0.4136	0.4855
		Responsibility	0.5222	0.0542	0.0983
		Role	0.6875	0.2796	0.3975
		Skill	0.6784	0.3764	0.4842
		Talent	0.6363	0.1764	0.2763
	S1 & S2	Character	**0.6322**	**0.6577**	**0.6447**
		Knowledge	0.5662	0.6031	0.5840
		Responsibility	0.3874	0.2279	0.2870
		Role	0.5138	0.6491	0.5736
		Skill	0.5691	0.4368	0.4943
		Talent	0.6046	0.2280	0.3312
	S1, S2 & S3	Character	0.8097	**0.7641**	0.7862
		Knowledge	0.8546	0.7444	**0.7957**
		Responsibility	0.5929	0.3471	0.4378
		Role	0.6759	0.7519	0.7119
		Skill	0.8181	0.5777	0.6772
		Talent	**0.8775**	0.3467	0.4971
8	S1	Character	0.7325	**0.4809**	**0.5806**
		Knowledge	0.5232	0.4649	0.4924
		Responsibility	0.6041	0.0855	0.1498
		Role	**0.7394**	0.4069	0.525
		Skill	0.6708	0.375	0.4810
		Talent	0.6571	0.2446	0.3565
	S1 & S2	Character	**0.7378**	0.608	**0.6666**
		Knowledge	0.4896	0.6450	0.5567
		Responsibility	0.3925	0.2584	0.3116
		Role	0.4856	**0.7072**	0.5758
		Skill	0.6258	0.4790	0.5426

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Talent	0.6341	0.2857	0.3939
	S1, S2 & S3	Character	0.8531	0.7393	0.7922
		Knowledge	0.8354	**0.7696**	**0.8012**
		Responsibility	0.5723	0.3522	0.4360
		Role	0.5757	0.76	0.6551
		Skill	**0.8607**	0.5811	0.6938
		Talent	0.8571	0.3529	0.5
9	S1	Character	**0.7051**	**0.6508**	**0.6769**
		Knowledge	0.6775	0.3992	0.5024
		Responsibility	0.5919	0.1607	0.2528
		Role	0.6810	0.3726	0.4817
		Skill	0.6620	0.4155	0.5106
		Talent	0.6760	0.3809	0.4873
	S1 & S2	Character	**0.7011**	**0.7393**	**0.7197**
		Knowledge	0.5436	0.5502	0.5469
		Responsibility	0.4126	0.2792	0.3330
		Role	0.4691	0.685	0.5569
		Skill	0.6463	0.5033	0.5659
		Talent	0.6463	0.4380	0.5221
	S1, S2 & S3	Character	0.8472	**0.8037**	**0.8249**
		Knowledge	0.7659	0.6941	0.7282
		Responsibility	0.6342	0.4061	0.4951
		Role	0.6112	0.7863	0.6878
		Skill	**0.8918**	0.6168	0.7292
		Talent	0.7674	0.5	0.6055
10	S1	Character	0.7031	**0.5487**	**0.6164**
		Knowledge	0.6105	0.4157	0.4946
		Responsibility	**0.7698**	0.1121	0.1957
		Role	0.5029	0.3021	0.3775
		Skill	0.6579	0.3710	0.4745
		Talent	0.7021	0.3437	0.4615
	S1 & S2	Character	**0.7103**	0.6477	**0.6776**
		Knowledge	0.5766	0.5522	0.5641

Continued on next page

Table 7.2 – *Continued from previous page*

Execution	Stage	Class	Precision	Recall	F-1
		Responsibility	0.4583	0.2128	0.2906
		Role	0.4743	**0.6801**	0.5589
		Skill	0.5833	0.4686	0.5197
		Talent	0.6666	0.3870	0.4897
	S1, S2 & S3	Character	**0.8693**	**0.7355**	**0.7968**
		Knowledge	0.8302	0.6509	0.7297
		Responsibility	0.6035	0.3078	0.4077
		Role	0.5747	0.7616	0.6551
		Skill	0.8453	0.5909	0.6956
		Talent	0.8305	0.4851	0.6125

It is observed in Table 7.2 that the performance in recognizing key phrases is always higher when using the three stages than when using only stage 1 and stage 2. In addition, the table shows that in most of the executions the Recall is high for the *Character* class, which is easily explained as the key phrases in this class are primarily made up of unigrams and bigrams. The *Role* class also stands out, with a high recall. The explanation for this is that more than 50% of the key phrases that make up this class are bigrams, and they have the POS-tag NNP (proper noun in singular form) same which facilitates their correct identification. On the other hand, the *Talent* class tends to have a high Precision over the others in most of the executions the This is because most of the key phrases that make up the class are bigrams and trigrams, which have the POS-tag NN (singular noun). Also, the *Character* class is the second to show better Precision. Finally, the *Character* class has a higher F1 measure than the other classes, this is because it was the only class with a High Precision and Recall.

Although in Table 7.2 we can observe the obtained results of each class in each execution for its Precision, Recall, and F1, it is hard to recognize the performance in general for each category or class in each of the experiments using the different stages. For that reason, we have averaged the performance of each class in each stage, and the results are shown in Table Table 7.3.

Table 7.3 shows that Stage 3 plays an important role in recognizing key phrases. The main reason for having a notable improvement here is the fact of using poincaré embeddings, since in stages one and two the most common and highly secure key phrases have been tagged, so in stage 3 the amount of untagged information it has decreased considerably and

Table 7.3 Performance of classes when using different stages of the workflow process

		Classifier Model		
Class	**Measure**	**Stage 1**	**Stage 1 & 2**	**Stage 1 & 2 & 3**
Character	Precision	0.7089	0.6833	0.8508
	Recall	0.5846	0.6635	0.7672
	F-1	0.6381	0.6710	0.8057
Knowledge	Precision	0.6351	0.5773	0.8390
	Recall	0.4208	0.5874	0.7095
	F-1	0.5037	0.5800	0.7681
Responsibility	Precision	0.5847	0.4186	0.6107
	Recall	0.1048	0.2678	0.3673
	F-1	0.1760	0.3250	0.4578
Role	Precision	0.6341	0.4581	0.5777
	Recall	0.3428	0.6876	0.7627
	F-1	0.4433	0.5480	0.6553
Skill	Precision	0.6990	0.6438	0.8627
	Recall	0.3999	0.4902	0.5963
	F-1	0.5068	0.5553	0.7046
Talent	Precision	0.7163	0.6880	0.8484
	Recall	0.3032	0.3603	0.4354
	F-1	0.4224	0.4695	0.5719

thus it is easier to analyze the remaining data and determine through the closest words to a root word potentially identified as belonging to a category or class. To further simplify the reading of the classifier's performance, we have graphed the results in Figure 7.4.

Figure 7.4 shows a comparison of the model performance when implementing different stages of the classifier for every class.

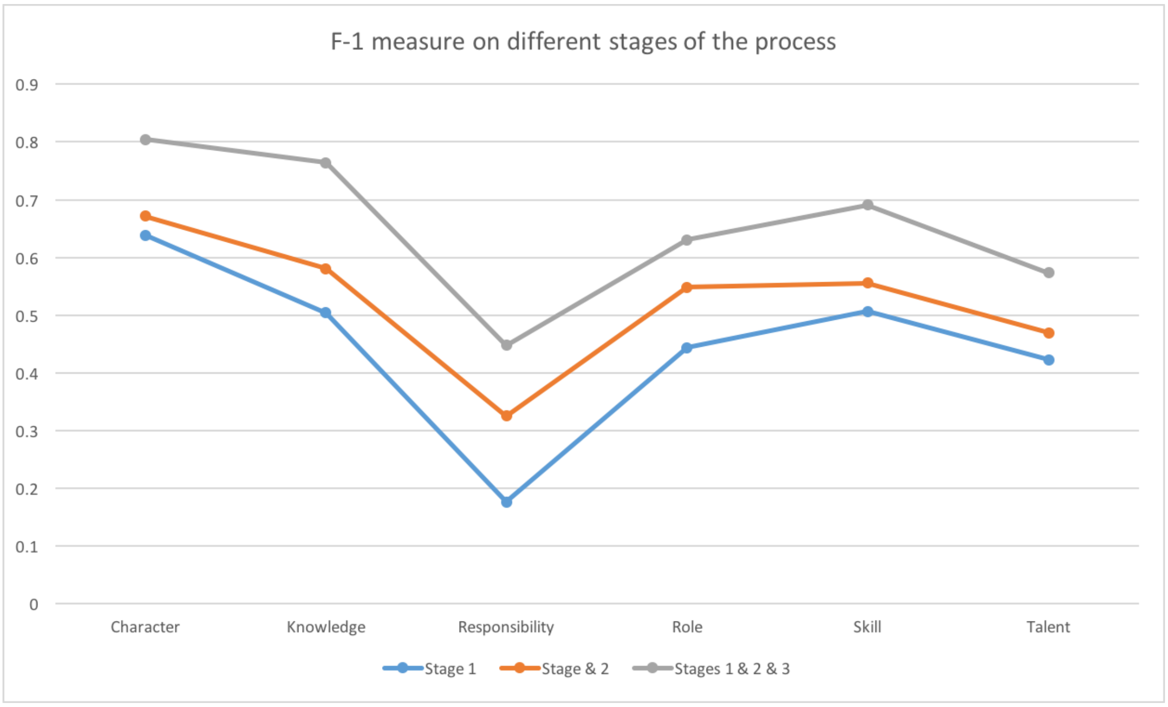


Fig. 7.4 Classifier performance when using only the first stage of the diagram

In stage one all those key phrases that exist in the training corpus are tagged, that is why the recognition level is not so high, in addition to that the Precision in this stage is generally high as observed in Table 7.3. However, the second stage allows POS structures to be recognized based on the POS structures of each class in the training corpus, thus increasing the Recall level and decreasing Precision. Finally, the third stage allows to create a better balance between Precision and Recall, when using the Poincare Embeddings since they establish a relationship between a root word with an assigned class and allows to know which words around it belong to the same class and therefore to the same key phrase.

7.4 Conclusions

Now it is time to compare the results of the proposed classifier using the complete process (3 stages) against the classifiers previously trained in the previous chapters (Stanford, CRF ++, MITIE, and our proposed graph structure). We carry out a comparison contrasting the results by class and by classifier, making use of the three performance measures (Precision, Recall, and F-1). This comparison is clearly shown in Table 7.4, where the best results obtained by class and by measure are highlighted in bold letters.

Table 7.4 Final results by classifier and by class of the entire experiment

Model	Class	Measures		
		Precision	Recall	F-1
Stanford	Character	0.8772	0.5221	0.6513
	Knowledge	0.7132	0.5832	0.6490
	Responsibility	0.5038	0.3359	0.3882
	Role	0.8072	0.6232	**0.6883**
	Skill	0.7254	0.4334	0.5400
	Talent	0.6672	0.2357	0.3494
CRF++	Character	**0.9343**	0.2962	0.4447
	Knowledge	0.7603	0.3349	0.4631
	Responsibility	0.6055	0.1340	0.2185
	Role	**0.8717**	0.3530	0.4953
	Skill	0.8438	0.2633	0.3992
	Talent	**0.9127**	0.0992	0.1763
MITIE	Character	0.6585	0.4002	0.4962
	Knowledge	0.6298	0.4850	0.5469
	Responsibility	0.4506	0.2171	0.2911
	Role	0.7484	0.5825	0.6511
	Skill	0.5784	0.3636	0.4430
	Talent	0.4801	0.1343	0.2081
Graph Structure	Character	0.6094	0.3889	0.4724
	Knowledge	0.7311	0.3458	0.4686
	Responsibility	0.3881	0.0482	0.0855
	Role	0.5695	0.3835	0.4553
	Skill	0.6620	0.2388	0.3497
	Talent	0.5363	0.1900	0.2772
Poincaré Embeddings Classifier	Character	0.8508	**0.7672**	**0.8057**
	Knowledge	**0.8390**	**0.7095**	**0.7681**
	Responsibility	**0.6107**	**0.3673**	**0.4578**

Continued on next page

Table 7.4 – *Continued from previous page*

Model	Class	Precision	Recall	F-1
	Role	0.5777	**0.7627**	0.6553
	Skill	**0.8627**	**0.5963**	**0.7046**
	Talent	0.8484	**0.4354**	**0.5719**

The results in Table 7.4 show that the CRF++ classifier achieves a higher *Precision* than the other classifiers in almost all classes. With the exception of the *Knowledge*, *Responsibility* and *Skill* classes, where our classifier obtained the best results, placing us in second place of *Precision*. It means that our model is correct about 76.4% of the time when predicting a class or category for a key phrase. Which implies that it produces few false positives.

Regarding the Recall measure, our proposed classifier model manages to obtain the best performance in each of the classes or categories, placing us in first place with 60% on average and thus surpassing the Stanford classifier that had shown the best performance over the other classifiers in previous experiments. It means that of all the key phrases tagged with any of the 6 classes, our classifier manages to correctly identify about 60

Finally, we can see that our classifier outperforms the other classifiers in general with a higher F-1 measure in each of the classes, except for the *Role* class, where the Stanford classifier achieves the best performance. This poor performance in the *Role* class by our classifier could be because this particular class is 51% composed of bigrams. And since the *Skill* class is 53% made up of bigrams, our classifier could get confused when considering a key phrase as *Skill* when it was actually *Role*.

However, we can observe a remarkable performance, demonstrating the advantage of using the Poincaré Embeddings for the representation of words. And at the same time its features of obtaining the closest words given a root word, considering its semantic relationship.

We achieved the implementation in our classifier of the features observed in chapter 3, about the POS structures and the N-grams that make up each class of our dataset. Because of that, we were able to identify possible key phrases in new Job Descriptions. To make sure that these were really key phrases, we employed the Poincaré embeddings, which managed to encode the key phrases of our Gold Standard efficiently. Our proposed model showed a higher performance than Stanford, CRF ++, MITIE and, our Graph Structure proposed in chapter 5.

Chapter 8

Conclusions

In this chapter, we summarize the findings obtained during this research project. We discuss the hypotheses that guided this work. In addition, we also briefly explain what we did to find an answer to these starting points. So we generally address the steps that were necessary to answer and direct the investigation until its completion. We also mention the possible paths that the work carried out during the research could take.

8.1 Research Work Overview

Our research work was born from the necessity to classify textual information through key phrases that distinguish one text from another. In this sense, it becomes complicated when we want to know if two texts or more address the same topic, even without containing exactly the same key phrases syntactically speaking. It means that two texts could be addressing the same topic using similar concepts. For that reason, we have considered analyzing the texts through their key phrases and at the same time finding some degree of similarity in the key phrases. As a result of this, we formulated the following hypotheses that guided our research work:

- **Hypotheses 1**: The classification of texts can be done by identifying their most relevant concepts or key phrases. It implies refining the recognition of key phrases, considering that two key phrases can be semantically similar even if they do not use the same terms.
- **Hypotheses 2**: Using word embeddings to capture the semantic relationships between words can improve the identification of key phrases.

To answer these hypotheses, we had to carry out a series of tasks that allowed us to find the best way to conduct this research. The tasks we perform are listed below:

- We carry out an in-depth study on the state of the art in text classification and key phrase extraction.
- We investigate methods for the recognition of named entities, and in this way, we deal with these proposals for the recognition of key phrases.
- We collect text documents from the page https://www.jobs.ie/ which are Job Descriptions in the Information Technology area.
- We developed a web system with which we could manually tag the key phrases of the set of documents that we collected previously. And in this way reliably extract the key phrases that made up our Gold Standard or training set. These key phrases were tagged using the following classes: Role, Knowledge, Skill, Character, Responsibility, and Talent.
- We performed experiments on our Gold Standard using the NER classifiers: Stanford, CRF ++, and MITIE.
- We treated key phrases as N-grams so that we were able to determine how many words each class's key phrases are typically made of. This helped improve key phrase recognition with a lower margin of error.
- We applied a POS tagging to each of the key phrases and analyzed the features of the key phrases for each class. With this, it was possible to determine some rules for the identification of key phrases from a specific class.
- For each of the words that make up our Gold Standard, we obtained their synonyms through a Thesaurus, and with these synonyms, we generated new key phrases from the original ones.
- To make sure that the new key phrases were reliable (phrases that are actually used in texts over the Internet) we performed a search for each of them on the Internet with the Google results API. So we only keep those key phrases that have the greatest use on electronic texts on the Internet.
- Although the new key phrases that we generated through the use of synonyms could exist on the Internet, we employed a second selection filter that consists of using the Wu&Palmer semantic similarity measure. In this way, we discarded key phrases that are not closely related to the initial concept of the original key phrase.

- Based on the analysis obtained with the use of POS tagging, we applied these characteristics observed in the original set as rules to preserve key phrases from the new set created.
- A final selection filter was applied on the resulting set, which was manually discarding those key phrases that had some syntactic error. Thus we obtained the real set of new key phrases similar to the original set.
- To measure the quality of the new set of key phrases created, we trained a classifier containing the new key phrases and compared its performance with that of the classifier that only contains the set of the Gold Standard. We explained the results obtained from this experiment through the use of the Jaccard Similarity Coefficient.
- We proposed a graph-based enriched data structure for key phrase storage and recognition. We explained the construction process of this graph structure with our Gold Standard.
- We used our graph structure to recognize named entities and compared the results obtained against the classifiers: Stanford, CRF ++, and MITIE.
- We studied the use of models for word representations such as Word2Vec, Glove, and Poincaré Embeddings.
- We ran experiments on our Gold Standard to discern the performance of each of these models for word representations.
- We found that Poincaré embeddings are better at capturing semantic relationships between words as they employ a hyperbolic space. For this reason, we discarded the implementation of Word2Vec and Glove since being Euclidian embeddings, the relationships between words are not captured so well in these models.
- We utilized the Poincaré Embeddings to improve the key phrase recognition of our proposed graph structure. Through the mapping of the closest words given a root word with the Word Embeddings, the analyzed POS characteristics, and the study of the N-grams, we improved the recognition of key phrases in text documents.

Our work starts from the Web tagging system that we developed to extract the key phrases that were necessary to generate our Gold Standard as explained in *Appendix A* and thus be able to carry out the first experiments. Subsequently, an analysis was carried out on the set of key phrases of our Gold Standard. So we were able to observe specific characteristics for each of the categories or classes. This was achieved by representing key phrases with N-grams and POS tagging. These characteristics or features later served us to use them as rules in our classifier proposal, since they served as filters to discard or identify key phrases more precisely. In addition, we used the characteristics of the Poincaré Embeddings to refine the recognition of the key phrases that our classifier recognized.

Multiple experiments were conducted to determine which models are better than others and to determine which features were useful to us. Thus, we realized that the results varied between each class, and this is partly because some classes, such as *Knowledge*, *Character*, and *Role*, have more incidents in the texts that we tagged. But mainly it is because the key phrases of these classes are composed of few words (*unigrams, bigrams, trigrams*) and also because their POS tagging usually has a specific pattern (*NN, JJ NN, NNP NNP*), so that is easier to recognize them. In addition, it was possible to determine that the Poincaré Embeddings offered us better results than Word2Vec or GloVe when obtaining the closest words to a specific word. And so we can validate new key phrases when analyzing them word by word, identifying if the set of words that make it up are close to the words or terms of the key phrases of the Gold Standard set. The results obtained in our experiments were shown to have superior performance than the Stanford, CRF ++, and MITIE classifiers.

Our research work was carried out on text documents of the Job Descriptions type, which were specific to the Information Technology area. So our classifier performs well in this type of document, but in the same way, it can be extended to another type of Job Descriptions using the logic and techniques that we detail. Although it would be necessary to have a specific Gold Standard for the type of documents to be treated, our work also proposed a web-based tagger system if no tagged corpus already exists for a specific case or problem to be solved. This would be a possible limitation, as it would take a long time to manually tag a specific set of documents, just as it happened in our case. If more time were available, the Gold Standard could be extended further and thus improve the performance of our classifier method.

There is an interesting path this research work could take in the future. If we consider that our classifier works very well in recognizing key phrases for Job Descriptions, then big companies that offer thousands of jobs annually could save a lot of time in finding the right workers for those jobs. So it would only be enough to train the classifier on a specific set of Job Descriptions provided by the company, and the search would be carried out in the Résumés of the professional profiles that best meet those characteristics that the company needs. In this way, a lot of time would be saved in the interview and selection processes by the Human Resources staff in charge of hiring the employees for each job position.

References

Aguilar, G., López-Monroy, A.P., González, F.A. and Solorio, T. (2019) Modeling noisiness to recognize named entities using multitask neural networks on social media *arXiv preprint arXiv:1906.04129*

Akbik, A., Bergmann, T. and Vollgraf, R. (2019) Pooled contextualized embeddings for named entity recognition in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* pp. 724–728

Alzaidy, R., Caragea, C. and Giles, C.L. (2019) Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents in: *The World Wide Web Conference* WWW '19 p. 2551–2557 Association for Computing Machinery, New York, NY, USA

Ando, R.K. (2007) Biocreative ii gene mention tagging system at ibm watson in: *Proceedings of the Second BioCreative Challenge Evaluation Workshop* vol. 23 pp. 101–103 Centro Nacional de Investigaciones Oncologicas (CNIO) Madrid, Spain

Augenstein, I., Das, M., Riedel, S., Vikraman, L. and McCallum, A. (2017) Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications *arXiv preprint arXiv:1704.02853*

Bari, M.S., Joty, S. and Jwalapuram, P. (2020) Zero-resource cross-lingual named entity recognition in: *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 34 pp. 7415–7423

Bennani-Smires, K., Musat, C., Hossmann, A., Baeriswyl, M. and Jaggi, M. (2018) Simple unsupervised keyphrase extraction using sentence embeddings *arXiv preprint arXiv:1801.04470*

Bird, S., Klein, E. and Loper", E. (2009) *"Natural Language Processing with Python"* "O'Reilly Media"

Buscaldi, D., David Hernandez, S. and Charnois, T. (2017) Classification of Keyphrases from Scientific Publications using WordNet and Word Embeddings in: *VADOR (Valorisation et Analyse des Données de la Recherche) 2017* Toulouse, France

Chandrasekaran, D. and Mago, V. (2020) Evolution of semantic similarity–a survey *arXiv preprint arXiv:2004.13820*

Chinchor, N. and Robinson, P. (1997) Muc-7 named entity task definition in: *Proceedings of the 7th Conference on Message Understanding* vol. 29 pp. 1–21

Clark, K., Luong, M.T., Manning, C.D. and Le, Q.V. (2018) Semi-supervised sequence modeling with cross-view training *arXiv preprint arXiv:1809.08370*

David. C. Lay, S.R.L. and McDonald, J.J. (2016) *Linear Algebra and Its Applications* PEARSON, United States of America

Dernoncourt, F., Lee, J.Y. and Szolovits, P. (2017) Neuroner: an easy-to-use program for named-entity recognition based on neural networks *arXiv preprint arXiv:1705.05487*

Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2018) Bert: Pre-training of deep bidirectional transformers for language understanding *arXiv preprint arXiv:1810.04805*

Dingare, S., Nissim, M., Finkel, J., Manning, C. and Grover, C. (2005) A system for identifying named entities in biomedical text: how results from two evaluations reflect on both the system and the evaluations *Comparative and functional genomics* **6**(1-2), pp. 77–85

Etaiwi, W., Awajan, A. and Suleiman, D. (2017) Statistical arabic name entity recognition approaches: A survey *Procedia computer science* **113**, pp. 57–64

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S. and Yates, A. (2005) Unsupervised named-entity extraction from the web: An experimental study *Artificial intelligence* **165**(1), pp. 91–134

Feng, Y., Bagheri, E., Ensan, F. and Jovanovic, J. (2017) The state of the art in semantic relatedness: a framework for comparison *The Knowledge Engineering Review* **32**, p. e10

Finkel, J.R., Grenager, T. and Manning, C. (2005) Incorporating non-local information into information extraction systems by gibbs sampling in: *Proceedings of the 43rd annual meeting on association for computational linguistics* pp. 363–370 Association for Computational Linguistics

Gollapalli, S.D., Li, X.L. and Yang, P. (2017) Incorporating expert knowledge into keyphrase extraction in: *Thirty-First AAAI Conference on Artificial Intelligence*

Grishman, R. and Sundheim, B. (1995) Design of the muc-6 evaluation in: *Proceedings of the 6th conference on Message understanding* pp. 1–11 Association for Computational Linguistics

Habibi, M., Weber, L., Neves, M., Wiegandt, D.L. and Leser, U. (2017) Deep learning with word embeddings improves biomedical named entity recognition *Bioinformatics* **33**(14), pp. i37–i48

Hao, Z., Lv, D., Li, Z., Cai, R., Wen, W. and Xu, B. (2021) Semi-supervised disentangled framework for transferable named entity recognition *Neural Networks* **135**, pp. 127–138

Harispe, S., Ranwez, S., Janaqi, S. and Montmain, J. (2017) Semantic similarity from natural language and ontology analysis *CoRR* **abs/1704.05295**

Harris, Z.S. (1954) Distributional structure *<i>WORD</i>* **10**(2-3), pp. 146–162

Jeffrey Pennington, Richard Socher, C.D.M. (2014) Glove: Global vectors for word representation in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* pp. 1532–1543

Joachims, T. (1996) A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report Carnegie-mellon univ pittsburgh pa dept of computer science

King, D.E. (2009) Dlib-ml: A machine learning toolkit *Journal of Machine Learning Research* **10**, pp. 1755–1758

Kuru, O., Can, O.A. and Yuret, D. (2016) Charner: Character-level named entity recognition in: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* pp. 911–921

Lafferty, J., McCallum, A. and Pereira, F.C. (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data

Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S. and He, L. (2020) A survey on text classification: From shallow to deep learning *arXiv preprint arXiv:2008.00364*

Liao, W. and Veeramachaneni, S. (2009) A simple semi-supervised algorithm for named entity recognition in: *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing* pp. 58–65

Liu, Z., Chen, X., Zheng, Y. and Sun, M. (2011) Automatic keyphrase extraction by bridging vocabulary gap in: *Proceedings of the Fifteenth Conference on Computational Natural Language Learning* pp. 135–144 Association for Computational Linguistics

Merriam Webster (2020) Merriam webster: Dictionary [online] https://www.merriam-webster.com/dictionary/

Mikheev, A. (1999) A knowledge-free method for capitalized word disambiguation in: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics* pp. 159–166 Association for Computational Linguistics

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013a) Efficient estimation of word representations in vector space *arXiv preprint arXiv:1301.3781*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013b) Distributed representations of words and phrases and their compositionality *arXiv preprint arXiv:1310.4546*

Miller, G.A. (1995) Wordnet: A lexical database for english. *Communications of the ACM* **38**, pp. 39–41

Mohamed, M. and Oussalah, M. (2014a) Identifying and extracting named entities from wikipedia database using entity infoboxes *International Journal of Advanced Computer Science and Applications* **5**(7)

Mohamed, M. and Oussalah, M. (2014b) Identifying and extracting named entities from wikipedia database using entity infoboxes *(IJACSA) International Journal of Advanced Computer Science and Applications.* **7**, pp. 164–169

Mothe, J., Ramiandrisoa, F. and Rasolomanana, M. (2018) Automatic keyphrase extraction using graph-based methods in: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* pp. 728–730

Munro, R. and Manning, C.D. (2012) Accurate unsupervised joint named-entity extraction from unaligned parallel text in: *Proceedings of the 4th Named Entity Workshop* pp. 21–29 Association for Computational Linguistics

Nadeau, D., Turney, P.D. and Matwin, S. (2006) Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity in: *Conference of the Canadian society for computational studies of intelligence* pp. 266–277 Springer

Nickel, M. and Kiela, D. (2017) Poincaré embeddings for learning hierarchical representations in: *Advances in neural information processing systems* pp. 6338–6347

Nigel Collier, Patrick Ruch, A.N., (Ed.) (2004) *JNLPBA '04: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications* Association for Computational Linguistics, USA

Palmer, D.D. and Day, D. (1997) A statistical profile of the named entity task in: *Fifth Conference on Applied Natural Language Processing* pp. 190–193

Paolozzi Stefano, F.F. and Grifoni, P. (2009) *"Handbook of Research on Digital Libraries: Design, Development, and Impact"* "Yin-Leng Theng, Schubert Foo, Dion Goh and Jin-Cheon Na"

Papagiannopoulou, E. and Tsoumakas, G. (2020) A review of keyphrase extraction *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **10**(2), p. e1339

Patil, N., Patil, A.S. and Pawar, B. (2016) Survey of named entity recognition systems with respect to indian and foreign languages *International Journal of Computer Applications* **134**(16)

Petasis, G., Vichot, F., Wolinski, F., Paliouras, G., Karkaletsis, V. and Spyropoulos, C.D. (2001) Using machine learning to maintain rule-based named-entity recognition and classification systems in: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics* pp. 426–433 Association for Computational Linguistics

Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. (2018) Deep contextualized word representations *arXiv preprint arXiv:1802.05365*

R. Grishman, B.S. (1996) Design of the muc-6 evaluation in: *Proceedings Sixth Message Understanding Conference (MUC-6)* pp. 1–11

Ritter, A., Clark, S., Etzioni, O. *et al.* (2011) Named entity recognition in tweets: an experimental study in: *Proceedings of the conference on empirical methods in natural language processing* pp. 1524–1534 Association for Computational Linguistics

Ruan, S., Li, H., Li, C. and Song, K. (2020) Class-specific deep feature weighting for naïve bayes text classifiers *IEEE Access* **8**, pp. 20151–20159

S. Tardif, J.R. Curran, T.M. (2009) Improved text categorisation for wikipedia named entities in: *Proceedings of the Australian Language Technology Workshop* pp. 104–108

Sahrawat, D., Mahata, D., Zhang, H., Kulkarni, M., Sharma, A., Gosangi, R., Stent, A., Kumar, Y., Shah, R.R. and Zimmermann, R. (2020) Keyphrase extraction as sequence labeling using contextualized embeddings in: *European Conference on Information Retrieval* pp. 328–335 Springer

Scott, S. and Matwin, S. (1998) Text classification using wordnet hypernyms in: *Usage of WordNet in Natural Language Processing Systems*

Shaalan, K. (2014) A survey of arabic named entity recognition and classification *Computational Linguistics* **40**(2), pp. 469–510

Sharnagat, R. (2014) Named entity recognition: A literature survey *Center For Indian Language Technology*

Sketch Engine (2020) Sketch engine: Pos tagger [online] https://www.sketchengine.eu/my_keywords/pos-tagger/

Slimani, T. (2013) Description and evaluation of semantic similarity measures approaches *arXiv preprint arXiv:1310.8059*

Spirling, A. and Rodríguez, P.L. (2019) Word embeddings what works, what doesn't, and how to tell the difference for applied research

Spirling, A. and Rodriguez, P.L. (2019) Word embeddings: What works, what doesn't, and how to tell the difference for applied research center for Data Science, New York University, Data Science Institute, Vanderbilt University and Instituto de Estudios Superiores de Administracion

Stanford University (2014) Glove: Global vectors for word representation [online] https://nlp.stanford.edu/projects/glove/

Wang, S., Zhou, W. and Jiang, C. (2020) A survey of word embeddings based on deep learning *Computing* **102**(3), pp. 717–740

Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C. and Nevill-Manning, C.G. (2005) Kea: Practical automated keyphrase extraction in: *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific* pp. 129–152 IGI global

Yadav, V. and Bethard, S. (2019) A survey on recent advances in named entity recognition from deep learning models *arXiv preprint arXiv:1910.11470*

Yadav, V., Sharp, R. and Bethard, S. (2018) Deep affix features improve neural named entity recognizers in: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics* pp. 167–172

Zong, W., Wu, F., Chu, L.K. and Sculli, D. (2015) A discriminative and semantic feature selection method for text categorization *International Journal of Production Economics* **165**, pp. 215–222

Appendix A

Web-based Tagger System

The proposed collaborative tagging system help users in the task of highlighting important information in plain text files. Additionally, it allows converting tagged texts into a structured format. The web-based system is proposed in order to exploit the relevant content information provided by tagger users, since actual collaborative tagging systems suffer from issues such as tag scarcity or ambiguous labeling. Approaches such as the proposed here can facilitate to obtain better quality in tags and in any domain, allowing to achieve significant improvements in information extraction through named entities extraction, avoiding the noise of information overload.

Since most of the textual data exist in an unstructured form it is important to produce structured data ready for post-processing, which is crucial to many applications of text mining such as text categorization, entity extraction, learning relations between named entities, etc. That is why the proposed web-based tool aims to help users in the task of highlighting the relevant information in plain text files and then by producing a structured version of the data ready to be used as corpora in the training of some models such as NER classifiers. There are so many reasons to have the tool tagger based on the web because it allows access to information at any device with internet connection. Also, it facilitates a huge number of users who can tag different text files simultaneously, increasing the number of tagged texts to be used as training data.

System architecture

Workflow

1. User first need to authenticate

2. If the authenticated user is an administrator, then the user is able to create, modify and delete as many users and labels as necessary; and also he can assign permissions of which labels can use every user depending on the domain they will be working.

3. If the user is authenticated as a tagger, then the user can upload plain text files in his account, as well as search and open a specific plain text file to start tagging the keywords in that document.

4. After that, the user is able to download the tagged plain text files in 3 different formats: InlineXML, Stanford highlighted (tab-separated columns) and a special format where the file includes the words preceding and proceeding the keywords tagged, so it can help to understand the context in which the keywords are taking place.

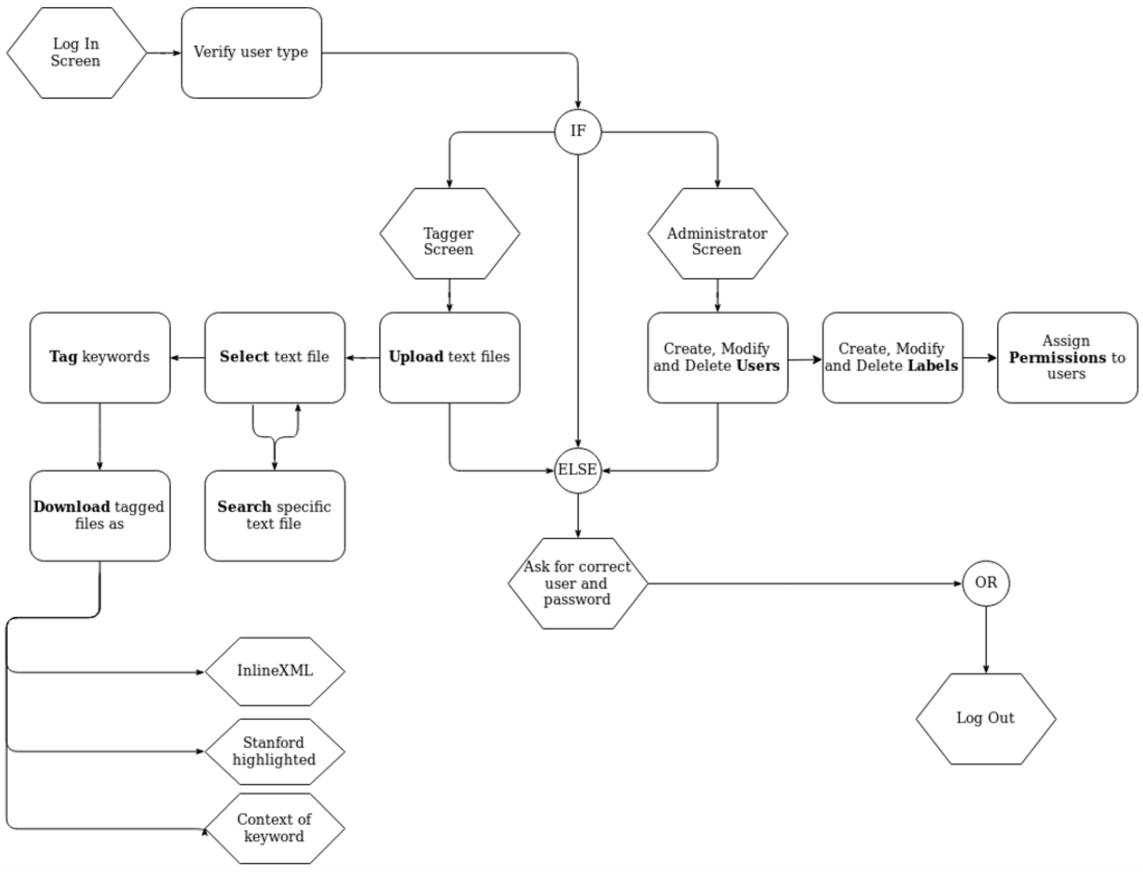


Fig. A.1 System Architecture

User authentication

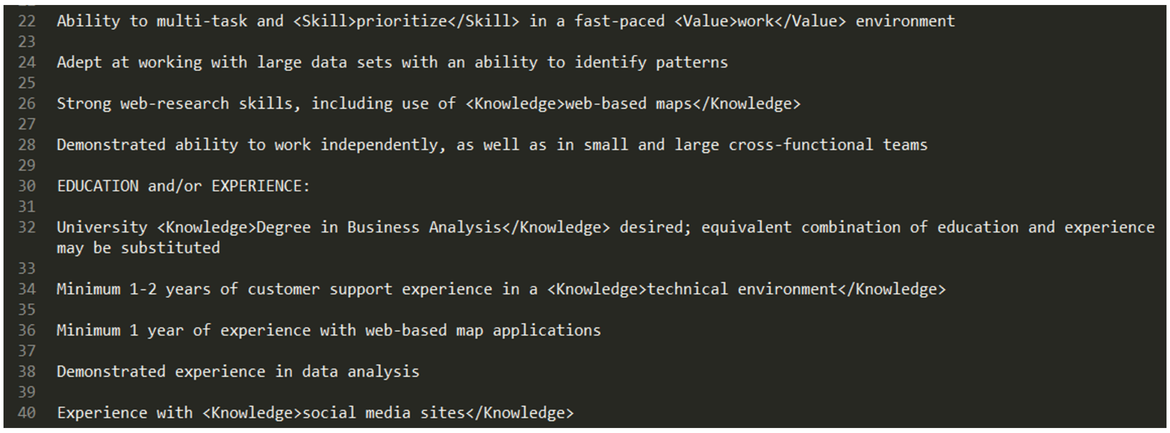
As there is a huge number of people tagging text files, it is necessary to authenticate the user session to know what plain text files and classes are allowed to use for a specific user. On the other hand, it facilitates access from any device, it means, once a text file was uploaded, the user can use it at any time, in any place and any device.

Tagger user

Taggers are main generators of high quality tagged data, since they have a good understanding in finding key concepts in textual data, differentiating from one and another context.

Once they are logged in the system, they are allowed to: upload, search and select specific plain text files, tag keywords or concepts in the text and then download tagged files. Alternatively, tagged files are available to download in three formats:

1. *InlineXML format:* It is a conversion of the original text into an XML representation where the output file contains the complete original text, but where each previously tagged concept appears between tags (similar to the HTML style) where the tag is the same that was assigned by the tagger user.

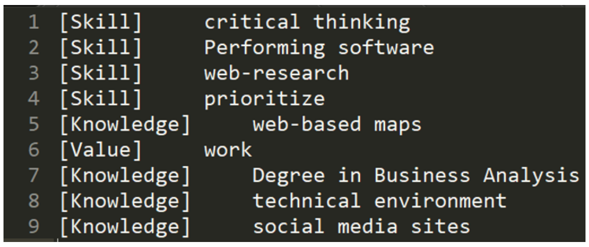


```
Ability to multi-task and <Skill>prioritize</Skill> in a fast-paced <Value>work</Value> environment

Adept at working with large data sets with an ability to identify patterns

Strong web-research skills, including use of <Knowledge>web-based maps</Knowledge>

Demonstrated ability to work independently, as well as in small and large cross-functional teams

EDUCATION and/or EXPERIENCE:

University <Knowledge>Degree in Business Analysis</Knowledge> desired; equivalent combination of education and experience
may be substituted

Minimum 1-2 years of customer support experience in a <Knowledge>technical environment</Knowledge>

Minimum 1 year of experience with web-based map applications

Demonstrated experience in data analysis

Experience with <Knowledge>social media sites</Knowledge>
```

Fig. A.2 InlineXML format of named entities in web-based tagger

2. *Stanford format (tab-separated columns):* The original text is converted into a structured format established by the NLP Group, where the first column contains the tokens of the plain text file and the second column contains the belonging to each token. The first thing that the web-based tool makes is tokenizing the whole text into words and punctuation. After each token is stored in the first column of the output file, in this way if the token is part of a tagged concept then the second column will store the class

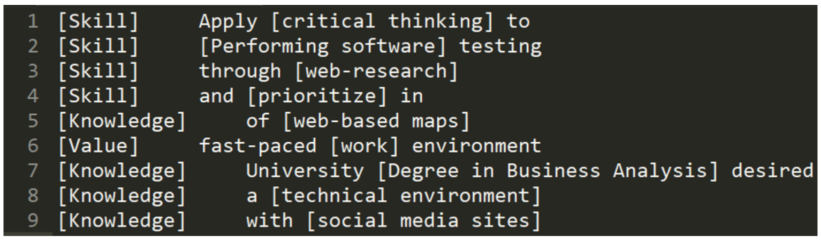
assigned to that concept, in another way the token will have the value of 0 as the class in the second column.



```
[Skill]     critical thinking
[Skill]     Performing software
[Skill]     web-research
[Skill]     prioritize
[Knowledge]     web-based maps
[Value]     work
[Knowledge]     Degree in Business Analysis
[Knowledge]     technical environment
[Knowledge]     social media sites
```

Fig. A.3 Stanford format

3. *The context of Keyword:* This format is similar to the tab-separated columns format, however, only the tagged concepts are exported. In this format, the first column contains the class and the second column contains the tagged concept as well as the words that proceed and precede to that concept. It is because the meaning of the concept can be different depending on the context in which it is identified.



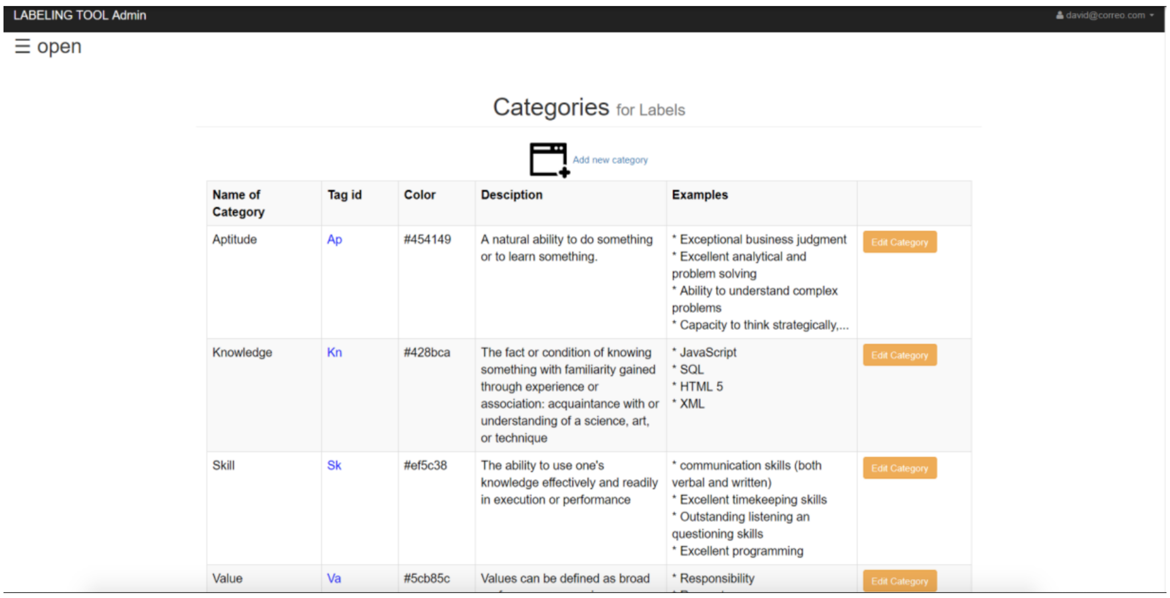
```
[Skill]     Apply [critical thinking] to
[Skill]     [Performing software] testing
[Skill]     through [web-research]
[Skill]     and [prioritize] in
[Knowledge]     of [web-based maps]
[Value]     fast-paced [work] environment
[Knowledge]     University [Degree in Business Analysis] desired
[Knowledge]     a [technical environment]
[Knowledge]     with [social media sites]
```

Fig. A.4 Special format to know the context

Administrator user

An administrator is allowed to create, modify and delete users, as well as administrate and assign classes (kind of labels) to each tagger. The administrator has an important role because he allows users to have access only to certain classes, and he can create as many as classes are required for users, allowing to get tagged texts with high quality in tags through human feedback. Additionally, the administrator is allowed to view the plain text files and is allowed to change the user tagger password if the tagger asks for that. In Figure A.5 the administration panel for classes is shown. When a new class is created it needs five features to be considered:

1. *Name:* It denotes the name of the class and will be used to format the output files in each one of the three available.



Categories for Labels

Name of Category	Tag id	Color	Desciption	Examples	
Aptitude	Ap	#454149	A natural ability to do something or to learn something.	* Exceptional business judgment * Excellent analytical and problem solving * Ability to understand complex problems * Capacity to think strategically,...	Edit Category
Knowledge	Kn	#428bca	The fact or condition of knowing something with familiarity gained through experience or association: acquaintance with or understanding of a science, art, or technique	* JavaScript * SQL * HTML 5 * XML	Edit Category
Skill	Sk	#ef5c38	The ability to use one's knowledge effectively and readily in execution or performance	* communication skills (both verbal and written) * Excellent timekeeping skills * Outstanding listening an questioning skills * Excellent programming	Edit Category
Value	Va	#5cb85c	Values can be defined as broad	* Responsibility	Edit Category

Fig. A.5 Administration panel for classes

2. *Short identifier:* It can be a little contraction to represent the class when the original name is too long.

3. *Color:* The color is used to highlight the tagged concepts by the tagger user in the text (displayed in section seven). It facilitates the reading to the users.

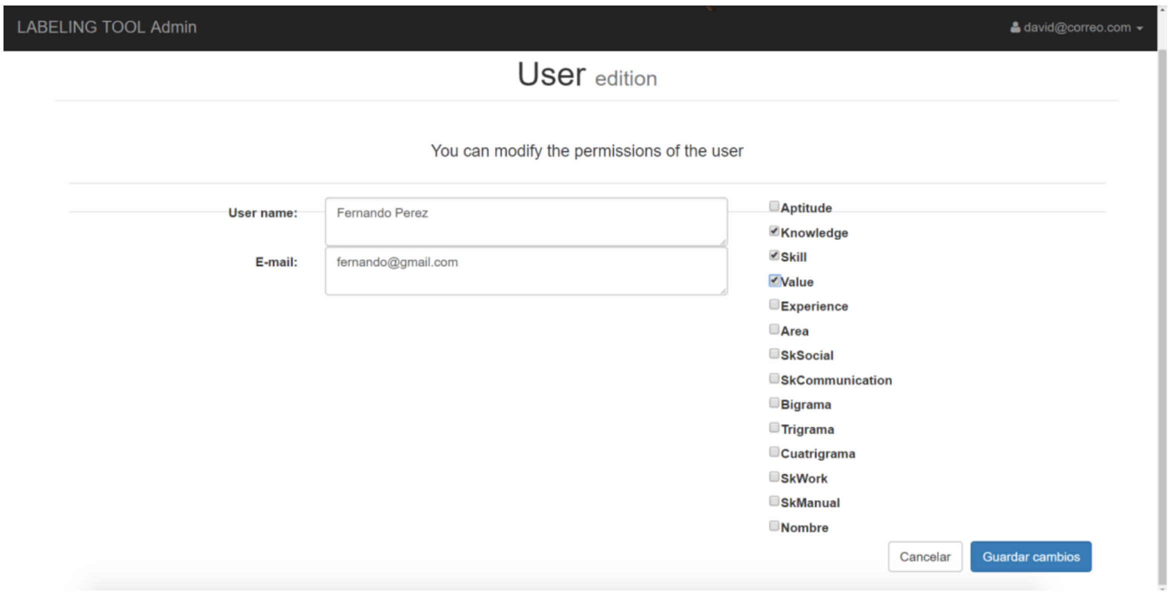
4. *Description:* A short description needs to be provided in order to guide users to understand the meaning of the class, since sometimes some classes can be similar, e.g. skill and aptitude.

5. *Examples:* In order to facilitate help for taggers, some examples should be included.

In Figure A.6 the interface corresponding to the administrator user is shown. In this screen, an administrator can assign permissions of classes to every tagger. This is because not all users will be working in the same domains, so they can request the creation and assignment of new classes to expand and improve the accuracy of the labeling for each text.

Web-based tagger interface

1. The first section (denoted by the circle with the number 1 inside) indicates if the selected plain text file is available to be tagged by other users since every owner decides if he allows each one of his files to be tagged collaboratively. If the owner does not allow his files to be tagged collaboratively, then each of his files will only be



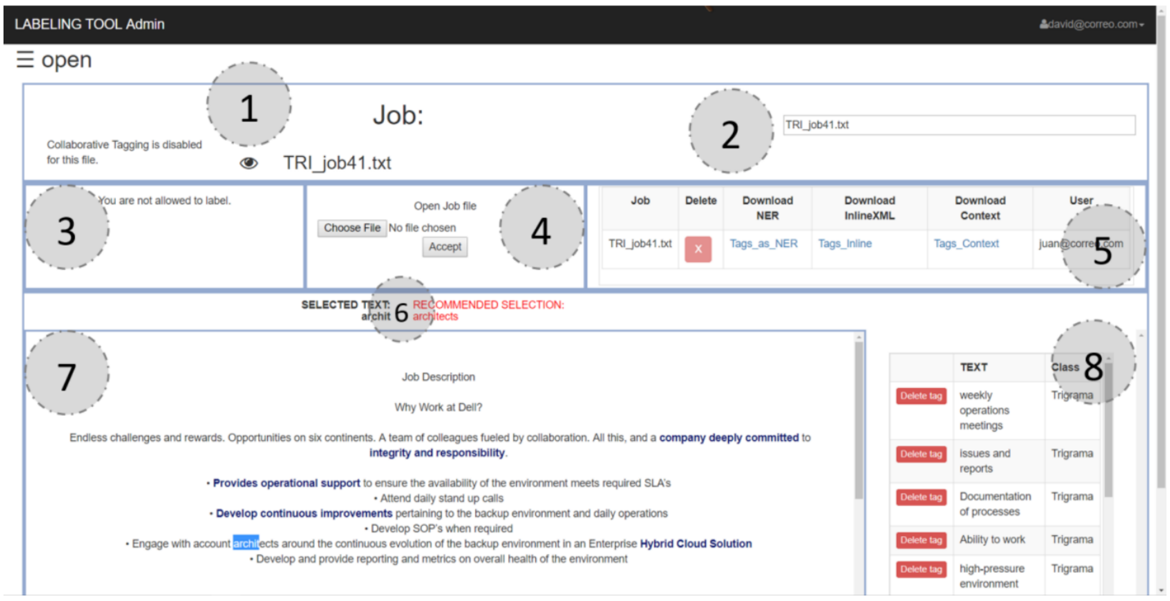
User edition

You can modify the permissions of the user

User name:	Fernando Perez
E-mail:	fernando@gmail.com

- ☐ Aptitude
- ☑ Knowledge
- ☑ Skill
- ☑ Value
- ☐ Experience
- ☐ Area
- ☐ SkSocial
- ☐ SkCommunication
- ☐ Bigrama
- ☐ Trigrama
- ☐ Cuatrigrama
- ☐ SkWork
- ☐ SkManual
- ☐ Nombre

Fig. A.6 Panel to assign tagging permissions for users



Job:

Collaborative Tagging is disabled for this file.

TRI_job41.txt

TRI_job41.txt

You are not allowed to label.

Open Job file

No file chosen

Job	Delete	Download NER	Download InlineXML	Download Context	User
TRI_job41.txt	X	Tags_as_NER	Tags_Inline	Tags_Context	juan@correo.com

SELECTED TEXT: archit RECOMMENDED SELECTION: architects

Job Description

Why Work at Dell?

Endless challenges and rewards. Opportunities on six continents. A team of colleagues fueled by collaboration. All this, and a **company deeply committed** to **integrity and responsibility**.

• **Provides operational support** to ensure the availability of the environment meets required SLA's
• Attend daily stand up calls
• **Develop continuous improvements** pertaining to the backup environment and daily operations
• Develop SOP's when required
• Engage with account architects around the continuous evolution of the backup environment in an Enterprise **Hybrid Cloud Solution**
• Develop and provide reporting and metrics on overall health of the environment

	TEXT	Class
Delete tag	weekly operations meetings	Trigrama
Delete tag	issues and reports	Trigrama
Delete tag	Documentation of processes	Trigrama
Delete tag	Ability to work	Trigrama
Delete tag	high-pressure environment	Trigrama

Fig. A.7 Web-based tagger interface

visible in reading mode to other users, however, any user can download the generated formats of any file, because it can be used as training data.

2. Section two contains a search field in order to help the users in finding a specific plain text file.

3. Section three contains the classes (as buttons) allowed to each user in the task of tagging plain text files, however, if the user selects a plain text file that belongs to other user and it is not enabled to be tagged collaboratively, then the user will not have active buttons to tag this plain text file.

4. Section four allows the users to upload plain text files in order to be tagged. Every tagger user can upload files as needed.

5. Section five is one of the most important sections since it offers the possibility to download the formatted plain text files in each one of the three available formats. This section also displays the information of the current plain text file selected, such as the name, the owner and the possibility to delete the file in case that the owner is the current user.

6. Section six have been integrated in order to provide help to each user at the time of highlighting the most important concepts in the text. When a user selects one keyword or concept in the text, he can omit part of the complete real concept so in that case, the web-tool offers an alternative to the highlighted text, such as in Figure A.7, where the user selected arch' and the web-tool offers the correct alternative for this case: architects.

7. Section seven is the most important of all since it displays the content of the selected plain text file to be tagged. Additionally, when the user has tagged a key concept then it is highlighted in the text with font bold and with the color assigned to the class with which the concept was tagged. This special feature helps to improve the reading of the text.

8. In section eight, a list with the tagged key concepts as well as the class assigned to each one are shown. Additionally, the user can delete one or more of the tagged concepts with the possibility of reassigning it to other class.

Appendix B

Detailed Results from Experiments

Due to the large number of results obtained from the experiments, we show tables with the results in detail in this section. These tables are the complete version of those displayed in the chapters, which are merely illustrative with graphics.

Table from chapter 4

Original dataset VS Expanded dataset

To know how accurately the original model versus the expanded model is, the V-fold cross-validation technique is be used. For this work, the 70descriptions were taken to train the first classifier and the remaining 30performance. These steps were executed 10 times by randomly taking 70using the remaining 30process was used but using the expanded set. The results for the experiments are measured in terms of Precision, Recall and F-1 score

Table from chapter 5

Graph Structure VS Stanford NER, CRF++, and MITIE

Performance comparison between our proposed Graph Structure against Stanford NER, CRF++, and MITIE classifiers.

Table B.1 Results obtained in 10 executions for each model

Execution	**Model**	**Precision**	**Recall**	**F-1**
1	Original dataset	0.4705	0.3264	0.3854
	Expanded dataset	0.4602	0.3314	0.3853
2	Original dataset	0.5069	0.3807	0.4348
	Expanded dataset	0.4883	0.3760	0.4248
3	Original dataset	0.5035	0.3227	0.3933
	Expanded dataset	0.4997	0.3210	0.3909
4	Original dataset	0.5026	0.3480	0.4113
	Expanded dataset	0.4907	0.3498	0.4084
5	Original dataset	0.4952	0.3409	0.4038
	Expanded dataset	0.4938	0.3596	0.4161
6	Original dataset	0.5349	0.3620	0.4318
	Expanded dataset	0.5217	0.3611	0.4268
7	Original dataset	0.4877	0.3278	0.3921
	Expanded dataset	0.4846	0.3420	0.4010
8	Original dataset	0.4249	0.3473	0.3822
	Expanded dataset	0.4149	0.3451	0.3768
9	Original dataset	0.5006	0.3611	0.4196
	Expanded dataset	0.4893	0.3684	0.4204
10	Original dataset	0.4835	0.3274	0.3904
	Expanded dataset	0.4688	0.3295	0.3870

Table B.2 Results of the different classifiers using the V-fold Cross Validation method to obtain the Precision, Recall and F1 measures.

Execution	Model	Class	Measures		
			Precision	Recall	F-1
1	Stanford	Character	0.8275	0.4137	0.5517
		Knowledge	0.5517	0.5708	0.6538
		Responsibility	0.4276	0.4276	0.3216
		Role	0.7836	0.7836	0.6767
		Skill	0.7835	0.3948	0.5250
		Talent	0.5250	0.2	0.3333
	CRF++	Character	0.9444	0.1428	0.2481
		Knowledge	0.7257	0.3167	0.4409
		Responsibility	0.5104	0.0963	0.1620
		Role	0.7105	0.3347	0.4550

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Skill	0.9130	0.2530	0.3962
		Talent	1	0.0740	0.1379
	MITIE	Character	0.6111	0.3793	0.4680
		Knowledge	0.6592	0.5006	0.5690
		Responsibility	0.4488	0.2517	0.3225
		Role	0.7325	0.5675	0.6395
		Skill	0.5309	0.3743	0.4390
		Talent	0.3684	0.0909	0.1458
	Graph Structure	Character	0.5769	0.375	0.4545
		Knowledge	0.7445	0.3886	0.5107
		Responsibility	0.3488	0.0413	0.0739
		Role	0.5508	0.3982	0.4623
		Skill	0.6666	0.2254	0.3369
		Talent	0.5	0.2098	0.2956
2	Stanford	Character	0.8988	0.5517	0.6837
		Knowledge	0.7447	0.6441	0.6908
		Responsibility	0.4357	0.3081	0.3609
		Role	0.7973	0.7573	0.7768
		Skill	0.7105	0.4106	0.5204
		Talent	0.7222	0.3145	0.4382
	CRF++	Character	0.9591	0.3154	0.4747
		Knowledge	0.7428	0.3704	0.4943
		Responsibility	0.6233	0.1668	0.2632
		Role	0.8333	0.5882	0.6896
		Skill	0.9005	0.2687	0.4139
		Talent	1	0.0781	0.1449
	MITIE	Character	0.7037	0.3931	0.5044
		Knowledge	0.6270	0.5270	0.5726
		Responsibility	0.4662	0.2430	0.3194
		Role	0.6991	0.7205	0.7096
		Skill	0.6727	0.3544	0.4642
		Talent	0.5405	0.1709	0.2597
	Graph Structure	Character	0.5772	0.4701	0.5182

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Knowledge	0.7479	0.3626	0.4884
		Responsibility	0.3106	0.0382	0.0681
		Role	0.5242	0.4556	0.4875
		Skill	0.6951	0.2321	0.3480
		Talent	0.5227	0.1782	0.2658
3	Stanford	Character	0.8712	0.4916	0.6285
		Knowledge	0.7336	0.5603	0.6354
		Responsibility	0.6049	0.3361	0.4321
		Role	0.8053	0.5555	0.6575
		Skill	0.7674	0.3908	0.5179
		Talent	0.7037	0.1743	0.2794
	CRF++	Character	0.9318	0.2192	0.3549
		Knowledge	0.7711	0.3137	0.4460
		Responsibility	0.5947	0.1396	0.2261
		Role	0.8421	0.2735	0.4129
		Skill	0.8395	0.2488	0.3838
		Talent	0.7857	0.0866	0.1560
	MITIE	Character	0.7473	0.3796	0.5035
		Knowledge	0.6746	0.5084	0.5798
		Responsibility	0.4587	0.2058	0.2842
		Role	0.7437	0.5509	0.6329
		Skill	0.6203	0.3075	0.4112
		Talent	0.4285	0.1111	0.1764
	Graph Structure	Character	0.6931	0.3210	0.4388
		Knowledge	0.7409	0.3405	0.4666
		Responsibility	0.5161	0.0513	0.0934
		Role	0.5957	0.3906	0.4719
		Skill	0.6926	0.2381	0.3544
		Talent	0.5128	0.1587	0.2424
4	Stanford	Character	0.8571	0.6131	0.7148
		Knowledge	0.7377	0.5250	0.6135
		Responsibility	0.5544	0.3030	0.3918
		Role	0.7894	0.6	0.6818

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Skill	0.7526	0.5207	0.6156
		Talent	0.7272	0.3243	0.4485
	CRF++	Character	0.86	0.3028	0.4479
		Knowledge	0.8099	0.2606	0.3943
		Responsibility	0.7891	0.1274	0.2194
		Role	0.9029	0.4407	0.5923
		Skill	0.8235	0.3218	0.4628
		Talent	0.9444	0.2151	0.3505
	MITIE	Character	0.6144	0.3642	0.4573
		Knowledge	0.6322	0.4447	0.5221
		Responsibility	0.4427	0.1666	0.2421
		Role	0.7861	0.6097	0.6868
		Skill	0.5612	0.3429	0.4257
		Talent	0.3225	0.1282	0.1834
	Graph Structure	Character	0.6708	0.3758	0.4818
		Knowledge	0.7188	0.3044	0.4277
		Responsibility	0.3888	0.0551	0.0966
		Role	0.5342	0.3714	0.4382
		Skill	0.6344	0.2776	0.3862
		Talent	0.4347	0.2597	0.3252
5	Stanford	Character	0.8901	0.5510	0.6806
		Knowledge	0.7213	0.5802	0.6431
		Responsibility	0.5873	0.3306	0.4230
		Role	0.8449	0.4866	0.6175
		Skill	0.7110	0.4338	0.5389
		Talent	0.7058	0.2424	0.3609
	CRF++	Character	0.9122	0.3513	0.5073
		Knowledge	0.7435	0.3459	0.4721
		Responsibility	0.6590	0.1161	0.1975
		Role	0.8356	0.2629	0.4
		Skill	0.8861	0.2416	0.3797
		Talent	0.9375	0.1401	0.2439
	MITIE	Character	0.6847	0.4468	0.5407

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Knowledge	0.6148	0.4984	0.5505
		Responsibility	0.4735	0.2390	0.3177
		Role	0.7070	0.5138	0.5951
		Skill	0.5664	0.3922	0.4635
		Talent	0.4857	0.1634	0.2446
	Graph Structure	Character	0.6111	0.4429	0.5136
		Knowledge	0.7327	0.3730	0.4944
		Responsibility	0.4328	0.04	0.0732
		Role	0.6142	0.3739	0.4648
		Skill	0.7671	0.2522	0.3796
		Talent	0.5769	0.1388	0.2238
6	Stanford	Character	0.9158	0.6901	0.7871
		Knowledge	0.7733	0.5764	0.6605
		Responsibility	0.5503	0.3549	0.4315
		Role	0.7760	0.6592	0.7129
		Skill	0.7436	0.3933	0.5145
		Talent	0.75	0.2105	0.3287
	CRF++	Character	0.9677	0.4195	0.5853
		Knowledge	0.7926	0.3637	0.4986
		Responsibility	0.6682	0.1805	0.2842
		Role	0.9058	0.3377	0.4920
		Skill	0.9245	0.2080	0.3396
		Talent	0.8181	0.0725	0.1333
	MITIE	Character	0.5929	0.4718	0.5254
		Knowledge	0.6727	0.4829	0.5622
		Responsibility	0.5205	0.2418	0.3302
		Role	0.7454	0.5347	0.6227
		Skill	0.6696	0.3340	0.4457
		Talent	0.6111	0.0973	0.1679
	Graph Structure	Character	0.6021	0.4028	0.4827
		Knowledge	0.7950	0.3355	0.4718
		Responsibility	0.4571	0.0633	0.1112
		Role	0.5365	0.3981	0.4571

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Skill	0.6690	0.1979	0.3054
		Talent	0.625	0.2	0.3030
7	Stanford	Character	0.9142	0.4238	0.5791
		Knowledge	0.7356	0.6113	0.6677
		Responsibility	0.4781	0.2909	0.3617
		Role	0.7754	0.5894	0.6697
		Skill	0.7610	0.3755	0.5029
		Talent	0.7692	0.1785	0.2898
	CRF++	Character	0.9302	0.2547	0.4
		Knowledge	0.7920	0.3571	0.4923
		Responsibility	0.5279	0.1131	0.1863
		Role	0.9487	0.2879	0.4417
		Skill	0.8275	0.2390	0.3709
		Talent	0.8888	0.0645	0.1203
	MITIE	Character	0.6666	0.3624	0.4695
		Knowledge	0.6190	0.4026	0.4879
		Responsibility	0.4898	0.1627	0.2443
		Role	0.7840	0.5542	0.6494
		Skill	0.6240	0.3340	0.4351
		Talent	0.3703	0.0925	0.1481
	Graph Structure	Character	0.6571	0.4480	0.5328
		Knowledge	0.7530	0.3621	0.4891
		Responsibility	0.3660	0.0460	0.0818
		Role	0.5827	0.3292	0.4207
		Skill	0.6474	0.2003	0.3060
		Talent	0.5862	0.1452	0.2328
8	Stanford	Character	0.8108	0.4838	0.6060
		Knowledge	0.6577	0.6344	0.6459
		Responsibility	0.3559	0.3492	0.3525
		Role	0.8546	0.5547	0.6727
		Skill	0.6575	0.4567	0.5390
		Talent	0.4516	0.1772	0.2545
	CRF++	Character	0.9487	0.2936	0.4484

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Knowledge	0.7159	0.3835	0.4994
		Responsibility	0.4475	0.0902	0.1502
		Role	0.9705	0.3535	0.5183
		Skill	0.7437	0.2952	0.4227
		Talent	1	0.0736	0.1372
	MITIE	Character	0.5454	0.3333	0.4137
		Knowledge	0.5662	0.4704	0.5139
		Responsibility	0.3970	0.2385	0.2980
		Role	0.8536	0.5	0.6306
		Skill	0.5034	0.3887	0.4387
		Talent	0.4827	0.1555	0.2352
	Graph Structure	Character	0.625	0.3906	0.4807
		Knowledge	0.6587	0.3603	0.4658
		Responsibility	0.2845	0.0510	0.0865
		Role	0.5445	0.3900	0.4545
		Skill	0.5873	0.2681	0.3681
		Talent	0.4166	0.1530	0.2238
9	Stanford	Character	0.8834	0.5449	0.6740
		Knowledge	0.7407	0.5912	0.6576
		Responsibility	0.5388	0.3349	0.4130
		Role	0.7976	0.6359	0.7076
		Skill	0.6959	0.4947	0.5783
		Talent	0.7454	0.3203	0.4480
	CRF++	Character	0.9454	0.3495	0.5104
		Knowledge	0.7619	0.3625	0.4913
		Responsibility	0.5579	0.1356	0.2182
		Role	0.9222	0.3610	0.5189
		Skill	0.8175	0.2679	0.4036
		Talent	0.8518	0.0974	0.1749
	MITIE	Character	0.7307	0.4634	0.5671
		Knowledge	0.6072	0.5452	0.5745
		Responsibility	0.3527	0.1639	0.2239
		Role	0.7061	0.6962	0.7011

Continued on next page

Table B.2 – *Continued from previous page*

Execution	Model	Class	Precision	Recall	F-1
		Skill	0.4970	0.4166	0.4533
		Talent	0.7241	0.1826	0.2916
	Graph Structure	Character	0.5384	0.3414	0.4179
		Knowledge	0.7060	0.3453	0.4637
		Responsibility	0.3921	0.0480	0.0856
		Role	0.5521	0.4225	0.4787
		Skill	0.6453	0.2450	0.3552
		Talent	0.6382	0.2272	0.3351
10	Stanford	Character	0.9024	0.4567	0.6065
		Knowledge	0.7350	0.5381	0.6214
		Responsibility	0.5042	0.3228	0.3936
		Role	0.8472	0.6099	0.7092
		Skill	0.6700	0.4627	0.5474
		Talent	0.5714	0.2150	0.3125
	CRF++	Character	0.9433	0.3125	0.4694
		Knowledge	0.7476	0.2743	0.4013
		Responsibility	0.6762	0.1742	0.2770
		Role	0.8446	0.29	0.4317
		Skill	0.7613	0.2881	0.4180
		Talent	0.9	0.09	0.1636
	MITIE	Character	0.6881	0.4076	0.512
		Knowledge	0.6252	0.4690	0.5360
		Responsibility	0.4554	0.2571	0.3286
		Role	0.7256	0.5774	0.6431
		Skill	0.5382	0.3912	0.4530
		Talent	0.4666	0.1505	0.2276
	Graph Structure	Character	0.5416	0.3209	0.4031
		Knowledge	0.7130	0.2852	0.4074
		Responsibility	0.3839	0.0470	0.0839
		Role	0.6594	0.3053	0.4174
		Skill	0.6145	0.2510	0.3564
		Talent	0.55	0.2291	0.3235