Institute of Mathematics and Statistics
University of São Paulo

# Comparing vector document representation methods for authorship identification

Pamela Revuelta Quintanilla

Dissertation presented for the degree
of Master in Sciences

**Program:** Computer Science

**Advisor:** Prof. Dr. Flavio Soares Correa da Silva

During the development of this work, the author received CAPES financial assistance

São Paulo, December 2020

# Comparing vector document representation methods for authorship identification

This version of the master thesis by Pamela Revuelta Quintanilla includes the modifications suggested by the Judging Committee members during the defense of the original document. A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

# Comparing vector document representation methods for authorship identification

Judging Committee:

- Prof. Flavio Soares Corrêa da Silva - IME-USP
- Prof. Ricardo Luís de Azevedo da Rocha - POLI-USP
- Prof. Roberto Cássio de Araújo - Mackenzie

# Acknowledgements

Primero quiero agradecer a la "Universidad de São Paulo" y en especial al "Instituto de Matemática e Estatística" por acogerme.

Es protocolar agradecer al orientador, en una tesis de pos-graduación, pero en esta ocasion no solo quiero agradecer a mi orientador, el Prof. Flavio Soares Corrêa da Silva, sino tambien a Flavio como persona; por su paciencia, su comprension y por su orientacion. Tambien quiero agradecer a todas las personas que me acompañaron a lo largo de este camino, a lo largo de la realizacion de esta tesis de maestria. Partiendo desde el inicio de este sueño, a mis padres, porque a pesar del miedo y las circunstancias, siempre me apoyaron. Agradecer a mi hermanita, por ser mi hermana y amiga. Junto con ello agradecer a toda mi familia por las risas, los buenos deseos y su gran comprensión

Y como no agradecer a todos los amigos que conoci en este camino recorrido, amigos de diferentes culturas y formas de pensar. Amigos que me brindaron su apoyo y espero conservarlos siempre en mi vida. Pero como olvidar a mis amigos de toda la vida, que estuvieron desde antes de esta aventura y estaran hasta que Dios lo disponga. Gracias amigos por sus palabras de aliento, por su tiempo, por su paciencia, por darme una mano y mas que eso. Estar en los momentos en los que los necesite.

Me pregunto, como no agracerte a ti que me has acompañado hasta el final. Hasta la ultima linea, hasta el ultimo minuto, hasta el ultimo aliento. Y no nesecito decir tu nombre, porque tu sabes quien eres.

Finalmente agradecer a Dios, y si has llegado hasta aqui te preguntaras porque al final a Dios. Porque aunque no quede nada, ni nadie, el siempre estara para ti. Dios estara contigo desde el primer momento hasta el fin, si tu lo quieres asi. Se que Dios no leera estas lineas pero yo siempre le estare agradecida. Y aprobecho este ultimo parrafo para transmitir un mensaje de fé. Fé que me enseño mi mandre. Gracias Mamá.

# Abstract

Quintanilla, P.R. **Comparing vector document representation methods for authorship identification**. 2020. Thesis (Master) - Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2020.

Over the years the information available in online media has had a great increase. In this sense, the automation of processing languages natural for large amounts of information gained importance, for example, text classification task. It can be used to identify the author (Authorship Identification); however, it requires Machine Learning techniques to identify the author, these techniques have given good results in NLP. In addition, Machine Learning receives the feature vector of the texts, which is extracted using vector document representation methods. The methods proposed for this research are grouped into three different approaches: i) methods based on vector space models, ii) methods based on word embeddings, and iii) methods based on graph embeddings, for this approach, we first model the texts as graphs. On the other hand, not all the methods are used for different languages because they can have different efficiency depending on the language of the analyzed texts. Therefore, the objective of this research is to compare several of these methods using literary texts in English and Spanish. In this way, we analyze whether the methods are efficient to represent several languages or its performance depends on the characteristic of every language. The results showed that the methods of Graph embeddings achieved the best performance for both languages, being that English reached a fairly high success rate. On the other hand, the other methods achieved good performance for English, however, the results for Spanish were not optimal. We believe that the results in Spanish were worse due to the morphological, lexical, and syntactic complexity that this language presents in comparison to English. For this reason, different approaches were compared for the mathematical representation of texts that try to cover the different aspects of a language.

**Keywords:** Authorship attribution, Feature extraction, Text classification, Machine Learning, Complex networks, Word embedding and Graph embedding.

# Summary

Quintanilla, P.R. **Comparando métodos de representação vectorial de documentos para identificação de autoria**. 2020. Tese (Mestrado) – Instituto de Matemática e Estatísica, Universidade de São Paulo, São Paulo, 2020.

Com o passar dos anos, as informações disponíveis na mídia online tiveram um grande aumento. Nesse sentido, ganhou importância a automatização de processamento de linguagens natural para grandes quantidades de informação, por exemplo, a tarefa de classificação de textos. Esta tarefa pode ser usada para identificar o autor, atribução de autoria, mas precisa de técnicas de Aprendizado Máquina para identificá-lo, o que têm dado bons resultados no PLN. Além disso, Aprendizado Máquina recebe o vetor característico dos textos os quais são extraídos utilizando métodos de representação vetorial de documentos. Os métodos propostos para esta investigação estão agrupados em três abordagens: i) métodos baseados em modelos de espaço vetorial, ii) métodos baseados em *Word embeddings*, e iii) métodos baseados em *Graph embeddings*, para esta abordagem, primeiro modelamos os textos como grafos. Por outro lado, nem todos os métodos são usados para diferentes idiomas, porque pode ter diferentes eficiências, dependendo do idioma dos textos analisados. Então, o objetivo desta pesquisa é comparar vários desses métodos utilizando textos literários em inglês e espanhol. Desta forma, nós analisamos se os métodos são eficientes para representar várias linguagens ou seu desempenho depende das características de cada linguagem. Os resultados mostraram que os métodos de *Graph embeddings* obtiveram bom desempenho para as duas linguagens, sendo que para o inglês alcançaram uma taxa de sucesso bastante elevada. Por outro lado, os demais métodos obtiveram bom desempenho para o inglês, porém os resultados para o espanhol não foram os ideais. Acreditamos que os resultados em espanhol foram piores devido à complexidade morfológica, lexical e sintática que este idioma apresenta em comparação ao inglês. Por esse motivo, foram comparadas diferentes abordagens para a representação matemática de textos que procuram abranger os diferentes aspectos de uma língua.

**Palavras-chave:** Atribuição de autoria, Extração de características, Classificação de texto, Aprendizado màquina, Redes complexas, Word embedding e Graph embedding.

# Contents

# List of Abbreviations

| | |
|---|---|
| AI | Artificial Intelligence |
| BERT | Bidirectional Encoder Representations from Transformers |
| CBOW | Continuous(common) bag of words algorithm |
| FM | Frequency Model |
| G2V | Graph2Vec |
| IDF | Inverse document frequency |
| IR | Information retrieval |
| KNN | K-nearest neighbors algorithm |
| NB | Naive Bayes classifiers |
| NLP | Natural Language Processing |
| N2V | Node2Vec |
| POS | Part of speech |
| SVM | Support-vector machines |
| TF | Term frequency |
| TF-IDF | Term frequency Inverse document frequency |
| VSM | Vector Space Model (Term vector model) |
| W2V | Word2Vec |
| WWW | World Wide Web |

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Context and motivation

The amount of information available on the Internet has grown enormously over the past decades. An important part of this information is available in textual documents. These documents include online news websites, user comments on various social networks such as Facebook, Twitter, Instagram or YouTube, blog posts, literature books with various topics in online libraries, scientific articles available in research journals, among many other examples. For a user, the manipulation of large amounts of information becomes a difficult task because it requires a lot of time and resources, for example, a first case could be of the organization of several related texts, in which the person would have to read each one to know the correct way to order them in a database. The accomplishment of this task becomes a tedious activity for the user because of the length and quantity of texts. The second case might consist on analyzing the evolution of literary texts over time, in this sense, it would be necessary to consider the different literary movements that have existed over the years. Analyzing a large number of books throughout history would be challenging for one or more people because it would require reading and analyzing them. These studies also include identifying the author through the writing style, of one or more literary books. Both organizing several related texts, as well as analyzing the evolution of literary movements, involves the joint work of several experts who know literary movements, styles and writing patterns.

Due to this great increase of textual information, the need to use automatic methods to understand, sort, and classify such information arises [RARP12]. For the reasons shown above, the NLP (Natural Language Processing) [NOMC11] becomes a widely used area for information processing. For instance, for the two cases shown above (organization of several related texts and literature analysis over time), we can apply different techniques of NLP. For the first case (organization of several related texts), tasks such as automatic document summarization [CJ15] and text classification [Kor12] are the most appropriate methodologies for the problem of text organization. While for the second case (literature analysis over time), the authorship identification task [Juo08] looks for document writing patterns to get special characteristics to associate an author with his literary production.

In addition to the above-mentioned NLP applications, there are also sentiment analysis [KZM14], topic labeling [HLZB18], authorship attribution [Sta09a], spam detection [CKP+15], text classification [Kor12], among others. Referring specifically to text classification, offers a good framework to become familiar with textual data processing, for this reason, in this work, we will tackle this application of NLP. The text classification [Kor12], which consists of assigning a label or category to a text considering its content. Several tasks are related to text classification, which varies according to the classification criteria (author, genre, epic, literary movement, study area, among others). In this master's research, it has been considered the author of the text as a classification criteria (authorship attribution), which associates each document with its author.

The best methods for text classification are the systems based on machine learning, that is why these have giving achieved significant results. For the training of machine learning algorithms, there should be two input parameters, which are the category or class of the document and the feature vector representing the text. That is to say, the reason why these techniques receive the feature vector and not the document itself is that a computer is capable of understanding binary language, but is impossible for it to understand each word of a text naturally (morphological, lexical, syntactic and semantic aspects). For this sense, the encoding of texts into numerical vectors has been one of the main research focuses related to NLP and text analysis; furthermore, the vector representation gives us the ability to perform meaningful analytics. Moreover, to classify texts it is required to have a vector representation of each document, which extracts the most representative characteristics of the text. Each property of the vector representation is a characteristic, and each characteristic represents attributes and properties of documents including their content and meta-attributes, such as document author, length, source, and publication date, among others. Last, The process of converting texts into numbers is called Vectorization or Feature Extraction [Sta09b].

Then, to capture an optimal representation of documents, we need to analyze and compare different methods for the representation of texts. The objective of these methods is to obtain a vector representation, which is capable of maintaining the text structure and the existing semantics between each word of the text. The vector representation methods could be divided into two methodologies, such as sparse vector representation and dense vector representation.

The first methodology for document vectorization called "Sparse vector representation" expresses that the vector's dimension is equal to the size of the text vocabulary, that is, if the word appears in the dictionary, it will be counted. The Sparse vector representation is based on the classic vector space model (VSM) [Com18]. VSM was successfully used for information retrieval and document-indexing tasks. This model has different variations, such as frequency model or bag-of-words (BOW) and TF-IDF model [SWY75a]. BOW counts how many times a word appears in a document, while TF-IDF is a statistical measure that evaluates how important is a word in a collection of document. Both variations are simple to understand, however, they have several weaknesses, such as vectors with high dimensionality (for large corpus), as well as they ignore the order and semantic of words.

Unlike the first methodology that does not has fixed size vectors, then, the second methodology (Dense vector representation) emerged to overcome the high dimensionality problems of sparse vector representation. The Dense vector representation looks to capture the meaning, semantic relationships and different contexts in which a word can be used. The Dense vector representation mainly includes Word and Sentence embeddings models, whose objective is to find the mathematical representation of each word or sentence that comprises a text. For this master's work, we experimented with Word embeddings models (Word2vec) [MCCD13], which argues that the vectors of similar words are close to each other, considering similar characteristics for their grouping, such as the context.

Another approach to extract characteristics from a document is to use concepts related to graphs and complex networks [COT+11]. Complex networks are graphs with special properties. Recently, complex networks have been used with success in different areas, including tasks related to NLP. Anything in the real world could be modeled as a graph, then, the documents can also be modeled as graphs. Furthermore, as described in the previous paragraph, there are methods to represent words as numerical vectors, so there are also methods that extract a representative N-dimensional vector of a graph, called Graph embeddings. Therefore, we proposed to model texts as graphs (where nodes and edges could be generated in various ways) and then, to extract the representative vectors of such graphs using graph embedding and node embedding techniques. In other words, we applied the most recent concepts of complex networks and Graph embeddings techniques (Graph2vec and Node2vec) for the classification task (authorship attribution).

The dataset is composed of documents, which will be used to evaluate the vector document representation methods using the authorship attribution task. To choose which documents be part of our dataset, two characteristics were considered. First, we selected a dataset composed of literary texts. This type of text represents a challenge due to the different characteristics, such as a period of time, literary movement and writing style, among others. Second, we choose texts written in two languages in order to analyze whether the results of the vector representation methods of texts maintain their performance in texts written in both languages. For this analysis, we considered two language families: Romance languages (Spanish) and Germanic languages (English). These languages have different linguistic structures, some are richer morphologically, others lexically, and others syntactically. To preserve the different linguistic aspects of both languages, it was proposed to use complex graphs and networks. In contrast, VSM and word embedding models do not preserve some of these linguistic characteristics.

For this Master's work, we compared the performance of various vector document representation methods (classic vector space models, word embeddings, and graph embeddings), for the document classification task by the author using texts in English and Spanish. On the other hand, in order not to cease our results of the performance of the vector document representation methods, it was decided to experiment with different classification algorithms: Decision Tree, K-Nearest Neighboors (KNN), Gaussian Naive Bayes (Gaussian NB) and Support Vector Machine (SVM).

## 1.2 Hypothesis

It is possible to use a vector document representation method to adequately represent literary texts written in English or Spanish for the task of authorship identification. Regardless of each language, such methods can be effective in capturing the main characteristics and peculiarities of each language. That is, could it be that a vector document representation method is efficient to represent several languages or its performance depends on the characteristic of every language.

## 1.3 Objectives

The main objective of this master's Work is to compare various methods to represent texts for authorship recognition and in this way find which methods give us good results independent of the language. We aim to evaluate the performance of these methods for both Spanish and English literary texts.

The specific objectives are:

- Analyze the difference in the performance of the methods of representation of texts when experimenting with text in different languages.
- Create a database for Spanish classified by the author with literary texts.
- Analyze the performance, strengths and weaknesses of traditional vector space models.
- Experiment and discuss which parameters are the most suitable to generate representative vectors when using word embeddings techniques
- When using dense vector representation techniques, experiment with different sizes of the feature vector.
- Compare the performance of techniques of vector representation of graphs with vectors extracted by traditional techniques of representation of texts.
- Analyze whether the performance of the proposed methods of vector representation of documents is independent or show many variations in the results.
- Experiment with different classifiers, so as not to divert the results of the vector document representation method to the performance of the classifier.

## 1.4 Organization of this Dissertation

This document is organized as follows:

- Chapter 2: We described the linguistic concepts.

- Chapter 3: We described the most important concepts related to NLP, authorship attribution and the methods for vector representation of documents. Likewise, we explained the concepts of graphs and complex networks and their applications for the representation of texts.
- Chapter 4 : We explained the most important works related to authorship attribution and the methods for the vector representation of documents.
- Chapter 5: We detailed the methodology developed for this Master's work.
- Chapter 6: We discussed and analyzed the results we obtained.
- Chapter 7: We described the conclusions and future work options for this Master's work.
- Appendix A: We describe the list of authors with the list of books selected for English and Spanish.
- Appendix B: We list the stopwords used for pre-processing in English and Spanish.

# Chapter 2

# Linguistic Concepts

As we described in the previous chapter (Chapter 1), the objective of this Master's work is to analyze the difference in the performance of the vector document representation method using text in different languages (Spanish and English). For this reason, not only a computational approach but also a linguistic approach should be involved on the vector representation methods of documents. Next, a brief contrastive analysis (contrastive linguistics) of both languages will be conducted and its contrasts in each level of language.

## 2.1 Contrastive linguistics

Contrastive linguistics is a branch of general linguistics, it is also called differential linguistics. This concept is a linguistic approach that compares and describes in detail the structural similarities and differences between two or more languages (in this case English and Spanish). This branch of linguistics considers the phonological, morphological, lexical-semantic, syntactic and pragmatic levels. We consider contrastive descriptions at the morphological, lexical-semantic and syntactic levels. We do not consider the phonological level because we do not study the sounds. We only study both languages in their written representation.

Before analyzing each level of the linguistic structure of both languages. We must understand the origin of these languages. These two come from Proto-Indo-European languages, which means that we will recognize some aspects shared by both languages, especially at the morphological and syntactic level (grammatical level) [Man02]. For example, 40% of the vocabulary (lexical level) of English is made up of words of French origin. Both Spanish and French are Latin languages. Then, we will find lexical coincidences between both languages. On the other hand, English is a Germanic language, related to German, Swedish or Danish. Spanish is a Romance language, like French, Portuguese, Italian or Catalan.

We will describe the existing contrasts in each level of language [Man02] , chosen for this thesis: morphological (Section 2.1.1), syntactic (Section 2.1.2), lexical-semantic (Section 2.1.3) and pragmatic (Section 2.1.4).

### 2.1.1 Morphological level

Morphology is a part of grammar, which studies words and their modifications in isolation. There is the inflectional and derivational morphology:

- **Inflectional Morphology:** It studies the changes in words that modify them to fit a syntactic context. These morphemes are added to a lexeme (root) and do not change the syntactic category of the root. Lexemes and morphemes are the least significant units called monemes. In Figure 2.1, a lexeme or root and its different morphemes can be seen.

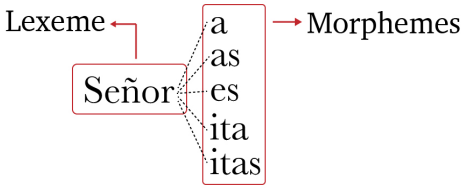



**Figure 2.1:** *Lexeme and Morphemes of the word Señor in Spanish.*

Although both languages have inflectional morphemes to indicate plural, Spanish applies the singular-plural distinction of the following syntactic categories: nouns (*perro-perros*), adjectives (*marron-marrones*), articles (*el-los*) and all the different forms of the verbal derivation (*como-comamos, comes-coméis, come-comen*). English only flexes nouns (dog-dogs). This explanation is reflected in Table 2.1

| **Types of words** | | **Spanish** | **English** |
|---|---|---|---|
| **Noun** | Number | perro-perros | dog-dogs |
| | Gender | perro-perra | dog |
| **Adjective** | Number | rojo-rojos | red |
| | Gender | rojo-roja | red |
| **Article** | Number | el-los | the |
| | Gender | el-la | the |

**Table 2.1:** *Inflectional Morphology for nouns, adjectives and articles.*

The greatest difference in inflectional morphology is observed in the verbal paradigm. In English, regular verbs usually have between four to five forms. For example: see, sees, seeing, saw, seen. However, other verbs have between one to eight forms, which are be, am, is, are, was, were, been and being. In contrast, Spanish verbs have a singular-plural version of the first person (como-comemos), the second person (comes-coméis) and third person (come-comen). English only modifies the third person singular (she eats). All this information is contained in Table 2.2.

On the one hand, English only distinguishes two tenses, present and past; because future tense uses the auxiliary "will". On the other hand, Spanish has four verb tenses such as present, future and two types of past tense, past imperfect (*comía*) and indefinite (*comí*). Furthermore, Spanish has two verb modes: indicative and subjunctive, having 47 forms of the Spanish verbs.

| Person | Number | Spanish | English |
|---|---|---|---|
| **1º Person** | Singular | como | eat |
| | Plural | comemos | eat |
| **2º Person** | Singular | comes | eat |
| | Plural | comen | eat |
| **3º Person** | Singular | come | eats |
| | Plural | comen | eat |

**Table 2.2:** *Inflectional morphology for verbs (person and number).*

Finally, we can see them in inflectional morphology, which is where we find more differences. Since Spanish has more morphological variants than English, then the morphological information of Spanish is richer than English.

- **Derivative Morphology**: : These morphemes change the syntactic category from the root to join it or alter its insignificant meaning. With the derivation, we can create words from others. For example, for verbs: ($cortar > cortadura$), adjectives ($alto > altura$), among others.

  Although both languages use this mechanism, Spanish: (i) Diminutive: ($silla > sillita, chico > chicuelo, perro > perrazo$); (ii) Augmentative ($silla > sillon$); (iii) Pejorative ($periodico > periodicucho$).

  The only diminutive exists in English as derivative morphemes but in a restricted way. Again, Spanish has an inventory of derivative morphemes superior to English.

- **The Composition**: Another important process is the composition, which is on the border between morphology and syntax (morphosyntactic). For example: $boygirl = boy + girl$, $paraguas = para + aguas$, among others.

  In Spanish, there are four categories: $verb+noun$, $noun+noun$, $noun-i+adjective$ and $noun + adverb$. However, English has more than eleven categories, some of them are $noun+noun$, $adjective+noun$, $adjective+noun-ed$, $noun+verb-ing$, $particle + verb$, $noun + verb - er$, $noun + adjective$, $adverb + noun$, $verb + noun$, $verb + particle$, among others.

  In English, this resource is more exploited than in Spanish. Therefore, it is observed that more categories to combine words, due to sequences of names that are successively modified can be accumulated. The largest and most productive category is $noun + noun$. In Spanish, we would have to resort to the preposition *"de"*, which can indicate the same type of little-specified relationship between *nouns*. In Spanish, the most frequent form of compound is $verb + noun$.

### 2.1.2 Syntactic level

This level is responsible for the study of the rules that a language handles. The syntax, not only studies the ways in which words are combined (sentences and syntagmas), but also the relationships between them. Due to the complexity of this level, we will describe the syntactic differences in two sections:

- **By grammatical category:** Both languages have similar grammatical categories (noun, adjective, verb, adverb, preposition and conjunction). But despite this, some divergences can be found: (i) **Difference in Noun**, is related to the division between countable and uncountable nouns, besides English has a higher number of uncountable nouns unlike Spanish, which has more countable nouns than uncountable; (ii) **Difference in Adjective**, the adjectives in English come first then the noun (*blue house*), but in Spanish, they usually come after the noun (*casa azul*), although there are exceptions where the adjective can be before or after the noun, but modify the meaning, for example *clase media*, *media clase*, *el mismo presidente*, *el presidente mismo*; (iii) **Difference in Article**, although determined and indeterminate articles fulfill the same functions, at the time of making translations it may or may not keep the same type of article, for example: *"He has a long nose"* to *"Tiene la nariz larga"*.

- **By the Order of Words and Constituents:** Although in general lines, in this aspect both languages are very similar. Considering the idea that when a language has more morphological information, it needs less syntactic information; and vice versa. So, as Spanish has more morphological information, it uses fewer syntactic rules, that is, it has a freer word order than English. Some consequences of this are: (i) **The Presence/Absence of the Subject**, the verb in English as it does not have detailed morphological information, it must include the pronominal form of the subject (*We sing today*). In Spanish the details of the subject are implicit in the verb (*Cantamos hoy*); also, (ii) **Order subject and verb**, English not allows the placement of the subject in other places in the sentence unlike Spanish (*Ya vienen las vacaciones*, versus *Las vacaciones vienen ya*). Therefore, we should talk about the equivalence between word senses.

### 2.1.3 Lexical-Semantic Level

When we speak of lexicon we refer to the vocabulary of the language, when we speak of semantic it refers to the meaning of the words. The task of finding lexical equivalents between any two words in different languages is difficult, due to polysemy. Polysemy is the ability for a word or phrase to have multiple meanings. For example, say that the word "gato" in Spanish translates to English is "cat" is false. In Table 2.3, we see that the different meanings of the word "gato" and the word "cat".

| Meaning of Cat | Meaning of Gato |
|---|---|
| 1. Domestic feline | 1. Domestic feline |
| 2. Feline mammal (lion) | 2. Crane to lift cars |
| 3. Harpy | 3. Tic-Tac-Toe (Mexico or Chile) |

**Table 2.3:** *Polysemic word comparison (cat-gato).*

So, we should talk about the equivalence between word meanings. Even more, that a word can mean something different depending on cultural and socio-cultural models, subjective connotations, among others. That is why the semantic comparison between languages offers a high level of complexity.

### 2.1.4 Pragmatic Level

This level of language is concerned with how context influences the interpretation of meaning. Because to understand a sentence or phrase, pragmatics takes into consideration the extralinguistic factors that condition the use of the language. That is, all those factors that are not referred to in a formal study, such as the knowledge shared by the speakers, communicative situation, intentionality of the author, interpersonal relationships, among others.

Furthermore, for interpretation of the meaning of a sentence, an understanding of the social uses of words is necessary because when contrasting two languages on the pragmatic level, we inevitably go beyond the purely linguistic realm to include sociocultural aspects of the broader and more inclusive nature. On the other hand, a sentence is the syntactic unit of complete sense and shares a semantic content, but its meaning and proper interpretation not only depends on its content but also requires a defined linguistic context to be interpreted. That is, the same sentence can have different intentions or interpretations in different contexts (it can be literal, ironic or metaphorical). It will not be semantic information, but inferred and contextual information, deduced jointly from the context and words.

In conclusion, to properly analyze, interpret and understand a sentence or document at a pragmatic level, the grammatical decoding that gives it the semantic content is necessary and the context is also necessary for correct inference. We will describe the context in more detail below.

#### Context

In real life, context is a set of circumstances that surround a situation and without which it cannot be properly understood. In the framework of a document, the context is words that occur in the proximity of other words. From a linguistic approach, the context is the set of linguistic elements that include, precede or follow a word or sentence and that can determine its meaning or its correct interpretation. But, for your better understanding, the context should be understood as a situation since it can include any extralinguistic aspect.

At time of communication, people make deductions and inferences from what is being said in a conversation or linguistic interaction to create a linguistic context in which the following statements are properly interpreted. The implicatures or implicit information are the meanings in addition to the literal or explicit meaning that the receiver of a message infers. In addition, they are also obtained from the recognition of the intention of the speaker, considering: the literal meaning of the statement, the knowledge shared by the speaker and listener, the situational context and the intention of the speaker.

Finally, the pragmatic level presents special characteristics, which are complex and difficult to interpret, even for the human being, due to this particularity, we have not reviewed the pragmatic level because it would exceed the scope of this Master's work.

# Chapter 3

# Computational Concepts

In this chapter, we will present the fundamental concepts from the main areas related to this work; authorship attribution, document classification, and methods for mathematical document representation. We present definitions, such as applications of authorship attribution in Section 3.1. Section 3.2 comprises the main concepts of document classification, methods, algorithms and applications. Before analyzing the methods for obtaining the feature vector of the documents, it is important to understand what the vector representation of documents means (Section 3.3). Also, this chapter describes several methods in order to obtain the feature vector that best represents a text. First, Section 3.4 presents some of the traditional methods (Vector space model). Section 3.5 describes word embeddings techniques, that can capture the meaning, semantic relationships and different contexts in which a word is used. Finally, considering the last vector representation approach, where the texts are first modeled as graphs, and then, the representative vectors of such graphs are extracted. To model texts as graphs, we previously studied graphs and complex networks in Section 3.6 and Section 3.7 explains the methods to extract the feature vector of such graphs.

## 3.1 Authorship attribution

Natural language processing (NLP) is a multidisciplinary field that involves Artificial Intelligence (AI) and Linguistics. NLP studies the interactions between computers and linguistic signs (sound, writing, gestures and signs). Likewise, NLP focuses on the analysis of human communications and, specifically, their language. This automatic analysis on any type of text allows to classify, organize, search or discover non-explicit information. The most important NLP applications are mentioned as follows: automatic translation of documents [SPdVLO19]; conversational systems that allow interaction between humans and machines [ML18]; morphological, syntactic and semantic labeling [HLZB18]; sentiment analysis of texts [KZM14]; automatic summarization of texts [CJ15]; information retrieval [UF16]; document classification [Kor12]; authorship attribution [Juo08]; among others. The main objective of these applications is to analyze large amounts of information, and in this way, obtain relevant information that clearly and concisely represents the content of all these textual documents.

For this master's research, we focused on authorship attribution and document classification. First, authorship attribution, which can be defined as a multi-class text classification problem. Second, document classification is required to obtain the vector representations of source documents to be used as feature vectors for several machine learning algorithms. In this sense, we proposed to evaluate different methods of mathematical representation of documents for authorship recognition tasks using document classification algorithms.

Authorship attribution (or authorship recognition) consists in assign to a text the most probable author of a set of candidate authors [Sta09a]. These methods have been of considerable interest for various applications, such as the classification of literature books, identification of patterns, detection of plagiarism in text segments or full texts, problems related to copyright, among other applications [Mar].

Several works have focused on probabilistic and statistical models for this task. Probabilistic models maximize the probability of identifying the real author of a text, while statistical models focus on finding attributes and features to characterize different writing patterns and styles. Due to the large increase in textual information on the Internet (books, emails, blog posts, among others), the need to find methods to analyze this large number of documents becomes important. For this reason, the stylometry research area emerged, in charge of finding writing patterns in texts. These patterns are grouped into the following categories: lexical, character, syntactic, and semantic features. We will briefly explain these stylometric attributes below [Sta09a, Mar]:

- Lexical features: These features are related to word, sentence lengths, word frequencies, vocabulary size and variations, among other features.
- Character features: These features consider the text as a sequence of characters. Common features are the following: alphabetic counts, the frequency of punctuation marks, frequency of letters, among other features.
- Syntactic features: These features are related to the structure of texts. These approaches include the analysis of sentences, part-of-speech (POS) tags of each word, and how the sentences are decomposed into its constituent parts. POS tags indicate the grammatical class of each word from texts. Words could be classified as nouns, adjectives, verbs, adverbs, etc.
- Semantic features: These features consider the semantic relations between the words composing a text. Synonymy or antonym relationships are commonly used for this approach.

Finally, when we talk about authorship attribution, we must not only consider the linguistic characteristics of documents (we had detailed in Section 2.1) but also consider algorithms for text classification (Section 3.2). They allow us to classify the documents by the author and will receive as input parameters the feature vector of the documents.

## 3.2 Document Classification

The classification of documents is the process that consists of assigning labels or categories to the documents hierarchically, according to the principles of origin, order or other factors. We can classify any type of document: academic, literary, among others. This task has several applications, such as sentiment analysis, topic tagging, spam detection, among others [KTA+16].

Some of the document classification methods are rule-based, others are in machine learning, and there are also hybrid approaches. Methods based on machine learning learn to perform classifications according to past observations (look at Figure 3.1). That is, a training is performed with the labels associated with each text. Where the label, represents the category of the text. To train the classifier, the feature vector must first be extracted from the document. Then, the algorithm receives the training data, which are the feature vector (numerical representation of the document) and its label (category or class). Once trained with a sufficient amount of data, a model is generated. This model is expected to make good predictions for the categories for which it was trained. Finally, new documents are used for the prediction phase. From the documents, the vector of characteristics is extracted to be introduced in the classification model and obtain its possible label. Sections 3.4, 3.5 and 3.7 describe the methods used in this work to obtain the vector of document characteristics.

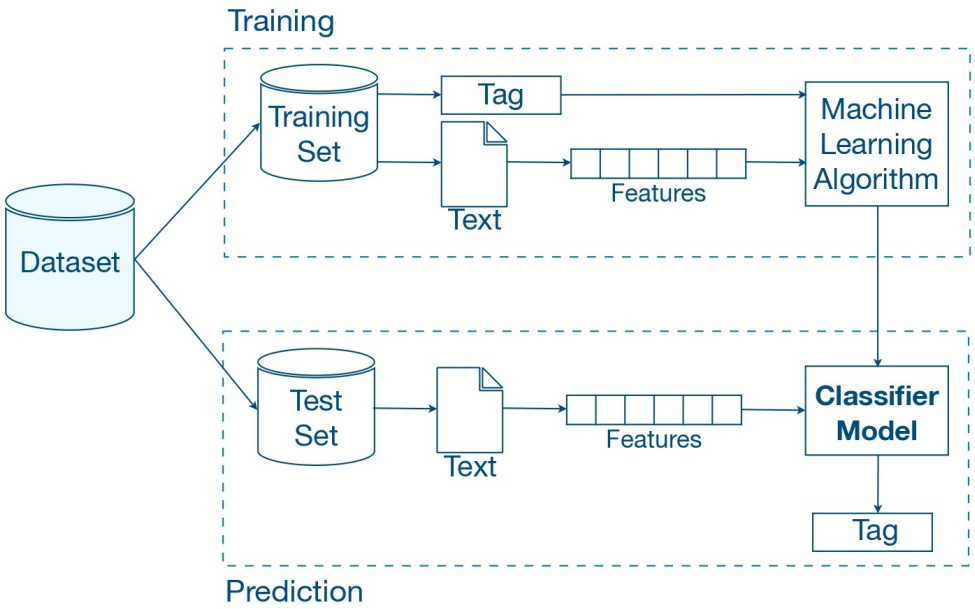



**Figure 3.1:** *Document classification process, training and prediction.*

Furthermore, there is a great number of algorithms for document classification, such as linear regression, logistic regression [VE18], Random Forest [Pal05], K Means Clustering [KMN+02], Decision Tree [RM05], KNN (K-Nearest Neighbors) [Alt92b], Gaussian Naive Bayes [MM09], SVM (Support Vector Machine) [BGV92], among others. Supervised learning models will be used in this Master's work [KPC95], next we are going to detail the classification algorithms that will be used in this Master's work:

### 3.2.1 Decision Tree

Decision tree is a predictive model that uses a directed tree like to support in decision-making. Decision tree is an algorithm that contains conditional control statements, which have a group of hierarchical decisions where the paths taken at internal nodes are the split criteria and the external nodes are final decisions [RM05]. The objective is to help identify a strategy most likely to succeed; therefore, certain criteria are considered such as: causes and consequences, possible event outcomes, resource costs, utility, among others.

The process of training and learning of Decision tree [RM05] creates a model, which predicts the value of a target variable by learning decision rules which are inferred from the features [Qui14]. This model groups the independent variables, grouping from the most common characteristics to the most specific ones, creating a series of branches to the dependent variables. Inside the decision tree models, there are **classification trees** where the dependent variables (target variables) are discrete set of values. The leaves represent class labels. The branches or paths represent combinations of features that lead to those class labels. So, the resulting classification tree is an input for classification.

Decision tree is graphical and easy to understand the model but there is a disadvantage, it works well with training data, but when new test data is included the results may not be as favorable. For example, Figure 3.2, where a tree is modeled to answer the question (where the possibility of playing soccer is evaluated).


**Figure 3.2:** *Example of Decision tree (Deciding whether or not to play soccer).*

### 3.2.2 KNN (K-Nearest Neighbors)

KNN [Alt92b] is a supervised Machine Learning algorithm. This algorithm is based on the idea of similarity (distance, proximity, or closeness) assuming that there are similar things near or with similar characteristics. If it is used with discrete values it is for classification and if it is used with continuous values it is for regression. Furthermore, KNN does not need any training data for model generation, therefore all training data are used in the testing phase.

To classify an unknown element, the KNN method first initializes $K$ choosing the number of neighbors. Then, the distance between the $q$ query element and all the elements of the dataset is calculated. Then, it is ordered by distances from smallest to largest and the first $K$ elements from the sorted collection is chosen. This subset represents the k elements closest to the $q$ query element. (i) In classification's case, the class that is associated with the query element corresponds to the majority class, which is observed in the selected k-set [Alt92a]. (ii) In regression's case, the mean of the labels of the selected k-set is returned. Figure 3.3 'shows the steps of the algorithm KNN.

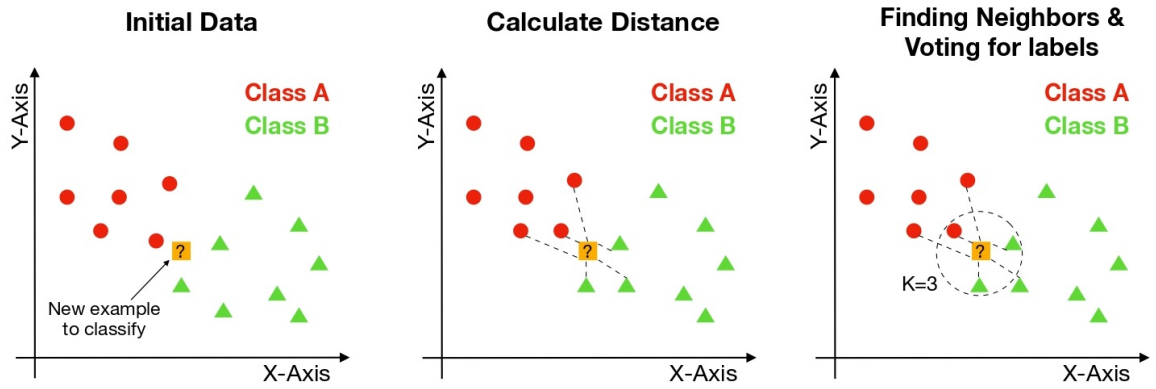



**Figure 3.3:** *KNN classification process.*

Finally, because all training data are used in the testing phase so the training phase is fast, but the testing phase is slow and expensive (memory, time and processing resources). In the worst case, if KNN will scan all the data it needs more time and therefore more memory for storing training data. For these reasons, KNN tends to work best on small datasets with a small number of features.

### 3.2.3 Gaussian Naive Bayes (Gaussian NB)

Naive Bayes [Ras14] is a statistical classification method, which used Bayes' probability theorem for predicting unknown classes. NB is suitable for a large amount of data, achieving high precision and speed. Likewise, based in Bayes' theorem with strong (naive) assumptions of independence between its features [Zha04] Born Naive Bayes classifiers, which is a supervised learning. This classifier assumes that the effect of a feature on a class is independent of other features. This assumption is called conditional class independence, which simplifies the calculation and is therefore considered naive.

Naive Bayes, can be extended to continuous data, that is, a real-valued attributes. The most common for NB is to assume a Gaussian distribution. Finally, this modification of naive Bayes is called **Gaussian Naive Bayes** [MM09]. Various functions can be used to estimate the distribution of the data, but the normal distribution or Gaussian distribution without covariance is the easiest to work. Distribution Gaussian only needs to estimate the mean and the standard deviation of your training data. Therefore, Gaussian Naive Bayes is an algorithm for classification, where the probability of the characteristics is assumed to be Gaussian.

### 3.2.4 Support Vector Machine (SVM)

SVM [BGV92] is a supervised machine learning algorithm for classification (binary classifier) and regression analysis. In both cases, there is the first training phase, which receives the data in pairs. $[(elemento_1, clase_1), (elemento_2, clase_1), ...(elemento_n, clase_1), ...]$. In its second phase or phase of use, SVM becomes a "black box" that provides a response to an input. SVM can be used for both continuous and discrete data.

Given the training data (labeled sentences), it constructs hyperplane or line that divides a space into two subspaces (As can be seen in Figure 3.4). This hyperplane is a decision boundary with the margin's property of separation between two classes being maximum [HHHH09]. One subspace contains vectors that belong to a group. The other subspace contains vectors that do not belong to that group. As an advantage, SVM does not need much training data to provide accurate results like Naive Bayes. Nevertheless, it requires more computational resources.

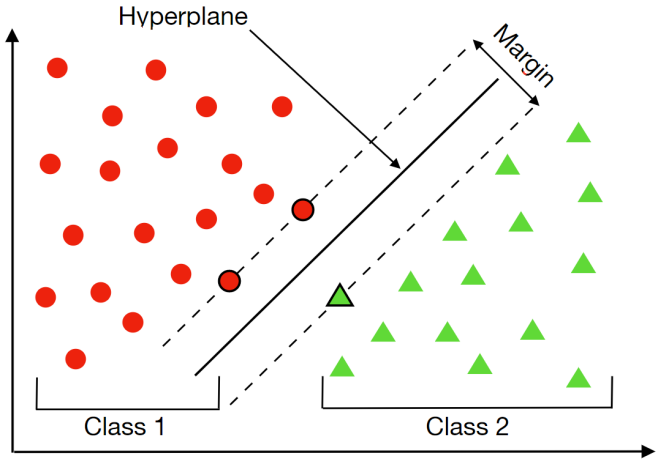



**Figure 3.4:** *Schematization of SVM for two class (Hyperplane and margin).*

## 3.3 Vector Representation of Documents

Documents are highly dimensional unstructured data, due to the number of letters and words that it usually contains. Therefore, a more summary way must be sought to represent them. The representation of documents is the transformation of the text into another form of representation that is more appropriate depending on the problem to be solved, that is, it is an alternative representation of the documents that facilitate their analysis. This representation is performed through a set of phases that contribute to certain simplifications and generalizations to presenting a logical view of the documents that allow their analysis and comparison.

For example, the vector space model (Section 3.4) represents textual documents through vectors of terms, but it does not allow to represent semantic relationships between words. On the other hand, semantic vector spaces are based on the idea that the meaning of a word can be learned from a linguistic environment (Word Embeddings and Graph Embeddings are in Sections 3.5 and 3.7).

## 3.4 Vector Space Model

Vector space model (VSM) or term vector model is one of the most used models for mathematical document representation. It is also used in filtering, retrieving, indexing and calculating the relevance of information. VSM [Com18] is an algebraic model for representing text documents as vectors of identifiers. At first, it emerged to be used in information retrieval [HSMN12], but it is also used in information filtering, automatic summaries, indexing and relevancy rankings, among others. Since it was born to be used for information retrieval, then documents and queries are represented as n-dimensional vectors. For each document we have: $d_j = (w_{1,j}, w_{2,j}, \ldots, w_{t,j})$, where $j$ is the document's number and $t$ term's number. Finally the query: $q = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$, where n is the term's number of query.

VSM is based on the notion of similarity. Therefore, vector operations are performed to compare documents with queries. This leads the model to calculate the similarity between the document $d_j$ and query $q$. Cosine similarity is commonly used to compare these vectors. In the Equation 3.1 the cosine similarity is used between the document $d_j$ and the query $q$. Where the ratio between the inner product of the document vector and the query vector and the product of the norm of the document vector by the norm of the query vector is calculated.

$$\cos(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}} \tag{3.1}$$

As we saw lines above in this model a document is represented by a vector of indices of terms extracted from the text. In Figure 3.5 a three-dimensional vector space is shown because for the best visualization of the example the vocabulary is only made up of three terms; therefore, three dimensions are denoted. We also observe four three-dimensional vectors that correspond to four documents, $D_1, D_2, D_3, D_4$; denoted $\vec{D_1}, \vec{D_2}, \vec{D_3}$, and $\vec{D_4}$. The weights of $D_1$ are $(w_{11}, w_{12}, w_{13})$. The term *weights* represent the document's orientation and placement in the vector space. As well as documents, queries are also represented as three-dimensional vectors, $Q = (w_1, w_2, w_3)$.

The similarity between document $D_1$ and query $Q$ is given by the similarity of the cosine (Equation 3.1) between its vectors. If the similarity of the cosine is 1, it means that $Q$ and $D_1$ are close, or that are the same. In contrast, the closer it gets to 0, the farther $Q$ is from $D_1$.

In Section 3.4.1 the bag-of-words model used to represent documents and queries is described. Also have been developed other more modern models, which use (term) weights, such as TF-IDF weighting (Section 3.4.2).

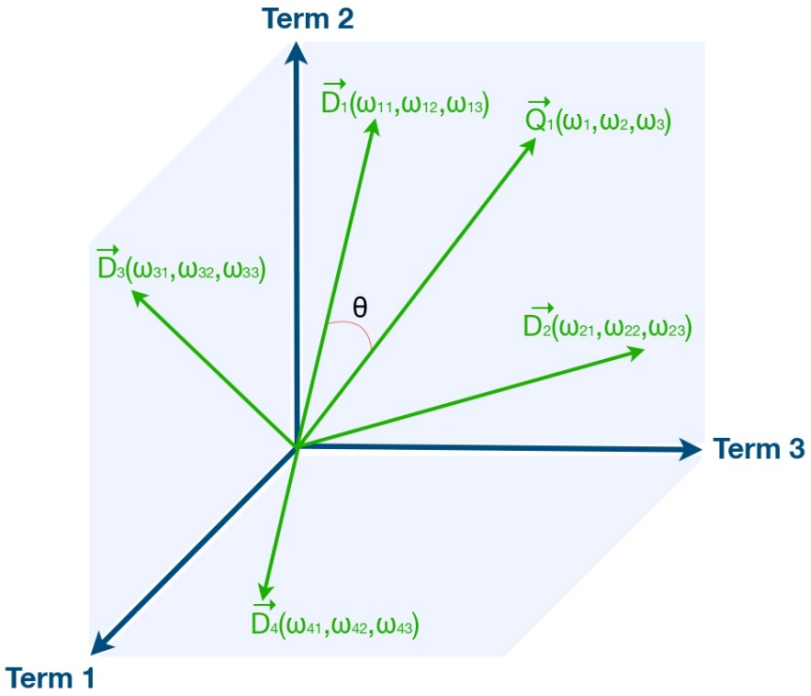



**Figure 3.5:** *Vector Space Model.*

### 3.4.1 Frequency Model

Frequency Model or Bag-of-word (BOW) is a method to obtain a simplifying representation of any text. This model has been used in many areas, such as document summarization, multi-label classification, among others. The Frequency Model represents a document as a vector. This vector is constructed from a set of words of the text called also vocabulary. First, the number of different terms in a document is counted and stores the frequency of each of these terms. Formally the vector is $d = (fw1, fw2, ..., fwj)$, where $j$ is the number of unique terms present in the document. Therefore, the dimensionality of the vector is the number of terms in the vocabulary (or the number of different words presented in the documents). While the vocabulary has more words, the characteristic vector will be larger. Although a single word is present once in the document, it is considered for the vector representation of the document.

Currently, there are other improved adaptations based on Vector Space Model, which use a weighting measure such as TF-IDF seen in the following Section (Section 3.4.2)

### 3.4.2 Term Frequency–Inverse Document Frequency (TF-IDF)

TF-IDF stands for "Term frequency-inverse document frequency". This model is a statistical measure or numerical that analyzes, how relevant or important is a word to a document in a collection of documents or corpus. Salton, Wong and Yang [SWY75a] are the proponents of TF-IDF, for various applications, it is currently often used in text summarization, stop-words filtering, information retrieval, among others.

Theoretically, TF-IDF value has an increment proportionally to the number of repetitions of a word in the document, however, is offset by the frequency of the word in the document collection. That is, TF-IDF is calculated by multiplying two measures: how many times a word in a document is repeated, and the inverse document frequency of the word across a set of documents. On the other hand, It is important to note that some words appear more than others, that is, they are more frequently than others.

From a practical perspective, this model represents the texts as vectors of identifiers in a multidimensional linear space. Formally, the vector $d_j = (w_{1j}, w_{2j}, ..., w_{nj})$ represents a document or sentence, where $n$ represents the total number of unique words that are presented in the documents and $0 \leq w_{wij} \leq 1$ is the contribution of the term $t_i$ for representation of the document $d_j$.

As mentioned above, TF-IDF is calculated from two measurements. The first measure, term frequency (TF), $tf(t, d)$, which is the number of times the term $t$ appears in the document $d$ or frequency of the word. Some documents are larger than others, raising the possibility that a word is repeated much more times in long texts than shorter ones. Therefore, we seek to normalize $tf$, that is, the frequency of the term is divided by the document length (number of terms in the document). Next, TF is calculated as follows:

$$tf(t, d) = \frac{f(t, d)}{|d|} \tag{3.2}$$

where $f(t, d)$ is the number of times the term $t$ appears in the document $d$ and $|d|$ is the number of terms of $d$, that is, the vocabulary.

The second measure is the inverse document frequency (Idf), $idf(t, D)$, where $t$ is the term and, $D$ the total number of documents, that is, the corpus. Idf is used to determine how common or rare a word is in the corpus. This metric calculated as the logarithm of the total number of documents $|D|$ (cardinality of $D$), divided by the number of documents that contain a word $DF$, as shown in:

$$Idf(t, D) = log\Big(\frac{|D|}{DF}\Big) \tag{3.3}$$

where $DF$ is the number of documents in which the term $t$ appears. Finally, TF-IDF is computed in Equation 3.4, where the Equations 3.2 and 3.3 are used.

$$TF - IDF(t, d, D) = Tf(t, d).Idf(t, D) \tag{3.4}$$

The representative vector created using TF-IDF, returns an improved approximation than BOW (Section 3.4.1). Both models are based on linear algebra and allow computing the similarity between queries and documents, representing the texts from a lexical aspect. Despite giving representative results in the area of NLP, they present highly dimensional problems, since the size of the representative vector is the number of unique words in the document. Also, it lacks semantic sensitivity, because the documents are not associated with their context. Due to these disadvantages, new methods have emerged that seek to consider the semantics of documents to create the feature vector. These methods are described in Section 3.5.

## 3.5 Word embeddings

The techniques described above (Section 3.4) have the main disadvantage that they ignore the semantics and syntax of the words. Another disadvantage is the high dimensionality of data, because the more terms the documents have, the larger the feature vector. In contrast, for Word Embeddings a fixed size is established.

Word embeddings are models for word vector representation, this model search that words with similar meaning have similar representative vectors. Therefore, in a semantic vector space, words with similar meanings should be located closer to each other. This model represents each word with a real-valued vector. For example, Figure 3.6, the word *house* has a semantic meaning related to the word *apartment*. Hence, the vector representation of word *house* should occupy a position close spatial to the word *apartment*. Formally, the cosine of the angle between these vectors should be close to one. It will allow words that are used in similar ways to have similar vector representations, and naturally capturing their meaning. Looking for the word to be represented using its context. This is an advantage over the TF-IDF model, where different words have different representations, regardless of how they are used.

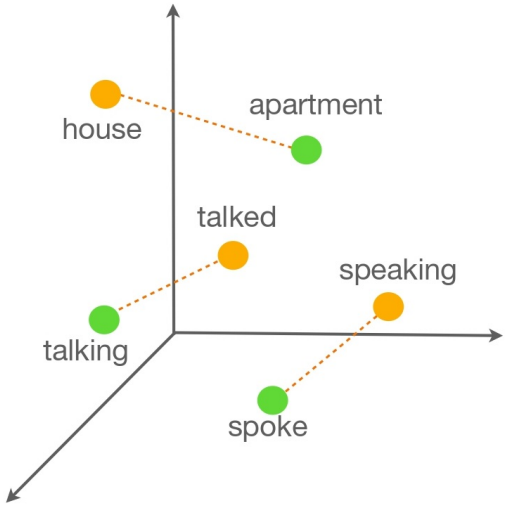



**Figure 3.6:** *Semantic representation of a word in word embeddings.*

Word embeddings models are used with success in several NLP applications such as document classification [Seb02], question answering [TKL+03], name entity recognition [TRB10], etc. There are several methods of Word Embeddings, such as Word2Vec, FastText, GloVe, and BERT. But, we studied Word2Vec (Section 3.5.1).

### 3.5.1 Word2Vec

The Word2Vec model [MCCD13], converts text into a numerical representation (vectors). These vectors are distributed in a vector space, where the vectors of similar words are close to each other (they are grouped), considering similar characteristics for their grouping such as the context. As we were saying of a vector space, the association of words with other words is shown by mathematical similarities.

In detail, Word2Vec [MSC+13] is a neural network, which learns to predict words using its context (neighboring words). Word2Vec is a shallow neural network that has three layers, input layer-hidden layer and output layer. The input layer is the collection of documents, that is, the context. The output layer is the target word. But, to obtain the feature vector (numerical representation) of the target word, the training process is repeated, so that words with similar meanings begin to the group, achieving that the weights of the neural network stabilize and thus obtain the vector. In other words, the vectors are formed by the weights of neurons of the hidden layer because from a collection of $k$ neurons we would have $k$ weights, which allow us to obtain a feature vector of size $k$.

To exemplify the model, Figure 3.7 shows an example of this neural network. Let us suppose we have the input sentence "My pet is cheerful and playful". The neural network tries to learn features (weights $W$ and $W'$), which look at words in a window, for example, "my pet is" and it tries to predict the next word "cheerful". Hence, with the input words "my", "pet", and "is"; the training process adjusts the weight of the network, so the probability of output "cheerful" is maximized; as compared to other words, which are in the vocabulary. As the training procedure repeats this process over a large number of sentences, the weights stabilize. As shown above, these weights are then used, in this example, as the vectorized representation of la word "cheerful".

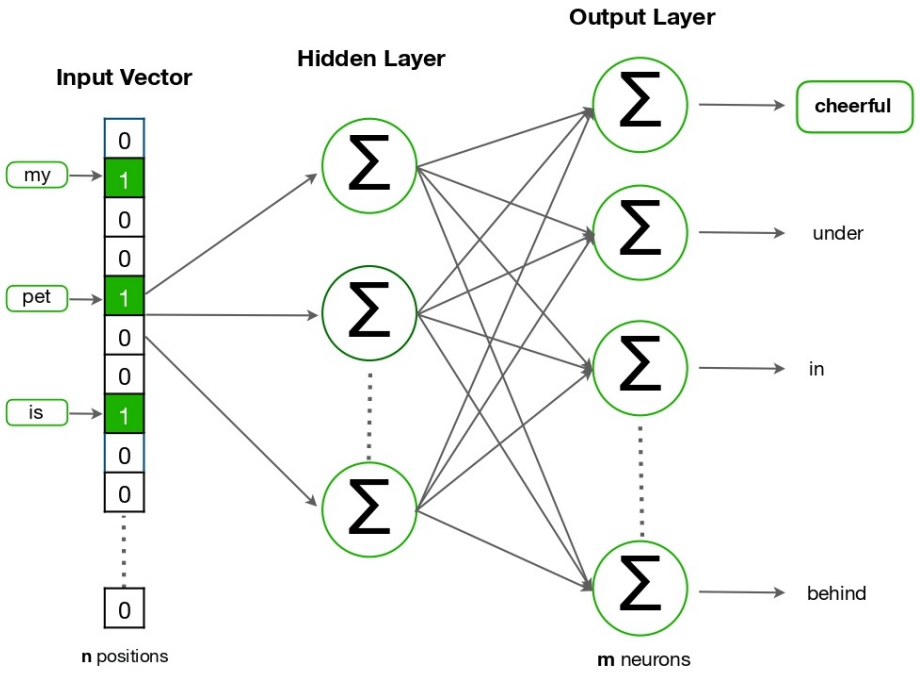



**Figure 3.7:** *Example of Word2Vec neural network.*

Word2Vec has two methods: Common Bag Of Words (CBOW) and Skip Gram. CBOW uses the context of each word (neighboring words) as the input and tries to predict the target word. The Skip-gram model reverses the use of word target to predict the context words. Due to its naturalness, CBOW is the fastest and best to predict frequent words in the corpus, in addition to using little memory. On the other hand Skip Gram is better with rare or infrequent words in the corpus, but is slower than CBOW and uses more memory than its.

## 3.6 Graphs and complex networks

The study of complex networks has almost a century of history, but since the late 50's, thanks to the study of two mathematicians, Erdös and Rényi, a great advance has been made in graph theory from the viewpoint of the formulation of mathematical algorithms, achieving a revolution when it comes to modeling these problems, differing from the classical way that used to be done, thus establishing the theory of random networks. It is true that not all networks in real life are random, but the model proposed by these two scientists established the first sensible and rigorous method of study for complex networks.

Thanks to the advances that have taken place recent years, the study of networks has advanced drastically, now being able to model and program networks and graphs with a large amount of data and with numerous tools that allow the detailed study of these networks and these data.

Currently, complex networks are studied thanks to their relationship with science and all the areas in which they can be applied. Many fields and systems that exist in reality can be modeled and studied through complex networks, graphs that are generated using nodes or vertices and arcs, which through certain indices and characteristics, allow to study them from various points of view. There are many examples of complex networks in real life, such as the Internet, the Worldwide Web (WWW) or social networks, studied from many different aspects.

### 3.6.1 Networks applied to Language Studies

Recently, a growing body of literature has been employing complex networks to model and analyze human languages. According to findings from studies considering several languages, human languages are also a complex system, with some properties emerging from experience, and social interaction, for example. Therefore, the models and tools of complex networks constitute a relevant methodology to study the language. In this context, a network $N$ is given by $N = (V, E)$, where the set of nodes $V$ represents linguistic units and the set of edges $E$ represents relationships among those units, as seen in Figure 3.8. Some examples of linguistic units are words, phonemes, and morphemes. The relationships among those units could be extracted from different linguistic levels, such as co-occurrence, syntactic, or semantic. The co-occurrence relationships represent the order of the words in a sentence.

In this model, two words are connected if they are adjacent in at least one sentence. In the syntax-based models, each edge represents a syntactic dependency, which connects a head to a modifier word. Finally, the extraction of semantic relationships requires a deeper analysis. Despite the differences, complex networks modeling any of those relationships display some properties found in many real networks. For instance, show that the co-occurrence networks of human language display the small-world and scale-free properties. Their networks presented more than 450,000 nodes and the relationships were obtained from an extract of the British National Corpus. In addition, extracted syntactic dependency networks from three languages (European Languages),

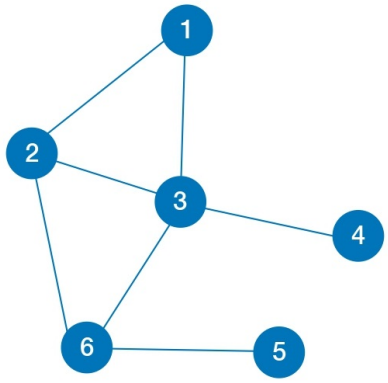



**Figure 3.8:** *Network $N$, where $N = (V, E)$. $V$ represents nodes and $E$ represents relationships between nodes.*

namely, Romanian, Czech, and German. They found the presence of small-world structures and discovered that their degree distributions follow a power law. Taken together, these findings suggest that human language presents a structure that could be described by universal patterns [KE07]: Node degree, that is the number of edges connected to that node; PageRank; betweenness, it measures the number of conversations that can pass through a node on the network; ShortestPaths, a path that links two nodes $i$ and $j$ with a minimum length.

## 3.7 Graph embeddings

Graph embeddings are methods whose objective is to extract the properties of a graph to convert them into a vector or set of vectors of fixed size. Such representations must preserve the topology of the graph, relationships between the nodes, as well as important information about graphs, subgraphs and nodes [NCV+17].

### 3.7.1 Node2Vec

Node2vec [GL16], is a method that returns numerical representations (vector modeling) of each node of a graph, optimizing the neighborhood preserving. Node2vec treats a graph as a piece of text and nodes as text tokens, although the text is a linear sequence and the graph is a complex structure. On the other hand, Node2vec looks for nodes that share similar roles and have similar representative vectors

The Deepwalk method, which is based on the BFS and DFS methods to perform random walks, was previously proposed. The sampled BFS is a local microscopic view and the sampled DFS is a global macroscopic view. Node2vec combines the BFS and DFS methods to sample the nodes in the graph, as shown in Figure 3.9. The goal is to find a flexible notion of a node's network neighborhood. As a result, perform several random walks, which efficiently explore diverse neighborhoods of a given node.

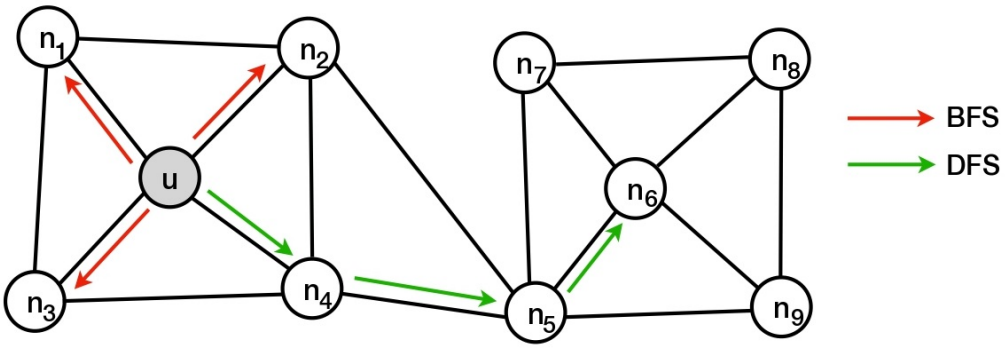



**Figure 3.9:** *BFS and DFS used in Node2vec.*

### 3.7.2 Graph2Vec

Graph2vec converts nodes and edges into a vector space representation. The objective of this method is to generate low dimensional vectors preserving the topological properties of the graph (related to graph structure and interconnections between nodes) [NMV+17].

Graph2vec represents entire graphs as fixed-length feature vectors – inspired by neural document embeddings models, the authors extend the model to learn graph embeddings. As Node2Vec is an analog to Word2Vec, Graph2Vec is an analog to Doc2Vec. Graph2Vec proposes to view an entire graph as a document. See the subgraphs around every node as words that compose the document. Moreover, add document embedding neural networks to learn representations of entire graphs. An input of algorithms is a set of labeled or unlabeled graphs, in the case of unlabeled graph nodes are labeled with their degree (or you can provide a set of features for each node). The main motivation behind this idea is that structurally similar graphs will be close to each other in the embedding space.

# Chapter 4

# Related works

For this research, we performed a comparison between several document vector representation methods for the authorship recognition task. In most Natural Language Processing related applications, the vectorization stage is necessary, where the input texts are converted into their mathematical representation or a more appropriate model. Over the years, researchers have proposed several methods to find the most suitable representation of a document. These methods could be divided into two large groups: representations based on high-dimensional vectors (classical methods like Vector Space Model) and methods based on dense vectors with fixed dimensions (Word embeddings and Document embeddings). Another more recent approach widely studied is modeling texts as graphs or networks. Researchers used different measurements and concepts related to complex networks to find interesting patterns derived from a set of texts. These concepts are useful for various applications such as document summarization, keyword extraction, analysis of author writing for authorship attribution, topic labelling, automatic correction of texts, among other applications.

We describe in Section 4.1 the main works related to authorship recognition. Then, in Section 4.2 we describe the most relevant works that dealt with the traditional methods of vector space models, subsequently in Section 4.3 we briefly detail the works and applications related to word and sentence embeddings. Finally, in Section 4.4, we explain how the concepts of graphs and complex networks were used to model texts for several applications.

## 4.1 Authorship attribution

The objective of authorship attribution methods is to extract textual features that are capable of distinguishing the authors of a set of texts. The research area in charge of finding features that quantify the authors' writing style was called stylometry. The different features that can be extracted from the texts can be grouped into the following categories: lexical and character features, which consider the texts as sequences of words or characters, while the syntactic and semantic features require deeper linguistic analysis and advanced computational methods for its development [Sta09a].

One of the first works to use lexical features was [Men87], where the length of sentences and words was used to identify the authorship of a set of documents. Various works used as lexical attributes the sentence and word length, word frequencies, character frequencies and vocabulary richness. Other works studied the importance of the use of function or common words, which includes articles, prepositions, and pronouns. According to [Sta09a] these words can capture the writing patterns of the authors. Other investigations considered the most frequent words as features related to an author's writing, therefore, they considered each text as a vector of frequencies. In this way, it was possible to use machine learning methods to classify these extracted vectors. Several authors compared the results of evaluating both function words (stopwords) and content words (most frequent words) as features. The authors concluded that frequency-based features could reveal certain stylistic patterns in texts.

Other authors considered analyzing the texts as sequences of characters. In this way, they established the following features: alphabetic counts, the frequency of punctuation marks, frequency of letters, digit character count. Another approach was to extract the frequencies of n-grams characters, that is, they selected subsets of n adjacent characters to be considered as features of each text. Several authors concluded that the most common n-grams of characters are relevant for authorship attribution [Sta09a].

Other research focused on syntactic approaches that included the extraction of rewrite rule frequencies. These approaches measure how a sentence could be decomposed into its constituent parts. In this sense, they analyzed sentence and phrase structures from source documents. Other works applied part-of-speech (POS) tagging to find the grammar categories of each word (nouns, adjectives, verbs, adverbs) and in this way, they analyzed the proportion regarding the use of these categories by each author. Recent works considered semantic attributes, where they analyzed the semantic relationships between the words of a text. Synonymy relationships and semantic dependencies were commonly used for these approaches [Sta09a, GAPDSP18].

## 4.2 Classic techniques and vector space models

The need to represent texts as numerical vectors arose from the information retrieval area. The objective of this area is to find data of an unstructured nature (generally texts) that satisfies an information need within large data collections (stored on the Internet). In this way, to effectively retrieve the most relevant documents, the texts were transformed into a logical or mathematical representation of them [Cho10].

The first models were models based on set theory, where documents are represented as a set of words or phrases. The most important models are the Boolean model, Extended Boolean model, and Fuzzy model. In Boolean models, the documents to be searched and the user's query is conceived as a set of terms. Therefore, the retrieval is based on the presence or absence of the query terms in the documents. Although they are easy to implement, these models have several weaknesses: they can retrieve many or few documents, all terms have the same weight, and it is difficult to establish a ranking among the retrieved documents [LMG09, War92].

From the disadvantages of previous models, the algebraic models emerged, where the documents and queries are represented as vectors, matrices or tuples. The best known models were the vector model (bag of words model), and the generalized vector model. For the bag of words model or frequency model, each component of the document vector reflects the importance or weighted frequency of each term (word) in the document [SWY75b]. The computation of the importance of each term played a fundamental role in making the similarity between the more reliable documents. For this reason, other more efficient methods were proposed for weighting the terms of each document vector. One method that presented several improvements was the TF-IDF weighting model.

The TF-IDF model was developed as an improvement on the Boolean and frequency-based models. This weighting scheme measures how relevant a word is in a document collection. The TF-IDF value increases proportionally to the number of times a word appears in the document, but it is compensated by the word frequency in the document collection [SB88]. Even though this model had significant improvements over previous models, it still has some limitations:

- Long documents are underrepresented as they contain few common values
- Search words must match the words in the document
- Semantic sensitivity, documents with similar contexts but with different vocabulary will not be associated
- High dimensionality for large document collections

To reduce the dimensionality problems presented in the previous approaches, the LSA (Latent semantic analysis) method was proposed [DDF+90]. This method assumes that words close in meaning occur in similar text segments. Like the previous methods, a matrix containing word counts per document was constructed over a text collection. In that matrix, rows represent unique words and columns represent each document. After the matrix was constructed, a mathematical technique called singular value decomposition (SVD) [GR71] was applied for that matrix. This technique was used in order to reduce the number of rows while preserving the similarity structure among the columns of the matrix. The new values resulting from this reduced matrix were considered as representative vectors of each text. Therefore, the documents could be compared by considering some similarity measurement between the vectors representing such documents. The similarity measurements (for example cosine similarity) could have values between 0 and 1, where values close to 1 represent similar documents, while values close to 0 represent different documents.

## 4.3 Word and document embeddings

First, in Section 4.3.1 we will describe the main methods related to Word Embeddings. Later, in Section 4.3.2 we will discuss about Document Embedding methods.

### 4.3.1 Word embeddings representations

Due to the limitations of the sparse vector representations, methods based on word and document embeddings were developed. Word embeddings are a set of methods that assign a representative vector for each word in the document. Such vectors store semantic information, therefore, they can be associated or dissociated to other vectors (words) according to different contexts [Bak18]. Word embeddings have been useful for several important NLP related applications such as:

- Identification of synonyms, where the similarity between the vectors of two words is greater than a previously defined threshold.
- Semantic clustering of a set of similar words (words whose feature vector is similar).
- The vector of each word can be used to generate the representative vector of a complete text for text classification and text clustering approaches.

The first word embedding approaches that emerged were Word2Vec, GloVe, FastText, among others. The Word2Vec model, proposed by [MSC+13], is composed by a neural network with three layers. The main idea of this algorithm was to learn word representations that can predict a word given its surrounding or context words. This model was quite useful to detect synonyms and to recommend words from an incomplete sentence. The authors of [MSC+13] even evaluated this method for the word analogy task, where, from the relations "Germany-Berlin", and "France-X", Word2Vec was able to correctly predict the word "Paris".

The models for learning word representations are mainly divided into two large groups: global matrix factorization methods and local context window methods. Word2Vec is a model based on local context window methods. The GloVe model (Global Vectors for Word Representation) [PSM14] was proposed as an improvement to the Word2Vec method because it incorporates both characteristics of the methods to learn word representations: it is based on local statistics (local context information of words), and it also incorporates global statistics to obtain the representative word vectors. This method was successfully evaluated for the following applications: word analogy task, word similarity, and named entity recognition.

Most Word embeddings methods ignore word morphology (internal structure of words), therefore, this limitation could be a problem for morphologically rich languages containing a large vocabulary with many rare or uncommon words. [BGJM16] proposed the FastText method with the aim of overcoming such limitations. This model is an extension of Word2Vec, and each word is represented as a set of character n-grams(subwords of n letters): corpus words are divided into several n-grams and they are trained in the neural network. After the training stage, all extracted n-grams are associated with their respective word embedding representation. Therefore, the vector representation of a word is based on the sum of the vectors of the n-grams comprising such words. In comparison to other methods, FastText has the following strengths: the training stage is faster than other methods, and it can obtain vector representations for words that did not exist in the training data. Also, this method is good to represent rare or uncommon words.

### 4.3.2 Document embeddings representations

The goal of methods based on document embeddings is to map sentences, paragraphs or whole texts into informative vector representations. We will now describe the most popular approaches related to document embeddings. Several of these methods are inspired by approaches based on word embeddings, and some of them can be considered as generalizations of the Word2Vec model.

The simplest methods for obtaining document embeddings were based on averaging word embeddings. Therefore, given a document, it was feasible to apply some arithmetic operations on the vectors of the words that comprise the document to obtain a unique vector representing the entire document. The operations that were applied were average and sum. These methods were quite useful for representing documents, however, models based on word embeddings need to be optimized for task sentence representation. For that reason, [KBDR16] trained word embeddings directly for the purpose of being averaged. They trained a neural network that learns word embeddings by predicting, given a sentence representation, its surrounding sentences. This model was successfully evaluated on 20 datasets for different applications.

The authors of [PGJ17] proposed the Sent2Vec model, which is a combination of the classic Word2Vec model and the method based on averaging word vectors. Word2Vec is extended to include word n-grams, and it is also adapted to optimize the generated word embeddings in order to average them to yield the representative document vectors. One of the strengths of this method is its low computational cost during the training and inference stage of sentence embeddings. Therefore, it is a very useful method to learn sentence representations from extremely large datasets.

One of the first attempts to generalize the Word2Vec method to obtain document vector representations was the Paragraph Vector model (or Doc2Vec) proposed by [LM14]. They trained the vector representations for predicting words in a paragraph. [LM14] concatenated the paragraph vector with the representative vectors of the word comprising the paragraph, and then they predicted the following word in the given context. The word and paragraph vectors were trained using the neural network algorithms stochastic gradient descent and back propagation. This method was evaluated for the following NLP applications: sentiment analysis, information retrieval and document similarity.

The Doc2VecC (Document vector through corruption) algorithm, proposed by [Che17], combined the previous model based on paragraph vectors (Doc2Vec) with methods that average word embeddings to get document vector representations. This algorithm used a neural network composed of an input layer, a projection layer and an output layer to predict the target word. Where, the embeddings of adjacent words provide local context, while the vector representation of the complete document represents the global context. Different from the paragraph vectors, Doc2VecC represents each document as an average of the embeddings of words randomly sampled from the document. Initially, this method was evaluated for sentiment analysis, document classification, and semantic relatedness tasks. The results showed that this method achieved a better performance than the Doc2Vec model.

[KZS+15] proposed the model called Skip-thought vectors, which used the Word2Vec method in the following way: they extended this method to sentences and they proposed an encoder-decoder model. The encoder trains the input sentence and generates a vector for that sentence. Then, two decoders that have as input parameter the generated vector from the previous step. The first decoder attempts to predict the previous sentence, while the second decoder tries predicting the next sentence. [KZS+15] used recurrent neural networks (RNN) to construct the encoder and decoder. The encoder uses a word embedding layer in order to convert each word from the input sentence to its corresponding word embedding. The decoders also used this embedding layer. This method was successfully evaluated for several NLP tasks such as semantic relatedness, paraphrase detection, image-sentence ranking, question-type classification, and sentiment and subjectivity analysis. However, the training stage of the Skip-thought model is very slow as many deep neural language models. In this sense, [HCK16] proposed the FastSent method, which is a significantly simpler variation on the Skip-thought method but with much lower computational expense: given the vector representation of a sentence in context, this method simply predicts the adjacent sentences. The FastSent generated vectors were evaluated for six sentence classification tasks: paraphrase identification, movie review sentiment, product reviews, subjectivity classification, opinion polarity, and question type classification.

The latest models we review for this research are the BERT (Bidirectional Encoder Representations from Transformers) [DCLT18] and the Sentence-BERT(SBERT) [VSP+17]. These models achieved a better performance compared to other methods because they are based on contextual word embeddings. For example, for the sentences "I like apples", and "I like Apple Macbooks"; the other methods will produce the same representation for the word "apple", while the BERT-based methods can differentiate the context of that word and generate two vector representations (for the fruit context and computer context). These models are capable not only for capturing the word polysemy, but also they can capture other relevant information that would produce more accurate feature representations.

BERT is an encoder that obtains bidirectional representations from transformer networks. This method first creates an embedding for each unique word in a sentence (like Word2Vec). However, the same words can appear in more than one sentence (even in different contexts). Then, BERT creates embeddings for each word pair in the sentence, but this algorithm considers how close these words are to one another in the sentence. Therefore, the BERT method was an excellent model to extract the feature vectors of words comprising the documents, also that has been pre-trained with a gigantic corpus because it was born initially to solve different NLP tasks. The Sentence-BERT method was proposed as an improvement of BERT in order to generate semantically meaningful sentence embeddings. Both BERT and SBERT were evaluated for the following tasks: question answering, language inference, and document classification.

## 4.4 Complex networks for text representation

The modeling of texts as graphs and the use of complex network concepts have meant a great advance for different applications in the NLP area. Recent studies have used such representations for various applications such as automatic language understanding and generation, sentiment analysis, word sense disambiguation, text summarization, keyword extraction, text classification, authorship attribution, semantic analysis, among others. For example, for document summarization tasks [TA17, TA18], documents are represented as sentence networks, where each node is a sentence and edges are connected according to a similarity measure. In these works, complex network metrics are applied in order to find the most relevant sentences (nodes) that could compose a final summary. For the keyword extraction, documents are represented as word co-occurrence networks, where most important words from such networks are considered as keywords. Several works used complex network measurements to represent texts achieving excellent results.

In this sense, the concept of linguistic networks emerged, where research on statistical physics is used in the study of languages [COJT+11]. A linguistic network could be composed by a group of interconnected syllables, words, sentences or paragraphs. The relationships (edges) between these text segments (nodes) were established in several ways.

According to [COJT+11], linguistic networks could be classified into semantic and superficial networks. Semantic networks contain semantic relationships between words such as synonymy, antonymy, hypernymy and hyponymy. These networks are constructed from dictionaries, lexicons or thesaurus. Superficial networks are based on: the internal structure of words (for example morphological relationships), position of words in the text, syntactic structures, among other factors.

For semantic networks, the nodes are represented by words from any language, and two nodes are connected if they express similar concepts [COJT+11]. Thesaurus is commonly used for constructing these networks. A thesaurus is a list of controlled words or terms used to represent concepts [Rei96]. For English, the WordNet database [Mil98] is an excellent tool to construct semantic networks. WordNet is a lexical database that groups English words into sets of synonyms, providing short and general definitions and it can store the semantic relationships between sets of synonyms [Mil95].

There are several examples of superficial networks. For example, the syllables that compose any language could be used to build a network, where nodes represent the syllables and the edges represent that two syllables co-occur in the same word [COJT+11]. Similar to this idea, word co-occurrence networks or word adjacency networks are constructed to represent the proximity of words in documents. For such networks, words represent the nodes, and two words are connected if they are neighbors in the text [CS01].

Word co-occurrence networks were useful to represent the technical and literary texts for several languages [CLA+06]. The topological properties of these networks were used as feature vectors for several classification tasks [dACA16]. In the same way, these networks were used for authorship characterization, where network measurements served to represent stylistic features from English famous writers [MHA16, APNOJ07].

The authors of [Edm98] used word co-occurrence networks to identify synonyms in a given context. In another application, the text quality essays written by scholar students were evaluated using the measurements extracted from the networks. The network measurements correlated with text quality scores assigned by humans [ANOJC07]. In the same way, these networks were used to evaluate the quality of machine translations. The network measurements from the original text and the translated text were compared. The authors evaluated the difference between the original text and the translation [AAP+08].

Word co-occurrence networks were also used for keyword extraction. [VOGMB19] extracted several network measurements for each word, and they considered as keywords the words associated with the top values of the considered measurements. A sentence network was proposed by [AOJdFCN09, TA17] for the document summarization task. The nodes of such a network are sentences, and two sentences are connected by a similarity metric. [AOJdFCN09, TA17] employed several network measurements in order to rank each node (sentence). The most ranked sentences were considered to compose a final summary.

Other important networks were lexical and syntactic networks. Lexical networks were useful to the construction of spellcheckers, where each edge is linked with the orthographic distance between words [CTM+07]. The syntactic networks were used for the study of language acquisition [ANOJC07].

In relation to the focus of our work, the authors of [SKD19] discuss the correspondence or relationship between the authorship of a document and the structure of the network generated from the document. First, they generate from the document an adjacency network of words, that is say, a graph. Then, they extract some properties of the graphs to form the characteristic vector of each document, such properties are: Vertex degree, Clustering coefficient, Average shortest path length, Assortativity coefficient and Modularity. Finally, they normalize the vector, to go to Data grouping and classification methods and assign the author to each document.

## 4.5 Graph embeddings

The methods of graph embeddings consist of obtaining from a graph a vector or set of vectors representing such graph. This representation must capture the topology of the graph, relationships between the nodes and other information that is relevant to the graph, subgraphs and nodes. The more properties of the graph that the vector captures, the better the graph will be represented for several applications. The graph embeddings can be grouped as follows [GF18]:

- Vertex or node embeddings, where we obtain a representative vector for each node of the graph
- Graph embeddings, where a single vector can represent all the properties of a graph.

There are many challenges for extracting the embeddings from a graph. Below we mention the most relevant challenges [GF18]:

- It is necessary that such vectors represent the topology of the graph, the connections between each vertex, as well as the neighborhood of each node.
- Networks representing real world applications are generally large, including thousands or millions of nodes and connections. It is a real challenge to implement a method that generates a vector representation for these types of graphs with a low computational cost maintaining the topological properties of such networks.
- It is also necessary to know what size or dimension the embedding vector of each graph will have. High dimensional representations will preserve much more information, however, they may require too high computational times for their implementation.

According to [GF18], the methods for obtaining the vector representations of graphs can be divided into three categories: methods based on factorization, methods based on random walks, and methods based on deep learning. Methods based on matrix factorization represent the connections between nodes using matrices and then they factor such matrices to obtain the embedding representation. The most common matrices to represent the connections are the following: node adjacency matrix, the Laplacian matrix, probability matrix, among others. These methods are based on the properties found in each matrix of the graph. On the other hand, random walks are quite useful for simulating various topological properties of the graph, such as centrality measures and similarity metrics. They are also useful because they partially traverse the graph to obtain meaningful measurements when the graph is too large to fully analyze. For this reason, some methods use algorithms based on random walks to obtain the representative vectors of each node of the graph. Finally, deep learning-based methods use deep neural networks in graphs to apply dimensionality reduction.

Recently, several studies are focusing on developing methods to represent graphs as vectors due to their large number of applications, such as i) network compression, where graph embeddings can be interpreted as a summarization of the graph [WCZ16]; ii) visualization, since embedding represents the graph in a vector space, dimensionality reduction techniques such as PCA (Principal Component Analysis) can be applied to visualize the graph [PARS14]; iii) clustering, where clustering algorithms such as k-means can be used on the embedding of the nodes to group them and visualize the clusters found by these methods. In the same way, the embeddings of a set of graphs can be used by k-means to group graphs with similar properties [WS05]; iv) node and graph classification, where the embeddings can extract the features of each node or graph based on the structure of the graph. For example, in a word network (where each node represents a word), each word can be classified according to its grammatical category, in this sense, the embedding of each node could be used with machine learning algorithms to determine whether a word is a noun, verb, adjective or adverb [GF18].

From the mentioned applications, we think that it is possible to model literary texts as graphs (networks of words or networks of sentences) and then obtain the embeddings vectors of each generated graph. Finally, these vectors can be used as input to several machine learning algorithms. In the next chapter, we will describe the use of these methods in more detail.

# Chapter 5

# Methodology

In Chapter 3 and Chapter 4, we described several methods commonly used for the mathematical representation of documents. These methods included traditional vector space methods and word and graph embeddings models. The above-mentioned were successfully used to represent texts for several NLP tasks. We also studied concepts related to complex networks, where a document could be represented as a graph of words, sentences or paragraphs; and edges could be established in several ways. Recently, many works have focused on representing texts as networks, achieving excellent results. They found that networks are a powerful tool to represent everything in our real world. Due to the good performance achieved using complex network concepts, novel works proposed to compress the properties of such networks in vector representations [SKD19]. In this way, related concepts to graph and node embeddings emerged.

The main goal of this Master work's is to perform a comparison between several methods for vector representation of documents for the task of authorship recognition. Therefore, given a set of features extracted from literary texts and a set of candidate authors, we used several supervised classifiers to associate each author with his corresponding book. We selected two datasets for authorship attribution containing literary books written in English and Spanish. We generated several feature vectors for each literary text with the aim of determining which vectors best represent the writing patterns of each author. In the same way, we intended to analyze whether the proposed methods of vector representation of texts are capable of adequately capturing and representing the particular characteristics of each language (characteristics related to the lexical, syntactic and semantic level of each text).

We analyzed the strengths and weaknesses from the most traditional vector space models to the most recent methods of graph and node embeddings. Many vector representation methods have been extensively studied. For example, methods like vector space models (frequency and TF-IDF model) and word embeddings (Word2vec) have been successfully employed to represent texts in different NLP tasks. Another objective of this work is to analyze if those new models of text representation more sophisticated (Graph embedding), they achieve represent texts as feature vectors. For Graph embedding we use Graph2vec and Node2vec. These models seek to be better than the traditional methods, such as vector space models and word embedding.

In order to graphically understand the entire process of the proposed methodology, Figure 5.1 shows the architecture for this research.

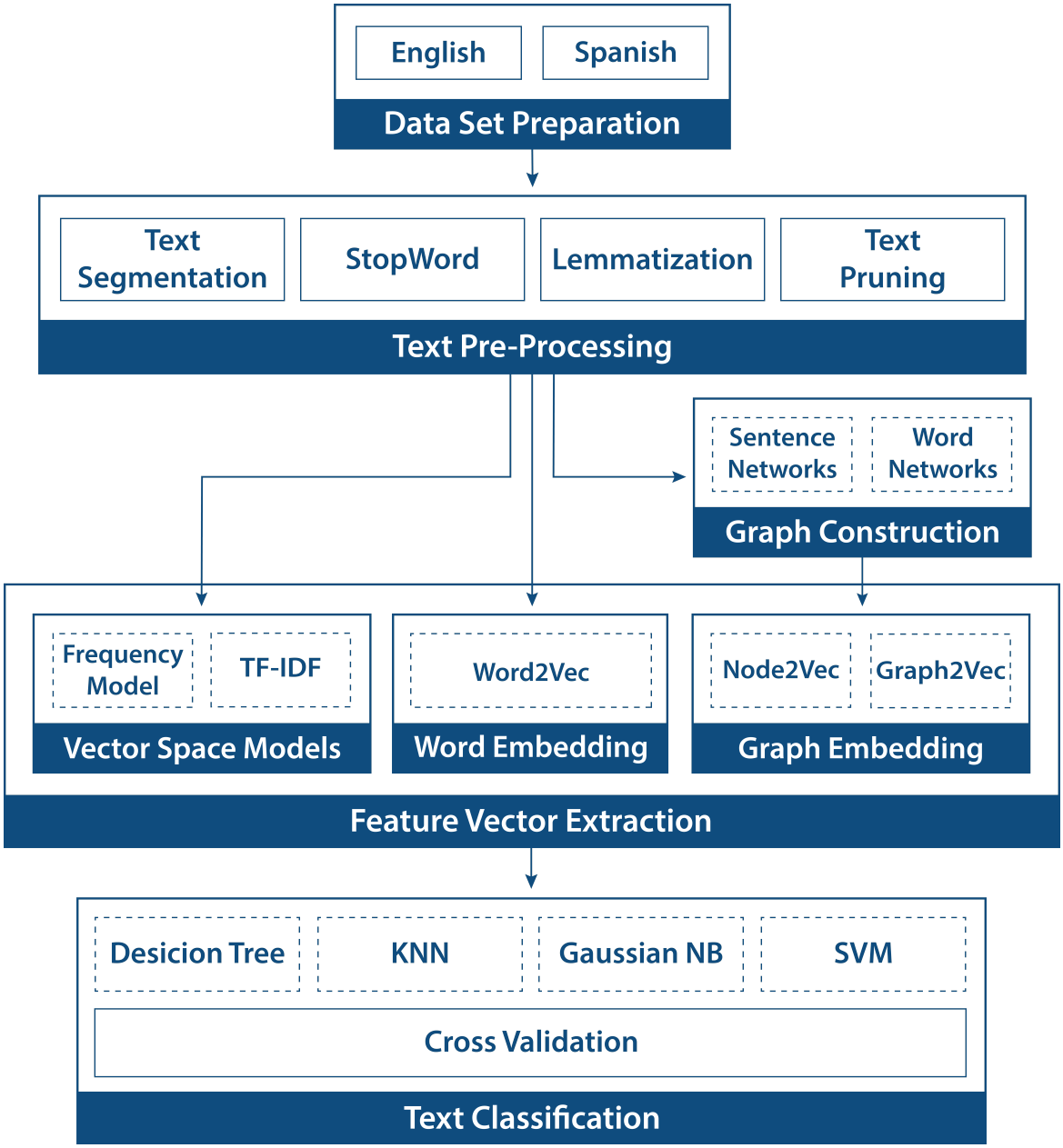



**Figure 5.1:** *Architecture for this master research.*

The proposed methodology for comparing the performance of vector document representation methods using authorship attribution is organized as follows:

1. **Dataset preparation:** We selected two datasets of authorship recognition for text written in English and Spanish. Section 5.1 describes the selection process for both datasets.

2. **Text pre-processing:** This step is responsible for preparing the source texts to be used for vector extraction algorithms. This preparation includes text segmentation, stopword removal, lemmatization, and text prunning. Section 5.2 shows the text pre-processing step.

3. **Feature vector extraction** (Section 5.3): The main phase of this work includes the methods for representing texts as feature vectors. Each one of these models includes several ways to extract the feature vectors of documents. The proposed methods are listed as follows:

   (a) Vector space models: Section 5.4 shows the process for vector extraction using the Frequency and TF-IDF model.

   (b) Word embeddings: Section 5.5 describes one of the first word embedding models, Word2vec.

   (c) Graph embeddings: Section 5.6 shows the construction process of document networks. We modeled texts as word co-occurrence and sentence networks. This step also includes the modeling of such networks as vectors using graph and node embeddings.

4. **Classification:** The feature vectors obtained in the previous step are used as input for traditional machine learning methods. Section 5.7 describes in detail this step.

## 5.1 Datasets

For this Master work, we selected two datasets (in English and Spanish) for the authorship attribution task. Authorship attribution is a method, which assigns a text document the most likely writer or author from a set of candidate authors [Mar].

The selected datasets were downloaded from the Gutenberg Project [Har71], which is an online library containing more than 60,000 free eBooks in different languages. To collect the datasets, we considered the following criteria:

- **English dataset:** We extracted from the Gutenberg library the English books following the works of [SKD19, FdANSQM+18]. This dataset comprises 78 books containing 13 authors and 6 books per author. The complete list of the books and authors is detailed in Appendix A.1.

- **Spanish dataset:** We collected this dataset from the Gutenberg library because we did not find available datasets for Spanish. We first collected all the Spanish documents from Guttenberg, and then we looked for those authors with the largest number of books. After removing several texts, our Spanish dataset comprises 105 books containing 21 authors and 5 books per author. The complete list of books is detailed in Appendix A.2.

Among the criteria for selecting the authors we have (i) Each author studied has at least 5 books, (ii) Each book has an average of 57643 words for English texts and 33451 for Spanish texts and (iii) Availability, each book is available to the public.Below we will show in Table 5.1 some statistics related to our database: the average number of sentences, average number of words, maximum and minimum number of sentences, maximum and minimum number of words.

| Statistical data | English | Spanish |
|---|---|---|
| **Maximum number of sentences** | 14717 | 10800 |
| **Minimum number of sentences** | 1562 | 186 |
| **Average number of sentences** | 5547 | 3042 |
| **Maximum number of words** | 154206 | 99242 |
| **Minimum number of words** | 22986 | 1458 |
| **Average number of words** | 57643 | 33451 |

**Table 5.1:** *Statistical data from the English and Spanish dataset*

When the dataset selection process is performed, the larger the size of the documents, the less marked the authors' styles are (the more the styles resemble each other) and the less variety of epochs the dataset has, it will be more difficult to find the author for each document. So, to make a fair comparison between the performance of the methods used when using texts in English and Spanish, we proceeded as follows. On the one hand, documents written in English have more words than documents written in Spanish. On the other hand, documents in Spanish are from nearby times, being more difficult to classify by author and thus we managed to equalize the difficulty identifying the author in both cases.

With these dataset where each document is cataloged with its class, in this case the class is the name of its author. The entire process for the extraction of feature vector is conducted. However, before performing this process, the texts to be used must be prepared, as seen in Section 5.2.

## 5.2 Text Preprocessing

Before extracting the representative vectors for each document, we need to transform the texts of dataset in a more convenient way. This process is called text pre-processing. This phase comprises the following pre-processing steps: text segmentation, stopword removal, lemmatization, and text prunning. The definition and implementation of the previous steps are described as follows:

- **Text segmentation:** We used the Python *NLTK* - Natural Language Toolkit [Bir06] for text segmentation. This step consists of dividing the texts into sentences. A sentence is defined as any text segment, which is separated by a period, the exclamation or question mark. We applied this step because the following vector representation techniques need as input the documents divided into sentence vectors like graph embeddings. This method is described in the following sections.

- **Stopwords and punctuation removal:** In this step, we removed the stopwords, which are usually filtered out in several NLP tasks and do not contribute semantic content. The stopword list generally contents prepositions, adverbs, and articles. An example of stopword elimination for English is shown in Table 5.2, the words to be deleted are crossed out.

| Orginal Text | Text without stopwords |
|---|---|
| This is a very bare and incomplete way of putting the case. The human soul moves in many channels, and Mr. Casaubon, we know, had a high sense of justice | This ~~is a very~~ bare ~~and~~ incomplete way ~~of~~ putting ~~the~~ case. ~~The~~ human soul moves ~~in~~ many channels ~~, and~~ Mr~~.~~ Casaubon ~~,we~~ know ~~, had a~~ high sense ~~of~~ justice |

**Table 5.2:** *Example of stop word removal (Text taken from: Middlemarch. Author: George Eliot)*

For this research, we considered removing the stopwords considered by the Python NLTK toolkit [Bir06]. This library has a list of stopwords stored in 16 languages. We also removed punctuation marks. In the Aprendex B we can see the complete list of storpwords and punctuation marks for English and Spanish. In Table 5.2 we can see an example of removing stop words from a text.

- **Lemmatization:** This step is responsible for transforming the words into their canonical form (or lemmas). For example, plural nouns converted to their singular version while verbs are transformed into their infinitive forms. An example of stemming for English is shown in Table 5.3, the changes are highlighted.

| Text without stopwords | After Lemmatization |
|---|---|
| This bare incomplete way putting case. human soul moves many channels Mr Casaubon know high sense justice | This bare incomplete way **put** case. human soul **move** many channels Mr Casaubon know high sense justice. |

**Table 5.3:** *Example of Lemmatization (Text taken from: Middlemarch. Author: George Eliot)*

For the implementation of this step, we used the WordNet Lemmatizer from NLTK library [Bir06], for English text. For Spanish texts, we used the Spacy library [HM17]. This library contains several models to predict named entities part-of-speech tags and syntactic dependencies. Spacy was trained for 16 languages. They can include word vectors (which will be used as features during training) and other pre-trained representations.

- **Text pruning:** The books we selected vary a lot in relation to their size. The book size was considered in relation to its number of words. There are numerous books with thousands of words and there are plenty of books containing few words. Due to this unbalanced problem in relation to the size of such books (number of words), we need to apply a text pruning step. For this step we first get the size $M$ of the smaller book ($M$ is the number of word). Then, we select the first $M$ words for each book from datasets. We performed two types of test for this step: i) applying the text pruning for each book, and ii) considering the entire content of each book. In the Chapter 6 (Results) we detailed the performed tests.

After conducting all previous steps for the English and Spanish dataset, we stored the pre-processed versions of such datasets. We did this procedure because the pre-processing stage is a time consuming task. In this way, To avoid repeating several times the text pre-processing methodology, we simply used the dataset containing the pre-processed texts.

## 5.3 Feature vector extraction

We considered the pre-processed datasets to generate the feature vector. In order to obtain the feature vector that represents each book from the selected datasets, we developed three techniques. These three different techniques are: i) **traditional vector space models** (Section 5.4), ii) we used more recent methods called **word embeddings** (Section 5.5). iii) For the third method, we converted the pre-processed text into networks. Such networks were used as inputs for **graph embedding models** (Section 5.6). In the following sections, we will describe with more detail the above three ways for document representation. In Figure 5.2 we can observe these three paths and the methods experienced in each path.

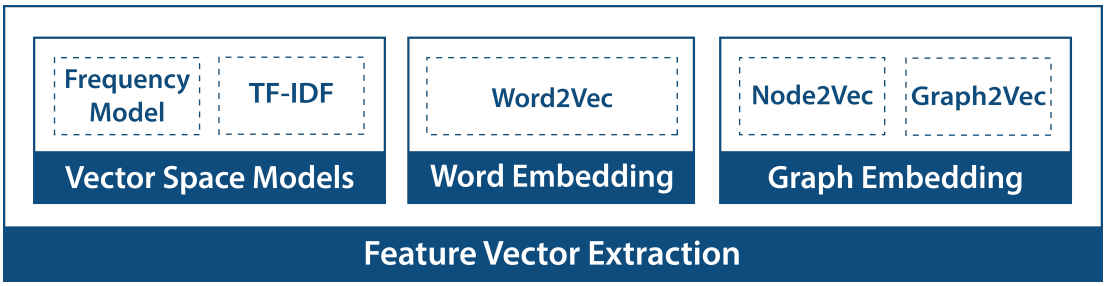



**Figure 5.2:** *Feature Vector Extraction*

## 5.4 Vector space model

The first methods we proposed are based on the traditional vector space models, such methods have been used with success in text clasification due to their simplicity and good results they obtained. These methods represent the texts as high sparse numerical vectors of $N$ dimensions, where $N$ is the total number of unique words from the corpus of texts. The most widely used approaches are the frequency and TF-IDF models. Next we will explain how we used both approaches:

### 5.4.1 Frequency-based approaches

Each source text from the pre-processed dataset is converted into a frequency vector of $N$ dimensions, where $N$ is the vocabulary size from dataset. We used the Feature Extraction module forms the Python skLearn library [PVG+11] for the implementation of this model.

Due to its simplicity, this model has some weaknesses. For example, very frequent words do not necessarily denote importance in a text. Words that appear with high frequency in several texts are very common words that would not allow to differentiate one document from the other. For this reason, the TF-IDF method was proposed, which gives more relevance to the most important words for a specific document in relation to the complete corpus.

### 5.4.2 TF-IDF based approaches

This method was proposed as an improvement of the frequency-based model. Similar to the previous method, pre-processed text is transformed into a N-dimensional vector. Each element from such vector represents a weight based on the relevance of a word in a document collection. More specifically, $Tf$ represents the Term Frequency while $Idf$ is the Inverse document frequency. The $Tf$ value estimates how often a term occurs in a document, and the $Idf$ value estimates whether the word is common or rate throughout all document collections. The weight of each vector represents the product between the $Tf$ with the $Idf$ value. We used the Python skLearn module [PVG+11] for the computation of this model.

## 5.5 Word embedding

The vector space model (seen in the Section 5.4) has the following main weaknesses as follows (i) generated high dimensional space vectors, (ii) the word order is not considered and (iii) ignores the semantics of words. For example, in these models, synonyms have very different vector representations, however, it would be more convenient that these words had similar vector representation.

The word embedding models were proposed in order to overcome the weaknesses of vector space models. Word embeddings consider the semantics of words because they are capable of capturing the context of a word in a document in relation to the other words. Another strength of these models is the dimension of vectors. We performed several experiments to determine the ideal size for the representation of documents from the dataset, review Section 6.3. In this research, we used Word2Vec for text representation.

### 5.5.1 Word2Vec

As we described in Chapter 3, Word2Vec is a word embedding model for obtaining the feature vector of words. We used the Gensim library [ŘS10] from Python for training this model for the documents of the pre-processed dataset. For the training process, the Gensim's Word2Vec model receives as input the pre-processed documents composed by lists of words. In addition to the documents, the Word2Vec technique has some input parameters. These parameters seek to configure the training of the model to optimize result. The list of the main parameters is described below:

- Dimension of the feature vectors.
- Maximum distance between the current and predicted word within a sentence.
- Frequency threshold to ignore all words lower than this value.
- Unsupervised learning technique to find the words most related to a given word (Skigram and CBOW algorithms).

We evaluated several combinations of these parameters in order to get the feature vectors that best represent the document collections. After selecting the optimal parameters for the Word2Vec algorithm, we conducted a set of steps to get the feature vectors of words.

**Algorithm 5.5.1** : Calculate **Word2Vec** of all Documents

**Require:** $D = \{d_1, d_2, \ldots, d_i\}$, where $i$ is the number of documents

1: **for all** $D$ **do**

2: $L_i = \{W_{i1}, W_{i2}, \ldots, W_{in}\}$, where, $L_i$ is the list of unique word for document $d_i$

3: $FL_i = \{(W_{i1}, F_{i1}), (W_{i2}, F_{i2}), \ldots, (W_{in}, F_{in})\}$, where $W$ is the word and $F$ is the frequency for this word in document $d_i$. $F_{i1}$ is the highest frequency.

4: $L(set)_i = \{W_{i1}, W_{i2}, W_{i3}, \ldots W_{ik}\}$, where, $L(set)_i$ is the subset of the K-highest frequency words.

5: $LV_i \leftarrow$ Word2Vec$(L(set)_i)$, where, $LV_i$ is the list of feature vectors.

6: $V_i \leftarrow average(LV_i)$, where, $V_i$ is the average of all these feature vectors.

7: **end for**=0

As we shown in Algorithm 5.5.1, the following steps were performed for each document in order to get their feature vectors: i) We first get the list of unique word from each document. ii) Then, the words are ordered according to their frequency for each document. the most frequent words were considered in the top of this list. iii) From this ordered list, we extracted a subset of the K-highest frequency words. For this research we performed different values of K. iv) We got the feature vectors of each of the previously selected words by using the Word2Vec model previously trained with all documents. v) We finally averaged all features vectors from each word to get a single vector that represents the document. vi) We also performed an additional step, where we considered the entire words from a document to get the representative vector of such document. Therefore, we averaged the feature vectors of Word2Vec from all words of the document.

## 5.6 Graph embedding

Recently, several works focused on representing text documents as graphs or networks [SKD19]. Research showed that networks are an efficient way to capture the structure and relationships between words or sentences in a document. For this reason, recent methods for modeling networks as vectors have been little studied for text classification. These novel methods are called Graph and Node Embeddings, where we can obtain a vector that characterizes a network (Graph2Vec) or several vectors representing each node from network (Node2Vec) respectively.

For this research, we first modeled the pre-processed texts as networks, and then we used graph and node embedding methodologies to get the representative vectors of each document. In Chapter 6, we discussed if such models are better than simpler models like vector space models or word embeddings. In the following sections we explained the network construction process (Section 5.6.1), the Node2Vec (Section 5.6.2) and Graph2Vec (Section 5.6.3) techniques.

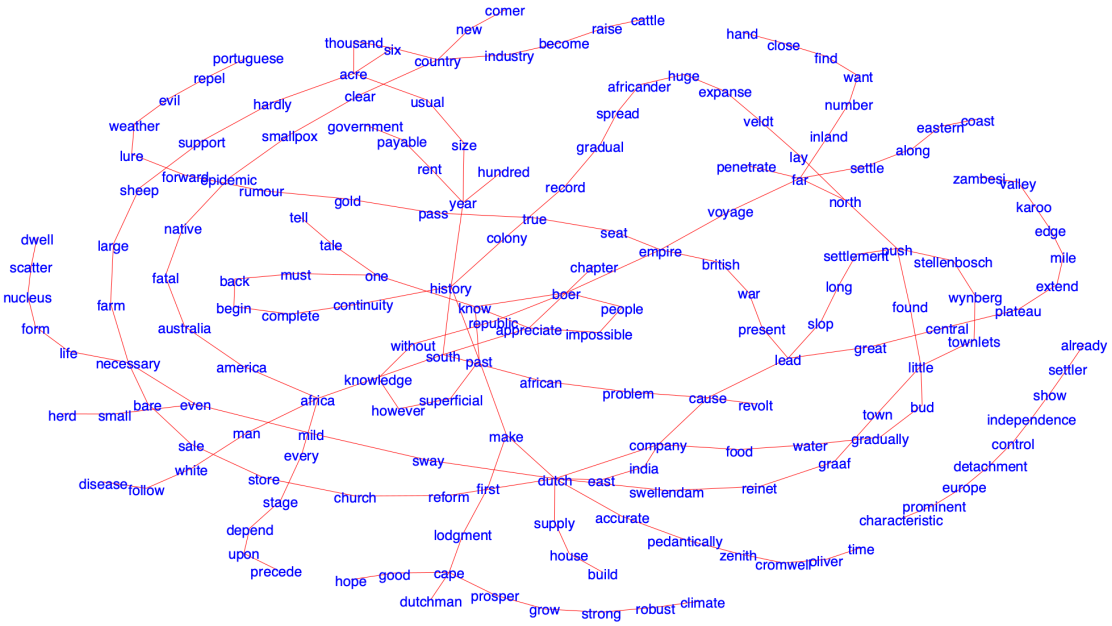
### 5.6.1 Graph construction

The first step, is the construction of the network that represents the document. Each text could be divided into sentences, and sentences are composed of words. Such words could be classified as nouns, verbs, adverbs, etc. Based on these components, we can model the texts in several network representations. We proposed two network types: undirected word co-occurrence and sentence networks. For the creation and use of networks, we used the Igraph library of Python [CN+06]. The network construction process is explained as follows:

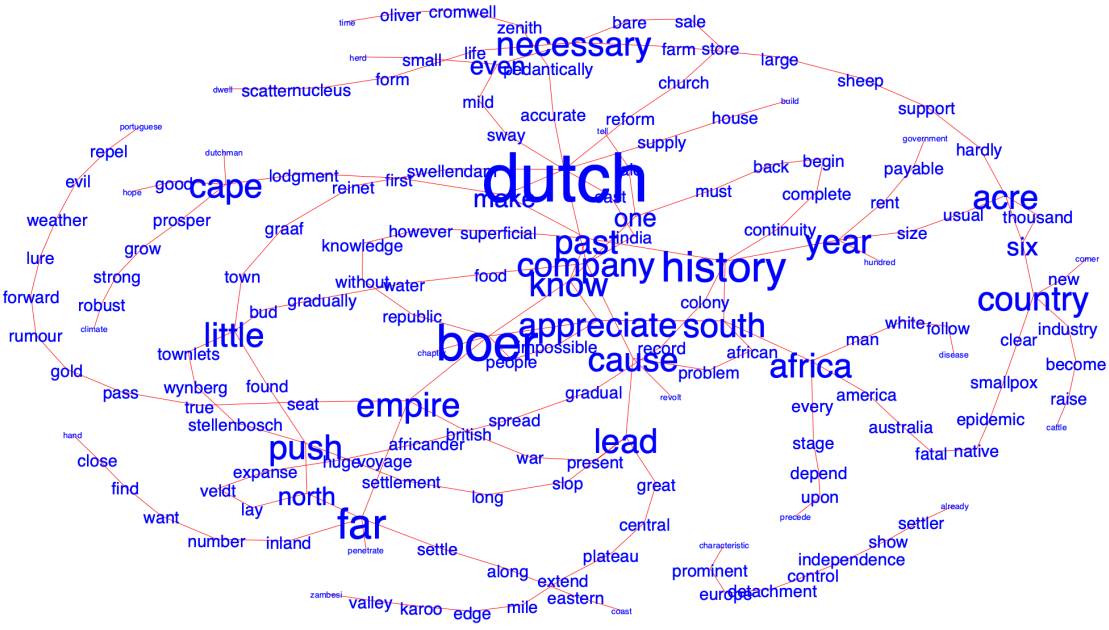
- Word co-occurrence networks: For this type of network, each node is represented by a word of the document. Considering that a word can appear more than once in a document, regardless of the number of times it appears, a single node is created for each unique word. The edges of the network could be established in the following ways:
  - The first way considers that if two words are neighbors in the text, then an edge is placed between two nodes that represent the words. This means that, in any sentence, if a word is next to another word, an edge would be established between the nodes representing such two words. For example, for the sentence *"Today we are learning machine learning concepts"*, the network is composed by 6 nodes *(today, we, are, learning, machine, concepts)* and the following 6 edges: *(today, we), (we, are), (are, learning), (learning, machine), (machine, learning), (learning, concepts)*.
  - The second way, consist on enrich word relationships using word embeddings. Here, we calculate the similarity between the vectors of each pairs of words of the network, and then we selected those pair of words with the highest similarities. We added new edges to these selected words. We used the Word2Vec model for calculation vector similarities.

  Figure 5.3 shows an example of a word co-occurrence network.

- Sentence networks: In this type of network, nodes represent each sentence of document and the edges can be established in the following ways:
  - **Common word between two sentences:** If two sentences have words or nouns in common, they increase an edge that would join the nodes representing these sentences. After performing this process on the entire document, we needed to apply a cleaning process for the graph, since it must be a simple undirect graph. A simple graph means that it accepts a single edge joining any two vertices, which is equivalent to saying that any edge is the only one that joins two specific vertices. Undirect graphs are graphs where edges have no direction. In these graphs, for an edge between the nodes $i$ and $j$, we have $(i, j) = (j, i)$. This graph creation criteria is interesting in this study, because it shows that two arbitrarily chosen sentences by having some common words represent that they are trying some related criteria and that would help to extract their feature vector.




**(a)** *Network visualization considering an equal size for each node (word).*




**(b)** *Network visualization considering the size of each node is proportional to the node degree. The most frequent words can be clearly seen.*

**Figure 5.3:** *Two examples of network visualization of a word co-occurrence network extracted from the book "The War in South Africa: Its Cause and Conduct" written by Arthur Conan Doyle. For visualization, we considered the first 15 sentences. Each node represents a word and two words are connected if they are adjacent in the text.*

– **Similarity between two sentences:** In order to construct this type of graph, the feature vectors of each sentence should first be extracted using traditional methods such as (i) TF-IDF representation, or (ii) Frequency count. Therefore, an edge arises between two sentences if the similarity between the two vectors of such sentences is greater than a predefined threshold. We also considered a simple undirect graph.

In Figure 5.4 we can see an example of a sentence network where each node represents a sentence and the edges are established based on considering the common words between sentences.

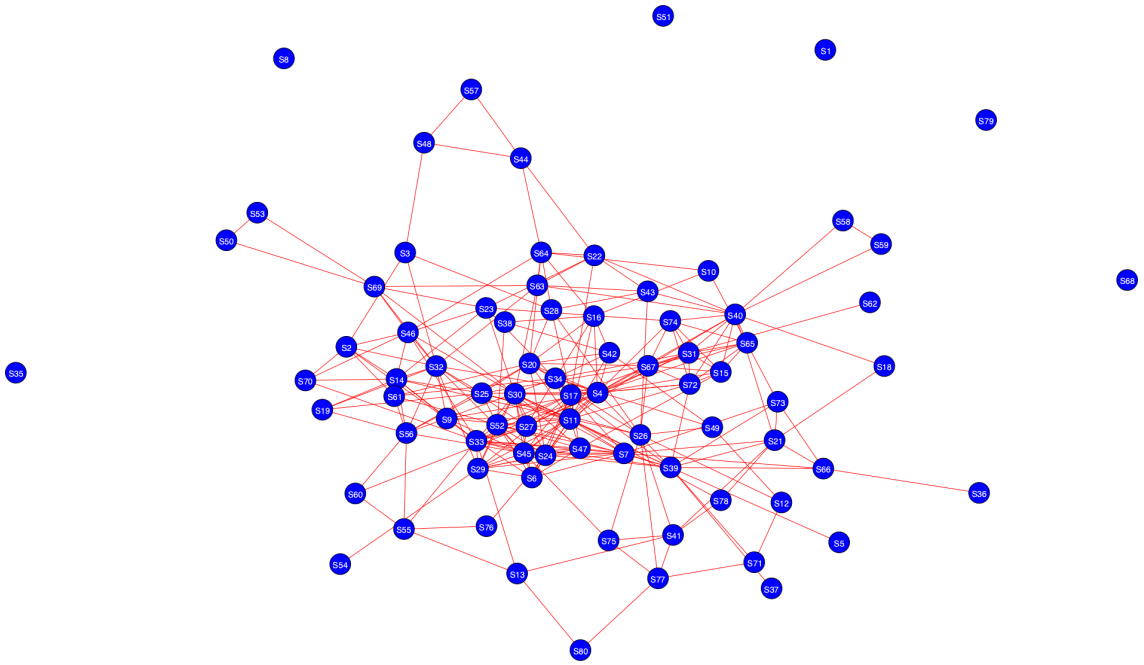


**Figure 5.4:** *Sentence network extracted from the book "The War in South Africa: Its Cause and Conduct" written by Arthur Conan Doyle. For visualization, we considered the first 80 sentences. Each node represents a sentence, while two sentences are connected if they have at least one word in common.*

### 5.6.2 Node2Vec

This method we proposed is based on node embeddings called Node2Vec. The main objective of this method is to get a feature vector for each network node. We used the library [GL16] for the implementation of this method. Similar to Graph2Vec, we need the network representation of the documents to be used as input for the Node2Vec method. As we mentioned earlier, we must create the network to use graph vector representation methods, in this case Node2Vec. For this method we only experimented with sentence networks (View Section 5.6.1).

We evaluated different parameters to get the best representation of each document. It was also important to select parameters whose computational time would not be very high. This technique requires the following parameters:

- The dimension size of the feature vector
- The number of paths to be generated for each node
- The path size, which is related to the number of nodes in each walk

The selection of the best parameters is discussed in Chapter 6. After finding the optimal parameters, we used the Node2Vec algorithm to extract the feature vector for each node from network. Then, we obtained the feature vector of each network by averaging the vectors of their composing nodes. The generated vector of each network was used for the text classification step (Section 5.7).

### 5.6.3 Graph2Vec

As we mentioned earlier, for the vector representation of documents using graph embeddings, we first must create the network based on the pre-proccessed version of each document. Then, from this network we can extract a feature vector based on graph or node embeddings. To get the vector representation of a document, we performed the following steps:

1. We transformed the documents into networks (word co-occurrence or sentence networks) as we mentioned in Section 5.6.1. We performed several tests to get the best representation of a document.

2. The Graph2Vec method requires a set of input parameters for its execution. The first parameter is the number of iterations required to execute each subgraph per node. This step is needed to enrich the set of words that are input parameters for the Doc2Vec algorithm [Che17]. The Doc2Vec model is a sentence embedding method that is used internally by Graph2Vec.

3. After enriching the graph, we considered two other input parameters of Graph2Vec: the number of dimensions of the feature vector, and the number of iterations that the Doc2Vec method must perform.

4. As we mentioned in the previous steps, we used the Doc2Vec method to train the model that will generate feature vectors based on the enriched network considering the above-mentioned parameters.

5. Finally, we get the feature vector of each document based on the vector obtained for each network. We used the obtained vector for the text classification step (Section 5.7).

In summary, Graph2Vec considers the following parameters (i) number of iterations required to execute each subgraph per node, (ii) dimensions of the feature vector and (iii) the number of iterations that executes the Doc2Vec method. In Chapter 6 we discussed the results obtained for the different experiments that we performed using these parameters.

## 5.7 Document classification

In this last step, we used the extracted features in the previous sections for the text classification task. As we mentioned before, for this research we proposed three techniques for feature extraction: i) features based on vector space models (frequency and TF-IDF model), ii) features based on word embeddings (Word2Vec) and iii) features based on graph and node embeddings (Node2Vec and Graph2Vec). These features are used as input for traditional machine learning methods. We used the following supervised classifiers [KPC95]:

- Decision Tree (DT) [Qui14].
- Gaussian Naive Bayes (Gaussian NB) [Zha04].
- K-Nearest NeighborsClassifier (kNN) [Alt92a].
- Support Vector Machine (SVM) [HHHH09]

The features extracted from the datasets of authorship recognition are used for the classification process. For the datasets we used, the class labels represent the authors of each dataset. English dataset is composed by 13 different class labels while the Spanish dataset is composed by 21 different class labels, where 13 and 21 represent the number of authors for each dataset.

For several classification tasks, the datasets are often divided into training and test datasets. While the training dataset is used to create the model, the test dataset serves for evaluating the model performance. However, the datasets we selected are not divided into training and test datasets. For this reason, we used the K-fold cross validation (View Section 6.1.1) procedure to split the original dataset into training and test datasets.

In Chapter 6 we discussed the results we obtained for document classification using the different feature extraction methods for text representation.

# Chapter 6

# Results and Discussion

In this chapter, we explained the main results obtained from the evaluation of methods for feature vector representation of English and Spanish documents. Our objective is to compare the performance of several mathematical representation's methods of texts for authorship recognition, where each document represents a literary book and each class represents an author.

The current chapter is organized as follows. In Section 6.1, we describe some initial considerations such as the cross validation method, which was used to evaluate all methods and details about the use of pruning in our dataset. In Section 6.2 are shown the results of frequency count and TF-IDF based methods. Section 6.3 describes the results of Word2Vec model. Sections 6.4 and 6.5 explain the results obtained from the graph embedding technique Node2Vec and Graph2Vec. Finally, in Section 6.6 is shown a comparative approach among all methods used in this Master's work.

## 6.1 Initial considerations

For this research we used two different datasets. The English dataset has 78 books, and the Spanish dataset has 105 books. In Appendices A we show the complete list of books that comprises these datasets. The database is not divided into training dataset and test dataset. So, for the purposes of this research, it is necessary to divide the datasets into these two sets and then we apply a cross validation method, which helps us to obtain the accuracy rate of each proposed method. The Cross Validation Method is described in Section 6.1.1.

Moreover, in Section 6.1.2. We will describe the pruning process that was conducted in our dataset. As there are some documents larger than others and to standardize the size of the documents, the experiments were performed with pruning and without pruning.

### 6.1.1 Cross Validation Method

Cross-validation is a statistical method employed to compute the performance of machine learning models [Bro00]. Cross-validation help us to test the model's ability to predict new data that was not used in the training stage, that is, with independent dataset.

Here, we use K-fold cross validation for the evaluation of each method. K-fold cross validation first partitions the data into multiple subsets $k$. Then, it proceeds to train with all groups except one. The remaining pool is used for testing. This process is repeated for each subset of the dataset. At the end, all the results of the iterations are averaged. To exemplify this process, see Figure 6.1), where five iterations are performed, that is, 5-*Folds*. After performing the iterations, all these are averaged.

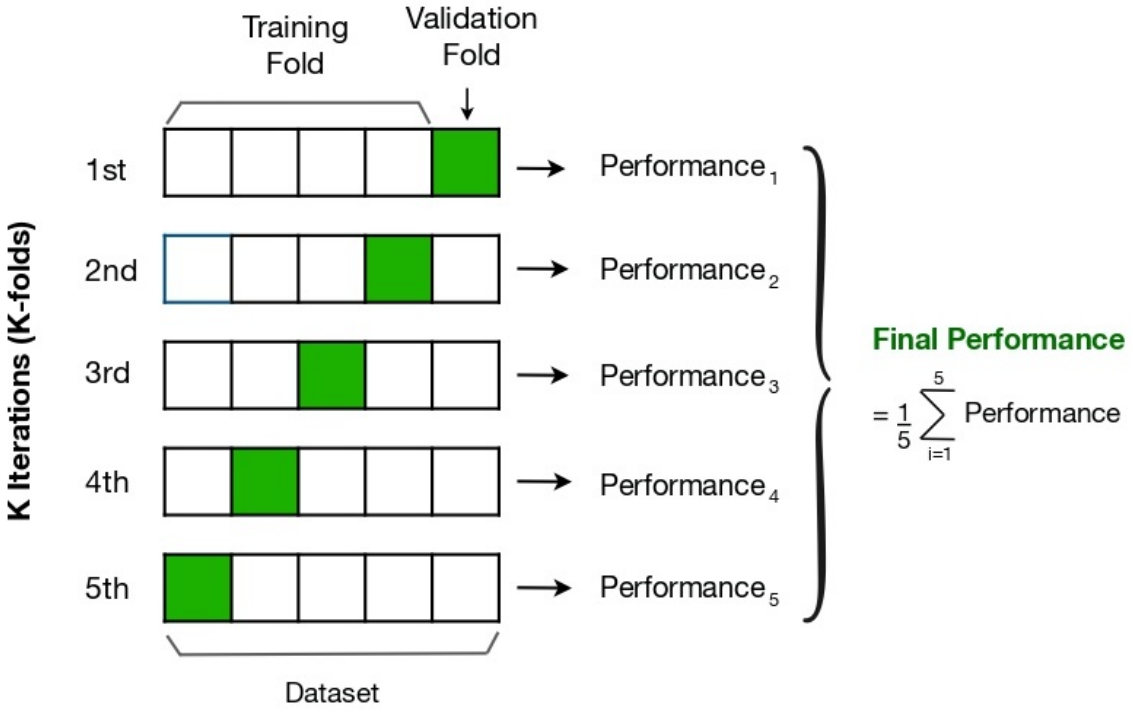



**Figure 6.1:** *K-fold cross validation.*

In each test (iteration) which is executed by the K-fold cross validation it is evaluated using the accuracy metric. This measure is outlined in the Equation 6.1. Accuracy is the most intuitive performance measure and it is a relationship between the correctly predicted observation and the total number of observations. That is, how close the result of a measurement is to the true value (true positives among the total samples). It is represented by the ratio between the real positives predicted by the algorithm and all the positive cases. Where the total of samples is the sum of true positives and false positives.

$$Accuracy = \frac{true\_positives}{true\_positives + false\_positives} \tag{6.1}$$

### 6.1.2 Pruning Stage

As we described in Section 5.2, our dataset goes through a pre-processing and one of these stages is the pruning stage. Due documents in our dataset have different sizes (the size measures by the number of word), then the pruning process was conducted. Another reason why pruning was used is because of the high dimensionality of traditional techniques (Frequency and TF-IDF Model). To perform the pruning, the document with the fewest number of words was selected, then all the documents were only allowed to keep this number of words. Last, it is important to note that to perform our experiments, the complete dataset was used and the dataset after the pruning process, that is why all the experimented methods have results with pruning and no pruning.

## 6.2 Experiments with Frequency and TF-IDF Models

In this section we displayed the results based on frequency counts and TF-IDF method for the selected databases for English and Spanish. As previously shown, the representative vectors obtained by frequency (Section 3.4.1) and TF-IDF (Section 3.4.2) models have high dimensionality. High dimensionality means that the number of dimensions are staggeringly high, since the dimension or size of vector works is based on the count of unique words. Both the Frequency and TF-IDF models consider the number of unique words (vocabulary) for the vector size (dimension), therefore, the number of dimensions is the same considering the Frequency or the TF-IDF method. In other words, as explained above (Section 6.1.2) pruning was used to try combating the high dimensionality of the feature vector.

Therefore, Table 6.1 shows the dimension's summary that we considered to create the feature vectors that represent the documents for both English and Spanish. The size of the feature vectors in English without pruning (where the entire document is considered) is 58511 for both Frequency method and TF-IDF method. Applying the pruning stage the number decreased by 30% resulting in a size of 37268, but it is still a vector with a very high dimensionality. For Spanish, the number of dimensions is 21740 with pruning and 106717 with no pruning, in this case it decreased by 80%, but despite the size of the vector, it still has a high dimensionality. As can be seen, the number of dimensions of the vector obtained from traditional vector representation techniques such as frequency and TF-IDF models is very high. This is one of the main reasons we evaluate other techniques to reduce their high dimensionality of the feature vectors of documents. By having fewer dimensions, less execution time is expected.

| Language | Pruning | No pruning |
|---|---|---|
| **English** | 37268 | 58511 |
| **Spanish** | 21740 | 106717 |

**Table 6.1:** *Dimensions of features (number of unique words) vectors for frequency count and TF-IDF method*

In Table 6.2 we show the results obtained from evaluating the frequency counting technique to generate the feature vectors that represent each dataset document. For English, the best classification method is SVM with an accuracy of 89, 23%. The worst case is obtained using the Decision Tree classifier with 46, 15%. Additionally, better results were obtained using pruning on our dataset than not using it. For Spanish, the best result is 69, 52% with the SVM classifier and the worst result is 39, 5% with Decision Tree classifier. Unlike the English books that the use of pruning before extracting the feature vector improves the results, however, for Spanish documents, the best option is to consider the complete document, that is, do not use pruning. In conclusion, in both cases (English and Spanish) the best classifier is SVM and the worst is Decision Tree. In another way, we can see despite the high dimensionality in English, acceptable results are obtained, nevertheless, for Spanish books the results are not very good using frequency count. For the previously mentioned, in the next sections we will analyze another technique usually used to extract the feature vector from documents.

| **Classifier** | **English** | | **Spanish** | |
|---|---|---|---|---|
| | **Pruning** | **No pruning** | **Pruning** | **No pruning** |
| **Decision Tree** | 49,23 | 46,15 | 39,05 | 40,00 |
| **KNN** | 58,46 | 60,77 | 39,05 | 50,48 |
| **Gaussian NB** | 50,00 | 50,00 | 54,29 | 53,33 |
| **SVM** | **89,23** | 81,54 | 65,71 | **69,52** |

**Table 6.2:** *Result (accuracy) of frequency count for Dataset in English and Spanish*

The results based on TF-IDF experiments are shown in Table 6.3 for English and Spanish database. For English, the best result was obtained using SVM, with an accuracy of 80, 00% and the worst ranked was KNN with 26, 92%. Then, analyzing the use of the complete document or the pruned document for feature vector extraction, we can see that the results are the same. So, if we use the pruned document for English texts, the results will be obtained in less time. For Spanish, the best result is 60, 95% with the SVM classifier, and the worst result is 39, 05% with Decision Tree. In the case of Spanish, it shows better results with the use of the pruned document than with the complete document (without pruning). In another way, the results with TF-IDF model of English are better than Spanish, as with the Frequency Model.

| Classifier | English | | Spanish | |
|---|---|---|---|---|
| | **Pruning** | **No pruning** | **Pruning** | **No pruning** |
| **Decision Tree** | 48,46 | 49,23 | 39,05 | 39,05 |
| **KNN** | 30,77 | 26,92 | 55,24 | 47,62 |
| **Gaussian NB** | 49,23 | 48,46 | 51,43 | 47,62 |
| **SVM** | **80,00** | **80,00** | **60,95** | 55,24 |

**Table 6.3:** *Result of TF-IDF for Dataset in English*

Comparatively, for both the Frequency method and the TF-IDF method, the classification method that gives us the best results is SVM, independent of Language. Likewise, the classifier that gives us the worst results is the Decision Tree, except for the results of the TF-IDF model with English, where the worst is KNN. So we can conclude that SVM is the best classifier for the task of authorship attribution with our dataset of literary documents. Because the results vary a lot between classifiers, at the classifier selection stage, special care must be taken with this choice because a bad choice of classifier can lead to bad results.

From the linguistic viewpoint, both frequency model and TF-IDF are methods that consider only the **lexical level** to generate the characteristic vectors. The lexical level mainly represents the vocabulary of language (see Section 2.1.3). As well, Frequency model and TF-IDF ignore the **syntactic level** (see Section 2.1.2) since the order is lost of the words, which represents their grammar. These techniques also lose the semantic information (**semantic level**) since it only considers the number of times a word appears in the document or in the collection of documents and not the meaning of these words. Since in the case that two words are linguistically similar (synonyms), they should have close or similar vectors. However, since that these models ignore semantics, words with similar meanings are not necessarily represented by similar vectors. Regarding the **morphological level** (see Section 2.1.1), which studies words and their modifications in isolation, in the pre-processing stage the lemmatization (view Section 5.2) is performed where only the root (lexeme) of the word is saved; therefore, almost all the morphological information is lost.

Both frequency counts and TF-IDF results are better in English than Spanish. One of the possible reasons is when the lemmatization is applied a large part of the morphological information is lost in both languages, because the Spanish language has more morphological information and it loses more information than English. Lemmatization is computationally important because if this process is not performed, there would be many versions of the same word, then, the computer could understand that they are different words and not the same word with variations of gender or number, among other variations. In addition, we observed in Table 6.1, the vectors would have a higher dimensionality. If the dimensionality of a vector is high, then the classification will have a high computational cost, and it could lower the performance of the classifier. For this reason, the lemmatization process is important, despite the loss of relevant information (at morphological level).

In next sections we will analyze different more modern techniques for extraction of feature vectors with Word Embedding techniques (Section 6.3) and Graph Embedding techniques (Sections 6.4 and 6.5). These techniques were used with the aim of improving the performance of vector representation methods for both Spanish and English books, and specifically improving the results for texts in Spanish.

## 6.3 Word2Vec experiments

As we discussed in the previous section, only the lexical level was considered to represent each document, which is not enough in most cases, a clear example is Spanish that loses a lot of morphological and lexical information by only using the lexical aspect. We can also conclude from the study of previous techniques, that due to their high dimensionality when classifying with these vectors, they will have a high computational cost. In order to solve the above-mentioned two problems, we experimented with word embedding techniques that consider the semantic level of words and have a fixed size vector feature.

Consequently, one of the main word embedding techniques is the Word2Vec model. As we mentioned in Section 3.5.1, Word2Vec considers several parameters that must be considered to configure execution of this technique in search of the best results. The parameters that were used are mentioned as follows:

- **Dimensions:** This parameter defines the size of the vector. We experimented with 50, 100, 150, 200, 300 and 400 dimensions.
- **Min count:** All words with a frequency lower than this parameter are not considered. It was experimented with values of 10 to 60 that restrict more words up to the value one that considers all the words.
- **Number of Word:** It is the number of words that is considered to form the feature vector of the document. Before selecting the words, they are ordered from highest to lowest frequency and the first n words are considered most frequently. In this parameter, it is experienced from 10, 20, 30, ... to all the words.
- **Algorithms:** Consider two algorithms to train CBOW (0) and Skip-gram model. CBOW uses the context of each word and tries to predict the target word and Skip-gram model reverses the use of word target to predict the context words.

We outlined in Table 6.4 the experimentation process for the evaluation of Word2Vec, with different dimensions for the feature vector. Regarding the configuration and choice of parameters *"Min count"*, and *"Number of Word"* and, the best values of parameter's variation were considered for each case. It was also considered the best result between experimenting with the Skip-Gram Algorithm and the CBOW algorithm. In addition, Table 6.4 is subdivided into two tables, where Table 6.4a shows the results for English books and Table 6.4b summarizes the obtained results for Spanish. Furthermore, the results are shown, considering the complete document and including the pass' process of the documents through the pruning stage as explained in the Section 5.2. Last, we considered four classifiers (described in Section 3.2) to be used to extract the percentage of success of our proposed feature vectors.

According to results of dataset in English presented in Table 6.4a, better results were obtained using the SVM classifier with an accuracy of 93.08%. Followed by Gaussian NB classifier together with KNN classifier with an accuracy of 83.08% and 83.07% respectively. Finally, the worst results were obtained by Decision Tree Classifier with accuracy rate of $57,70\%$. In the case of English, it shows better results with the use of the pruned document (93.08%) than with the complete document, that is, without pruning (max. 83.08%). Also for English documents, when comparing the best results of Word2Vec and vector space model (described in the previous Section 6.2), Word2Vec achieved a certainty percentage of 93.08%, and vector space model 89.23%.

From the results for dataset in Spanish (Table 6.4b), KNN classifier achieved the best accuracy rate, $69,52\%$. Followed by Gaussian NB classifier with $65,71\%$ and SVM Classifier with $63,81\%$. Finally, Decision Tree Classifier obtained the accuracy value of 59.05%. Regarding the use of pruning process in experimentation with documents in Spanish, for every case of Word2Vec was obtained excellent results using the complete document; that is, without pruning. In best case, we obtained without pruning $69,52\%$ and with pruning $45,71\%$. On the other hand, when comparing the results of Word2vec with vector space model, for Spanish do not show significant differences, because the accuracy in both cases is 69.52%.

Regarding the number of dimensions, it shows better results using 150 dimensions for both English and Spanish. By using Word2Vec models, this methodology reduced the size of the dimensions about 95% compared to traditional techniques based on vector space models. As can be seen in the results of Table 6.1 for English documents, there are 37268 dimensions considering pruning and 58511 dimensions without applying pruning; and for Spanish documents, there are 21740 dimensions considering pruning and 106717 dimensions; in contrast, with the use of 50 to 400 dimensions considered in these experiments. In this sense, the computational cost was significantly reduced considering the word embedding models.

With respect to use of pruning to reduce the computation time due to the large size of the documents. In all the experiments for documents in Spanish using Word2Vec, it is better not to use pruning, that is to say to use the complete document. However, for English, only the SVM classifier shows the best result using pruning, when comparing this result using pruning (93.08%) with the no pruning result (92.74%) it obtained a difference of 0.5% . Therefore, it is observed that in most cases for both languages it is better to use the whole document than to use the pruned document. Since the Word2Vec method is based only on the feature vector of each word. Then by reducing the size of the document we lose words, likewise the representative power of this method is decreased. Even more for Spanish, more words are needed for the characterization of Spanish documents.

**English**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 50,00 | 51,54 |
| **100** | 53,07 | 56,15 |
| **150** | 53,85 | 56,92 |
| **200** | 54,62 | **57,70** |
| **300** | 52,05 | 52,54 |
| **400** | 50,01 | 51,03 |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 58,46 | 56,15 |
| **100** | 73,85 | 77,69 |
| **150** | 75,38 | 82,31 |
| **200** | 78,46 | **83,07** |
| **300** | 77,43 | 82,05 |
| **400** | 71,01 | 80,00 |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 61,54 | 63,08 |
| **100** | 76,15 | 76,92 |
| **150** | 80,00 | 80,00 |
| **200** | 80,77 | **83,08** |
| **300** | 77,32 | 80,06 |
| **400** | 74,28 | 80,00 |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 61,54 | 60,77 |
| **100** | 90,77 | 87,69 |
| **150** | **93,08** | 91,53 |
| **200** | 93,08 | 91,54 |
| **300** | 93,08 | 92,74 |
| **400** | 92,30 | 90,00 |

**(a)** *Word2Vec for English*

**Spanish**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 37,14 | **59,05** |
| **100** | 40,00 | 58,10 |
| **150** | 42,86 | 55,24 |
| **200** | 41,90 | 53,33 |
| **300** | 42,86 | 53,33 |
| **400** | 41,90 | 55,24 |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 41,90 | 59,05 |
| **100** | 42,86 | 64,76 |
| **150** | 44,76 | **69,52** |
| **200** | 45,71 | 69,52 |
| **300** | 43,81 | 68,57 |
| **400** | 44,76 | 67,63 |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 35,24 | 60,95 |
| **100** | 36,19 | 60,00 |
| **150** | 36,19 | 60,95 |
| **200** | 33,33 | 63,81 |
| **300** | 38,10 | **65,71** |
| **400** | 37,14 | 61,90 |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 38,10 | 62,86 |
| **100** | 37,14 | 60,00 |
| **150** | 40,00 | **63,81** |
| **200** | 36,19 | 63,81 |
| **300** | 39,05 | 62,86 |
| **400** | 40,00 | 62,86 |

**(b)** *Word2Vec for Spanish*

**Table 6.4:** *Results obtained based on Word2Vec for English and Spanish literary books*

One of the main differences between vector space models and Word Embedding models is that Word Embedding considers the semantics of the words to extract the characteristic vector of each one of them. On the contrary, as we saw in Section 6.2, the vector space models consider only the lexical level to extract the characteristic vectors. In our methodology (Section 5.5), we tried to combine these two approaches (lexical y semantic), we first extracted the feature vector using Word2Vec from each word in the document (providing the semantic information to our feature vector) and then we considered averaging the vectors representing the most frequent words in the document (providing the lexical information). This is because for some methods the most frequent words can be considered as keywords. Since we have lexical and semantic information in the feature vectors generated by Word2Vec, this could be the reason that the documents in English dataset show us better results. Unlike the Spanish documents, in which the results do not improve compared to the frequency model, because for the documents in Spanish it is not enough to consider the lexical and semantic information.

In summary, according to the Table 6.4, the results for documents in English show greater accuracy than the results for Spanish dataset. These findings show us that, the semantic information provided by Word2Vec along with lexical information of the space model vector is not enough to obtain an optimal characteristic vector in Spanish and compensate for the loss of information suffered by texts in Spanish when lemmatization is performed. We mentioned in the previous sections, that when we lemmatizing Spanish, we lose morphological information that is more important for Spanish than for English. Another reason, why in Spanish the results are not very good, is due to the fact that space model vector and word embedding techniques consider only the words to represent document as isolated entities, and not as a set of interconnected and related words for different reasons.

Finally, for the reasons mentioned above, to improve the results for Spanish documents, in the next section we will analyze more sophisticated techniques based on Graph Embedding that search to give notions of the relationships and contexts of the words.

## 6.4 Node2Vec experiments

In order to improve the results, even more so for the document in Spanish, we evaluated the performance of Graph Embedding methods for the vector representation of documents. As we saw in the previous sections, the Spanish datasets lose morphological information when lemmatizing information that is relevant to its vector representation process. So, in order to try to compensate for this loss of information, the use of graph and complex networks is considered to represent the documents as vectors. Applying concepts of graphs and complex networks help us to preserve the context, order and structure of the document, that is to say, the relationships between words and sentences.

In this section, we evaluated approach of using the concept of graphs to represent a document, in this case, the characteristic vector of each node is extracted, this method is called Node2Vec. Each document is modeled to graph first, after extracting the vector of each node, finally, the characteristic vectors of all the nodes are averaged.

Before displaying and analyzing the results, Node2Vec has several parameters that must be considered for its execution. The parameters are the following: (i) the number of dimensions, for this experimentation we use 50 and 100 dimensions; (ii) the number of paths to be generated for each node, where we experiment with values 5, 10, 15 and 20; (iii) the path size (number of nodes in each walk), for this parameter we was experimented with values of 10 to 40, increasing from five to five. In search of finding the optimal vector, for the last two parameters mentioned, the best values of parameter's variation were considered.

The results obtained from the evaluation of Node2Vec are shown in Table 6.5, with 50 and 100 dimensions. In addition, this Table is subdivided into two tables, where Table 6.5a shows the results for English books and Table 6.5b the obtained results for the books in Spanish are displayed. Moreover, for Node2Vec we only show results considering the pass' process of the documents through the pruning stage, for both English and Spanish documents. The reason why the complete document was not experimented, that is, without pruning, is because of the high cost and computational time of Node2Vec. Finally, as with the other experiments, we considered four classifiers (Section 5.7).

**English**

| **Decision Tree Classifier** | |
|---|---|
| **Dimensions** | **Pruning** |
| **50** | 13,33 |
| **100** | 16,19 |
| **KNN Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 18,10 |
| **100** | 13,33 |
| **Gaussian NB Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 18,10 |
| **100** | 15,48 |
| **SVM Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 19,05 |
| **100** | 21,90 |

**(a)** *Evaluation of the node2Vec model for English documents*

**Spanish**

| **Decision Tree Classifier** | |
|---|---|
| **Dimensions** | **Pruning** |
| **50** | 15,24 |
| **100** | 12,38 |
| **KNN Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 17,14 |
| **100** | 13,33 |
| **Gaussian NB Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 19,05 |
| **100** | 13,33 |
| **SVM Classifier** | |
| **Dimensions** | **Pruning** |
| **50** | 15,24 |
| **100** | 14,29 |

**(b)** *Evaluation of the node2Vec model for Spanish documents*

**Table 6.5:** *Results obtained based on Node2Vec for English and Spanish literary books*

For the results of English texts presented in Table 6.5a, the best result is 21.90% with SVM classifier. Followed by Gaussian NB Classifier and KNN Classifier with an accuracy of 18.10% for both cases. Finally, Decision Tree Classifier with 16.19%. However, none of the classifiers get good results.

On the other hand, the results of the Spanish dataset shown in Table 6.5b, where the best classifier is Gaussian NB Classifier with an accuracy of 19.05%. Followed by KNN Classifier with 17.14%. Finally, the worst results were obtained by Decision Tree classifier together SVM classifier with an accuracy of 15.24%. As with the results for English, Node2Vec method give us bad results.

In conclusion, considering the Node2vec algorithm, the vector is based on the connections that each node has with the other nodes from the network. So, the reason why the results using Node2Vec are bad, could be that the information to generate the characteristic vector of the document is lost when all nodes are averaged. A suggestion for better results would be that only the vectors of the nodes that represent words with high frequency could be averaged.

## 6.5 Graph2Vec experiments

In search of improving the results obtained with Word2Vec and Node2Vec, that is, mainly improving the results with texts in Spanish, we evaluated another approach that uses the concept of graphs to represent a document, this method is called Graph2Vec, the characteristic vector of the entire graph is extracted. In this way, with Graph2Vec we seek to preserve the relationship between the words and the context of the document, when extracting the characteristic vector from it, in addition to the semantic and lexical information extracted when using Graph Embedding techniques.

As with Node2Vec, each document is modeled to graph first, the graphs can be modeled in different ways, because the nodes can represent words or sentences. After obtaining the graph, Graph2Vec is used to represent a graph in compressed form, by using a characteristic vector. Therefore this proposal seeks to obtain better results to compensate the loss at the morphological level of Spanish documents. Likewise, this method guarantee to consider the semantic and lexical information of each document. Regarding the experiments, we first experimented with generating sentence networks from the documents to use Graph2Vec. Another group of experimentation was performed using word networks, where Graph2Vec was later used.

### 6.5.1 Graph2Vec Results with Sentence Networks

In this section we experimented with sentence networks generated from the documents to use Graph2Vec and to extract the characteristic vector. As we can see in Section 5.6.1, when representing a document as a graph, nodes can represent words, as well as nodes can be sentences.

**English**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 16,67 | 16,61 |
| **100** | **20,51** | 20,48 |
| **200** | 19,23 | 19,16 |
| **300** | 19,23 | 19,13 |
| **350** | 19,23 | 19,17 |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 12,82 | 12,76 |
| **100** | 15,38 | 15,29 |
| **200** | 19,23 | 19,15 |
| **300** | **20,51** | 20,39 |
| **350** | 12,82 | 12,79 |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 23,08 | 22,91 |
| **100** | 24,36 | 23,99 |
| **200** | **26,92** | 26,75 |
| **300** | 24,36 | 24,11 |
| **350** | 20,51 | 20,29 |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 19,23 | 19,14 |
| **100** | 17,95 | 17,76 |
| **200** | 21,79 | 21,60 |
| **300** | **23,08** | 22,98 |
| **350** | 15,38 | 15,21 |

**(a)** *Graph2Vec for English*

**Spanish**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 17,14 | 18,70 |
| **100** | 15,24 | 18,10 |
| **200** | 12,38 | 13,33 |
| **300** | 16,19 | **19,50** |
| **350** | 16,19 | 17,90 |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 16,19 | 17,60 |
| **100** | 16,19 | **23,81** |
| **200** | 18,10 | 19,05 |
| **300** | 17,14 | 20,60 |
| **350** | 18,10 | 20.01 |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 17,14 | 18,9 |
| **100** | 18,10 | 20,00 |
| **200** | 14,29 | 16,19 |
| **300** | 19,05 | **22,11** |
| **350** | 18,10 | 19,80 |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 21,90 | 24,20 |
| **100** | 20,95 | 22,86 |
| **200** | 20,00 | **25,71** |
| **300** | 19,05 | 22,10 |
| **350** | 20,00 | 22,90 |

**(b)** *Graph2Vec for Spanish*

**Table 6.6:** *Results based on Graph2Vec using sentence networks for English and Spanish books.*

In the Table 6.6 shows the results of Graph2Vec using sentence networks with different dimensions. The results for the English documents can be seen in Table 6.6a. Moreover, in Table 6.6b, the results of Graph2Vec with sentence networks for Spanish books are displayed.

For English (Table 6.6a) best accuracy result is 26.92%, with the Gaussian NB classifier and considering 200 dimensions for the feature vector. Followed by SVM Classifier with an accuracy of 23.08% and 300 dimensions. Finally, note that KNN and Decision Tree Classifier gives us the worst results with an accuracy of 20.51% and likewise 300 and 100 dimensions respectively. Regarding the use of the pruning process, the results with pruning give better results than the results without pruning for English books.

Regarding the results of Table 6.6b, where sentence networks were used to generate the graph and then apply Graph2Vec with books in Spanish. In this case, a better accuracy is 25.71% was obtained using 200 dimensions, being the best classifier for this case SVM. Then, there is the KNN Classifier with an accuracy for English of 23.81% with 100 dimensions. Followed by an accuracy of 22.11% with 300 dimensions for the Gaussian NB Classifier. Lastly, the worst classifier is Decision Tree with an accuracy of 19.50% for 300 dimensions. It is also important to note that the results without pruning give better results than the results with pruning.

In conclusion, for both English and Spanish texts the results have declined compared to vector space model and Word2Vec. The reason why these bad results are shown may be that when constructing the graphs based on the documents, the sentence was selected as the minimum unit, that is, each node represents a sentence. Then, these networks of sentences, although they preserve the relationship between sentences, lose the relationship between words, which seems to be very important when representing documents as feature vectors. Consequently, lexical information is lost, because the contribution or count of each word is lost. In addition, semantic information is lost, because the meaning of each word is not saved, considering the sentence as a minimum unit. Therefore, the concept of context is also lost by not being able to save the relationship between the words. For all the aforementioned, it searches to experiment for better results, with another form of network construction to achieve a more adequate representation of the document. In next section we will show the results using another form of network construction.

### 6.5.2 Graph2Vec Results with Word Networks

In the process of representing documents as graphs, just as the nodes of the graph can represent sentences, so the nodes of the graph could be words. Then, each document would first be like a network of words, and then the Graph2Vec method is used to extract the characteristic vector of each graph, that is, of each document. Therefore, in this section we show the results using Graph2Vec with word networks, to obtain better results. In the Table 6.7 are displayed the result of Graph2Vec with word networks for documents in English and Spanish. This table is divided in two. First, Table 6.7a which shows the results for documents in English. Lastly, in Table 6.7b, which concentrates the results of the Spanish dataset.

In Table 6.7a, we can see the Graph2Vec results with word networks for English books. The best accuracy result is 93.59% with 350 dimensions which was obtained with SVM Classifier. Followed by KNN classifier which result is 80.77% with 350 dimensions. Then, Gaussian NB classifier is 74.36% with 150 dimensions. Finally, the worst classifier is Decision Tree Classifier with 50 dimensions and an accuracy of 62.82%. Furthermore, regarding the use of pruning, as with Word2Vec, in most experiments it is better not to use pruning, except for results with SVM classifier that shows the best result using pruning because when comparing the result from using pruning, because when comparing the result of using pruning 93.59% with the no pruning result 91.03% it obtained a difference of 3%. On the other hand, we obtained 93.59% which is the best accuracy result in English texts, we can infer that the vector representation of English documents has maintained favorable results in most techniques. Since, with Word2Vec an accuracy of $93,08\%$ was obtained and with Frequency model $89,23\%$. Thus, showing that for English, these techniques provide enough lexical and semantic information when obtaining the feature vectors. However, using Graph2Vec with word networks the results improve slightly, improving 0.6% in relation to the best result obtained previously (Word2Vec).

In another way, for Spanish documents we have Table 6.7b, which results have varied positively with respect to other techniques. The best accuracy result of $81,90\%$, was obtained by SVM classifier with 350 dimensions. Followed by KNN Classifier with $64,76\%$ for 50 dimensions. Next, we obtained the same result of $60,95\%$ for Gaussian NB Classifier with 150 and 300 dimensions. Finally, the worst classifier for this experimental group is Decision Tree Classifier with a value of $52,38\%$ for 300 dimensions. Regarding to the use of pruning, in all the experiments for Spanish documents using Graph2Vec with word networks, it is better not to use pruning; that is to say, to use the complete document. The reason why better results are shown without pruning, it may be that when losing a part of the document, words and relationships (relationships between words) are lost, which is relevant for building the word network and provides essential information to extract the feature vector.

In contrast to the English results, the best accuracy results of Spanish texts are shown a significant improvement when we compared $81,90\%$ versus $69,52\%$. The first value was obtained by Graph2Vec with word networks and the second value was obtained by Vector space model improving 17.80%. Then, Graph2Vec with word networks shows us better results for both languages, but much more for Spanish compared to other techniques.

About the number of dimensions, Graph2Vec and Word2Vec reduce the size of dimensions compared to traditional techniques like vector space model. Where the best accuracy value for English is obtained with 350 dimensions, unlike 37268 dimensions for vector space model. Then, in the case of Spanish experiments, the best result is with 100 dimensions, but for vector space model it is with 21740. Therefore, Graph2Vec in addition to improving the accuracy for both English and Spanish, the computational cost is significantly reduced when processing vectors with fewer dimensions. On the other hand, regarding the classifiers, for both English and Spanish results, SVM gives us the best results, unlike Decision Tree Classifier which shows the lowest results.

**English**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 35,90 | **62,82** |
| **100** | 37,18 | 53,85 |
| **150** | 39,74 | 50,00 |
| **200** | 55,13 | 55,13 |
| **300** | 41,03 | 50,00 |
| **350** | 43,59 | 50,00 |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 66,67 | 79,49 |
| **100** | 64,10 | 79,49 |
| **150** | 71,79 | 80,77 |
| **200** | 78,21 | 78,21 |
| **300** | 74,36 | 78,21 |
| **350** | 78,21 | **80,77** |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 53,85 | 73,08 |
| **100** | 62,82 | 64,10 |
| **150** | 56,41 | **74,36** |
| **200** | 73,08 | 70,51 |
| **300** | 65,38 | 69,23 |
| **350** | 69,23 | 71,79 |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 71,79 | 87,18 |
| **100** | 84,62 | 89,74 |
| **150** | 89,74 | 85,90 |
| **200** | 91,03 | 88,46 |
| **300** | 92,31 | 91,03 |
| **350** | **93.59** | 89,74 |

**(a)** *Graph2Vec for English*

**Spanish**

| Decision Tree Classifier | | |
|---|---|---|
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 23,81 | 44,76 |
| **100** | 20,95 | 47,62 |
| **150** | 20,00 | 47,62 |
| **200** | 24,76 | 46,67 |
| **250** | 27,62 | 42,86 |
| **300** | 27,62 | **52,38** |
| **KNN Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 40,00 | **64,76** |
| **100** | 38,10 | 60,95 |
| **150** | 39,05 | 62,86 |
| **200** | 37,14 | 61,90 |
| **250** | 36,19 | 59,05 |
| **300** | 40,95 | 60,95 |
| **Gaussian NB Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 31,43 | 53,33 |
| **100** | 29,52 | 60,00 |
| **150** | 28,57 | **60,95** |
| **200** | 23,81 | 53,33 |
| **250** | 24,76 | 58,10 |
| **300** | 35,24 | **60,95** |
| **SVM Classifier** | | |
| **Dimensions** | **Pruning** | **No pruning** |
| **50** | 50,48 | 77,14 |
| **100** | 53,33 | **81,90** |
| **150** | 60,00 | 80,95 |
| **200** | 54,29 | 80,00 |
| **250** | 52,38 | 79,05 |
| **300** | 54,29 | 76,19 |

**(b)** *Graph2Vec for Spanish*

**Table 6.7:** *Results based on Graph2Vec with word networks for English and Spanish books.*

In Summary, using Graph2Vec with word networks, an improvement is shown. In case of English, the results improve slightly (0.6%). But in Spanish the results improve much more (17.80%). That is, where the question arises. Why does Graph2Vec with word networks better represent documents for Spanish?. Firstly, for Spanish documents, the lexical and semantic information is not enough to represent this language using vector representation methods. Since Spanish, with the lemmatizing process, it loses a lot of morphological information; so it is necessary to add information about the context, to improve the results. Secondly, it was found that one way to compensate for this loss of information by considering the context. In this sense, the graph based methods help us in the process of preserving this information (context, order and structure). Consequently, reinforce our supposition, using Graph2Vec with word networks obtained good results for both English and Spanish. Therefore, this method obtained more homogeneous results, and it can be considered the best option to be used when using different languages.

Finally, to close section Graph2Vec experiments, we concluded that the experiment with Graph2Vec (using word networks), achieved good performance for both languages. However, The results with Graph2Vec (using sentence networks) were not good. The reason for these bad results is due to the fact that the relevance of each word within the document is lost, because sentence networks consider the sentence and not the word as a minimum unit.

## 6.6 Final Results

For this research, we experimented different methods, whose results are displayed and analyzed in Sections 6.2, 6.3, 6.4 and 6.5. Firstly, we experimented with traditional techniques based on vector space model (Frequency and TF-IDF model), next, with techniques based on Word Embedding and finally, with Graph Embedding techniques. As a result, in this Section summarizes and analyzes the best results of all the proposed techniques.

In Table 6.8 we show the best results we obtained from all the proposed methods for this Master's work, for both the English and Spanish dataset. According to this table, the Graph2Vec method using word networks outperformed the other methods for both English (con **93.59%**) and Spanish (con **81.90%**) documents. First, We observed a considerable improvement Graph2Vec with word network (con **81.90%**) achieved for Spanish books compared to the traditional vector space models and Word embedding (**69.51%**). Second, for English documents, all methods except Node2Vec and Graph2Vec with sentence network had obtained an excellent performance. Additionally, Word2Vec also shows good results for English (**93.08%**), but for Spanish it doesn't give us good results (**69.52%**).

On the other hand, in Table 6.8 we also observe that the worst results are the methods Node2Vec and Graph2Vec with sentence networks. For English, Node2Vec give us better accuracy of (**21.90%**) and Graph2Vec with sentence network of (**26.92%**). For results in Spanish, Node2Vec shows (**19.05%**) and Graph2Vec with sentence network (**25.71%**). It is concluded that Node2Vec is the worst method for both languages.

| Vcctor Document Representation Method | English | Spanish |
|---|---|---|
| **Frequency Model** | 89,23 | 69,52 |
| **TF-IDF** | 80,00 | 60,95 |
| **Word2Vec** | **93,08** | 69,52 |
| **Node2Vec** | 21,90 | 19,05 |
| **Graph2Vec (sentence network)** | 26,92 | 25,71 |
| **Graph2Vec (word network)** | **93.59** | **81,90** |

**Table 6.8:** *Summary of the best results obtained for each proposed vector representation method.*

Regarding the high dimensionality problem of traditional techniques based on vector space model, we have the following statements. First of all, although the results with VSM were good for English (**89.23%**) and fair for Spanish (**69.52%**), the main problem with these techniques is their high dimensionality due to the fact that comparing the size of the vectors generated with the vector space model versus the vectors generated with Word and Graph Embedding, with these last techniques the size of the vector is reduced by up to 98%. Therefore, by reducing the dimensionality of the vector, the problem of high dimensionality is eliminated and the computational time is reduced when processing the feature vectors; moreover, resources are optimized using Embedding methods. Lastly, Embedding methods, despite experimenting with language differences, their number of dimensions is maintained, unlike with other techniques. To understand better all this analysis, look at Figure 6.2.

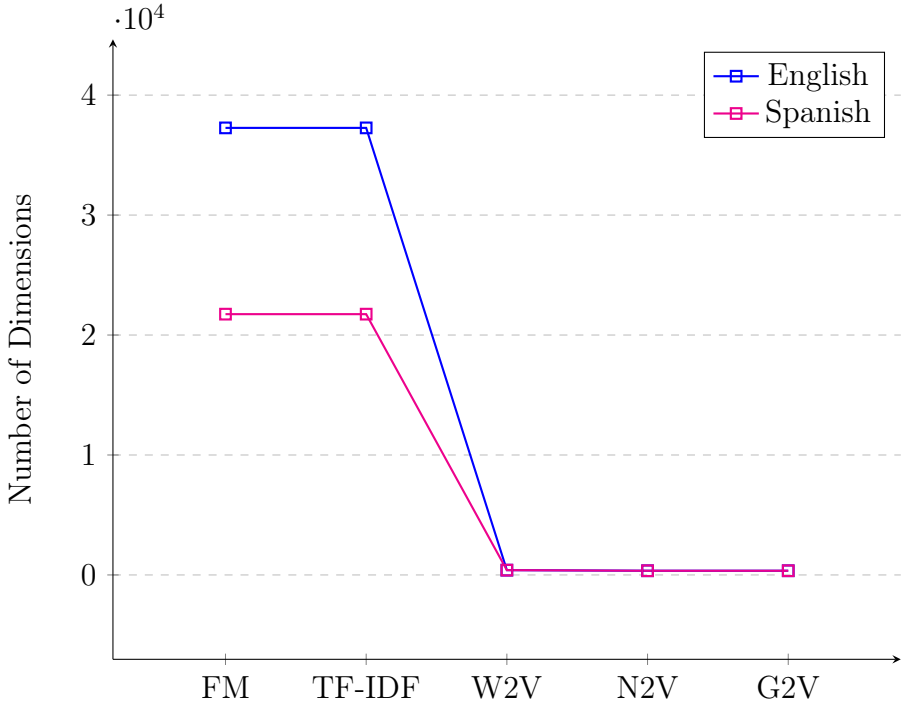
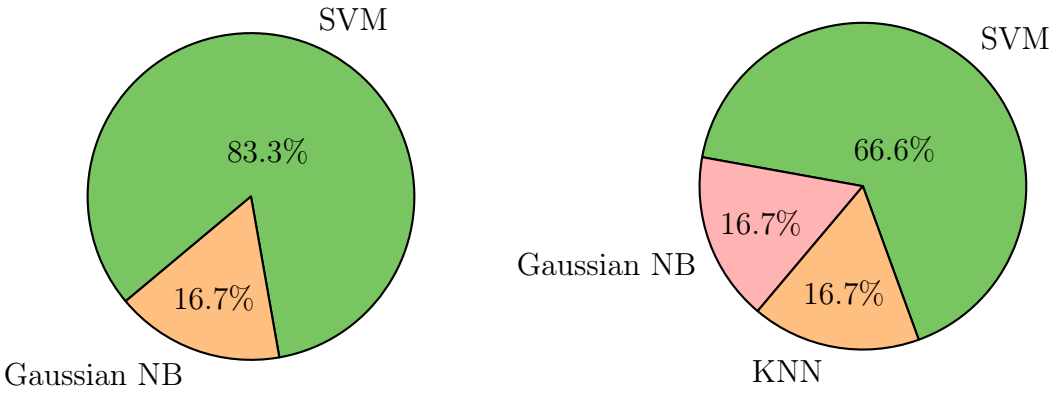



**Figure 6.2:** *Number of dimensions for each vector document representation methods*

On the other hand, in relation to the classifiers, in Figure 6.2 we also observed that in most cases SVM Classifier gives us the best results for English documents(83.3% of the times used), despite the different vector representation approaches. Unlike Decision Tree Classifier that in most cases show the worst results(0% of the times used). In addition, for Spanish SVM classifiers in most cases give us the best results (66.6% of the times used), unlike Decision Tree Classifier which shows mostly the lowest results (0% of the times used). Finally, in the scope of this work, with each of the experimented methods, in no case does the SVM classifier give the worst results, and likewise in no case does the Decision Tree Classifier give the best results.




(a) Ranking for English (b) Ranking for Spanish

**Figure 6.2:** *Ranking of the classification algorithms with the vector document representation method*

In conclusion, as we saw previously, in order to obtain good results with English, the lexical level must be minimally considered (Frequency and TF-IDF Model), it already offers significant results, but if lexical and semantic level (Word2Vec and Graph2Vec) is considered, it shows us excellent results. On the other hand, to show good results in Spanish it is necessary to consider the lexical level, semantic level and consider the context of the document. First, lexical information and use of context are considered when modeling the document as a graph. Second, when using Graph2Vec method, the semantic information is considered.

# Chapter 7

# Conclusions

Due to the great growth of textual information in digital media, the need to use various automatic methods to understand and organize this information arises. For this reason, there are countless investigations for manipulating large amounts of textual information. Under this context, the main objective of these investigations is that a computer system could capture the greatest amount of information and properties present in a text or sets of texts. The most commonly used method is to extract the numerical representation of the texts in the form of vectors of $n$ dimensions.

There are simple methods such as vector space models (frequency and TF-IDF based), to the more advanced methods such as word and sentence embeddings, which attempted to capture the semantics and context of the words.Other methods focused their analysis on the structure and organization of words in a text using concepts related to graphs and complex networks. In this master's thesis we conducted a comparative analysis of the strengths and weaknesses of each of these approaches and we evaluated their performance for the authorship attribution task. Similarly, we analyzed the performance of each method in text written in different languages (English and Spanish).

We evaluated the methods based on graph embeddings to obtain the feature vectors of each text (previously modeled as graphs of words or sentences). These methods, which emerged recently, have been successfully used for applications related to compression, classification and visualization of graphs. However, there are few studies that have used graph embeddings for text-related applications. These methods exceeded our expectations because they achieved an excellent performance for both English and Spanish texts.

The obtained results showed that most of the proposed methods ranged from good to excellent results when they were evaluated for English literary texts. However, the performance of most of these algorithms decreased significantly for Spanish texts. As already studied in the previous chapters, the Spanish language is a more complex language than the English language. For this reason we believe that the performance of the proposed methods was worse for texts in Spanish. It is also important to remember that all these methods were originally proposed for the English language, however, there are very few studies that have examined the Spanish language properly. Most of these investigations adapt methodologies that were successfully used for the English language.

In the following sections, we detailed the contributions, limitations, and future work related to this master's work.

## 7.1 Contributions

Below we briefly explain the main contributions of this master's work:

- Methods based on graph embeddings are being used for various graph-related applications. However, they have not yet been used for text-related applications. Therefore, this master's thesis is one of the first investigations that used the graph embedding approach to obtain the representative vectors of a set of documents. We have obtained excellent results using these algorithms, in this sense, we think that we can use these methodologies for various tasks related to NLP. Such representations can be used for classification, clustering and visualization of texts.

- According to the research we perform for authorship recognition, there are various methods to extract the attributes representing the writing style of an author. Most of these attributes consider both the morphological, lexical and syntactic aspects present in a text. However, semantic attributes have not yet been studied in depth, due to their complexity and possible high computational cost. The methods proposed in this thesis successfully addressed considerations related to the semantic level. The methods based on word embeddings and graph embeddings partially addressed these semantic considerations of the language.

- The proposed methods are not only useful for authorship recognition, but they could be efficient to study literary movements or analyze books according to the time in which they were written. It is likely that writers of the same era have similar writing patterns or that the topics of their books have been influenced by the events that happened at the time in which the authors lived. Therefore, we think that the writing styles for each literary movement or era could be recognized by the methods proposed in this master's work.

- To have a more complete analysis, we compared the proposed methods for different languages. Most of the authorship recognition works evaluated only texts for the English language. Likewise, we did not find research that investigated authorship recognition for texts in Spanish. Therefore, this master's thesis could initiate future research related to authorship recognition and other topics related to NLP for Spanish texts.

## 7.2 Limitations

Below, we briefly explain the limitations we found for this master's work:

- Existence of specialized corpus for other languages different from English. There are many methods and corporas for various applications for English. However, for other languages, such as Spanish, it becomes a difficult task to find a representative corpus for the applications, we intend to evaluate.

- High computational cost of some algorithms. For word and graph embeddings based methods, finding the ideal size of each vector is challenging. Vectors with high dimensions could store more information, however they would have a high computational cost for their computation. On the other hand, if we consider vectors with small dimensions, it is likely that we lose important information considering a low computational time.

- Another algorithm with high computational cost was Node2Vec, which took a long time to obtain the representative vectors of each node in the network. It also obtained the worst results.

- Although there is much research related to document representation, it is still very difficult for a computer system to understand all the characteristics of each language. All the proposed methods and those that we find in the literature are approximations at different levels of each aspect of the language. These aspects include the morphological, lexical and syntactic part (several works have approached it successfully), to deeper aspects such as the semantic and pragmatic level (which are a great challenge to model due to their high complexity)

## 7.3 Future works

Below we list the future works for this master's work:

- We propose to combine the methods of this research with author attributes that were used for the stylometry research area. We think that the combination of both approaches will improve for both English and Spanish texts.

- To evaluate the proposed methods not only for other authorship datasets, but also for text classification datasets. For example, it would be relevant to analyze texts grouped by literary genre (fiction, romance, or comedy) or by period (literary movement). We can also evaluate with other types of texts such as academic texts. For this work we only compare one dataset for each selected language.

- To extend the proposed methodology to various languages, such as Portuguese, French, German, among others. For example, since the Portuguese and Spanish are languages with similar characteristics (Romance or Latin languages), we think that our methods will possibly have similar performance for these two languages. It would also be interesting to work with languages that have different letters such as Japanese, Arab, Tibetan, among others.

- There are too many works related to word and sentence embeddings. Because the language is complicated to model computationally, various approaches have been proposed to try to capture all the peculiarities that each language has. For this reason, it would be interesting to thoroughly analyze each of these methods and use them together. In this way, we think that the representation of a text would improve significantly.

- Concepts related to complex networks have also been used successfully for various NLP applications. It would be important to deepen these concepts and use them for the authorship task. For example, the topological and metric properties of complex networks have been used successfully to find writing patterns of authors. Measures such as node degree, pageRank, betweenness, among other more advanced measures could capture stylistic patterns of each author.

- For future works, it would be very interesting to use, analyze and visualize a confusion matrix for all the methods used in both languages. This tool would allow us to determine where the system is confusing the classes, visualize the performance of the different vector document representation methods and work separately with different types of errors.

- It would be important to make a comparison of the computational times of each proposed method and the most relevant methods found in the literature. Using parallelization and GPU methods, we could speed up the execution time of each method.

# Appendix A

# Datasets for authorship recognition

As mentioned in Section 5.1, we downloaded our dataset from Project Gutenberg [Har71]. The selected books are associated with their author, who for the purpose of our thesis is very relevant. The documents are literary productions. To make up our dataset, we considered two language families: Romance languages (Spanish) and Germanic languages (English). In Sections A.1 and A.2, we describe the list of authors with the list of books selected for English and Spanish, respectively.

## A.1 English dataset

Below is the list of 13 authors selected for the English dataset:

- Allan Poe
- Arthur Conan Doyle
- Bram Stoker
- Charles Darwin
- Charles Dickens
- Daniel Defoe
- George Eliot
- Jane Austen
- Joseph Conrad
- Hector Hugh
- Mark Twain
- P.G. Wodehouse
- Thomas Hardy

On the Tables A.1 , A.2 and A.3 we can see the list of books in English that are part of our dataset. Its author and year of publication are also shown. Six books were chosen for each author. In total, our dataset in English contains 78 books.

| **Author** | **Book** | **Publication Date** |
|---|---|---|
| Allan Poe | The Narrative of Arthur Gordon Pym | 1838 |
| | The Works of Edgar Allan Poe - V1 | 1850 |
| | The Works of Edgar Allan Poe - V2 | 1859 |
| | The Works of Edgar Allan Poe - V3 | 1859 |
| | The Works of Edgar Allan Poe - V4 | 1859 |
| | The Works of Edgar Allan Poe - V5 | 1859 |
| Arthur Conan Doyle | Micah Clarke | 1889 |
| | The Adventures of Sherlock Holmes | 1892 |
| | The Refugees | 1893 |
| | The Exploits of Brigadier Gerard | 1896 |
| | The Lost World | 1912 |
| | The Valley of Fear | 1915 |
| Bram Stoker | The Mystery of the Sea | 1902 |
| | The Jewel of Seven Stars | 1903 |
| | The Man | 1905 |
| | The Lady of the Shroud | 1909 |
| | The Lair of the White Worm | 1911 |
| | Dracula's Guest | 1914 |
| Charles Darwin | The Structure and Distribution of Coral Reefs | 1842 |
| | Geological Observations on the Volcanic Islands | 1844 |
| | Geological Observations on South America | 1846 |
| | On the origin of species | 1859 |
| | The Expression of the Emotions in Man and Animals | 1872 |
| | The Different Forms of Flowers on Plants of the same Species | 1877 |

**Table A.1:** *DataSet English (Part 1)*

| Author | Book | Publication Date |
|---|---|---|
| Charles Dickens | The Pickwick Papers | 1836 |
| | Oliver Twist | 1837 |
| | Barnaby Rudge: A Tale of the riots of eighty | 1841 |
| | David Copperfield | 1849 |
| | A Tale of Two Cities | 1859 |
| | The Mystery of Edwin Drood | 1870 |
| Daniel Defoe | Captain Singleton | 1719 |
| | Memoirs of a Cavalier | 1720 |
| | The Life, Adventures of Robinson Crusoe | 1720 |
| | Colonel Jacque | 1722 |
| | The Fortune and Misfortune of Mall Flanders | 1722 |
| | Roxana | 1724 |
| George Eliot | Adam Bede | 1859 |
| | The Mill on the Floss | 1860 |
| | Romola | 1862 |
| | Felix Holt, the Radical | 1866 |
| | Middlemarch | 1871 |
| | Daniel Deronda | 1876 |
| Hector Hugh | The Rise of the Russian Empire | 1900 |
| | The Chronicles of Clovis | 1912 |
| | The Unbearable Bassington ‘ | 1912 |
| | When William Came | 1913 |
| | Beasts and Super-Beasts | 1914 |
| | The Toys of Peace and other papers | 1919 |
| Jane Austen | Sense and Sensibility | 1811 |
| | Pride and Prejudice | 1813 |
| | Mansfield Park | 1814 |
| | Emma | 1816 |
| | North anger Abbey | 1817 |
| | Persuasion | 1818 |

**Table A.2:** *DataSet English (Part 2)*

| **Author** | **Book** | **Publication Date** |
|---|---|---|
| Joseph Conrad | An Outcast of the Islands | 1896 |
| | Lord Jim | 1900 |
| | Nostromo: A Tale of the Seaboard | 1904 |
| | Under WesternEyes | 1911 |
| | Chance: A Talein Two Parts | 1913 |
| | Victory: An Island Tale | 1915 |
| Mark Twain | Innocents Abroad | 1869 |
| | The Adventures of Tom Sawyer | 1876 |
| | The Prince and the Pauper | 1881 |
| | Life on the Mississippi | 1883 |
| | The Adventures of Huckleberry Finn | 1884 |
| | The Equator a Journey Around the World | 1897 |
| P.G. Wodehouse | Tales of St. Austin's | 1903 |
| | The Man with Two Left Feet | 1917 |
| | My Man Jeeves | 1919 |
| | The Adventures of Sally | 1921 |
| | The Clicking of Cuthbert | 1922 |
| | Right Ho. Jeeves | 1934 |
| Thomas Hardy | A Pair of Blue Eyes | 1873 |
| | Far from the Madding Crowd | 1874 |
| | The Hand of Ethelberta | 1876 |
| | The Return of the Native | 1878 |
| | Jude the Obscure | 1895 |
| | A Changed Man and other Tales | 1913 |

**Table A.3:** *DataSet English (Part 3)*

## A.2 Spanish dataset

The following list shows the authors (21 authores) selected for the spanish dataset:

- Adolf Friedrich von Schack
- Armando Palacio Valdés
- Bartolomé de las Casas
- Benito Pérez Galdós
- Concha Espina
- Eduardo Zamacois
- Emilia Pardo Bazán
- Jacinto Benavente
- Jacinto Octavio Picón
- Jaime Balmes
- José Maria de Pereda
- José Rizal
- Juan Valera
- Manuel Fernández y González
- Mariano José de Larra
- Marie Lebert
- Miguel De Unamuno
- Pío Baroja
- Ramon del Valle-Inclan
- Rubén Darío
- Vicente Blasco Ibañez

For the Dataset of Spanish five books were chosen for each author. In total, our dataset in Spanish contains 105 books. On the Tables A.4, A.5 and A.6 we can see the list of books in Spanish that are part of our dataset.

| Author | Book | Publication Date |
|---|---|---|
| Adolf F. von Schack | H. de la literatura y del arte dramático V1 | 1885 |
| | H. de la literatura y del arte dramático V2 | 1886 |
| | H. de la literatura y del arte dramático V3 | 1887 |
| | H. de la literatura y del arte dramático V4 | 1887 |
| | H. de la literatura y del arte dramático V5 | 1888 |
| Armando Palacio V. | La Espuma | 2004 |
| | Riverita | 2009 |
| | Aguas Fuertes | 2010 |
| | Marta y Maria | 2010 |
| | La Guerra Injusta | 2013 |
| Bartolome de las Casas | Historia de las Indias - V1 | 1875 |
| | Historia de las Indias - V2 | 1875 |
| | Historia de las Indias - V3 | 1875 |
| | Historia de las Indias - V4 | 1876 |
| | Historia de las Indias - V5 | 1876 |
| Benito Perez Galdos | La Fontana de Oro | 2004 |
| | La desheredada | 2008 |
| | Gloria (Parte 1) | 2015 |
| | La Familia de León Roch | 2015 |
| | El Amigo Manso | 2017 |
| Concha Espina | Dulce Nombre | 2013 |
| | Agua de Nieve | 2014 |
| | Ruecas de Marfil | 2015 |
| | Despertar para Morir | 2015 |
| | La Esfinge Maragata | 2016 |
| Eduardo Zamacois | La enferma | 2015 |
| | Incesto | 2015 |
| | La cita | 2015 |
| | De carne y hueso | 2016 |
| | El misterio de un hombre pequeñito | 2016 |
| Emilia Pardo Bazan | La Quimera | 2015 |
| | La Sirena Negra | 2016 |
| | Insolación y Morriña | 2016 |
| | Cuentos de Navidad y Reyes | 2017 |
| | Dulce Dueño | 2017 |

**Table A.4:** *DataSet Spanish (Part 1)*

| Author | Book | Publication Date |
|---|---|---|
| Jacinto Benavente | Heath's Modern Language Series | 2009 |
| | De Sobremesa; crónicas - V1 | 2017 |
| | De Sobremesa; crónicas - V2 | 2017 |
| | De Sobremesa; crónicas - V4 | 2018 |
| | De Sobremesa; crónicas - V5 | 2018 |
| Jacinto Octavio Picon | Cuentos de mi tiempo | 2008 |
| | Dulce y sabrosa | 2008 |
| | Lázaro | 2008 |
| | El enemigo | 2009 |
| | Vida y obras de don Diego Velázquez | 2009 |
| Jaime Balmes | Filosofia Fundamental - V1 | 2004 |
| | Filosofia Fundamental I-IV, V2 | 2005 |
| | Filosofia Fundamental, V3 | 2006 |
| | El Criterio | 2009 |
| | Filosofía Fundamental - V4 | 2009 |
| Jose Maria de Pereda | Escenas Montañesas | 2004 |
| | Los Hombres de Pro | 2005 |
| | La Montálvez | 2008 |
| | El Buey suelto | 2017 |
| | La Puchera | 2017 |
| Jose Rizal | El Consejo de los Dioses | 2005 |
| | Filipinas Dentro De Cien Años | 2005 |
| | Junto Al Pasig | 2005 |
| | El Filibusterismo | 2010 |
| | Noli me tángere | 2014 |
| Juan Valera | Doña Luz | 2005 |
| | Juanita la Larga | 2005 |
| | Cuentos y diálogos | 2008 |
| | A vuela pluma | 2011 |
| | Las Ilusiones del Doctor Faustino | 2016 |
| Manuel Fernandez y G. | Amparo | 2008 |
| | El Manco de Lepanto | 2009 |
| | Los Hermanos Plantagenet | 2012 |
| | La Vieja verde | 2014 |
| | La alhambra; leyendas árabes | 2015 |

**Table A.5:** *DataSet Spanish (Part 2)*

| Author | Book | Publication Date |
|---|---|---|
| Mariano Jose de Larra | Fígaro | 2010 |
| | El doncel de don Enrique el doliente V1 | 2016 |
| | El doncel de don Enrique el doliente V2 | 2016 |
| | El doncel de don Enrique el doliente V3 | 2016 |
| | El doncel de don Enrique el doliente V4 | 2016 |
| Marie Lebert | El Internet y los Idiomas | 2009 |
| | Una Corta Historia del eBook | 2009 |
| | Del Libro impreso al Libro digital | 2011 |
| | El eBook tiene 40 años | 2011 |
| | La web, una enciclopedia multilingüe | 2013 |
| Miguel de Unamuno | La tía Tula | 2013 |
| | Abel Sánchez: Una Historia de Pasión | 2013 |
| | Amor y Pedagogía | 2015 |
| | Niebla | 2015 |
| | Tres novelas ejemplares y un prólogo | 2017 |
| Pio Baroja | Las Inquietudes de Shanti Andía | 2014 |
| | Memorias de un Hombre de Acción: V4 | 2015 |
| | Memorias de un Hombre de Acción: V5 | 2015 |
| | El Sabor de la Venganza | 2017 |
| | La Isabelina | 2017 |
| Ramon del Valle-Inclan | Romance de lobos, comedia barbara | 2003 |
| | Sonata de estío | 2013 |
| | Sonata de primavera | 2013 |
| | Sonata de otoño | 2014 |
| | El Marqués de Brandomín | 2018 |
| Ruben Dario | Los Raros | 2015 |
| | Autobiografía | 2016 |
| | España Contemporánea | 2017 |
| | Parisiana | 2017 |
| | La Caravana Pasa | 2018 |
| Vicente Blasco Ibañez | El Préstamo de la Difunta | 2006 |
| | La Barraca | 2005 |
| | Los cuatro Jinetes del Apocalipsis | 2008 |
| | La Bodega | 2009 |
| | Los Enemigos de la Mujer | 2011 |

**Table A.6:** *DataSet Spanish (Part 3)*

# Appendix B

# Stopwords

By definition, Stopword are words that do not have a meaning by themselves (null meaning) but rather modify or accompany others, this group is usually made up of articles, pronouns, prepositions, adverbs and even some verbs. Next we will list the stopwords are filtered in the pre-processing stage, for English (Section B.1) and Spanish (Section B.2). Additionally in Section B.3 the punctuation marks that were removed in the text pre-processing are mentioned.

## B.1 English

The list below illustrates the stopwords in English that were used in the pre-processing of the texts. Therefore, all of these words have been eliminated from the analysis of writing style:

Don, needn't, you're, the, whom, shouldn't, yourself, for, when, ma, were, wasn't, between, don't, that, both, do, i, before, y, mightn't, now, myself, a, further, in, who, herself, is, be, how, where, by, and, am, than, himself, all, once, hadn't, wasn't, you, from, other, you'd, isn't, being, mustn't, this, wouldn't, they, shan't, an, will, ourselves, o, these, won't, them, same, into, couldn't, should, themselves, above, while, again, did, mustn't, its, any, at, should've, each, such, to, only, haven, can, wouldn't, what, if, why, d, didn't, about, been, off, 've, hasn't, more, aren't, he, she, own, ours, she's, here, yours, with, t, then, after, through, was, ain't, doing, him, it, or, they'll, are, re, because, no, had, needn't, her, on, hers, doesn't, shall, didn't, you've, so, does, few, we, won, has, his, up, very, of, most, weren't, hasn't, me, against, shouldn't, which, m, have, but, weren't, haven't, below, your, s, ll, our, as, theirs, down, just, their, itself, out, too, under, aren't, over, nor, couldn't, not, mightn't, until, during, its, having, doesn't, isn't, some, there, you'll, those, hadn't, yourselves, my.

## B.2 Spanish

Next we will show a list wich illustrates the stopwords in Spanish that were used in the pre-processing of the texts. Consequently, entirely these words have been excluded from the analysis of writing style:

He, hubieron, eran, estamos, tenido, tuvieses, fuesen, teníamos, habidos, habiendo, somos, tuve, estoy, había, esto, tú, habida, este, estar, esta, habrían, tenéis, unos, estaríais, seremos, todos, fui, lo, fuéramos, fuésemos, seas, suyos, se, habíais, estuvo, mis, una, mucho, sentidos, algunas, fueseis, tienes, hubimos, teniendo, antes, tendremos, nuestras, tuvieras, tuviesen, estaríamos, seáis, estuviéramos, tenidos, habían, hube, tuviéramos, hayan, sea, tuya, estarían, sois, estado, estará, sentida, muchos, eres, habré, y, son, otro, esa, tuvisteis, suyo, sean, tendría, estas, habríamos, sobre, otra, sí, fuese, hubisteis, hubieses, algunos, hubiéramos, todo, nosotras, durante, estuvieron, hubiésemos, como, tengamos, tuvieran, donde, tengo, estada, e, tenemos, estadas, qué, les, le, las, tuviese, tendré, desde, estuviera, han, estáis, están, suyas, mío, tendríais, estuvieran, estuviésemos, tened, algo, hay, fue, para, tendrán, entre, estaba, tuyas, tenías, tuviste, está, habías, también, estuve, hubierais, ha, tuyos, hubiera, tuvieseis, tendrá, estuvieseis, éramos, estuviesen, sentido, vuestra, sintiendo, fuerais, habrías, tenida, estuvimos, habrán, era, vuestras, estuviste, hasta, sentid, hubieran, esos, has, otras, ella, o, el, mías, fuimos, ese, estaréis, que, tengas, tenga, tanto, vosotros, tienen, quien, hayamos, nuestra, míos, habrá, en, estés, estados, ti, fuisteis, haya, te, suya, tuvierais, estuvieses, es, tiene, siente, vosotras, serás, estaría, tendrás, estuviese, yo, habidas, tuvieron, habréis, mi, estuvierais, hubiste, del, nada, fueran, hubieseis, hubiesen, más, con, sentidas, esté, ni, hubo, estuvieras, habido, porque, por, estos, habría, pero, cuando, estaré, tuvimos, nosotros, uno, ya, sería, nuestros, estaban, tenía, nos, hubieras, cual, teníais, estad, poco, mí, la, tuyo, estéis, estuvisteis, estén, fueron, contra, ellas, él, estemos, habrás, tengan, fueras, sus, un, eso, seréis, ellos, tenidas, seré, los, estábamos, seríamos, tuvo, habríais, estás, estarás, hayas, no, tengáis, estaremos, nuestro, tu, su, habéis, seamos, estarán, tenían, habremos, al, sin, tuviera, fueses, a, eras, muy, tendrían, fuiste, estando, tus, serán, hemos, esas, mía, seríais, fuera, tendréis, soy, quienes, vuestro, vuestros, tendrías, estarías, ante, será, habíamos, estabais, tuviésemos, me, hubiese, estabas, os, serías, hayáis, erais, tendríamos, otros, serían, de.

## B.3 Punctuation marks

This section lists the punctuation marks (Table B.1) that were removed for both databases (English and Spanish). Punctuation marks and stopwords were removed in pre-processing text stage. Next we list these punctuation marks below:

| Symbol | Punctuation Mark | Symbol | Punctuation Mark |
|---|---|---|---|
| . | Period | ; | Semicolon |
| ... | Ellipsis marks | ¡! | Exclamation mark |
| . | Square brackets | " " | Quotation marks |
| - | Hypen | ’ | Apostrophe |
| {} | Braces | : | Colon |
| , | Comma | () | Round brackets |
| ¿? | Question marks | ¨ | Umlaut marks |
| / | Slash | § | Paragraph marks |
| * | Asterisk | | |

**Table B.1:** *Puntuation Marks*

# Bibliography


[AAP+08] Diego R Amancio, Lucas Antiqueira, Thiago AS Pardo, LUCIANO da F. COSTA, Osvaldo N Oliveira Jr e Maria GV Nunes. Complex networks analysis of manual and machine translations. *International Journal of Modern Physics C*, 19(04):583–598, 2008. 34

[Alt92a] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992. 17, 49

[Alt92b] N.S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician - AMER STATIST*, 46:175–185, 08 1992. 15, 16

[ANOJC07] Lucas Antiqueira, M das Gracas V Nunes, ON Oliveira Jr e L da F Costa. Strong correlations between text quality and complex networks features. *Physica A: Statistical Mechanics and its Applications*, 373:811–820, 2007. 34

[AOJdFCN09] Lucas Antiqueira, Osvaldo N Oliveira Jr, Luciano da Fontoura Costa e Maria das Graças Volpe Nunes. A complex network approach to text summarization. *Information Sciences*, 179(5):584–599, 2009. 34

[APNOJ07] Lucas Antiqueira, Thiago Alexandre Salgueiro Pardo, Maria das Graças Volpe Nunes e Osvaldo N Oliveira Jr. Some issues on complex networks for author characterization. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 11(36):51–58, 2007. 33

[Bak18] Amir Bakarov. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*, 2018. 30

[BGJM16] Piotr Bojanowski, Edouard Grave, Armand Joulin e Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. 30

[BGV92] Bernhard E. Boser, Isabelle M. Guyon e Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. Em *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, páginas 144–152. ACM Press, 1992. 15, 18

[Bir06] Steven Bird. Nltk: the natural language toolkit. Em *Proceedings of the COLING/ACL on Interactive presentation sessions*, páginas 69–72. Association for Computational Linguistics, 2006. 40, 41

[Bro00] Michael W Browne. Cross-validation methods. *Journal of Mathematical Psychology*, 44(1):108 – 132, 2000. 52

[Che17] Minmin Chen. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*, 2017. 31, 48

[Cho10] Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010. 28

[CJ15] Michal Campr e Karel Jezek. Comparing semantic models for evaluating automatic document summarization. páginas 252–260, 09 2015. 1, 13

[CKP+15] Michael Crawford, Taghi Khoshgoftaar, Joseph Prusa, Aaron Richter e Hamzah Al-Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2:23, 10 2015. 2

[CLA+06] Silvia MG Caldeira, TC Petit Lobao, Roberto Fernandes Silva Andrade, Alexis Neme e JG Vivas Miranda. The network of concepts in written texts. *The European Physical Journal B-Condensed Matter and Complex Systems*, 49(4):523–529, 2006. 33

[CN+06] Gabor Csardi, Tamas Nepusz et al. The igraph software package for complex network research. *InterJournal, complex systems*, 1695(5):1–9, 2006. 45

[COJT+11] Luciano da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana e Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011. 33

[Com18] Chapter 11 - information retrieval: Concepts, models, and systems. Em Venkat N. Gudivada e C.R. Rao, editors, *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, volume 38 of *Handbook of Statistics*, páginas 331 – 401. Elsevier, 2018. 2, 19

[COT+11] Luciano da Fontoura Costa, Osvaldo N. Oliveira, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana e Luis E C Rocha. Analyzing and modeling real-world phenomena with complex networks : a survey of applications. *ADVANCES IN PHYSICS*, 60(3):329–412, 2011. 3

[CS01] Ramon Ferrer I Cancho e Richard V Solé. The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482):2261–2265, 2001. 33

[CTM+07] Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu e Niloy Ganguly. How difficult is it to develop a perfect spell-checker? a cross-linguistic analysis through complex network approach. *arXiv preprint physics/0703198*, 2007. 34

[dACA16] Henrique F de Arruda, Luciano da F Costa e Diego R Amancio. Using complex networks for text classification: Discriminating informative and imaginative documents. *EPL (Europhysics Letters)*, 113(2):28007, 2016. 33

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee e Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 32

[DDF+90] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer e Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990. 29

[Edm98] Philip Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. *arXiv preprint cs/9811009*, 1998. 34

[FdANSQM+18] Henrique Ferraz de Arruda, Filipi Nascimento Silva, Vanessa Queiroz Marinho, Diego Raphael Amancio e Luciano da Fontoura Costa. Representation of texts as complex networks: a mesoscopic approach. *Journal of Complex Networks*, 6(1):125–144, 2018. 39

[GAPDSP18] Helena Gomez Adorno, Juan Posadas Durán, Grigori Sidorov e David Pinto. Document embeddings learned on various types of n-grams for cross-topic authorship attribution. *Computing*, 100, 07 2018. 28

[GF18] Palash Goyal e Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. 34, 35

[GL16] Aditya Grover e Jure Leskovec. Node2vec: Scalable feature learning for networks. Em *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, páginas 855–864, New York, NY, USA, 2016. ACM. 25, 47

[GR71] Gene H Golub e Christian Reinsch. Singular value decomposition and least squares solutions. Em *Linear Algebra*, páginas 134–151. Springer, 1971. 29

[Har71] Michael Hart. Dataset gutenberg, @BOOKLET, 1971. 39, 73

[HCK16] Felix Hill, Kyunghyun Cho e Anna Korhonen. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*, 2016. 32

[HHHH09] Simon S Haykin, Simon S Haykin, Simon S Haykin e Simon S Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2009. 18, 49

[HLZB18] Shexia He, Zuchao Li, Hai Zhao e Hongxiao Bai. Syntax for semantic role labeling, to be, or not to be. Em *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 2061–2071, Melbourne, Australia, Julho 2018. Association for Computational Linguistics. 2, 13

[HM17] Matthew Honnibal e Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017. 41

[HSMN12] Eric H Huang, Richard Socher, Christopher D Manning e Andrew Y Ng. Improving word representations via global context and multiple word prototypes. Em *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, páginas 873–882. Association for Computational Linguistics, 2012. 19

[Juo08] Patrick Juola. Authorship attribution. *Foundations and Trends® in Information Retrieval*, 1:233–334, 03 2008. 1, 13

[KBDR16] Tom Kenter, Alexey Borisov e Maarten De Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016. 31

[KE07] Jinyun KE. Complex networks and human language. 02 2007. 25

[KMN+02] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman e A. Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002. 15

[Kor12] Vandana Korde. Text classification and classifiers:a survey. *International Journal of Artificial Intelligence Applications*, 3:85–99, 03 2012. 1, 2, 13

[KPC95] Julian Kupiec, Jan Pedersen e Francine Chen. A trainable document summarizer. Em *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, páginas 68–73. ACM, 1995. 15, 49

[KTA+16] Joo-Kyung Kim, Gokhan Tur, Celikyilmaz Asli, Bin Cao e Ye-Yi Wang. Intent detection using semantically enriched word embeddings. páginas 414–419, 12 2016. 15

[KZM14] Svetlana Kiritchenko, Xiaodan Zhu e Saif Mohammad. Sentiment analysis of short informal text. *The Journal of Artificial Intelligence Research (JAIR)*, 50, 08 2014. 2, 13

[KZS+15] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba e Sanja Fidler. Skip-thought vectors. Em *Advances in neural information processing systems*, páginas 3294–3302, 2015. 32

[LM14] Quoc Le e Tomas Mikolov. Distributed representations of sentences and documents. Em *International conference on machine learning*, páginas 1188–1196, 2014. 31

[LMG09] Arash Habibi Lashkari, Fereshteh Mahdavi e Vahid Ghomi. A boolean model in information retrieval for search engines. Em *2009 International Conference on Information Management and Engineering*, páginas 385–389. IEEE, 2009. 28

[Man02] Javier Valenzuela Manzanares. Linguistica contrastiva ingles-español: una visión general. 51:27–45, 2002. 7

[Mar] Vanessa Queiroz Marinho. *Development of new models for authorship recognition using complex networks*. Tese de Doutorado, Universidade de São Paulo. 14, 39

[MCCD13] Tomas Mikolov, Kai Chen, Gregory S. Corrado e Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. 3, 22

[Men87] Thomas Corwin Mendenhall. The characteristic curves of composition. *Science*, 9(214):237–249, 1887. 28

[MHA16] Vanessa Queiroz Marinho, Graeme Hirst e Diego Raphael Amancio. Authorship attribution via network motifs identification. Em *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, páginas 355–360. IEEE, 2016. 33

[Mil95] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 33

[Mil98] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998. 33

[ML18] Julia Masche e Nguyen-Thinh Le. A review of technologies for conversational systems. páginas 212–225, 06 2018. 13

[MM09] Ronei Moraes e Liliane Machado. Gaussian naive bayes for online training assessment in virtual reality-based simulato. *Mathware Soft Computing*, 16:123–132, 01 2009. 15, 17

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado e Jeff Dean. Distributed representations of words and phrases and their compositionality. Em *Advances in neural information processing systems*, páginas 3111–3119, 2013. 23, 30

[NCV+17] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu e Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017. 25

[NMV+17] Annamalai Narayanan, Chandramohan Mahinthan, Rajasekar Venkatesan, Lihui Chen, Yang Liu e Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. 07 2017. 26

[NOMC11] Prakash M Nadkarni, Lucila Ohno-Machado e Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 09 2011. 1

[Pal05] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing - INT J REMOTE SENS*, 26:217–222, 01 2005. 15

[PARS14] Bryan Perozzi, Rami Al-Rfou e Steven Skiena. Deepwalk: Online learning of social representations. Em *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 701–710, 2014. 35

[PGJ17] Matteo Pagliardini, Prakhar Gupta e Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*, 2017. 31

[PSM14] Jeffrey Pennington, Richard Socher e Christopher Manning. Glove: Global vectors for word representation. Em *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, páginas 1532–1543, 2014. 30

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 42, 43

[Qui14] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. 16, 49

[RARP12] Rafael Ribaldo, Ademar Takeo Akabane, Lucia Helena Machado Rino e Thiago Alexandre Salgueiro Pardo. Graph-based methods for multi-document summarization: exploring relationship maps, complex networks and discourse information. Em *International Conference on Computational Processing of the Portuguese Language*, páginas 260–271. Springer, 2012. 1

[Ras14] Sebastian Raschka. Naive bayes and text classification i - introduction and theory. 10 2014. 17

[Rei96] Joan M Reitz. *Online dictionary for library and information science: ODLIS*. Libraries Unlimited, 1996. 33

[RM05] L. Rokach e O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005. 15, 16

[ŘS10] Radim Řehůřek e Petr Sojka. Software Framework for Topic Modelling with Large Corpora. Em *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, páginas 45–50, Valletta, Malta, Maio 2010. ELRA. http://is.muni.cz/publication/884893/en. 43

[SB88] Gerard Salton e Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. 29

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002. 22

[SKD19] Tomasz Stanisz, Jarosław Kwapień e Stanisław Drożdż. Linguistic data mining with complex networks: a stylometric-oriented approach. *Information Sciences*, 482:301–320, 2019. 34, 37, 39, 44

[SPdVLO19] Xabier Soto, Olatz Perez-de Viñaspre, Gorka Labaka e Maite Oronoz. Neural machine translation of clinical texts between long distance languages. *Journal of the American Medical Informatics Association*, 26(12):1478–1487, 07 2019. 13

[Sta09a] Efstathios Stamatatos. A survey of modern authorship attribution methods. *JASIST*, 60:538–556, 03 2009. 2, 14, 27, 28

[Sta09b] Efstathios Stamatatos. A survey of modern authorship attribution methods. *JASIST*, 60:538–556, 03 2009. 2

[SWY75a] G. Salton, A. Wong e C. S. Yang. A vector space model for automatic indexing. 18(11):613–620, Novembro 1975. 2, 20

[SWY75b] Gerard Salton, Anita Wong e Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. 29

[TA17] Jorge Valverde Tohalino e Diego Raphael Amancio. Extractive multi-document summarization using dynamical measurements of complex networks. Em *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, páginas 366–371. IEEE, 2017. 33, 34

[TA18] Jorge V Tohalino e Diego R Amancio. Extractive multi-document summarization using multilayer networks. *Physica A: Statistical Mechanics and its Applications*, 503:526–539, 2018. 33

[TKL+03] Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes e Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. Em *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, páginas 41–47. ACM, 2003. 22

[TRB10] Joseph Turian, Lev Ratinov e Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. Em *Proceedings of the 48th annual meeting of the association for computational linguistics*, páginas 384–394. Association for Computational Linguistics, 2010. 22

[UF16] Rishabh Upadhyay e Akihiro Fujii. Semantic knowledge extraction from research documents. volume 8, 09 2016. 13

[VE18] Kimmo Vehkalahti e Brian Everitt. *Applying Logistic Regression*, páginas 123–137. 12 2018. 15

[VOGMB19] Didier A Vega-Oliveros, Pedro Spoljaric Gomes, Evangelos E Milios e Lilian Berton. A multi-centrality index for graph-based keyword extraction. *Information Processing & Management*, 56(6):102063, 2019. 34

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser e Illia Polosukhin. Attention is all you need. Em *Advances in neural information processing systems*, páginas 5998–6008, 2017. 32

[War92] Steven P Wartik. Boolean operations., 1992. 28

[WCZ16] Daixin Wang, Peng Cui e Wenwu Zhu. Structural deep network embedding. Em *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, páginas 1225–1234, 2016. 35

[WS05] Scott White e Padhraic Smyth. A spectral clustering approach to finding communities in graphs. Em *Proceedings of the 2005 SIAM international conference on data mining*, páginas 274–285. SIAM, 2005. 35

[Zha04] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004. 17, 49