UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DESING AND DEVELOPMENT OF A HATE SPEECH DETECTOR IN SOCIAL NETWORKS BASED ON DEEP LEARNING TECHNOLOGIES

DIEGO BENITO SÁNCHEZ

JUNIO 2019

TRABAJO DE FIN DE MÁSTER

Título:	Diseño y Desarrollo de un Detector de Discurso de Odio en Redes Sociales Basado en Técnicas de Aprendizaje Profundo
Título (inglés):	Design and Development of a Hate Speech Detector in Social Networks Based on Deep Learning Technologies
Autor:	Diego Benito Sánchez
Tutor:	Óscar Araque
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MÁSTER

DESING AND DEVELOPMENT OF A HATE SPEECH DETECTOR IN SOCIAL NETWORKS BASED ON DEEP LEARNING TECHNOLOGIES

DIEGO BENITO SÁNCHEZ

JUNIO 2019

Resumen

Recientemente, durante los últimos años, tanto el uso de Internet como la conectividad en redes sociales se han visto in incrementados. Las redes sociales son plataformas que facilitan la comunicación entre usuarios mediante diferentes interacciones. Desafortunadamente, las redes sociales también se han convertido lugares para la proliferación de discurso de odio.

El discurso de odio se ha convertido en un tema popular en los últimos años. Esto puede verse reflejado no solo en el aumento de la cobertura de este problema, sino también en la creciente atención política que está recibiendo. Ante la progresión de este fenómeno, instituciones, asociaciones de minorías internacionales, investigadores y redes sociales están tratando de reaccionar lo más rápido posible. Debido a la escala masiva de las redes sociales, se requieren métodos que detecten automáticamente el discurso del odio. Procesado del Lenguaje Natural (PLN) enfocado específicamente en este fenómeno es necesario, dado que filtros básicos de palabras no proporcionan suficiente remedio: un mensaje con discurso de odio podría estar influenciado por aspectos como el dominio, el contexto, el uso de multimedia (imágenes, vídeos, audios), etc.

Esta memoria es el resultado de un proyecto cuyo objetivo principal ha sido obtener un detector de discurso de odio con una perspectiva multilingüe, con el fin de eliminar toda forma de discurso de odio que pueda ocurrir en las redes sociales independientemente del lenguaje en el que se produzca. Para desarrollarlo, se han utilizado herramientas de aprendizaje automático supervisado, técnicas de PLN y Python como lenguaje de programación.

El sistema propuesto ha sido evaluado haciendo uso de dos casos de estudio, la participación en una competición reconocida a nivel internacional, como es SemEval, y enfrentado el sistema a un desafío transferencia de aprendizaje entre idiomas y rasgos del discurso de odio. La extensa experimentación llevada a cabo ha resultado en una posición muy honorable en la competición de SemEval y en una demostración de los beneficios que puede aportar la aplicación de la transferencia de Aprendizaje al problema de la detección del discurso de odio.

Palabras clave: Redes Sociales, Discurso de Odio, Aprendizaje Autómatico, Procesado del Lenguaje Natural, SemEval, Transferencia de Aprendizaje, Python, Scikit-learn, NLTK.

Abstract

Recently, during the last few years, activity over Internet and social network connectivity has been increased. Social networks are platforms that ease communication between users by means of different interactions. Unfortunately, social networks have also become places for hate speech proliferation.

Hate Speech has become a popular topic in recent years. This is reflected not only by the increased media coverage of this problem but also by the growing political attention it is receiving. Given the constant progression of this phenomenon, institutions, international minorities associations, researchers and social networks are trying to react as quickly as possible. Because of the massive scale of the social networks, methods that automatically detect hate speech are required. Natural Language Processing (NLP) focusing specifically on this phenomenon is required since basic word filters do not provide a sufficient remedy: a hate speech utterance might be influenced by aspects such us the domain, context, co-occurrence media objects (images, video, audio), etc.

This thesis is the result of a project whose main aim has been to obtain a hate speech detector with a multilingual perspective, in order to remove all shape of hate speech that can occur in social networks, independently the origin language. During the development phase, there have been used supervised machine learning tools, NLP techniques, and Python as programming language.

The proposed system is evaluated against two study cases, a participation in a internationally recognized competition, such as SemEval and facing the system against a Transfer Learning challenge across languages and hate speech traits. The extensive experimentation carried out has resulted in a very honorable position in the SemEval competition and in a demonstration of the benefits that can be brought by the appliance of Transfer Learning to the hate speech detection problem.

Keywords: Social Networks, Hate Speech, Machine Learning, Natural Language Processing, SemEval, Transfer Learning, Scikit-learn, NLTK.

Agradecimientos

En este apartado, me gustaría agradecer a las que han influido de manera positiva en mí durante los dos años que he estado estudiando el Máster y que han permitido que pueda finalizar esta titulación de la manera más satisfactoria posible.

En primer lugar, quiero dar las gracias a mis padres, por hacer posible que yo pudiera realizar estos estudios, tanto a nivel económico como a nivel anímico.

En especial, me gustaría agradecer a mi pareja, Belén, que siempre haya confiado en mí y me haya hecho creer en mis posibilidades cuando no todo venía de cara.

Por supuesto, considero que merecen una mención en este apartado todos mis compañeros y amigos que he tenido durante mi vida en la universidad, todos ellos han hecho que este sufrido proceso sea más ameno.

Además, quiero dar las gracias a mis compañeros del Intelligent Systems Group (GSI), que desde el primer día me han tratado como si fuera uno más y siempre estaban disponibles para las dudas y problemas que me surgían.

Por último, me gustaría agradecer la labor de Óscar Araque, mi tutor, ya que gracias a él he adquirido muchos conocimientos interesantes y me ha prestado una ayuda fundamental para realizar el trabajo.

Contents

Resumen **VII**

Abstract **IX**

Agradecimientos **XI**

Contents **XIII**

List of Figures **XIX**

1 Introduction **1**

1.1 Context 2

1.2 Motivation 5

1.3 Project Goals 6

1.4 Structure of this Document 6

2 State of Art **9**

2.1 Related Work 10

2.2 Features 15

2.2.1 General Features 15

2.2.2 Specific Hate Speech Detection Features 20

2.3 State of the art Performance 21

2.4 Datasets and Open Source Projects 23

3 Enabling Technologies **27**

3.1 Data Managing 28
3.1.1 Numpy 28
3.1.2 Pandas 28
3.2 Natural Language Processing 29
3.2.1 NLTK 29
3.2.2 Gensim 30
3.2.3 TextBlob 30
3.3 Machine Learning 31
3.3.1 Scikit-learn 31
3.3.2 Imbalanced-learn 32
3.3.3 GSITK 32
3.4 Hatebase API 34
3.5 Machine Learning Fundamentals 35
3.5.1 Logistic Regression 37
3.5.2 Support Vector Machines 38
3.5.3 Random Forest 39
3.5.4 Artificial Neural Networks 40
3.5.4.1 Perceptron 40
3.5.4.2 Multi-Layer Perceptron 41
3.6 NLP Fundamentals 42
3.6.1 BOW 43
3.6.2 TF-IDF 44
3.6.3 LDA 45
3.6.4 Word Embeddings 46
3.6.4.1 Word2Vec 47
3.6.4.2 GloVe 48

3.6.5 SIMON 49

4 Participation at SemEval-2019 Task 5 **53**

4.1 Introduction 54

4.2 Data 55

4.3 Evaluation 56

4.3.1 Task A 57

4.3.2 Task B 58

4.4 System Overview 58

4.4.1 Preprocessing 59

4.4.2 Feature Engineering 60

4.4.2.1 Statistical features 60

4.4.2.2 Content Analysis 60

4.4.2.3 Word Embeddings 61

4.4.2.4 Semantic Features 62

4.4.2.5 Linguistic Features 63

4.4.2.6 Feature Selection 64

4.4.3 Classification 64

4.5 Experiments 64

4.5.1 Sub-task A 65

4.5.2 Sub-task B 66

4.5.3 Discussion 67

5 Transfer Learning for Hate Speech Detection **71**

5.1 Introduction 72

5.1.1 Multilingual Word Embeddings 74

5.2 Data 75

5.2.1 English Data . . . 75
5.2.1.1 Davidson Data . . . 75
5.2.1.2 Waseem Data . . . 76
5.2.1.3 Stormfront Data . . . 76
5.2.2 German Data . . . 77
5.2.3 Indonesian Data . . . 77
5.2.3.1 Instagram Dataset . . . 78
5.2.3.2 Twitter Dataset . . . 78
5.2.4 Portuguese Data . . . 79
5.2.5 Italian Data . . . 79
5.2.5.1 Facebook Dataset . . . 79
5.2.5.2 Twitter Dataset . . . 80
5.3 Cross Language Transfer Learning . . . 80
5.3.1 Architecture . . . 80
5.3.2 Results . . . 83
5.3.2.1 Spanish . . . 84
5.3.2.2 German . . . 85
5.3.2.3 Indonesian . . . 86
5.3.2.4 Portuguese . . . 87
5.3.2.5 Italian . . . 87
5.4 Cross Domain Transfer Learning . . . 88
5.4.1 Results without Fine-tuning . . . 89
5.4.2 Results with Fine-tuning . . . 90

6 Conclusions **95**

6.1 Conclusions . . . 96
6.2 Achieved Goals . . . 97

6.3 Problems Faced 97
6.4 Future Work 99

A Ethical, Economical, Social and Environmental Impact **i**

A.1 Introduction i
A.2 Ethical Impact ii
A.2.1 Freedom of Speech ii
A.2.2 Information Privacy iii
A.3 Economic Impact iv
A.4 Social Impact v
A.5 Environmental Impact v

B Economic Budget **vii**

B.1 Introduction vii
B.2 Physical Resources viii
B.3 Human Resources viii
B.4 Indirect Costs ix
B.5 Taxes x

Bibliography **xi**

List of Figures

2.1 Number of publications per year (N = 51) (Figure from Fortuna et al. [42]) 10

2.2 Dataset sizes used in the papers (Elaborated from Fortuna et al. [42]) . . . 13

2.3 General features (Elaborated from Fortuna et al. [42]) 19

2.4 Dataset availability used in the papers (Elaborated from Fortuna et al. [42]) 23

2.5 Programming languages used in the projects 24

3.1 Hatebase community 34

3.2 Hatebase API 35

3.3 Overfitted versus regularized system 36

3.4 Logistic function 37

3.5 Maximum margin hyperplane 38

3.6 Kernel-trick 38

3.7 Effect of changing C parameter 39

3.8 Tree learned on titanic data 39

3.9 Perceptron architecture 41

3.10 Multi-Layer Perceptron 41

3.11 Comparison between activation functions 42

3.12 Word Embeddings patterns 47

3.13 Skip-gram model 48

3.14 CBOW model 48

3.15 Weighting function 49

3.16 Cosine similarity in a Word Embeddings model 50

3.17 Similarity features computation 51

4.1 System Overview 59

4.2 Keyword Matching 61

5.1 Transfer Learning process 72

5.2 Transfer Learning evaluation (Figure from Torrey et al. [89]) 73

5.3 Toy illustration of the method. Figure from Conneau et al. [27] 75

5.4 System trained with English data 81

5.5 English trained system transformed to predict hate speech in the target language 82

5.6 English trained system fine-tuned with target language data 83

5.7 Logistic Regression (left), Linear Support Vector Machine (SVM) (center), and Random Forest (right) curves for Spanish development set 84

5.8 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set 84

5.9 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for German annotation from experts group 1 85

5.10 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for German annotation from experts group 2 86

5.11 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Indonesian Instagram dataset 86

5.12 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Indonesian Twitter dataset 87

5.13 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Portuguese data 87

5.14 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Italian Facebook data 88

5.15 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Italian Twitter data 88

5.16 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English development set and transfer from aggressive to target learning 91

5.17 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English development set and transfer from target to aggressive learning 91

5.18 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English test set and transfer from aggressive to target learning . 92

5.19 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English test set and transfer from target to aggressive learning . 92

5.20 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish development set and transfer from aggressive to target learning 93

5.21 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish development set and transfer from target to aggressive learning 93

5.22 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set and transfer from aggressive to target learning . 93

5.23 Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set and transfer from target to aggressive learning . 93

CHAPTER 1

Introduction

This chapter introduces the context of the project, including a brief overview of all the different parts that will be discussed in the project. Special attention revolves around the hate speech definition statement which complexity may exceed prior expectations. After this, we describe the motivation for the development of this project in a few lines.

It also breaks down a series of objectives to be carried out during the realization of the project. Moreover, it introduces the structure of the document with an overview of each chapter.

1.1 Context

Nowadays, almost everyone has a device connected to the Net and if we focus on younger people, we find that almost everyone shares personal information on social networks [86]. This information is the result of many interactions between users and their activity on the Net like posting, friends or their network size.

Unfortunately, hate speech and other misuses are proliferating on the Internet. Hate speech is a complex phenomenon, intrinsically associated with relationships between groups, and also relying on language nuances. Hate speech authors justify this conduct based on the freedom of speech argument and debate over hate speech legislation and freedom of speech has been generated [48]. But this argument has no sense since freedom of speech is not understood if people are afraid to express themselves as they are.

The task to decide if a piece of text contains hate speech is not trivial, even for humans. Being subject to different interpretations and opinions, the manifestations of hate speech become difficult to define. For this reason, establishing homogeneous and uniform criteria that allows us to identify said discourse has become a controversial challenge. For that reason, we have compiled in Table 1.1 some definitions from different sources (surveys, social networks, minorities associations[1], and The European Union Commission).

Despite the similarities between the definitions, we conclude that there are some aspects that distinguish them. For example, Facebook and YouTube mention any content (not only written language), inferring that multimedia content could include hate speech. Also, Facebook makes a special mention of humor comments. In a similar way, Twitter has into account the context of the message. ILGA-Europe does not classify hate speech into groups, so any group could be a victim of hate speech. Having in mind all the subtleties cited above, we can give a complete hate speech definition, which is very similar to that stated by Paula Fortuna in her survey [42]:

> *"Hate Speech is any communication (text, image, video, etc.) that attacks, diminishes, incites violence or hate against individuals or groups, based on actual or perceived specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity, or any other".*

Though it is not explicitly mentioned, we see necessary including jokes (if they fulfill the previous specifications) as hate speech because even though they can be considered

[1] International Lesbian, Gay, Bisexual, Trans, Intersex Association (ILGA)

Table 1.1: Hate Speech Definitions

Source	**Definition**
EU Code of Conduct	"Public incitement to violence or hatred directed to groups or individuals on the basis of certain characteristics, including race, colour, religion, descent and national or ethnic origin." [26]
ILGA-Europe	"Public expressions which spread, incite, promote or justify hatred, discrimination or hostility towards a specific group. They contribute to a general climate of intolerance which in turn makes attacks more probable against those given groups." [50]
Schmidt et al.	"Any communication that disparages a person or a group on the basis of some characteristic such us race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic." [83]
Twitter	"Encourage violence against other people, attack or threaten them directly because of their race, ethnic origin, nationality, sexual orientation, gender, gender identity, religious affiliation, age, disability or serious illness. Taking into account the context in which this information is published"[2]
Facebook	"All content that directly attacks people because present what we call 'special traits': race, ethnicity, nationality, religion, class, sexual orientation, sex, sexual identity and disability or serious illness. We allow humorous, and educative purpose comments related to these topics." [1]
YouTube	"Content that promotes violence against or has the primary purpose of inciting hatred against individuals or groups based on certain attributes, such as: race or ethnic origin, religion, disability, gender, age, veteran status, sexual orientation/gender identity." [3]

harmless, the repetition of these jokes can become a way to reinforce hate speech attitudes without punishment. We also state that any group can be a target of hate speech instead of

creating a "protected groups" classification because new targets of hate speech can appear and those are undetectable unless the "protected groups" are redefined. Finally, we exclude offensive content in case of the context is not inciting to violence.

There is an additional consideration to complete this definition because hate speech is very dependent on the context. Offensive content does not necessarily have to be hate speech and hate speech does not have to be offensive. In light of this definition, there is a fine line between what is and what is not considered to be hate speech. For example, merely mentioning an organization associated with hate crimes ("Ku Klux Klan") does not constitute hate speech. Even an endorsement of the organization does not constitute a verbal attack on another group. For the same reason, an author's excessive pride in his own race (Aryan race) or group does not constitute hate speech. Another special case to notice, is that criticize a nation is allowed, while attack someone based on his national origin is not.

Although we have said that any group can be target of hate speech, historically these groups have been very well identified and Table 1.2 gathers up each type of hate speech that can be found in the actual literature.

Table 1.2: Types of Hate Speech and examples (Table from Silva et al. [85])

Categories	**Example of possible targets**
Race	nigga, black people, white people
Behaviour	insecure people, sensitive people
Physical	obese people, beautiful people
Sexual orientation	gay people, straight people
Class	ghetto people, rich people
Gender	pregnant people, cunt, sexist people
Disability	retard, bipolar people
Ethnicity	chinese people , indian people, paki
Religion	religious people, jewish people
Other	drunk people, shallow people

1.2 Motivation

The motivation of this project comes firstly by the increasing impact and popularity of hate speech topic. In fact, hate speech is prohibited by law, typified alongside the known hate crimes [2], therefore each country has its own laws approved against such crimes. Furthermore, The European Union Commission has founded several programs towards the fighting of hate speech (e.g., No Hate Speech Movement by the Council of Europe [70]) and pressured Facebook, Twitter, YouTube and Microsoft to remove any appearance of hate speech in their platforms in less than 24h [47]. Nevertheless, it seems that neither Facebook and Twitter are performing this task well enough [55, 19].

Secondly, the scientific study of hate speech, from a computer science point of view, is recent, and there is a general lack of data about hate speech. For this reason, the development and systematization of shared resources, such as guidelines, annotated datasets in multiple languages, and algorithms, is a crucial step in advancing the automatic detection of hate speech.

In addition, many companies and institutions might be very interested in hate speech mitigation over their platforms. After all, they provide a service which they profit, consequently, they assume public obligations with respect to the contents transmitted. Besides, their reputation mainly comes from the perceived image by users, so they cannot risk becoming known as *hate sites* [3]. Additionally, users can be interested in blocking discourse with hate speech to avoid being exposed to it. In this case, it is necessary taking steps to discourage online hate and remove hate speech within a reasonable time (as pretended by the European Union Commission).

For this reason, the aim of this project is to develop a hate speech detector using automated procedures in a social network context. These procedures are mainly supported by two fundamental fields, Machine Learning and Natural Language Processing. For that purpose, different techniques have been explored carrying out several study cases. In first place, we decided to participate in SemEval-2019 Task 5 [12]: Multilingual Detection of Hate Speech Against Immigrants and Women on Twitter. Then, in a second phase, a problem of Transfer Learning [89] across languages and domains was addressed. Next section, describes the expected results from this experimentation.

[2] `https://ucr.fbi.gov/hate-crime`

[3] A website that contains hate speech

1.3 Project Goals

As stated before, the final objective of this project is to obtain a hate speech classifier digging through different approaches using publicly available annotated data. In general issues, among the main goals of the project we can find:

- Review the current state of the art of the field in terms of available corpus with the aim of unifying the use of hate speech data.
- Develop a series of solutions capable of detecting hate speech in a social network context.
- Apply our solution into two study cases, an internationalized competition to detect hate speech, and a Transfer Learning problem.
- Explore the performance of our proposal in the Semeval participation
- Explore the performance of using architectures formed by traditional methods and more recent techniques in the NLP field for text classification.
- Explore the potential of Transfer Learning applied to the problem of hate speech detection across languages and hate speech traits.
- Conclusion extraction related to the results of the project and the presence of online hate speech.

1.4 Structure of this Document

In this section, we provide a brief explanation of the chapters included in this thesis. The structure follows this schema:

Chapter 1 explains the context in which this project is developed, mainly the hate speech phenomenon. Moreover, it describes the main goals to achieve in this project, as well as the motivation and the structure of this document.

Chapter 2 offers a brief review of the related published works found in the literature. Besides, some characteristics of the field and the datasets encountered are presented.

Chapter 3 provides a description of the main technologies on which this project relies, among them, we can find some Python libraries, NLP and Machine Learning technologies.

Chapter 4 describes the participation in SemEval-2019 Task 5.

Chapter 5 presents the Transfer Learning system evaluated in both, multilingual and cross-domain approach.

Chapter 6 discusses the conclusions drawn from this project, as well as the problems faced in its development. Finally, it focuses on the possible next step to be done as future work.

CHAPTER 2

State of Art

This chapter reviews the current state of the field, providing a general overview of previous approaches, including core algorithms, methods, and main features used. We first describe the work carried out by previous survey research which offers a general overview of the field of automatic hate speech detection. After this, we focus on the features used by other authors. Continuing, the main evaluation results are presented together with a discussion about their reliability. To finalize, this chapter also covers a description of the datasets and open source projects available to date.

2.1 Related Work

Hate speech detection is a new exploited field so the number of studies related to the domain is scarce. Figure 2.1 shows the number of publications per year since 2004, where can be seen that before 2014 the number of documents in relation to hate speech rarely surpassed two publications per year. However, since 2014 this number has been increasing until dozens of research papers in 2015 and 2016. The smaller value in 2017 is due to the search stopped in May 2017.

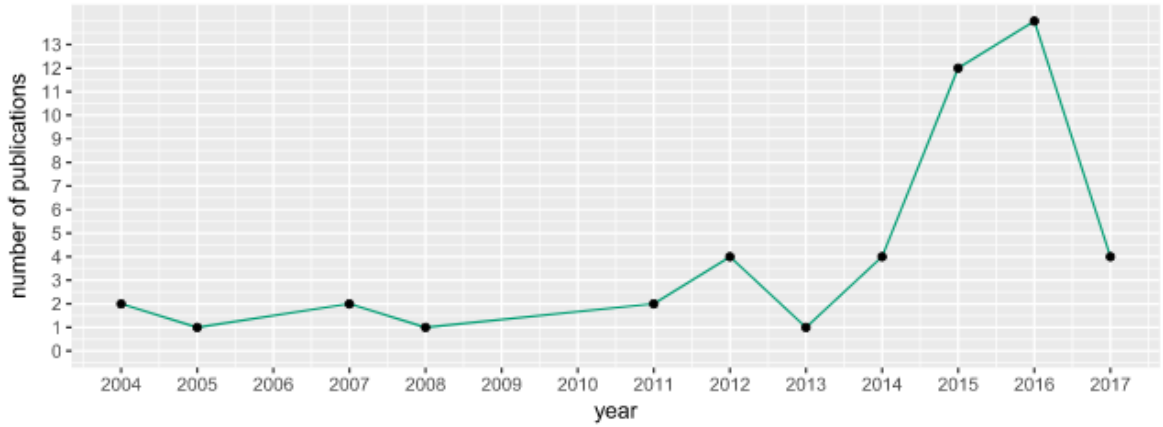

Figure 2.1: Number of publications per year (N = 51) (Figure from Fortuna et al. [42])

Most of our literature review from the field is referenced by previous survey research which provides a comprehensive, structured, and critical overview of the field of automatic hate speech detection using NLP techniques. One of them [83] was published in April 2017 and the other one [42] in July 2018, so the information they supply can be considered as actualized. Summarizing both, it could be said that firstly they define hate speech and compare hate speech with similar concepts (Hate, Cyberbullying, Discrimination, Abusive language, Extremism, Radicalization, etc.) and enumerate rules for hate speech identification. Then, they analyze the features used to solve the problem. Later, they focus on different classification methods used. Finally, they present existing data collections for this task, some applications, challenges, and problems faced.

Below we present a set of valuable characteristics extracted by the surveys that contain some interesting information. In the first place, we will talk about the keywords referred in the documents. These keywords were grouped and computed their frequencies. Table 2.1 shows the keywords with the highest frequencies. We conclude that a paper studies hate speech detection when it is related to:

- **Hate speech concepts:** cyberbullying, cyber hate, sectarianism, and freedom of speech.
- **Machine learning:** classification, sentiment analysis, filtering systems, and machine learning.
- **Social media:** internet, social media, social network, social networking, and hashtag.

Table 2.1: Keywords of the papers (Table from Fortuna et al. [42])

Keyword	**Frequency**
Cyberbullying	5
Social media	5
Classification	4
Internet	4
Freedom of speech	3
Hate speech	3
Machine learning	3
NLP	3
Sentiment analysis	3
Social network	3
Social networking (online)	3
Cyber hate	2
Filtering systems	2
Hashtag	2
Sectarianism	2

The found documents analyze datasets with messages that were collected from social networks. Table 2.2 presents the social networks used in the papers and their frequencies. Twitter is the most commonly used source, followed by general sites, YouTube and Yahoo!.

Table 2.2: Social networks used in the papers (Table from Fortuna et al. [42])

Keyword	**Frequency**
Twitter	5
Sites	5
YouTube	4
Yahoo! finance	4
American Jewish Congress (AJC) sites	3
Ask.fm	3
Blogs	3
Documents	3
Facebook	3
formspring.me	3
myspace.com	3
Tumbler	2
Whisper	2
White supremacist forums	2
Yahoo news	2
Yahoo!	2

The surveys also analyzed the number of instances used by each paper. An instance is an example that can be used by a model either to learn (if the instance belongs to the training set) or to predict (if the instance belongs to the test set). In the context of hate speech, an instance is a text message (it could also include meta or multimodal information, we will discuss this later), with a classification label. The total number of instances is the dataset size. Figure 2.2 shows a dataset histogram based on the dataset size. The number of instances has a wide range of magnitudes. Nevertheless, we can conclude that the majority

of papers use between 1,000 and 10,000 instances.

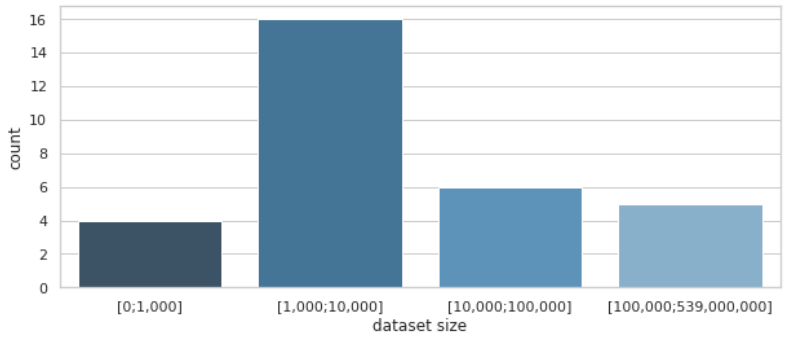

Figure 2.2: Dataset sizes used in the papers (Elaborated from Fortuna et al. [42])

We can analyze if the found documents focus on general hate speech or on more particular types of hate. Table 2.3 compile this information, and we can see that the majority (N = 26) considers general hate speech, however, there is a large number of papers (N = 18) that focus particularly on racism.

Table 2.3: Type of hate speech analyzed in the papers (Table from Fortuna et al. [42])

Hate type	**Frequency**
General hate speech	26
Racism	18
Sexism	6
Religion	4
Anti-semitism	1
Nationality	1
Other	1
Physisical/mental handicap	1
Politics	1
Sectarianism	1
Social and economic status	1

According to the task addressed, the most common approach found from the systematic literature review carried out by Fortuna et al [42] consists of building a Machine Learning model for hate speech classification. They also found that the most common algorithms used are SVM, Random Forest, and Decision Tree. Additional information about other algorithms is given in Table 2.4.

Table 2.4: Algorithms used in the papers (Table from Fortuna et al. [42])

Algorithms	**Frequency**
SVM	10
Random Forest	5
Decision Tree	4
Logistic Regression	4
Naive Bayes	3
Deep Learning	1
DNN	1
Ensemble	1
GBDT	1
LSTM	1
Non-supervised	1
One-class classifiers	1
Skip-gram model	1

2.2 Features

In this section, we analyze features used by other authors for hate speech detection. Finding the right features for a classification problem can be one of the more complex tasks when using machine learning. Therefore, we allocate this specific section to describe the features already used by other authors. We divide features into two categories: general features used in any field, which can be found in other data mining fields; and the specific hate speech detection features, which are found in hate speech detection documents and are intrinsically related to the characteristics of this problem. We present our analysis in the following sections. It is recommended a previous reading on Sections 3.5 and 3.6 for better understanding.

2.2.1 General Features

The majority of the papers found by the surveys try to adopt strategies already known in text mining for the specific problem of automatic detection of hate speech. We define general features as the features commonly used in text and multimedia content. We start by the most simplistic approach that uses dictionaries and lexicons.

Dictionaries. This approach consists of making a list of words (the lexicon or dictionary) that are searched and counted in the text. These frequencies can be used directly as features or to compute scores. There are several publicly available lists that consist of general hate-related terms[1]. Apart from sites that contain such lists, there are also sites which focus on lists that are specialized in a particular subtype of hate speech, such as ethnic slurs[2], Lesbian, Gay, Bisexual, and Transgender (LGBT) slang terms[3], or words with a negative connotation towards handicapped people[4].

Distance Metric: in online content, it is possible that the offensive words are obscured with a misspelling (intentional or not), often a single character substitution ("@ss", "sh1t", "nagger"). The minimum number of edits necessary to transform one word to another can be used as complement to dictionary-based approaches.

Bag-of-Words (BOW): is a model which represents a text inside a corpus, based on the word occurrences that appear in the text, when these words have been extracted from

[1] www.noswearing.com/dictionary, www.rsdb.org, www.hatebase.org

[2] https://en.wikipedia.org/wiki/List_of_ethnic_slurs

[3] https://en.wikipedia.org/wiki/List_of_LGBT_slang_terms

[4] https://en.wikipedia.org/wiki/List_of_disability-related_terms_with_negative_connotations

the whole corpus, instead of using a predefined set of words, as in the dictionaries procedure. After collecting all the words, the frequency of each one is used as a feature. BOW model is an orderless document representation, only the counts of words matter. As an alternative, the n-gram model can store spatial information.

N-grams: is a contiguous sequence of n items from a given sample of text. The items can be phonemes, syllables, letters, words or base pairs. As in the previous case, the n-grams are collected from a text corpus and the frequency for each n-gram inside each text is computed. The most common approach is to use word n-grams, though character n-gram features might be less sensitive to the spelling variation often faced when working with user generated content.

Term Frequency - Inverse Document Frequency (TF-IDF): is a numerical statistic applied to a BOW or n-gram model. It is intended to reflect how important a word is to a document within a collection or corpus. A high value of TF-IDF is reached when a word appears a lot in one document and does not appear in the others. In this way, the frequency of the term is offset by the number of documents in the corpus that contain the term, which help to identify important words having in mind the fact that some words appear more frequently by default (like stop-words).

Profanity Windows: is a mixture of dictionary and N-gram approaches. The aim is to check if a second person pronoun is followed by a profane word within the size of a window and then create a boolean feature if this condition is fulfilled. For example, considering "bastard" (a selection of words as in dictionary approach) as a profane word and a window size of 3 (number of terms to consider as in n-gram features), the following example complies the condition given: "You (1) stupid (2) bastard (3)", while the following one would not: "You (1) are (2) so (3) stupid (4), bastard (5)".

Part-of-Speech (POS): is a category of words which have similar grammatical properties. Words that are assigned to the same part of speech generally display similar behavior in terms of syntax (they play similar roles within the grammatical structure of sentences). Commonly listed parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, and sometimes numeral, article or determiner. These approaches consist in detecting the category of the word and extract a global feature for all texts and each category (frequency or ratio of appearance in the texts).

Lexical Syntactic Feature-based (LSF): it consists in capturing the grammatical dependencies within a sentence. Offensive words associated with another pejorative word or with a second pronoun becomes more offensive from users perception. For example, "you stupid" and "f***ing stupid" are much more insulting than "This game is stupid". So the

features obtained are pairs of words in which one word is offensive and depends on the other one in the sense we have seen in the examples.

Rule-Based Approaches: involves to identify and use a set of relational rules that collectively help a classifier to make decisions. Once the classifier is trained, to classify a new sentence it finds all rules whose words are contained in the sentence and then use an aggregation technique to choose a predominant class among those found by the rules.

Participant-vocabulary consistency: this method is used to characterize the tendency of each user to harass (bully score) or to be harassed (victim score), and the tendency of a key phrase to be indicative of harassment. In this problem, for each user, a bully score (b) and a victim score (v) are assigned. And for each feature, a feature-indicator (w) is associated that represents how much the feature is an indicator of a bullying interaction. Learning is then an optimization problem over parameter vectors b, v, and w.

Template Based Strategy: the basic idea is to build a corpus of words, and for each word in the corpus, collect K words that occur around. This gives information about the context. "W-1:go W0:back W1:to" is an example of template with K = 2 on the word "back".

Word Sense Disambiguation Techniques: this problem consists in identifying the sense of a word in the context of a sentence when it can have multiple meanings. This can be achieved by carrying out word clustering which consists of clustering words which are semantically similar and can thus bear a specific meaning. Latent Dirichlet Allocation (LDA) is a slight variant of the previous one which produces for each word a topic distribution indicating to which degree a word belongs to each topic instead of assigning each individual word to one particular cluster.

Typed Dependencies: is a representation which provides a simple description of the grammatical relationships in a sentence. Such relationships have the potential benefit that non-consecutive words bearing a (potentially long-distance) relationship can be captured in one feature. For instance, in "Jews are lower class pigs" a dependency tuple (pigs, jews) will denote the relation between the offensive term *pigs* and the hate target *Jews*.

Sentiments: hate speech may be considered as subjective content and relation between subjective content, sentiments, and emotions may appear. Hate speech is expected to have a negative polarity in most occasions, and using sentiment as a feature can be very helpful.

Knowledge-Based Features: detecting if a message is hateful or benign can be highly dependent on world knowledge, and it is therefore intuitive that the detection might benefit from including information on aspects not directly related to language. "Put on a wig and

lipstick and be who you really are" may not be categorized as some form of hate speech when only read in isolation, however, if this expression is directed towards a boy one may perceive it as an attack to the sexuality or gender identity of the boy being addressed. The basis of this model is to encode concepts that are connected by relations and stereotypes to form assertions, such as "a skirt is a form of female attire" or "lipstick is used by girls". This knowledge base allows computing the similarity of concepts of common knowledge with concepts expressed in user comments. Obviously, this approach only works for a very confined subtype of hate speech (i.e, anti-LGBT bullying). It could also be used for other types of hate speech but it would require domain specific assertions to be included and this would require a lot of manual work.

Word Embeddings: is the collective name for a set of language modeling and feature learning techniques in NLP where words or phrases are mapped to vectors of real numbers. These vectors are built in such a way that words with similar meaning have low distance in the vectorial space created and vice versa.

Meta-Information: information about a message is also a valuable source to hate speech detection. Having some background information about the user of a post may be very predictive. A user who is known to write hate speech messages (e.g. number of previous profanity words) may do so again. Knowing the gender of the user may also help because there have been seen that men are much more likely to post hate speech messages than women [98].

Multimodal information: Modern social media not only consists of text but also include images, video and audio content. This context outside a written user comment can be used as a predictive feature. In fact, it can be very helpful since, among hateful user posts illustrated by websites documenting representative cases of severe cyber hate[5], visual context plays a major role. This features can be employed based on image labels, shared media content, and pixel-level image features.

Other features: Other features used in this classification task were based on techniques such as **Named-Entity Recognition (NER)**, frequencies of **personal pronouns** in the first and second person, the presence of **emoticons** and **capital letters**. Characteristics of the message were also considered such as hashtags, mentions, retweets, URLs, number of tags, terms used in the tags, and link to multimedia content, such as image, video, or audio attached to the post. Figure 2.3 shows a classification diagram of all the features explained before.

[5]One example documenting disturbing cases of gender-based hate on Facebook: `www.womenactionmedia.org/examples-of-gender-based-hate-speech-on-facebook/-`

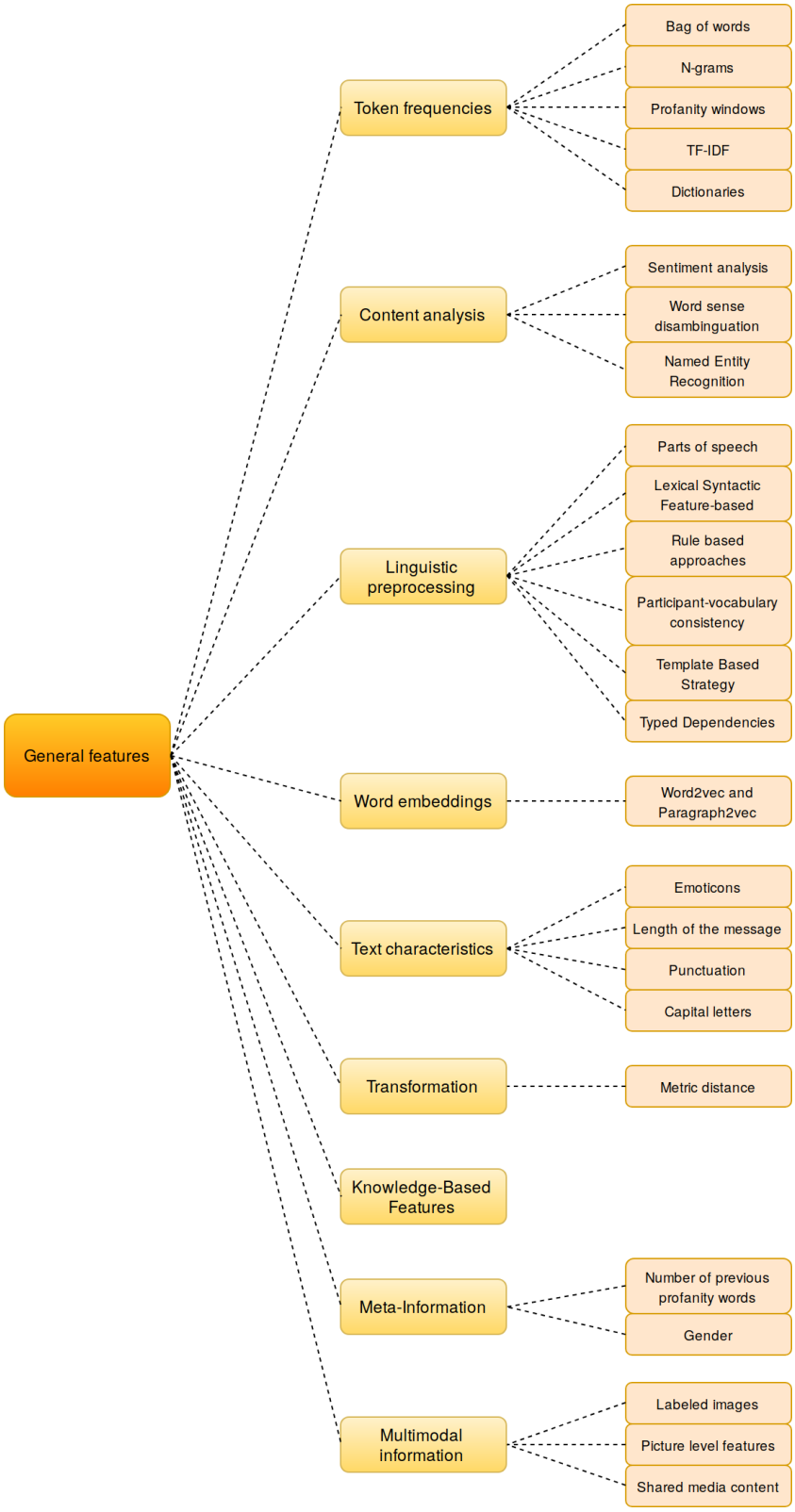

Figure 2.3: General features (Elaborated from Fortuna et al. [42])

2.2.2 Specific Hate Speech Detection Features

Complementary to the approaches commonly used in any other field, several specific features are being used to tackle the problem of hate speech automatic detection. We briefly present the approaches found.

Othering language: It is based on analyzing the contrast between different groups by looking at "Us versus Them". It describes "Our" characteristics as superior to "Theirs", which are inferiors. "Send them home" is an example of this type of language because it uses the opposition between "them" from "us" through the action of removing "them" to their "home".

Objectivity-Subjectivity of the language: In most cases, hate speech is present in subjective language, so detecting and erasing the objectives sentences can help in the analysis.

Focus on Particular Stereotypes: Hate speech often employs well-known stereotypes, especially in some types of hate speech such as racism or sexism. Given this, creating a language model for each stereotype is necessary for building a general model of hate speech.

Intersection of Oppression. It refers to the connection between several particular types of hate speech (For example, burka prohibition can be treated as Islamophobic, since this symbol is used by Muslims, or as sexist, because it is just worn by women). It can help to build a more general model of hate speech. For this purpose, it is necessary to feed the classifier with instances belonging to several categories (a balanced number of examples for each category is desirable) of hate speech.

2.3 State of the art Performance

In the collected papers, several metrics were computed to estimate the performance of the models. Precision, recall, and F-measure were the most common metrics and in some other studies, Accuracy and Area Under the Curve (AUC) were also considered. In Table 2.5, the results of the studies are presented in descending order of the F-measure value.

These results should be analyzed with some care because different datasets, definitions, and rules for hate speech identification are being used. In the table is shown a summary of the best results for each paper. It can be concluded that is it not clear which approaches perform better. On the one hand, the best results were achieved when deep learning was used either for feature extraction (word embeddings) or in the use as classification algorithms (DNN, CNN, etc.). On the other hand, using dictionaries as features seem not to work well. In light of the complexity of the hate speech domain, it could be argued that attending to word knowledge context instead of isolated keywords could help in the analysis. But due to what was commented previously, these results are not consistent. Comparative studies could help to understand this question.

Table 2.5: Evaluation Results from the Papers in the Metrics Accuracy (Acc), Precision (P), Recall (R), F-measure (F) and AUC, Respective Features, and algorithms Used (Table from Fortuna et al. [42])

Year	Acc	P	R	F	AUC	Features	Algorithms	Paper
2017	—	0.93	0.93	0.93	—	—	Logistic Regression, Random Forest, SVM, GBDT, DNN, CNN	[11]
2004	—	0.90	0.9	0.9	0.9	BOW, N-grams, POS	SVM	[44]
2017	—	0.91	0.9	0.9	—	TF-IDF, POS, sentiment hashtags, mentions, retweets, URLs, number of characters, words and syllabes	Logistic Regression, acsvm	[31]
2017	—	0.833	0.872	0.851	—	POS, sentiment analysis, wor2vec, CBOW, N-grams, text features	SVM, LSTM	[35]
2016	—	0.83	0.83	0.83	—	N-grams, length, punctuation, POS	Skip-bigram Model	[69]
2014	—	0.89	0.69	0.77	—	N-gram, typed dependencies	Random Forest, Decision Tree, SVM	[22]
2015	—	0.89	0.69	0.77	—	N-gram, typed dependencies	Random Forest, Decision Tree, SVM, Bayesian Logistic Regression, Ensemble	[20]
2016	—	0.72	0.77	0.73	—	User features	Logistic Regression	[98]
2016	—	0.79	0.59	0.68	—	BOW, dictionary, typed dependencies	SVM, Random Forest, Decision Tree	[21]
2015	—	0.65	0.64	0.65	—	Rule-based approach, sentiment analysis, typed dependencies	Non-supervised	[43]
2012	—	0.68	0.6	0.63	—	Template-based strategies, word sense, disambiguation	SVM	[95]
2016	—	0.49	0.43	0.46	0.63	Dictionaries	SVM	[90]
2015	—	—	—	—	0.8	paragraph2vec	Logistic Regression	[37]
2016	0.91	—	—	—	—	word2vec	Deep Learning	[100]
2013	0.76	—	—	—	—	N-grams	Naive Bayes	[56]
2016	—	0.73	0.86	—	—	Topic modelling, sentiment analysis, tone analysis, semantic analysis, contextual metadata	One-class Classifiers, Random Forest, Naive Bayes, Decision Trees	[4]
2004	—	0.93	0.87	—	—	BOW, N-grams, POS	SVM	[45]
2014	—	0.97	0.82	—	—	TF-IDF, N-grams, topic similarity, sentiment analysis	Naive Bayes	[59]

2.4 Datasets and Open Source Projects

In the majority of the papers, new different data was collected and annotated. However, only in a few studies data is made available for other researchers (label "own, available"), and only in one case an already published dataset is used ("published dataset"). The reduced number of datasets that are publicly shared is a relevant aspect in this area, making more difficult the comparison between different approaches. Figure 2.4 and Table 2.6 summarize the main information found. Despite the fact that some datasets and corpus for hate speech already exist, there are no established ones.

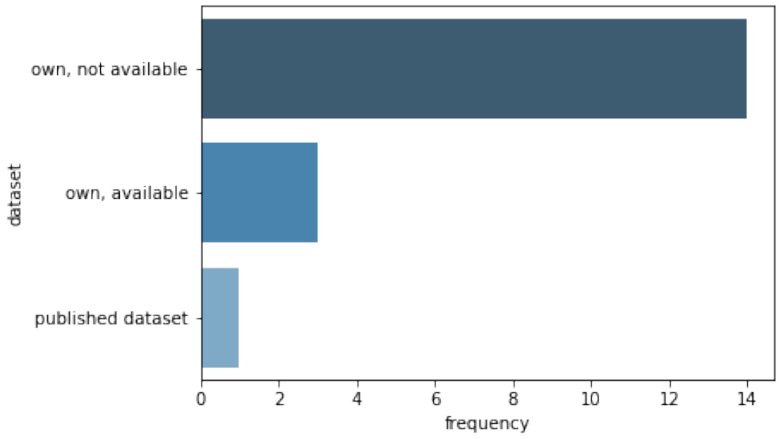

Figure 2.4: Dataset availability used in the papers (Elaborated from Fortuna et al. [42])

We had the goal to check if there are any projects available for automatic detection of hate speech that can be used or sources for annotated data. For this, we inspected GitHub using the expression "hate speech" in the available search engine. The search for projects in GitHub occurred in October 2018. A total of 180 repositories were found. We describe here the main conclusions from this research.

First, taking a first lookup we can say that the main approach found was to build models and classifying messages as hate speech. There is also a library for hate speech detection and projects that store datasets like the ones mentioned in Table 2.6. In what concerns to the programming language, Python is the most frequent language, followed by JavaScript and Java. Figure 2.5 presents the complete information about programming languages.

Regarding the datasets used by the projects, its source and language can be analyzed. Most projects do not provide any new data and there are also a few projects that use datasets already described in Table 2.6. Twitter is the social media most used although projects that use Facebook and Instagram have also been found. Projects that not use

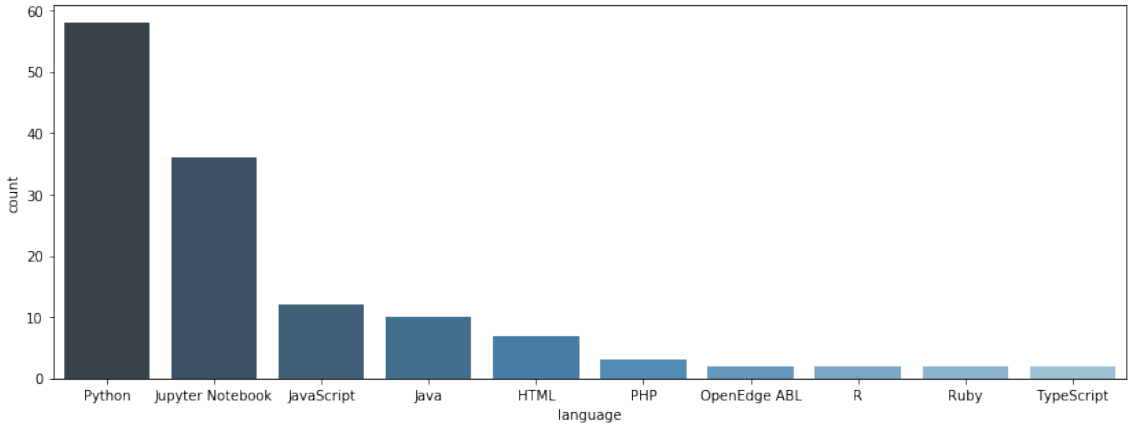

Figure 2.5: Programming languages used in the projects

social networks also appear, data collected from Wikipedia[6] or Reddit[7] is used instead. In most cases, projects use messages in English, but Italian, German and Indonesian are also used.

[6] https://www.wikipedia.org/
[7] https://www.reddit.com/

Table 2.6: Dataset and Corpus for Hate Speech Detection (Table from Fortuna et al. [42])

Name	Distribution	Year	Type	Number of instances	Classes Used	Language	Ref
Hate speech Twitter annotations	Github repository	2016	Dataset	16,914	Sexist, racist	English	[97]
Hate speech Stormfront annotations	Github repository	2018	Dataset	10,945	Hate speech, no hate speech	English	[71]
Hate speech Instagram annotations	Github repository	2018	Dataset	573	Hate speech, no hate speech	Indonesian	[78]
Indonesian Hate speech detection	Github repository	2018	Dataset	713	Hate speech, no hate speech	Indonesian	[7]
Hate speech identification	Available for the community	2015	Dataset	14,510	Offensive with hate speech, offensive with no hate speech, not offensive	English	[29]
Abusive language dataset	Not available	2016	Dataset	2,914	Hate speech, not offensive	English	[99]
German Hatespeech Refugees	Creative Commons Attribution-ShareAlike 3.0 Unported License	2016	Dataset	470	Hate speech, not offensive	German	[92]
Hate Speech and offensive language	Available for the community	2017	Corpus	24,783	Offensive with hate speech, offensive with no hate speech, not offensive	English	[30]
Italian Twitter Corpus of Hate Speech	Github repository	2017	Corpus	1,827	Hate speech, no hate speech	Italian	[39]
HateSpeech	Github repository	2017	Dataset	1,244	Hate speech, no hate speech	Portuguese	[33]

CHAPTER 3

Enabling Technologies

This chapter offers a brief review of the main technologies that have made possible the realization of this project. Throughout all this master thesis, Python was the programming language used for the implementation. In order to carry out most of the experiments, Jupyter notebooks have been used. Jupyter provides a rich framework for interactive computing, allowing the user to execute short code cells and outputting the results in the same window. The mentioned experiments need the previous tools to be enriched with some Python libraries which can be divided into data managing libraries, natural language processing libraries, and machine learning libraries. Following sections describe the fundamentals and usefulness of the different libraries that supported the development in this project. Finally, the chapter ends with a theoretical explanation about the fundamental concepts concerned with Machine Learning and Natural Language Processing fields.

3.1 Data Managing

This section encompasses libraries which ease data manipulation in Machine Learning related tasks.

3.1.1 Numpy

Numpy [1] [93] is the fundamental package for scientific computing with Python. Most of the remaining libraries use NumPy in their implementation. In relation to the usefulness in this project, Numpy was used due to its powerful high performance over multidimensional arrays and matrices, and the large collection of high-level mathematical functions to operate on these arrays. In addition, Numpy also contains tools for integrating C/C++ and Fortran code, along with broadcasting functions that transparently adapt dimensions between arrays.

3.1.2 Pandas

Pandas [2] [64] is an open source, BSD-licensed, Python data analysis library that provides fast, flexible, and expressive data structures. It is built on top of Numpy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. The two primary data structures of pandas are Series (1-dimensional) and DataFrames (2-dimensional).

- **Series** is a one-dimensional labeled object, capable of holding any data type (integers, strings, floating-point numbers, Python objects, etc.). It is similar to an array, a list, a dictionary or a column in a table. Every value in a Series object has an index.
- **DataFrames** is a two-dimensional labeled object with columns of potentially different types. It is similar to a database table, or a spreadsheet. It can be seen as a dictionary of Series that share the same index.

During the implementation, Pandas has been useful to structure datasets into unified arrangements as well as to apply operations on whole datasets. Here are some of the functions that Pandas implements:

[1] `http://www.numpy.org/`
[2] `https://pandas.pydata.org/`

- **DataFrame** object for data manipulation with integrated indexing.
- **Reading and writing data** in different formats: CSV and text files, Microsoft Excel, SQL databases and the fast HDF5 format.
- Intelligent **data alignment** and integrated handling of missing data
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large datasets.
- Columns can be inserted and deleted from data structures for **size mutability**
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on datasets.
- High performance **merging and joining** of datasets.

3.2 Natural Language Processing

The main information source employed in this project are text messages displayed in social networks. Consequently, NLP techniques acquire a role of utmost importance for the fulfillment of this thesis. Among the NLP tasks desired, text preprocessing, and feature extraction methods are the most valuable capabilities offered by the libraries described below.

3.2.1 NLTK

The Natural Language Toolkit (NLTK) [3] [60, 14] is the fundamental platform for building Python programs to work with human language data. It provides a suite of text processing tools for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. Here are some library highlights.

- It offers easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet [40].
- Lexical analysis: word and text tokenizer.
- n-gram and collocations.
- Part-of-Speech tagger.

[3] https://www.nltk.org/

- Tree model and text chunker.
- Named-Entity Recognition.

3.2.2 Gensim

Gensim [4] [79] is a free Python library designed to extract semantic topics from documents, as efficiently (computer-wise) and painlessly (human-wise) as possible. The algorithms in Gensim automatically discover the semantic structure of documents by examining statistical co-occurrence patterns within a corpus of training documents in an unsupervised way. Once these statistical patterns are found, any plain text documents (sentence, phrase, word...) can be succinctly expressed in the new semantic representation.

The essential algorithm used in this project is Word2Vec [66] which proposes two model architectures for computing continuous vector representations of words from very large corpus. These models are shallow, two layer neural networks are trained to reconstruct linguistic contexts of words. The resulting vector space, typically of several hundred dimensions, positions word vectors in the vector space such that words that share common contexts in the corpus are located in proximity to one another in the space.

Nevertheless, a requirement of these approaches is the use of large amounts of data and efficiently training neural networks in small datasets is still an open challenge. In light of such trend, it is not reasonable to train a model with the available hate speech wise data, pre-trained word vectors with general purpose collections billion words sized are used instead.

3.2.3 TextBlob

TextBlob [5] [61] is a Python (2 and 3) library for processing textual data. It is based on NLTK and Pattern [6] and plays nicely with both. It provides a simple API for dividing into common NLP tasks such as Part-of-Speech tagging, noun phrase extraction, sentiment analysis, classification, translation (powered by Google Translate), and more.

- Splitting text into words and sentences
- Word and phrase frequencies

[4] https://radimrehurek.com/gensim/
[5] https://textblob.readthedocs.io/en/dev/
[6] http://thepatternlibrary.com/

- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction
- WordNet integration

In the case of this project, the sentiment property from the toolbox was plenty exploited. Such property provides text polarity and subjectivity which can be directly included as features in our data processing pipeline. Initially was also thought to take into account the translation option but currently, it is failing due to deprecation issues.

3.3 Machine Learning

This section gives an overview of the libraries used for Machine Learning purposes. Such libraries are in charge of performing Machine Learning tasks, including classification taking matrix data as input which, in this case, has been obtained using the previously described technologies.

3.3.1 Scikit-learn

Scikit-learn [7] [73] is an open source, BSD-licensed, Python library providing simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy, and matplotlib. Scikit-learn implements a range of machine learning, preprocessing, cross-validation and visualization algorithms.

Scikit-learn can perform classification (identifying to which category an object belongs to), regression (predicting a continuous-valued attribute associated with an object), clustering (automatic grouping of similar objects into sets), dimensionality reduction (reducing the number of variables to consider), model selection (evaluating, validating, and selecting best hyper-parameters and models), and feature extraction.

This final work is focused on classification tasks, where the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where there is a

[7] `https://scikit-learn.org/stable/index.html`

limited number of categories and for each of the samples provided, the algorithm has to label it into the correct category or class.

Scikit also implements some evaluation metrics to measure how well a model performs. These functions compare a ground truth and a prediction, that is, the output of a fitted Machine Learning algorithm when faced with a test sample. Apart from the train/test approach, cross validation is also included into the Scikit-learn functionalities. In cross-validation, data is divided into k groups of samples, called folds, of equal sizes (if possible). The prediction function is learned using $k - 1$ folds, and the fold left out is used for test. This procedure is repeated k times until all folds are used as test.

3.3.2 Imbalanced-learn

Imbalanced-learn [57] [8] is an open-source Python toolbox aiming at providing a wide range of methods to tackle the curse of imbalanced datasets in Machine Learning. The implemented state-of-the-art methods can be categorized into four groups: (i) under-sampling, (ii) over-sampling, (iii) combination of over and under-sampling, and (iv) ensemble learning methods. The proposed implementation depends on Numpy, Scipy, and Scikit-learn and is distributed under MIT license.

The learning phase and the subsequent prediction of machine learning algorithms can be affected by the problem of data imbalance. The balancing issue corresponds to the difference between the number of samples in the different classes. In this situation, the decision function of the different learning models could be highly impacted, with an important imbalanced ratio, favoring the class with the greater number of samples, usually known as the majority class.

Since hate speech is a real, but limited phenomenon, is frequent facing with datasets where non-hateful samples are predominant. Therefore, we mostly exploit the under-sampling capabilities provided by the toolbox.

3.3.3 GSITK

GSITK [9] [9, 10] is a library developed by the GSI on top of Scikit-learn that eases the development process on NLP machine learning driven projects. It uses, Numpy, Pandas, NLTK, Gensim, and related libraries to easy development. GSITK manages datasets, features,

[8] https://imbalanced-learn.readthedocs.io/en/stable/
[9] https://github.com/gsi-upm/gsitk

classifiers, and evaluation techniques, so its scope is much larger than Machine Learning utilities. GSITK is divided into five main modules.

- **Datasets and DatasetManager.** These modules are responsible for managing datasets, allowing information loading in a quick manner. GSITK provides some datasets by default that can be directly employed, furthermore it enables to import your custom data by means of a Python module and a yaml file. Besides, as additional functionality, persistence is implemented in order to accelerate the reading of the dataset once it has been loaded before. All datasets are processed as a Pandas DataFrame.

- **Preprocess.** These modules provide tokenization and preprocessing methods for cleaning data before applying feature extraction and Machine Learning techniques. Among these preprocessing approaches, there is special attention under the one oriented towards the Twitter domain. User mentions normalization and flagging of hashtags, Universal Resource Locations (URLs), and all caps words are some of the principal utilities supported.

- **Features.** This module has the aim to extract features on top of Scikit-learn algorithms input format. Most of the components are built following the Scikit-learn API, permitting them being included as a part of a pipeline. Moreover, it contains feature extraction methods for sentiment analysis, as well as some deep learning capabilities, such us Word2Vec features.

- **Evaluation.** This module facilitates addressing exhaustive evaluations. GSITK evaluation combines one or more pipelines with one or more datasets. It computes standard evaluation metrics, including accuracy, precision, recall, and f-score for each pipeline-dataset combination with the aim of establishing comparison between different systems.

This project extensively exploits the preprocessing functions in order to make a Twitter-oriented preprocessing, as well as the features module to use word embeddings and similarity features.

3.4 Hatebase API

Hatebase [10] is a software platform built to help organizations and online communities detect, monitor and quarantine hate speech. Their natural language engine, Hatebrain, performs linguistic analysis on public conversations to derive a probability of hateful context. This analysis is based on a broad multilingual vocabulary based on nationality, ethnicity, religion, gender, sexual discrimination, disability, and class to monitor incidents of hate speech across 200+ countries. The regionalized vocabulary makes monitoring trends in hate speech usage and correlating with other datasets affordable to perform. All the Hatebase process is presented in Figure 3.1.

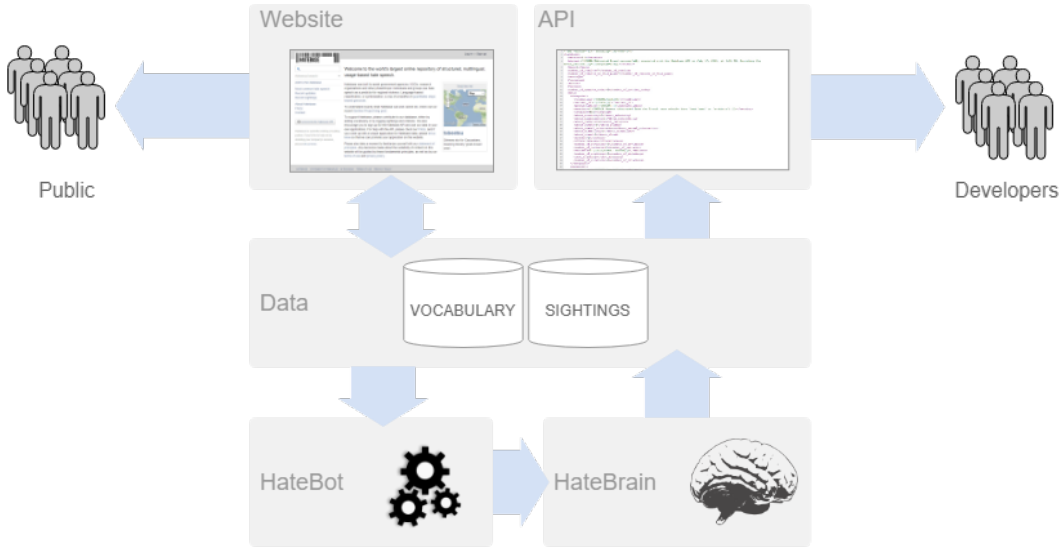

Figure 3.1: Hatebase community

All data is made available through the Hatebase web interface and API [11]. Supported queries include data on vocabulary and sightings. The read-only API returns XML or JSON formatted responses. As shown in Figure 3.2, querying the Hatebase API is a two-step process. The first step is an authentication handshake in which a one-hour valid token is received. The second step is the actual query itself, in which the authentication token is required. Specifically, in this project, the /get_vocabulary endpoint is consumed. This function allows users to download Hatebase's lexicon of multilingual hate speech filtering by language, nationality, ethnicity, religion, gender, sexual orientation, disability, and class.

[10] https://hatebase.org
[11] https://github.com/hatebase/Hatebase-API-Docs

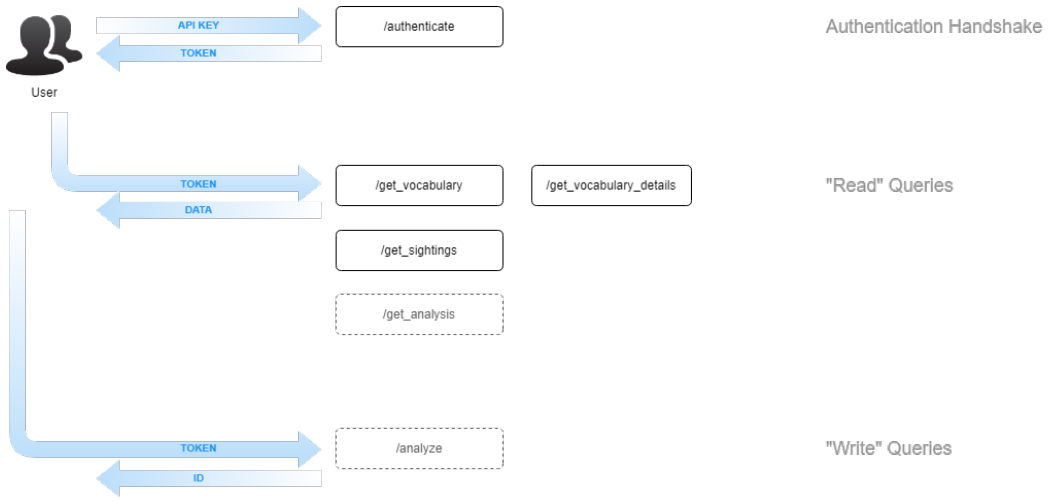

Figure 3.2: Hatebase API

3.5 Machine Learning Fundamentals

During the following sections, we review some basic notions about Machine Learning and Natural Language Processing technologies that will be used later in the project. In first place, Machine Learning theory is explained and then, in Section 3.6 the chapter ends with an NLP explanation.

Machine Learning is the scientific study of algorithms and statistical models that make possible computers have the ability to learn. More concretely, given a set of data Machine Learning goal is to obtain a computer program able to generalize behaviour relying on the data seen.

Machine Learning tasks are classified into several broad categories. This thesis focuses on supervised learning, in which the learning procedure consists of learning a model giving the algorithm a list of examples with the associated desired output and then, when faced with new input data responses with the most appropriate value what has been learned with the examples given.

Classification and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values, which is the problem we face in this study. Given a text example our system outputs a categorized value regarding the text hatefulness. Therefore, the values are restricted to presence of hate speech and absence of hate speech.

Classification models can be validated by accuracy estimation techniques like the holdout method, which splits the data in a training and test set and evaluates a model trained over

the training set with the test set. In comparison, cross-validation strategy divides the data into k folds, and iterates over these folds, taking in each iteration one of the sets as test data and taking the remaining k-1 sets as training data. Operating like this, every data sample acquires the train and test role. Finally, the estimation of the results is averaged in all tests to obtain the level of success of our model.

A few paragraphs above, we have said that with Machine Learning we are seeking to generalize. That is why when facing a Machine Learning problem we have to take care of avoiding overfitting issue. Overfitting is the effect of adapting too much a system to a set of data being unable to fit well on another set of data, generating so, a system with generalization lacking. Figure 3.3 illustrates an example of such problem in two dimensions comparing a pair of decision lines. The green line represents an overfitted model and the black line represents a regularized model. While the green line best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data, compared to the black line.

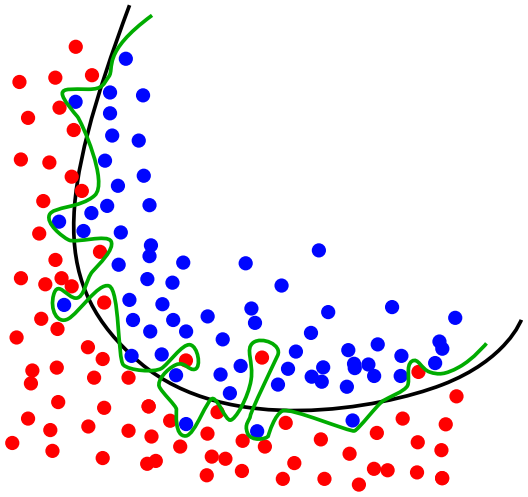

Figure 3.3: Overfitted versus regularized system

Another problem in Machine Learning is the selection of the best algorithm to perform a certain taska. This causes that we have to experiment with different models to select the algorithm best adapted to the problem addressed. In this project, we have evaluated the performance of three different types of algorithms: Logistic Regression, Support Vector Machines (SVM) with linear kernel, and Random Forest.

3.5.1 Logistic Regression

Regression name is a bit confusing since this algorithm is used for categorical data. Regression here refers to the internal operation of the model which runs a linear regression over the input training data. Following, we will see a shallow understandable explanation of the algorithm.

Let $\vec{x} = (x_1, x_2, ...x_M)$ the feature vector of a document and $y \in \{0, 1\}$ the label to predict. The objective is to learn a weighting vector $\vec{w} = (w_1, w_2, ..., w_M)$ which multiplied by the input vector $\vec{x}$ plus a bias term returns a value z that compared with a decision function (if $z >$ threshold y will be 1 and vice versa) gives us $y = 1$ if the document was categorized with the label $y = 1$ and $y = 0$ if the document was categorized as $y = 0$. So we have z like this:

$$z = \vec{w} \cdot \vec{x} + bias$$

Really we are not applying the threshold using z directly, however, an intermediate function is used. This is the logistic (the reason for naming it "logistic") or sigmoid function (Figure 3.4).

$$h(z) = \frac{1}{1 + e^{-z}}$$

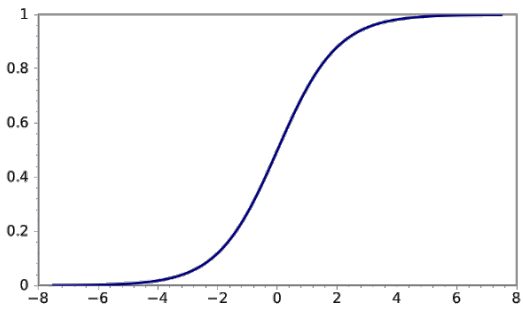

Figure 3.4: Logistic function

This way, we see if $h(z) >$ threshold (typically will be threshold=0.5), the final decision is $y = 1$ and backwards. Finally, weights and bias are learned to minimize the error between the output and the expected value.

3.5.2 Support Vector Machines

A support vector machine [28] is an algorithm that constructs a hyperplane with the aim of separating each class through said hyperplane. Again, we have $\vec{x}$, $\vec{y}$, and $\vec{w}$ in the same way than in Logistic Regression and the hyperplane can be written as the set of points $\vec{x}$ satisfying

$$\vec{w} \cdot \vec{x} - b = 0$$

In this case, $\vec{w}$ is defined so that distance between the hyperplane and the nearest point $\vec{x}$ from either class is maximized (Figure 3.5).

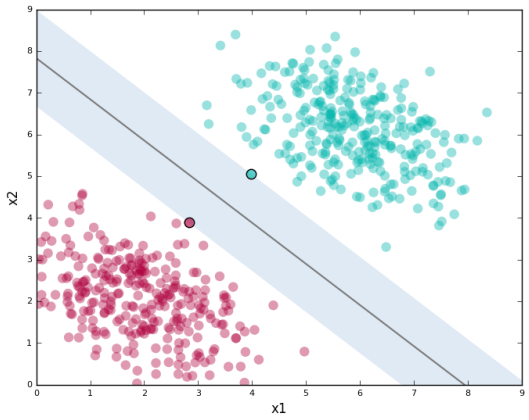

Figure 3.5: Maximum margin hyperplane

If classes are not linearly separable, the algorithms transform the input vectors to a higher dimensional space where data is linearly separable. This transformation is known as the kernel-trick (Figure 3.6).

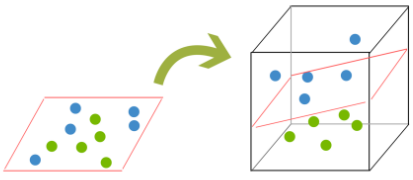

Figure 3.6: Kernel-trick

In order to avoid overfitting in the input data, SVM defines the Cost parameter (C) to allow misclassification of some instances of the training data (Figure 3.7).

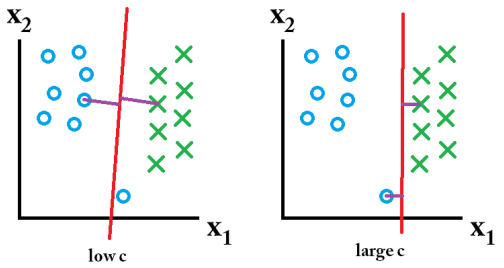

Figure 3.7: Effect of changing C parameter

3.5.3 Random Forest

Random Forest [18] is an ensemble learning method which uses a Decision Tree as base estimator. Various trees are trained and at prediction time the contribution of each tree is taken into account for the final decision. In order to understand Random Forest, first, we need to understand Decision Trees.

Decision Tree constructs a hierarchical model structure where each node represents a decision based on features values and final leave nodes represent the class selected relying on previous decisions. Figure 3.8 shows an example with Titanic data.

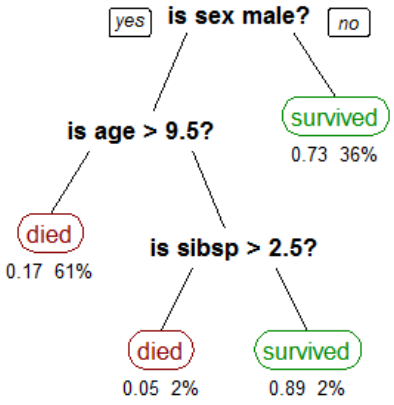

Figure 3.8: Tree learned on titanic data

Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring “best”. These generally measure the homogeneity of the target variable within the subsets. Gini impurity and information gain are the most famous approaches.

The technique used to construct the trees produce that they overfit [12] their training sets. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing variance. For that purpose, Random Forests work applying the general technique of bootstrap aggregating. Given a training set of feature vectors $X = \vec{x_1}, \vec{x_2}, .., \vec{x_N}$ with their respective annotations $\vec{y} = y_1, y_2, ..., y_N$ bagging repeatedly (B times) selects a random sample of X and features to construct a tree in each iteration. After training, predictions for unseen samples $\vec{x}$' can be made by averaging the predictions from all the individual trees on $\vec{x}$'.

3.5.4 Artificial Neural Networks

Neuron models [63] are another type of algorithms important for the thesis development, though they are not used to make predictions. Instead, they are used to extract features from text input, so it is not unwell include a description of the basics of Neural Networks.

3.5.4.1 Perceptron

The simplest model is a single neuron also known as the perceptron. The perceptron receives an input vector $\vec{x}$ and outputs a single binary value. The mapping function from input to output is the unitary step function (Heaviside step function) with the following condition.

$$f(x) = \begin{cases} 1 \quad if \quad \vec{w} \cdot \vec{x} + b > 0 \\ \\ 0 \quad if \quad \vec{w} \cdot \vec{x} + b < 0 \end{cases} \tag{3.1}$$

Where $\vec{w}$ is a vector of real-valued weights and b is the bias. Weights and bias must be learned in a way input vectors are mapped suitably to its corresponding output. The overall perceptron architecture is presented in Figure 3.9 [13].

In the context of neural networks, the mapping function is known as the activation function. Using the step function as the activation function presents several problems. First, it can only learn linear relationships between variables and target since all the operations carried out are linear functions. Besides, it is very abrupt and the output values are specific,

[12] Real implementations of Decision Trees provide parameters to avoid overfitting controlling the learning process, but generally, this assumption is considered to be true

[13] Usually the bias is modeled as a component of the weights vector, being the associated component from the input vector a one constant.

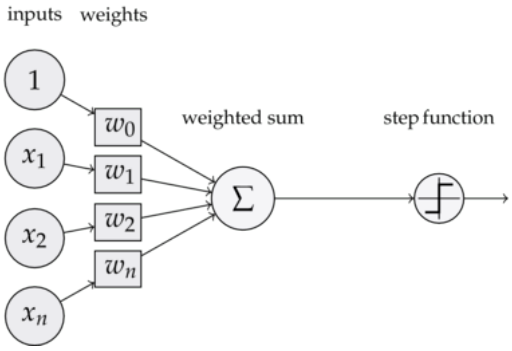

Figure 3.9: Perceptron architecture

so they are not real numbers inside a range which cannot be interpreted as probabilities. For fixing this problem, the sigmoid function (Figure 3.4) is added at the end of the perceptron. This model is equivalent to the logistic regression.

3.5.4.2 Multi-Layer Perceptron

Making use of the individual neuron or perceptron units which were described previously, it is possible to create more complex architectures. It is possible to combine lots of neurons arranging them in layers which take as input the output of the neurons in the previous layer. The naming of the layers is the following. Input layer is the one composed of the data which enters the network, output layer is the final layer where final predictions are yielded, and finally, intermediate neurons are part of the called hidden layers. A graphic representation of a simple multi-layer perceptron is shown in Figure 3.10.

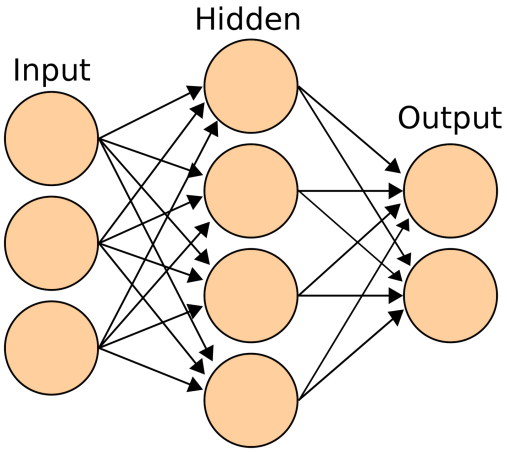

Figure 3.10: Multi-Layer Perceptron

Apart from Sigmoid function, another activation functions can be applied to neural

networks. Two of the most used ones are tanh (hyperbolic tangent) and ReLU (Rectified Linear Unit). A comparison is presented in Figure 3.11.

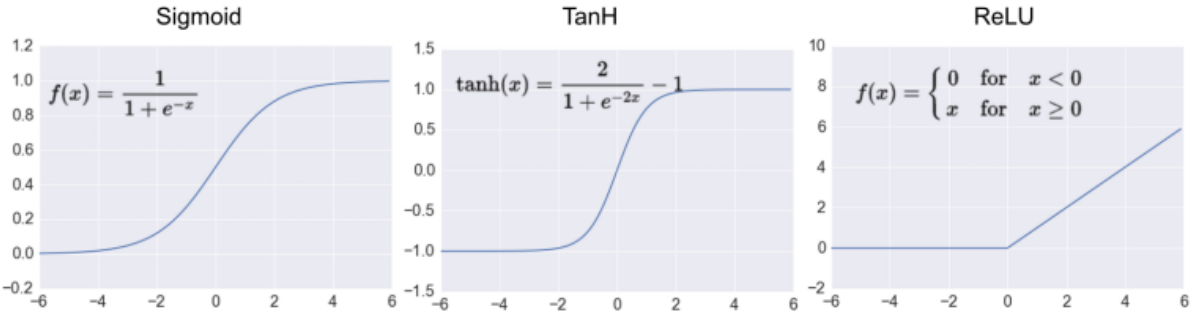

Figure 3.11: Comparison between activation functions

The process of learning the optimal weights of each neuron in the network is known as training. Training goal is to minimize the cost function. The cost function is an error measure which states how far the predictions are from the real values using the current weighting vectors. This error is computed in the output layer and by means of backpropagation neurons from the previous layers can know their contribution to the final output error. Backpropagation works using the chain rule deducting the weighting factors when propagating from the output to the input. Once all neurons know how they contribute to the error, they can update their weights to minimize such error. Neural networks follow the Stochastic Gradient Descent [17] minimization algorithm. It is applied in an iterative way, learning new weights in each iteration, until convergence is reached and the model achieves a decent performance over the training data.

3.6 NLP Fundamentals

Natural Language Processing (NLP) is a set of engineering techniques and methods concerned with the interactions between computers and human languages, in particular automated textual and linguistic analysis, generation, representation, and acquisition. There exists a broad variety of researched areas in NLP. These areas goes from machine translation [54] to question answering passing through Natural Language Understanding (NLU) [8] or Natural Language Generation (NLG) [80]. During the realization of this project, the task addressed is text classification.

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in NLP with broad applications such as

sentiment analysis [10] or spam detection [94]. In order to carry out a text classification based on machine learning techniques, the first step to tackle is feature extraction: a method to transform each text into a numerical representation in the form of a vector. Some approaches are shallowly explained in Section 2.2, but in the following sections, a more deeply explanation is given to the methods used in the thesis.

3.6.1 BOW

The BOW model is one of the most popular representation methods for text categorization. The key idea is to transform each document within a corpus to a vector of terms that occur in the corpus. In the identification of terms, a term can be represented by simple words or characters (1-gram) or a composed version of the previous terms (1,2,...,n-gram) that occur in the documents. Then, each term is used as an attribute of the text set represented in the attribute-value form resulting in a representation similar to that shown in Table 3.1.

Table 3.1: BOW representation

	t_1	t_2	...	t_m
d_1	a_{11}	a_{12}	...	a_{1m}
d_2	a_{21}	a_{22}	...	a_{2m}
...	...	...	...	...
d_3	a_{31}	a_{32}	...	a_{3m}

In Table 3.1 n documents are represented, and each document vector is composed by m terms $(a_{i1}, a_{i_2}, ...a_{im})$. The vector size is equal to the vocabulary size found by analyzing the corpus. An a_{ij} value refers to the frequency of appearance of term j in the document i.

The following models a pair of texts using Bag-of-Words:

```
(1) John likes to watch movies. Mary likes movies too.
(2) John also likes to watch football games.
```

Based on these two text documents, a list of words for each document is constructed

```
"John","likes","to","watch","movies","Mary","likes","movies","too"
"John","also","likes","to","watch","football","games"
```

Representing each Bag-of-Words as a JSON object:

```
(1) {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1}
(2) {"John":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};
```

Resulting in a document vector like this:

```
(1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
(2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
```

In order to normalize the output vectors, binary values can be used, so the value 1 means the presence of the term j in the document i, and the value 0 means the term is absent.

Bag-of-word model is an orderless document representation (only the counts of words matter). For instance, in the above example "John likes to watch movies. Mary likes movies too", the Bag-of-Words representation will not reveal that the verb "likes" always follows a person's name in this text. As an alternative, the n-gram model can store this spatial information. Applying to the same example above, a bigram model will parse the text into the following units and store the term frequency of each unit as before.

```
"John likes",
"likes to",
"to watch",
"watch movies",
"Mary likes",
"likes movies",
"movies too",
```

3.6.2 TF-IDF

A better practice when working with statistical measures is to take into account the frequency a term appears in a document and also the frequency this term is found in other documents. That is the basis of TF-IDF [5] representations whose main goal is to obtain the word importance used in a set of documents. It works in a similar way than BOW model, converting a text input as a vector representation, but in this case being its components the importance of each term.

The TF-IDF value increases proportionally to the number of times a word appears in the document, but this value is contrasted by the frequency of the word in the corpus, which helps us to adjust the importance of a word according to the fact that some words appear more frequently in general.

The first step, again, is creating a vocabulary from the corpus creating a dictionary

of terms that appear on it. Following, the term frequency is computed to represent each text in the vector space (as done in BOW model). In this way, each document within the corpus is represented by a vector with zeros on the terms that did not appear and the number of occurrences on the terms that did appear on it. Once we have the vectors, it is important normalizing them because the importance of a word appearing a number of times depends on the text length. Now we are in a good position to calculate the inverse document frequency.

The inverse document is defined as:$\log \frac{|D|}{1+|d:t\in d|}$ being $1 + |d : t \in d|$ the number of documents where the term t appears and $|D|$ the number of documents in the corpus. There are many versions of this function since it tries to get the impact of a term inside a corpus and in this case it smooths it by computing it into a logarithmic scale.

When we have both calculated term frequency and inverse document frequency, we multiply both values, and we obtain the TF-IDF value. So finally, a high TF-IDF is reached with a word with a high frequency in a document and with a low frequency in terms of the whole set.

3.6.3 LDA

One of the problems when working with Bag-of-Words and TF-IDF is the feature size, cause the resulting vectors are sparse (most of their components are zero and a few have non null values). More sophisticated approaches such us LDA [15] try to find short descriptions of the members of a collection that enable efficient processing of large collections while preserving the essential statistical relationships that are useful.

LDA is a generative probabilistic model for collections of discrete data such as text corpora. It is also a topic model that is used for discovering abstract (latent) topics from a collection of documents in an unsupervised way. The algorithm assumes a collection of k topics, each topic following a multinomial distribution over the vocabulary available in the corpus. Then, LDA represents each document as a mixture of those k topics in basis of the word probabilities of pertaining to a certain topic.

In more detail, LDA works with the following generative process for each document. First, draw a topic mixture for the document according to a Dirichlet distribution over a fixed set of k topics. Following, generate each word in the document by:

1. Selecting a topic according to the previous mixture drawn
2. Finally, using the topic to generate the word itself in line with the topic's multinomial

distribution

Assuming this generative model for a collection of documents, LDA then tries to backtrack from the documents to find a set of topics that are likely to have generated the collection. One way of doing this is known as collapsed Gibbs sampling. It consists in randomly assigning to each word for each document one of the k topics. Notice that this random assignment already gives you both topic representations of all the documents and word distributions of all the topics (albeit not very good ones). In order to improve previous allocation, for each word w in each document d we compute two things 1) p(topic t | document d) = the proportion of words in document d that are currently assigned to topic t, and 2) p(word w | topic t) = the proportion of assignments to topic t over all documents that come from this word w. Reassign w a new topic, where you choose topic t with probability p(topic t | document d) $\cdot$ p(word w | topic t) (according to our generative model, this is essentially the probability that topic t generated word w, so it makes sense that we resample the current word's topic with this probability). In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated. After repeating the previous step a large number of times, we'll eventually reach a roughly steady state where the assignments will be pretty good. Finally, these assignments can be used to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated to each topic (by counting the proportion of words assigned to each topic overall).

3.6.4 Word Embeddings

One essential technique used extensively in this project is creating the so-called Word Embeddings [91], which are mappings from words to vectors. Any operation that fulfills previous statement is called word embeddings. The simplest way to carry out this task can be done by one-hot encoding. With one-hot encoding, each word has a unique vector formed by zeros except a one in the position that represents that word in the vocabulary. Applying this method results in a vector size equal to vocabulary size, which may have extremely high dimensions when working with large sets of data. The same thing occurs when using TF-IDF vectors, so in order to solve this problem, dense (instead of sparse) word representations have recently appeared. Dense representations are the ones considered as real word embeddings and have become well-known during the last few years.

Dense representations are able to capture some of the regularities from the human lan-

guage, such as gender relationships, verb tenses, and countries and their respective capitals as shown in Figure 3.12. Next sections show some intuitions of how these vectors are extracted.

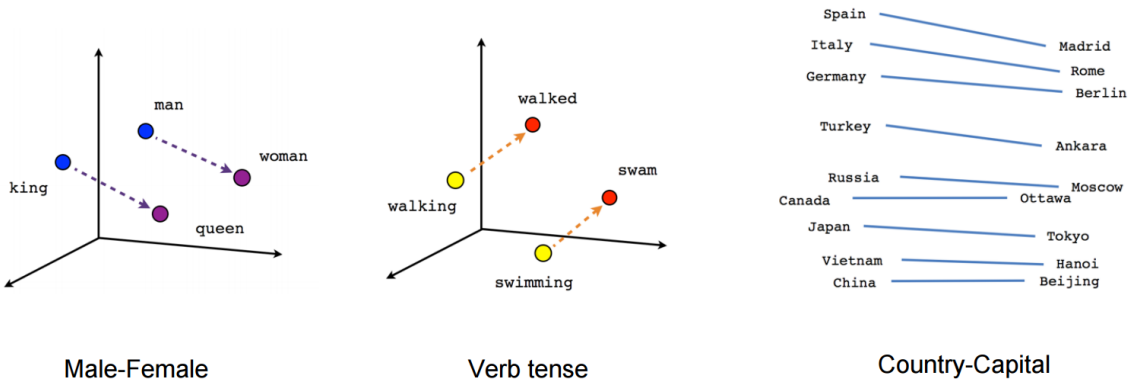

Figure 3.12: Word Embeddings patterns

3.6.4.1 Word2Vec

There are several approaches which can obtain these vectors with such properties, and almost all of them rely on unsupervised techniques. One of the most popular models is Wor2Vec [66], which learns the representations using a neural network architecture to solve a predictive task. Such neural network has a simple architecture consisting of an input layer, a projection layer (hidden layer), and an output layer. There are two variations of this structure: Skip-gram and Continuous Bag of Words (CBOW).

Skip-gram model topology is presented in Figure 3.13, being its aim predicting the context of a word (words before and after that word) given such word. The input layer consists of the word one-hot encoded, being V the vocabulary size. The hidden layer has N neurons with $N < V$ and the output layer are the one-hot encoded context words, with C as the context window size. Finally, the weights learned between the input and the hidden layer give the word vector of size N.

CBOW model inverts the topology of the skip-gram model (Figure 3.14), being its aim predicting a word given its context. In this case, the input layer consists of the one-hot encoded context words and the output layer is the word predicted also in one-hot encoded format. After training the network, the word vectors are provided by the weight matrix between the hidden layer and the output layer.

Although the training process relies on a neural network based supervised prediction model, the real training results are the vector representation of words instead of the neu-

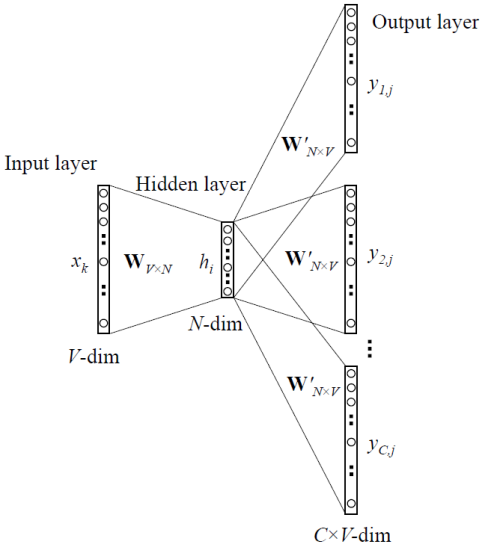

Figure 3.13: Skip-gram model

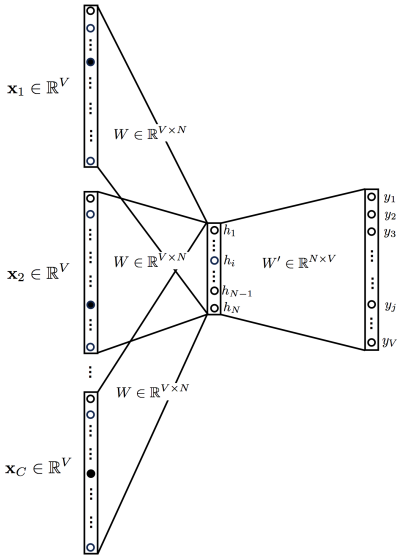

Figure 3.14: CBOW model

ral network prediction model. Because of such idea, the training of word embedding is unsupervised and can be applied in various textual corpus without labeled datasets.

3.6.4.2 GloVe

Global Vectors for Word Representation (GloVe) [74] is another approach to learn word representations. GloVe vectors are obtained through defining a function whose target is

obtaining ratio probabilities (Equation 3.2) of co-occurrence based on a word co-occurrence matrix. In order to better understand, we first must establish some notation. Let the matrix of word-word co-occurrence counts be denoted by X, whose entries X_{ij} tabulate the number of times word j occurs in the context of word i. Let $Xi = \sum_k X_{ik}$ be the number of times any word appears in the context of word i. Finally, let $P_ij = P(j \mid i) = X_{ij}/X_i$ be the probability that word j appears in the context of word i. Therefore, starting from equation 3.2 and deriving some calculus the equation 3.3 is obtained which gives us the word representations.

$$F(w_i, w_j, \vec{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{3.2}$$

$$w_i^T \cdot \vec{w}_k + b_i + \vec{b}_k = log(X_{ij}) \tag{3.3}$$

There is one problem with the equation above: it weights all co-occurrences equally. Unfortunately, not all co-occurrences have the same quality of information. Co-occurrences that are infrequent will tend to be noisy and unreliable, so we want to weight frequent co-occurrences more heavily. On the other hand, we don't want co-occurrences like "it is" dominating the loss, so we don't want to weight too heavily based on frequency. Through experimentation, GloVe authors found the following weighting function to perform relatively well (Figure 3.15):

$$weight(x) = min(1, (x/x_{max})^{\frac{3}{4}}) \tag{3.4}$$

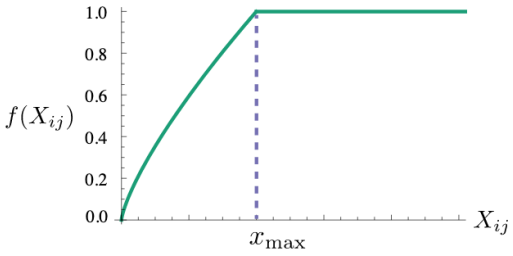

Figure 3.15: Weighting function

3.6.5 SIMON

SIMilarity-based sentiment projectiON (SIMON) [10] is a method to include external specific information together with the semantics given by a word embedding model. Arguably,

it is accepted that pre-trained word vectors do not enclose specific domain information, as the training process was carried out with general purposes with no target goals. In order to include specific information, additional information must be included in the feature extraction process. In this way, semantic similarity features exploit the aforementioned word embeddings regularities using a selection of words, namely, a lexicon vocabulary which contains the specific information (sentiment, subjectivity, etc.) we want to include in the analysis.

More concretely, this work proposes the representation of a certain word, that may be outside the lexicon vocabulary, by a projection to a set of domain-specific words extracted from a domain lexicon. Such projection is computed using the semantic similarity between words, which can be computed by means of a word embedding model. This way, the word embedding model can be exploited via the cosine similarity.

Formally said, cosine similarity is a measure of similarity between two vectors of a vectorial space. Given two word vectors, $\vec{w_i}$ and $\vec{w_j}$, the cosine similarity is represented using a dot product and a magnitude as

$$similarity = \frac{\vec{w_i} \cdot \vec{w_j}}{\| \vec{w_i} \| \cdot \| \vec{w_j} \|} \tag{3.5}$$

As stated, in general, word embeddings contain semantic and syntactic information, therefore, applying cosine similarity to a word embedding model results in a semantic similarity measure of words. Such idea is illustrated in Figure 3.16

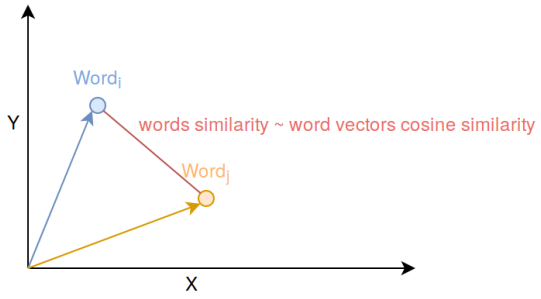

Figure 3.16: Cosine similarity in a Word Embeddings model

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

The process to generate the features is as follows. The method considers a selection of words S that constitutes a lexicon vocabulary to which the input documents are projected.

Given a text document (e.g., tweet), a similarity value between the input word vectors of that document and each of the words in S is computed. After iterating over all input words and all lexicon words, a matrix $I \times |S|$ is obtained, where I is the number of input words in a particular document and $|S|$ the lexicon vocabulary size. Following, the maximum pooling function is applied column-wise, obtaining the semantic similarity feature vector of dimensionality $|S|$. This process is presented in Figure 3.17.

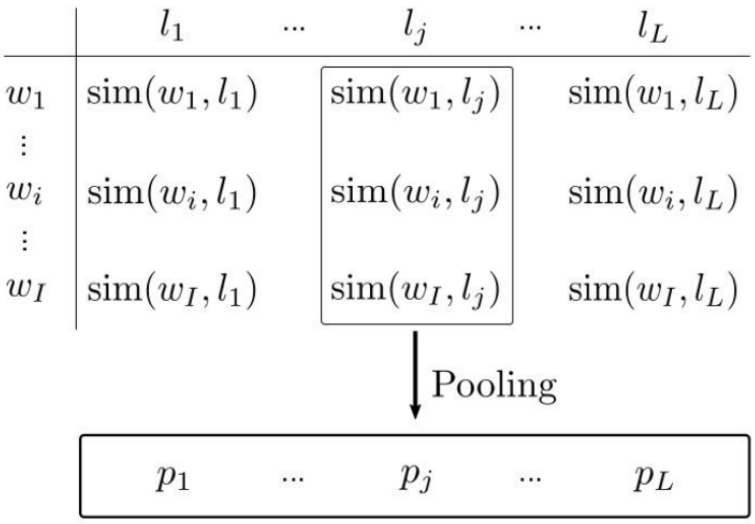

Figure 3.17: Similarity features computation

Additionally, SIMON incorporates a selection of lexicon words in two steps. First selection step is carried out before computing similarities and consists of filtering words by frequency of appearance in the dataset. The second step is done after the similarity computation and makes use of ANalysis Of VAriance (ANOVA) statistical test between features (which correspond to the selected words from the first step) and labels. The ANOVA or F-value measures for each feature its informativeness regarding the classification task (label to predict) of a certain dataset. Scikit-learn implements the one-way ANOVA test which can be calculated as:

$$F = \frac{\text{between-group variability}}{\text{within-group variability}} \tag{3.6}$$

$$\text{between-group variability} = N \cdot ((\bar{Y} - \bar{S})^2 + (\bar{X} - \bar{S})^2) \tag{3.7}$$

$$\text{within-group variability} = \frac{\sum_{i=1}^{N}((Y_i - \bar{Y})^2 + (X_i - \bar{X})^2)}{N - 2} \tag{3.8}$$

Being X the vector of the values of a specific feature, Y the vector of the values of the

target value (what we want to predict), $\bar{X}$ amd $\bar{Y}$ the respective mean of each vector, $\bar{S}$ the global mean (i.e. $\bar{S} = \frac{\bar{X}+\bar{Y}}{2}$), and N the total number of documents.

CHAPTER 4

Participation at SemEval-2019 Task 5

Such big impact has generated the hate speech phenomenon during the last few years that among the proposed tasks organized in SemEval-2019 hate speech appears as principal subject in one of those tasks. This chapter describes the GSI-UPM system for SemEval-2019 Task 5, which tackles multilingual detection of hate speech on Twitter. The main contribution of the participation is the use of a method based on word embeddings and semantic similarity combined with traditional paradigms, such us n-grams, TF-IDF, and POS. This combination of several features is finetuned through ablation tests, demonstrating the usefulness of different features. While our approach outperforms baseline classifiers on different sub-tasks, the best of our submitted runs reached the 5th position on the Spanish sub-task A.

4.1 Introduction

SemEval (Semantic Evaluation) is an ongoing series of evaluations of computational semantic analysis systems; it evolved from the Senseval word-sense evaluation series. The evaluations are intended to explore the nature of meaning in language. These exercises have evolved to articulate more of the dimensions that are involved in our use of language. They have adapted to investigate the interrelationships among the elements in a sentence (e.g., semantic role labeling), relations between sentences (e.g., coreference), and the nature of what we are saying (semantic relations and sentiment analysis).

The Task 5 from SemEval-2019 [12] focus the analysis towards the hate speech problem. The proposed task consists in Hate Speech detection in Twitter but featured by two specific different targets, immigrants and women, from a multilingual perspective, for Spanish and English tweets. The task will be articulated around two related subtasks for each of the involved languages: a basic task about Hate Speech, and another one where fine-grained features of hateful contents will be investigated in order to understand how existing approaches may deal with the identification of especially dangerous forms of hate, i.e. those where the incitement is against an individual rather than against a group of people, and where an aggressive behavior of the author can be identified as a prominent feature of the expression of hate. Participants will be asked to identify, on the one hand, if the target of hate is a single human or a group of persons, on the other hand, if the message author intends to be aggressive, harmful, or even to incite, in various forms, to violent acts against the target.

- **TASK A - Hate Speech Detection against Immigrants and Women:** a two-class (or binary) classification where systems have to predict whether a tweet in English or in Spanish with a given target (women or immigrants) is hateful or not hateful.
- **TASK B - Aggressive behavior and Target Classification:** where systems are asked first to classify hateful tweets for English and Spanish (e.g., tweets where Hate Speech against women or immigrants has been identified) as aggressive or not aggressive, and second to identify the target harassed as individual or generic (i.e. single human or group).

According to the FBI hate crime statistics [1], sexism and racism victims increased during 2017. For this reason, participating in SemEval-2019 Task 5 is such an interesting challenge

[1] https://ucr.fbi.gov/hate-crime/

and is supposed a good first contact with the hate speech domain. The system proposed relies on a supervised classifier using different text features combined with several strategies with the aim of finding an optimal performance. The remainder of this chapter is structured as follows. After this introductory section, Section 4.2 presents the data and its distribution. Following, the proposed evaluation metrics are described in Section 4.3. Then, Section 4.4 presents the proposed approach to afford the suggested problem, and finally, Section 4.5 concludes the chapter showing the experimental results and a final discussion.

4.2 Data

All data for the competition was collected from Twitter and manually annotated mainly via the Figure8 [2] crowdsourcing platform. Datasets were specially released for the competition according to the languages and targets involved. More specifically, it was organized in two datasets, both containing tweets about hate against women and immigrants, in English and Spanish, respectively. Each dataset was partitioned into train, development, and test sets. Data distribution for Task A is presented for each language in Table 4.1.

Table 4.1: Data distribution in Task A

Split	**HS**	**no HS**	**Total**
English			
Train	3,783 (42%)	5,217 (58%)	9,000
Development	427(43%)	573 (58%)	1,000
Test	1252 (42%)	1719 (58%)	2,971
Spanish			
Train	1,838 (41%)	2,631 (59%)	4,469
Development	222 (44%)	278 (56%)	500
Test	660 (41%)	940 (59%)	1,600

As seen, each category is fairly represented in all cases, so imbalanced data is not a problem to tackle within this task. Non hate speech instances are a bit more numerous in

[2] https://www.figure-eight.com/

the same proportion in all sets and both languages. This fact may lead to think that the splitting has been done randomly. Nevertheless, the majority class never gets to surpass a sixty percent of the total data.

On the other hand, if we focus on data distribution of Task B, that is to say, distribution of hateful Tweets in terms of aggressiveness and target goals, we can observe some notable differences across languages in Table 4.2.

Table 4.2: Data distribution in Task B

Split	**AG**	**no AG**	**TR**	**no TR**
English				
Train	1,559 (41%)	2,224 (59%)	1,341 (35%)	2,442 (65%)
Development	204 (48%)	223 (52%)	219 (51%)	208 (49%)
Test	590 (47%)	662(53%)	522 (42%)	730 (58%)
Spanish				
Train	1,485 (81%)	353 (19%)	1,117 (71%)	721 (39%)
Development	176 (79%)	46 (21%)	137 (62%)	85 (38%)
Test	474 (72%)	186 (28%)	423 (64%)	237 (36%)

In this case, there are different distributions along languages and sets, but different labels show a similar layout. This result goes in line with the work presented in [38], which states that directed hate speech is more informal, angrier, and often explicitly attacks the victim. Regarding the language, Spanish-speaking people tend to be more aggressive and more direct towards specific individuals. Seeing this skewed distribution, applying some balancing technique becomes quite appealing.

4.3 Evaluation

For the evaluation of the results of task A and B different strategies and metrics are applied in order to allow for more fine-grained scores.

4.3.1 Task A

Systems will be evaluated using standard evaluation metrics, including accuracy, precision, recall, and F1-score, but predictions are ranked by F1-score metric alone. The equations below show how the calculations are done. For better understanding, we will show the following definitions:

- **True Positive (TP):** is an instance correctly classified as hateful. For example, the target class is hateful and the model predicts it as hateful.
- **True Negative (TN):** is an instance correctly classified as not hateful. For example, the target class is not hateful and the model ranks it as not hateful.
- **False Positive (FP):** is a not hateful instance wrong classified as hateful. In this case, the target class is not hateful and the model says hateful.
- **False Negative (FN):** is a hateful instance miss classified as not hateful. In this case, the target class is hateful, but the model fails predicting it as not hateful.

Finally, the four mentioned metrics will be computed as follows [3]:

- **Accuracy:** proportion of correctly predicted samples (in binary classification, both true positives and true negatives) among the total number of cases examined.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.1}$$

- **Precision:** computes the proportion of instances predicted as positives that were correctly evaluated (it measures how right our classifier is when it says that an instance is positive).

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

- **Recall:** counts the proportion of positive instances that were correctly evaluated (measuring how right our classifier is when faced with a positive instance)

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

[3]Precision, Recall, and F1-score are metrics which can consider as positive instance any of the possible values of the target class. In this case, metrics are computed considering hateful tweets as positive class, then considering non-hateful tweets as positive class, and finally, the macro-average (unweighted mean without taking label imbalance into account) operation is applied to each metric

- **F1-score:** is the harmonic mean of precision and recall combining both values in a single number.

$$F1 - score = \frac{2 \cdot Precison \cdot Recall}{Precision + Recall} \tag{4.4}$$

4.3.2 Task B

In this task, systems will be evaluated on the basis of two criteria: partial match and exact match but predictions are ranked by exact match metric alone.

- **Partial Match:** each dimension to be predicted (Hate Speech HS, Target TR and Aggressiveness AG) will be evaluated independently of the others using standard evaluation metrics, including accuracy, precision, recall and F1-score as defined above. The report for each participant will include all the measures and a summary of the performance in terms of macro-average F1-score, computed as follows:

$$F1 - score = \frac{F_1(HS) + F_1(AG) + F_1(TR)}{3} \tag{4.5}$$

- **Exact Match:** all the dimensions to be predicted will be jointly considered computing the *Exact Match Ratio* [53]. Given the multi-label dataset consisting of n multi-label samples (x_i, Y_i), where x_i denotes the i-th instance and Y_i represents the corresponding set of labels to be predicted (HS $\in$ 0,1, TR $\in$ 0,1 and AG $\in$ 0,1), the Exact Match Ratio (EMR) will be computed as follows:

$$EMR = \frac{1}{n} \sum_{i=1}^{n} I(Y_i, Z_i) \tag{4.6}$$

  where Z_i denotes the set of labels predicted for the i-th instance and I is the indicator function, that is to say, I returns 1 if Y_i and Z_i are equal in all the dimensions considered (HS, TR, and AG) and 0 otherwise.

The reason for choosing EMR to rank the predictions comes from the willingness from the most difficult task of capturing the entire phenomena, and therefore to identify the most dangerous behaviours against the targets. When presenting the evaluation results we will discover the rigorousness of this metric.

4.4 System Overview

This section gives a completely explanation of the techniques utilized to tackle the problem we were aimed within the competition. Our system mainly relies on a supervised machine

learning algorithm. This final classification step is fed by a data processing pipeline formed by the preprocessing and the feature extraction modules. Regarding the implementation, Python has been used, including additional capabilities offered by the enabling technologies explained in the previous chapter, such as, Scikit-learn for the classification and feature extraction step, NLTK for text preprocessing purposes, and GSITK for general purposes. Figure 4.1 illustrates the system architecture from a global perspective.

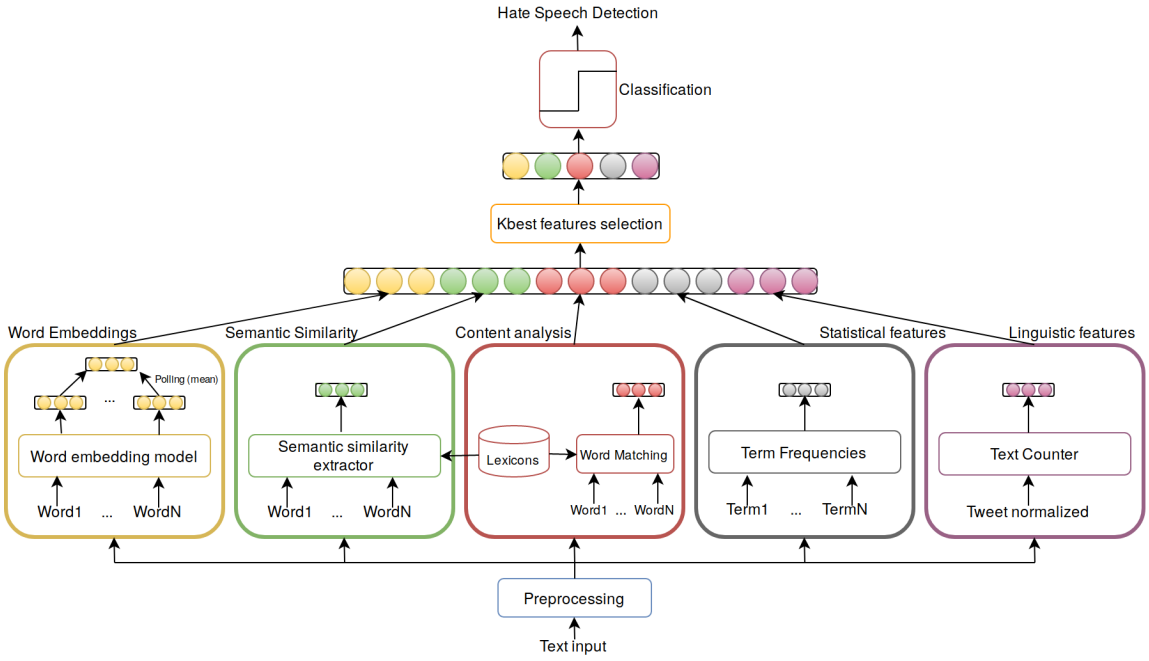

Figure 4.1: System Overview

4.4.1 Preprocessing

In this phase, the raw text is taken and cleaned using common NLP techniques [62]. We must have in mind that some existing words do not contribute any information, these are called stop words and are formed mainly by prepositions, pronouns and articles. These words need to be preprocessed so that our learning algorithm does not consider them. Besides, punctuation marks, special characters, URLs, and other stuff need also to be removed. Tweet preprocessing relies on tokenization, user mentions normalization, the appearance of hashtags, URLs, and all caps words flagged supported by the tools provided by GSITK.

It is also necessary to preprocess documents in another way, which consists in extracting the root or lemma of each word. This performance is necessary because there is more information on a specific concept than on its variations. Ignoring this variation could cause noise introduction and complexity addition. In the implementation of this project, we have

decided to use the Porter stemmer [76].

4.4.2 Feature Engineering

The next step we need to address is to obtain features from our unique information source input, the text message from the Twitter post. From a computer's point of view, the raw text gives no information at all. In order to get that information, we must perform some language processing techniques. Those consist in following different strategies to transform raw text data into a vector in such a way each position within the vector encompasses some kind of information which is expected to be helpful for the final decision phase. Those vectors are known as features and are the input format accepted to feed the final classification layer as seen in 4.1.

Different features have been taken into account during the feature engineering stage. Such features are divided into subcategories: statistical features, content analysis, word embeddings, semantic features, and linguistic features.

4.4.2.1 Statistical features

The first set of features considered are word and character n-grams frequencies evaluating both approaches, Bag-of-Words (BOW) and Term Frequency - Inverse Document Frequency (TF-IDF). The reason to include character n-grams comes from the Twitter domain, where texts are short and misspelling may occur; this effect can be attenuated at the character level [83]. Apart from the mentioned reasoning, previous research [65] has shown the effectiveness of character n-grams in the problem of offensive language.

Besides tokens included within the text corpus, the system also considers frequencies from words pertaining to external lexicons. Lexicons with hate speech terms [4], for sentiment [49, 58], and subjectivity [72] analysis domain were considered.

4.4.2.2 Content Analysis

As seen, sentiment and subjectivity information has been included. Hate speech can be considered as subjective content, and a relation between subjectivity, sentiments, and emotions can occur. Sentiments and subjectivity were included by means of word matching. For example, given a sentiment lexicon and input text, word matching counts each lexicon word

[4] https://hatebase.org/

in the input text as shown in Figure 4.2. Word matching can also be used with the TF-IDF measure.

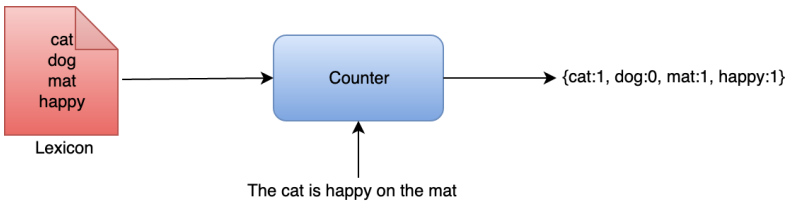

Figure 4.2: Keyword Matching

Besides, hate speech is expected to have a negative polarity, so text subjectivity and polarity provided by the TextBlob [61] library were included in the analysis.

Another interesting analysis was carried out using Latent Dirichlet Allocation (LDA). Topic modelling with LDA was added to the study. In our project, we have implemented it using the LatentDirichletAllocation module that provides scikit-learn, using parameters such as the number of topics we want to discover, priors of document-topic, and topic-word distributions.

Additionally, inside this set of features we include a method for focusing on topics from another perspective. We created a hashtag (which are really Twitter topics) vocabulary from the corpus given and computed frequencies using this lexicon as done with sentiment and remaining lexicons.

4.4.2.3 Word Embeddings

One of the problems in BOW models is that they do not have any knowledge about semantics of words. The similarity between two messages is calculated based on how many matching words there are in the messages (and their weights from TF-IDF). Therefore, we tried word embeddings which encodes words that are semantically similar with similar vectors. These vectors can be trained with the given data but there is still an open challenge on obtaining good vectors with low sized data. Due to this, is frequent to work with pre-trained word vectors learned over huge amounts of data. As said, pre-trained word vectors convert words into a vector space where semantically similar words tend to appear close by each other.

In this system, a vector is extracted for each word in the input text; then, as done in [9], the average pooling operation is performed on all word vectors, resulting in a vector of the same dimensions as the original ones. Several models have been tested during the development of this project due to language variety and vectors themselves will vary based

on the documents or corpora they are trained on.

For the Spanish case, 1,000,653 word vectors of dimension 300 trained on the Spanish Billion Words Corpus [23] were used. The skip-gram algorithm with negative-sampling was used, together with five as minimum word frequency, the 273 most common words were downsampled and the amount of "noise words" for the negative sampling was 20. The original corpus had a total of 1,420,665,810 raw words, 46,925,295 sentences and 3,817,833 unique tokens and after the model was applied a total of 771,508,817 raw words and 1,000,653 unique tokens were obtained.

For the English case, two models were evaluated. One of them was trained using GloVe and the other one using fastText. GloVe model was trained on data extracted from the Wikipedia [5] and Gigaword [6] containing 6 billion of raw words outputting a word representation of 300 dimensions (Other versions of 50, 100, and 200 are also available) and 400,000 as vocabulary size.

Last model proved was learned using the fasText approach which consists in an evolution over the previous explained Word2vec model. The gist of fastText is that instead of directly learning a vector representation for a word (as with word2vec), we learn a representation for each character n-gram. Each word is represented as a bag of character n-grams, so the overall word embedding is a sum of these character n-grams. This model resulted in a 2 million word vectors size trained [67] on Common Crawl [7] data.

4.4.2.4 Semantic Features

A central part of the system consists of a method [10] that exploits the semantic similarity measure that a word embedding model provides, via cosine similarity. In general lines, this approach uses a lexicon to which the input text is projected, employing the similarity measure obtained from an embedding model (See Section 3.6.5).

Typically, lexicons are used through word matching or BOW approaches as seen in Section 4.4.2.2. Using SIMON, lexicon information is extracted in a more sophisticated way and the final representation of a text is relies on how similar are the input words to lexicon words which can help the decisions of the final classifier.

In this work, the previously mentioned lexicons have been used, as well as a domain-oriented word selection, which have been extracted from the given dataset. In this last

[5] https://www.wikipedia.org
[6] https://catalog.ldc.upenn.edu/LDC2011T07
[7] http://commoncrawl.org/

approach, words were filtered by its frequency of appearance considering the document annotation, being the cutoff frequency an adjustable parameter.

4.4.2.5 Linguistic Features

The last set of features are related to linguistic aspects. Among all the approaches considered from this sub-module, Part-of-Speech (POS) stats stand out from the rest. POS features focus their analysis on lexical categories of words. The aim is collecting statistics about word categories from the input text and see if hateful tweets use some specific categories. The parts of speech analyzed include adjectives, adpositions, adverbs, conjunctions, nouns, numerals, pronouns and verbs. The tag set (categories) depends on the corpus annotation. Fortunately, NLTK defines a Universal tag set which is presented in the Table 4.3

Table 4.3: Universal tags defined by NLTK

Tag	**Meaning**	**English Examples**
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner, article	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
PRT	particle	at, on, out, over, per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

The proposed system also relies on lexical statistics, including the number of sentences and length from the tweet, as well as a count of several Twitter-related features such as hashtags, URLs, mentions, all caps words, emojis, and exclamations. A better way of

exploiting emojis essence consisting in creating an emoji vocabulary and applying it the previous explained lexicon-based methods [8] is also considered.

4.4.2.6 Feature Selection

Reminding Figure 4.1, an intermediate step called feature selection between feature extraction and classification phases can be found. At this point, we have transformed the input text into a feature output vector and what we pretend in this state is to select the most helpful characteristics to solve the addressed problem.

Some advantages come by using this feature selection step. Firstly, reducing the feature vector size can avoid the overfitting problem. Feeding the classifier with too many features, may cause that the parameters learned by the classifier adapt too much to the the training set, and the algorithm will only be able to predict some very specific results, failing to predict new ones.

Secondly, with a lower number of features complexity is reduced and features dropped are the less significant, those without meaningful impact on the final decision. This way, we are favouring the classifier labour letting it focusing in the most important data. For this project, the k best features according to the ANOVA statistical test are used.

4.4.3 Classification

Finally, the furthest step in the data processing pipeline makes use of a machine learning classifier. There are many options among machine learning models that can be used. In this project, we have evaluated the performance of three different types of algorithms: Logistic Regression, Support Vector Machines (SVM) with linear kernel, and Random Forest.

4.5 Experiments

This section presents the results obtained by the proposed system in the competition, considering both test and development phase submissions. The evaluation of the different feature extraction approaches has been done by using a holdout strategy. Using the training set for training the models and the development set for testing it.

[8] TF-IDF and BOW statistics can be computed over an emoji lexicon, however it is not possible to use such vocabulary in the semantic similarity approach because emojis are not part of a word embedding model (they do not have an a associated vector in the model)

In order to optimize the algorithm behaviour, the GridSearchCV tool provided by Scikit-learn has been used. This module allows you to know the hyper-parameters that give the best results to a classifier when classifying a dataset with certain features. Proceeding using grid-search is better than choose hyper-parameters randomly cause the value given to different hyper-parameters can influence the obtained results by our algorithms. Special attention has been paid in the regularization parameter of the algorithms: "C" parameter in the Logistic Regression and Linear SVM case and "maximum depth" of the trees in the Random Forest case. Finally, the system is trained, and the evaluation metrics are computed. This workflow has been repeated several times from the feature extraction step, changing the set of features in every iteration.

4.5.1 Sub-task A

We remind that the goal of this task is to classify both Spanish and English tweets as hateful or not hateful. Systems are evaluated using standard evaluation metrics, including accuracy, precision, recall, and F1-score, but predictions are ranked by F1-score metric alone.

As seen, Task A data was partitioned into train, development, and test sets. Train and development sets were used to obtain the best feature combination by training over the train set and testing over the development one. Finally, for the final submission, the predictions for the test set were obtained with a system trained over both train and development sets.

Test results, which represent the official submission, as well as development phase results are presented in Tables 4.4 and 4.5 respectively. Task organizers included two baselines [12] in the competition, a linear SVM based on a TF-IDF representation and a trivial model that assigns the most frequent label from the training set to all instances in the test set.

The Spanish-oriented system used to make the official submission, relies on linguistic features (excepting POS), semantic similarity with a domain-oriented lexicon, sentiments (using the sentiment vocabulary weighted by the TF-IDF measure), word embeddings, topic modeling (both LDA and hashtags), and word and character TF-IDF n-grams. These features are filtered according to the ANOVA F-test, selecting the best 3,000. Linear SVM has been the selected machine learning algorithm for classification. On the other hand, the English-oriented system considers the same feature set excluding word embedding representation; the number of selected features has been set at 17,500. In contrast to the previous system, a Logistic Regression model was used to perform the classification.

Team	Accuracy	Precision	Recall	F-score
English				
Best	0.506	0.65	0.566	0.457
SVM baseline	0.492	0.595	0.549	0.451
GSI-UPM	0.483	0.643	0.549	0.42
MFC baseline	0.579	0.289	0.5	0.367
Spanish				
Best	0.731	0.734	0.741	0.73
GSI-UPM	0.728	0.726	0.733	0.725
SVM baseline	0.705	0.701	0.707	0.701
MFC baseline	0.58	0.294	0.5	0.37

Table 4.4: Official test set results for Task A

4.5.2 Sub-task B

We also remind that the goal of this task is firstly to classify hateful tweets (i.e., tweets identified as hate speech against women or immigrants) as aggressive or not aggressive, and secondly to identify the target harassed as individual or generic (i.e., single person or group). Systems are evaluated by two criteria: partial match and exact match, but predictions are ranked by exact match metric alone.

For this task, the data has been delivered in the same way than sub-task A, so we emulated the same workflow than before, but in this case, considering solely hateful tweets. In this case, there are different data distributions 4.2, so we outlined the idea to balance aggressiveness and directed messages by oversampling hateful tweets with not hateful ones, assuming that not hateful tweets are not aggressive nor directed.

As done previously, Tables 4.6 and 4.7 present official and development results, respectively. The Spanish-oriented system in this task is identical to that from Task A, but finally selecting 2,500 features. For the English case, in light of aggressiveness and target tweets, a different combination of features have been chosen. In order to detect aggressive tweets,

Feature combination	Accuracy	Precision	Recall	F-score
English				
Official submission combination	0.777	0.774	0.780	0.775
Lexical, similarity, embeddings, and n-grams (1)	0.757	0.754	0.758	0.754
Bigrams, trigrams, similarity, and embeddings (2)	0.75	0.747	0.752	0.748
Embeddings, similarity, twitter stats, and LDA (3)	0.736	0.731	0.733	0.732
Spanish				
Official submission combination	0.856	0.856	0.852	0.853
Lexical, similarity, embeddings, and n-grams (1)	0.812	0.811	0.807	0.808
Bigrams, trigrams, similarity, and embeddings (2)	0.796	0.794	0.791	0.792
Embeddings, similarity, Twitter stats, and LDA (3)	0.784	0.781	0.782	0.782

Table 4.5: Development set results for Task A

all features except semantic similarity have been used, filtering the 32,500 best. Otherwise, for target messages, the complete set of features (sentiments and subjectivity were included by means of TF-IDF and semantic similarity) are used just considering the 2,500 best. Finally, different models were applied for each label, Logistic Regression for Target label and Linear SVM for the Aggressive one. The same algorithm selection was made in the Spanish case.

4.5.3 Discussion

In general terms, the results obtained are auspicious: the best submitted system achieved the 5th position in the Spanish Task A, 0.5% points under the best result obtained in the

Team	F-score(HS)	F-score(TR)	F-score(AG)	F-score (Avg)	EMR
English					
MFC baseline (Best)	0.367	0.452	0.445	0.421	0.58
GSI-UPM	0.421	0.686	0.556	0.555	0.312
SVM baseline	0.45	0.697	0.587	0.578	0.308
Spanish					
Best	0.729	0.798	0.737	0.755	0.705
GSI-UPM	0.725	0.79	0.735	0.75	0.624
SVM baseline	0.701	0.781	0.726	0.736	0.605
MFC baseline	0.37	0.424	0.413	0.402	0.588

Table 4.6: Official Results for Task B

Feature Combination	F-score(HS)	F-score(TR)	F-score(AG)	F-score (Avg)	EMR
English					
Official submission	0.775	0.811	0.723	0.770	0.665
(1)	0.754	0.797	0.712	0.755	0.641
(2)	0.748	0.788	0.699	0.745	0.628
(3)	0.731	0.767	0.687	0.728	0.611
Spanish					
Official submission	0.853	0.876	0.824	0.851	0.78
(1)	0.808	0.839	0.777	0.808	0.732
(2)	0.792	0.843	0.776	0.804	0.718
(3)	0.782	0.836	0.783	0.800	0.714

Table 4.7: Development Results for Task B

same task. For the Spanish Task B, the proposed model outperforms the baseline. In contrast to this, results in English Task A are lower than expected, where there was not any team that surpassed the 50% threshold in terms of F-score. As a general trend, test

set results are worse than development results, which may indicate that our systems suffer over-fitting, and cannot generalize properly. This observation is enforced by attending to the English Task B, where no system has surpassed the baseline.

Since the data distribution is equal along languages in Task A (Section 4.2), the difference in performance across languages may be due to Spanish-speaking people are more explicit when typing any utterance with hate speech goals. As previously mentioned, we have observed that this type of hate speech messages show more aggressiveness. Language characteristics may be involved since the Spanish language has a morphologically-richer nature than English.

The presented results constitute the outcome of exhaustive experimentation of a variety of feature combination tests. In contrast with earlier work, semantic similarity and word embeddings representations do not produce such high performance results when compared to other domains such as sentiment analysis [10] and sleep disorder detection [87] tasks. This circumstance suggests that hate speech detection is still an open challenge and more research must be done into the specific characteristics of such an exciting task.

Attending to the Spanish case, sentiment information and character n-grams were features that helped in a meaningful manner, confirming the issues raised in Sect. 4.4. For the English case, the improvement of the proposed features was incremental. While subjectivity and emojis had a relevant role in the results, this improvement was not as high as in the Spanish case. In light of the complexity of the hate speech domain, it could be argued that attending to word context instead of isolated words could help in the analysis. Indeed, n-grams include this type of information to some extent, but capturing the grammatical dependencies within a sentence [24] or template based strategies [96] could enhance the performance.

CHAPTER 5

Transfer Learning for Hate Speech Detection

The penultimate chapter reviews some experiments related to the Transfer Learning research problem. The selected use case discusses the feasibility of two approaches: multilingual hate speech detection, and cross-domain hate speech traits study. Overall, the chapter includes an introduction to the problem, a Transfer Learning statement, multilingual word embeddings fundamentals, a description of the data used, and finally, architectures, experiments, and results of both approaches.

5.1 Introduction

First of all, we must start introducing the Transfer Learning concept:

> *"Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned." [89].*

Previous statement says the following. We have a model trained with data annotated to solve a specific task, and by applying minor changes in such model we can use it to solve other specific tasks in relation to the former one. That is to say, the final goal is to broaden the scope of our algorithm to generalize from solving one problem to solve similar problems.

Figure 5.1 illustrates the whole process. The source learning task step corresponds to a traditional Machine Learning approach as the ones afforded in Chapter 4. Then, Transfer Learning module can be fed with new data from the target task, but it is not necessary neither such data matches to classification data. After Transfer Learning step we have obtained a system ready to use in the target class.

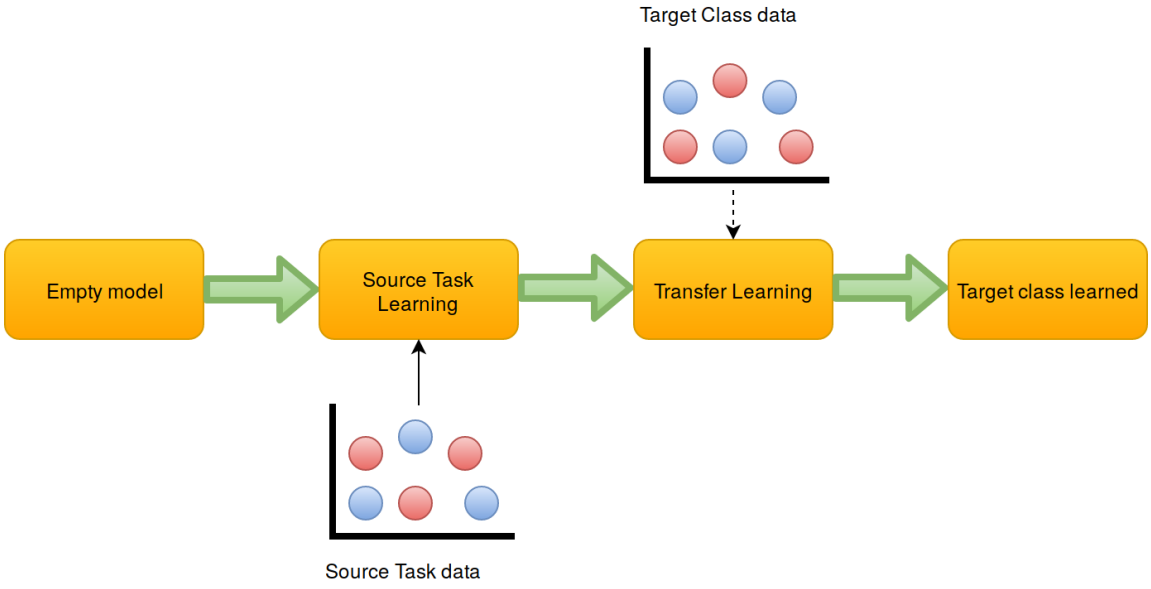

Figure 5.1: Transfer Learning process

If classification data is used in the transfer process, then the Machine Learning model is being fine-tuned. fine-tuning consists of keep training a previously trained model, taking the learned parameters (for example, weights from the neural network) as initialization to the new training process. Note that this is only applicable for some models and some parameters, and even some orders of parameter values. Indeed, from now on the traditional

optimization strategy of linear SVM has to be changed, so the Stochastic Gradient Descent implementation has to be used.

There are three common measures by which transfer can be evaluated. All of them are based on plotting the learning curves of both, a transfer learned system and a model learned from scratch. A learning curve is a graphical representation of how an increase in learning (measured on the vertical axis) comes from a greater number of training examples (the horizontal axis).

Figure 5.2 presents these three measures. First is the initial performance achievable in the target task using only the transferred knowledge, before any further learning (without fine-tuning) is done, compared to the initial performance of an ignorant agent. Second is the amount of time it takes to fully learn (with fine-tuning) the target task given the transferred knowledge compared to the amount of time to learn it from scratch. Third is the final performance level achievable in the target task compared to the final level without transfer and fine-tuning.

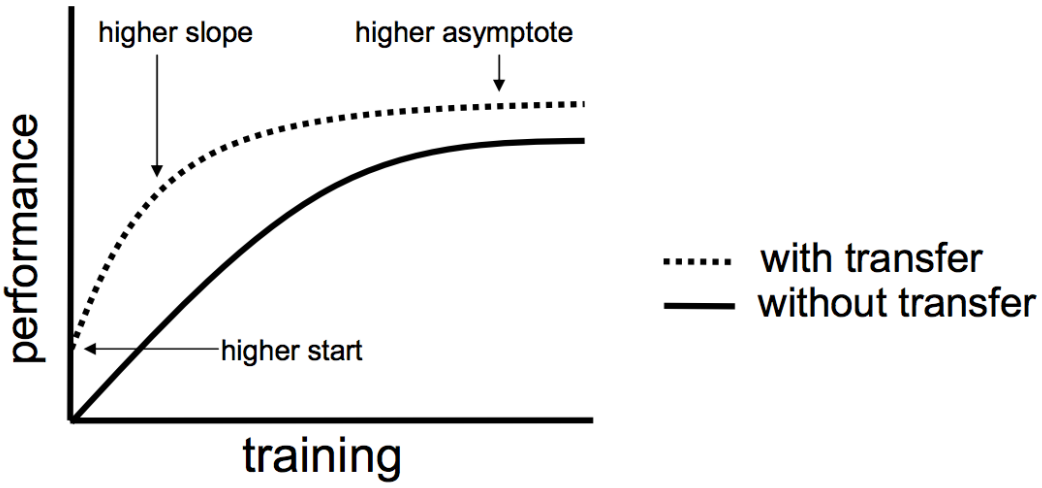

Figure 5.2: Transfer Learning evaluation (Figure from Torrey et al. [89])

In our specific case, we are going to test two Transfer Learning approaches. One of them is the use of Transfer Learning for multilingual hate speech detection. The idea is to train a hate speech system in a specific language, and use this system to serve as hate speech detector in any other language. The fundamentals of this idea are explained in Section 5.1.1, which relies on aligned word vectors and SIMON as feature extraction method.

The other study case consists of Transfer Learning across hate speech traits, the same from those used in SemEval-2019 Task 5, aggressiveness and target goal. The idea now is to use hateful tweets to detect aggressiveness with a system trained to classify the target

and vice versa.

5.1.1 Multilingual Word Embeddings

Usually, word embeddings are trained with a single language corpus. This way, we have a vector model for Spanish, another for English and so on. Initially, different language vectors do not share vectorial space (as shown in Figure 5.3 (**A**)) because the training process is different. Different data is used, initialization weights of the neural network are different, model hyper-parameters to train the model are also different.

Nevertheless, in [68] first noticed that continuous word embedding spaces exhibit similar structures across languages, even when considering distant language pairs like English and Vietnamese. The work presented in [27] proposes an unsupervised approach to exploit this similar shape by learning a linear mapping from a source to a target embedding space. The resulting monolingual embeddings are overlapped in the space, so the same word in different languages has the same vector (as shown in Figure 5.3 (**D**)).

Let $X = (\vec{x}_1, \vec{x}_2, .., \vec{x}_n)$ and $Y = (\vec{y}_1, \vec{y}_2, .., \vec{x}_m)$ be two sets of n and m word embeddings coming from a source and a target language respectively. Then, the optimal linear mapping is a matrix W which multiplied by X gives Y. Therefore, the goal is to learn a matrix W which minimizes

$$\|WX - Y\| \tag{5.1}$$

An illustration of the approach is given in Figure 5.3. In Figure 5.3 (**A**), two distributions of word embeddings initially separated are presented. Then, in Figure 5.3 (**B**) using adversarial learning two neural networks contest with each other in a zero-sum game framework. A network is trained to discriminate between elements randomly sampled from $WX = (W\vec{x}_1, W\vec{x}_2, .., W\vec{x}_n)$ and Y. The other network is trained to prevent the discriminator from making accurate predictions. As a result, the discriminator aims at maximizing its ability to identify the origin of an embedding, and the other model learns W preventing the discriminator from doing so by making WX and Y as similar as possible. Afterwards, in Figure 5.3 (**C**), we can see that the matrix W obtained with adversarial learning roughly aligns the two distributions, but the adversarial approach align words irrespective of their frequencies. However, rare words have embeddings that are less updated and are more likely to appear in different contexts in each corpus, which makes them harder to align. In order to refine the mapping, a synthetic parallel vocabulary is constructed using the W just learned. Subsequently, the Procrustes solution from [84] on this generated dictionary is applied. Considering the improved solution generated with the Procrustes algorithm, it is

possible to generate a more accurate dictionary and apply this method iteratively. Finally, in Figure 5.3 (**D**), the resulting vectors are aligned.

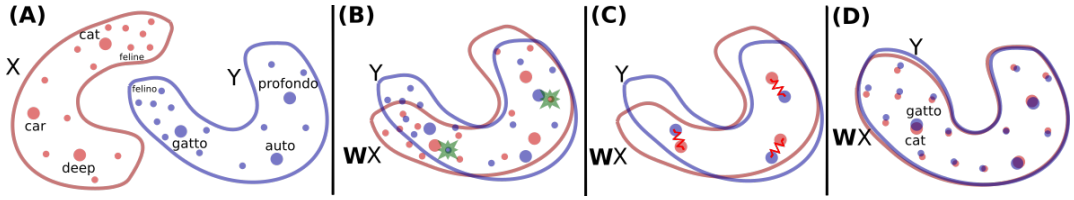

Figure 5.3: Toy illustration of the method. Figure from Conneau et al. [27]

Besides, they provide multilingual embeddings already aligned. In particular, they release fastText Wikipedia supervised word embeddings for 30 languages, aligned in a single vector space [1]. Having these vectors aligned, a system constructed with these word embeddings can be extrapolated from one language to another.

5.2 Data

This section focuses on the public data found to carry out the later experiments. We will follow a similar approach to that presented in Section 4.2, a brief description of the data will be given, as well as the corresponding label distribution. We are using data from six languages: English, Spanish, German, Indonesian, Portuguese, and Italian. Following sections describe data as mentioned for each language. Spanish language is not included since it is the same data used in the SemEval competition and it has been already outlined in Section 4.2.

5.2.1 English Data

English is the language in which more data is available (See Table 2.6). For this reason, we have made a selection of three well-established datasets in the hate speech field in addition to the SemEval data. Each of the three datasets is presented below.

5.2.1.1 Davidson Data

This section treats the data from Davidson et al. study [31]. They used the Twitter API to search for tweets containing terms from the *Hatebase.org* lexicon. Then, 25k tweets were selected and manually coded by CrowdFlower workers. Workers were asked to label

[1] https://github.com/facebookresearch/MUSE#Download

each tweet as one of three categories: hate speech, offensive but not hate speech, or neither offensive nor hate speech. Each tweet was coded by three or more people, using the majority decision for each tweet to assign a label. Only 6% (1,430 tweets) of tweets were labeled as hate speech. The majority of the tweets were coded as offensive (19,190, 77%) and the remainder were considered to be neither hate speech nor offensive (4,163, 17%).

5.2.1.2 Waseem Data

This data is part of the Waseem et al. research [98]. Their data set consist of tweets collected over the course of 2 months, again using the public Twitter search API looking for tweets containing common slurs and terms used pertaining to religious, sexual, gender, and ethnic minorities. Finally, they manually annotated 16,914 tweets, 3,383 (20%) of that for sexist content, 1,972 (12%) for racist content, and 11,559 (68%) for neither sexist or racist. [2]

5.2.1.3 Stormfront Data

The files provided within this dataset [32] were extracted from Stormfront [3], a white supremacist forum. Textual hate speech is annotated at sentence level, allowing to work with the minimum unit containing hate speech and reduce noise introduced by other sentences that are clean. A total number of 10,944 sentences have been extracted from Stormfront and classified as conveying hate speech or not, and into two other auxiliary categories.

- **Relation:** if the sentence is not an utterance of hate speech, but in combination with other sentences it is.

- **Skip:** sentences that are not written in English or that do not contain information as to be classified into hate speech or no hate speech.

Table 5.1 shows the distribution of the sentences over classes.

[2]The dataset they provide contains the TweetID instead including directly the text, so when downloading the tweets the distribution was a bit different due to tweet removal. 3,167 for sexist tweets, 1,935 for racist tweets, and 11,033 for non hateful tweets

[3]`https://www.stormfront.org/forum/`

Assigned label	sentences	%
Hate	1,196	11
No hate	9,507	87
Relation	168	1
Skip	73	1

Table 5.1: Distribution of sentences over categories in the Stormfront dataset

5.2.2 German Data

German data [81] is centered over the recent topic of the refugee crisis. They used Twitter as source with 10 hashtags [4] that can be used in an insulting or offensive way. The resulting corpus consists of 469 tweets, none of which contain links or pictures or are retweets or replies. They personally annotated the tweets based on their previous expertise. Data were split into six parts and each part was annotated by two annotators in order to determine if hate speech was present or not. The annotators were rotated so that each pair of annotators only evaluated one part. So there are two sets of annotators which annotation distribution is presented in Table 5.2

Annotator	#Hate (%)	#No Hate (%)
Experts set 1	110 (23)	359 (77)
Experts set 2	98 (21)	371 (79)

Table 5.2: Distribution of German data for each label and each expert set

5.2.3 Indonesian Data

We did not expect to found data in such exotic language as Indonesian, but surprisingly there is more than one available dataset for hate speech detection written in this language. In particular, there exists a dataset which uses Instagram [5] as source, meanwhile, there is

[4] #Pack, #Aslyanten, #WehrDich, #Krimmigranten, #Rapefugees, #Islamfaschisten, #RefugeesNotWelcome, #Islamisierung, #AsylantenInvasion, #Scharia

[5] `https://www.instagram.com/`

also another one extracted from Twitter.

5.2.3.1 Instagram Dataset

The dataset provided was used to detect hate speech on Instagram comments [77]. They crawled comments from nine suspected hateful accounts based on a keywords and hashtags search. As a result, they gathered 1,200 comments to be manually labeled by three annotators representing a certain diversity in terms of age and sex: a woman (25) and two men (25, 20). After the annotation process, the tweets where the annotation agreement was not 100% were removed due to ambiguity. This decision reduced the data to a total of 835 comments, consisting of 286 hateful comments and 549 labeled as not hateful ones. Finally, they decided to make available a balanced version of the data with 286 samples for each category.

5.2.3.2 Twitter Dataset

The data here provided [6] uses the more habitual social network, Twitter. The new dataset created aims to cover hate speech in general, including hatred against religion, race, ethnicity, and gender. In this case, they collected tweets using the Twitter Streaming API. These tweets were related to a political event, the Jakarta Governor Election 2017. This election was a potential source of hate speech data because one of its candidates came from a minority group in Indonesia, in terms of religion and race, while another candidate was a woman that potentially could trigger hate speech related to gender. After removing duplicated tweets, they had 1,100 tweets to be labeled manually.

The annotation process was done in binary format, by means of deciding if a tweet contained hate speech or not. Annotators were 30 volunteers who were all college students in Jakarta and surrounding areas with the age range of 17-24 years. Gender distribution was of 43.3% of men and 56.7% of women. The previously 1,100 collected tweets were divided into 22 sets, each one of them being of 50 tweets size. Each set would be annotated by three people from different religious, racial and gender backgrounds. Following the same approach than in the previous case, only 100% annotator consensus tweets were considered. Thus, the resulting tweets were 713 consisting of 260 (36%) tweets with hateful content and 453 (64%) tweets with hate absent.

5.2.4 Portuguese Data

The source of the data is the most accessed news site in Brazil [6]. During the research [34] where the data was generated, they noticed that the new categories with the most offensive comments are politics and sports, so the data collection was limited to those sections. At the end of the process, they obtained 10,336 comments posted for 115 news.

Because of human limitation, a sample of 1,242 comments was randomly selected for annotation. Following the standard procedure adopted for dataset annotation, each comment was annotated by three judges which were asked to whether it was offensive. In case of an affirmative answer, the annotator was also asked to categorize the offence as racism, sexism, homophobia, xenophobia, religious intolerance, or cursing. Two datasets were generated from the annotations. The first has all 1,242 instances and the class assigned to each comment was the one picked by at least two of the judges. The second is a stricter dataset, composed solely of the comments for which all three judges agreed as to identify whether or not the comment was offensive. In this project, we are using the first dataset of both previously presented, which presents the following binary distribution. 414 (33%) samples resulted to be offensive, while 828(66%) instances were not.

5.2.5 Italian Data

The last dataset used in this chapter was proposed for an Italian task [7] (similar to SemEval). Two research groups made an effort to create two datasets, in order to allow their exploitation in the task. The first dataset is a collection of Facebook comments developed by the group from CNR-Pisa and created in 2016 [35], while the another one is a Twitter corpus developed in 2018 by the group from the Computer Science of Turin University [75, 82]. Data was protected, so we were only able to access training data which consists of 3,000 tweets for both cases.

5.2.5.1 Facebook Dataset

Facebook data was retrieved from public pages of Italian newspapers, politicians, artists, and groups, which typically host discussions spanning across a variety of topics. Among all the available information only direct comments to the posts were considered. Five bachelor students annotated the data, assigning a class to each post among the two possibilities as

[6] https://g1.globo.com/

[7] http://www.di.unito.it/~tutreeb/haspeede-evalita18/index.html

usual: hate and no hate. Finally, the distribution turned out to be almost balanced: 1,618 (54%) for non hateful posts and 1,382 (46%) for hateful ones.

5.2.5.2 Twitter Dataset

Tweets were collected according to three specific targets: immigrants, Muslims and Roma. They first annotated 1,827 thanks to the labour of four independent annotators. The corpus was split in two, and each part was annotated by two annotators. The annotator pairs then switched to the other part, in order to provide a third (possibly solving) annotation to all those tweets where at least one category was labelled differently by the previous two annotators. Then, another data collecting was carried out filtering by words that more frequently occur in texts annotated as hate speech in this first dataset. The final version of the corpus consists of 6,009 tweets, having the ones for the training data in the task, the following distribution. 2,028 (%) inoffensive tweets and 972 (32%) tweets categorized as hateful ones.

5.3 Cross Language Transfer Learning

The first set of experiments address the problem of Transfer Learning across languages. Seeing data distributions for each language in Section 5.2, we can observe that except for English, hate speech publicly available data is very poor. For this reason, we can build a robust classifier with English data and making few changes, apply this model to new data from other languages. In the following section, we will see how to do this.

5.3.1 Architecture

The key idea, in order to achieve the desired transfer, is supported by the multilingual word embeddings. If we learn a system based on a method which in turn depends on a monolingual (English) word embedding model and such model is aligned with other different monolingual embeddings, we can export the learning to other languages by means of changing the original (English) word embedding model by the word vectors from the target language.

The first step is to generate the English system. Because of the data in target languages is annotated in a binary format, we must transform English data into the same binary format. Davidson data was encoded to have two categories grouping offensive (but not hate speech) tweets with no offensive ones in the no hate category. Racist and sexist samples from Waseem data were grouped into the hate speech class. Regarding the Stormfront

data, only samples labeled as hate and no hate were used. Remember that SemEval data was already encoded in a binary format. After cleaning the data and removing possible duplicates, a dataset of 63,472 instances was formed with 50,483 (80%) hateful examples and 12,989 (20%) no hateful ones. Finally, data was balanced using only a random subset of 12,989 instances from the no hate class.

Then, following a similar approach than the one presented in Section 4.4 and Figure 4.1, the English system is built. As opposed to the system presented in the competition, we now only consider SIMON sub-module in the feature extraction step, since we need a method based on word embeddings. Besides, the feature selection step has been removed. The lexicon used to feed the semantic similarity extractor is a mix of Hatebase terms and most frequent words from the English corpus recently generated. Figure 5.4 illustrates the system used to learn from English data with SIMON step zoomed. In fact, this system, with some changes, will be used in other languages.

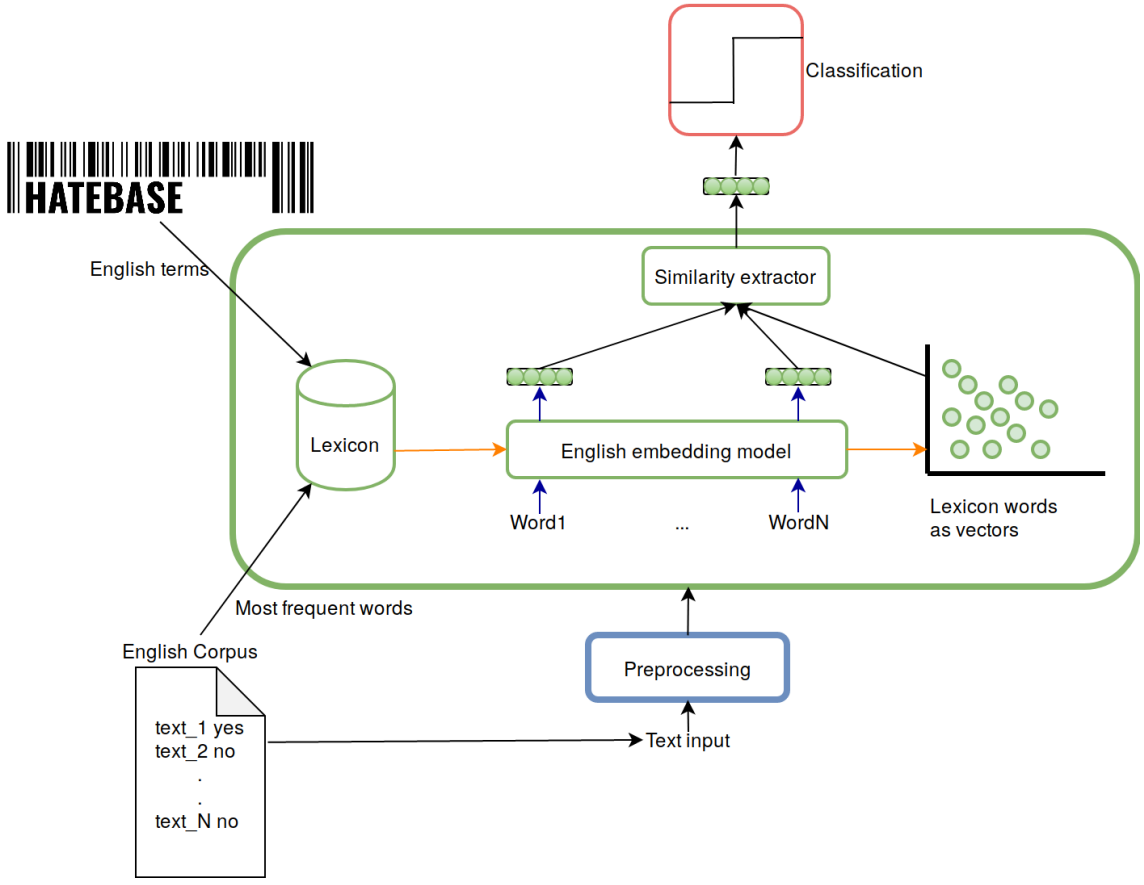

Figure 5.4: System trained with English data

SIMON implementation stores the lexicon word vectors during the training process, which facilitates the transfer implementation because word embeddings alignment makes

those vectors valid for all languages in which there exists a corresponding aligned vector. This fact makes ourselves to save from translating the lexicon or some similar approach. This causes that by only changing the word embedding model we can use this system in any other language as shown in Figure 5.5.

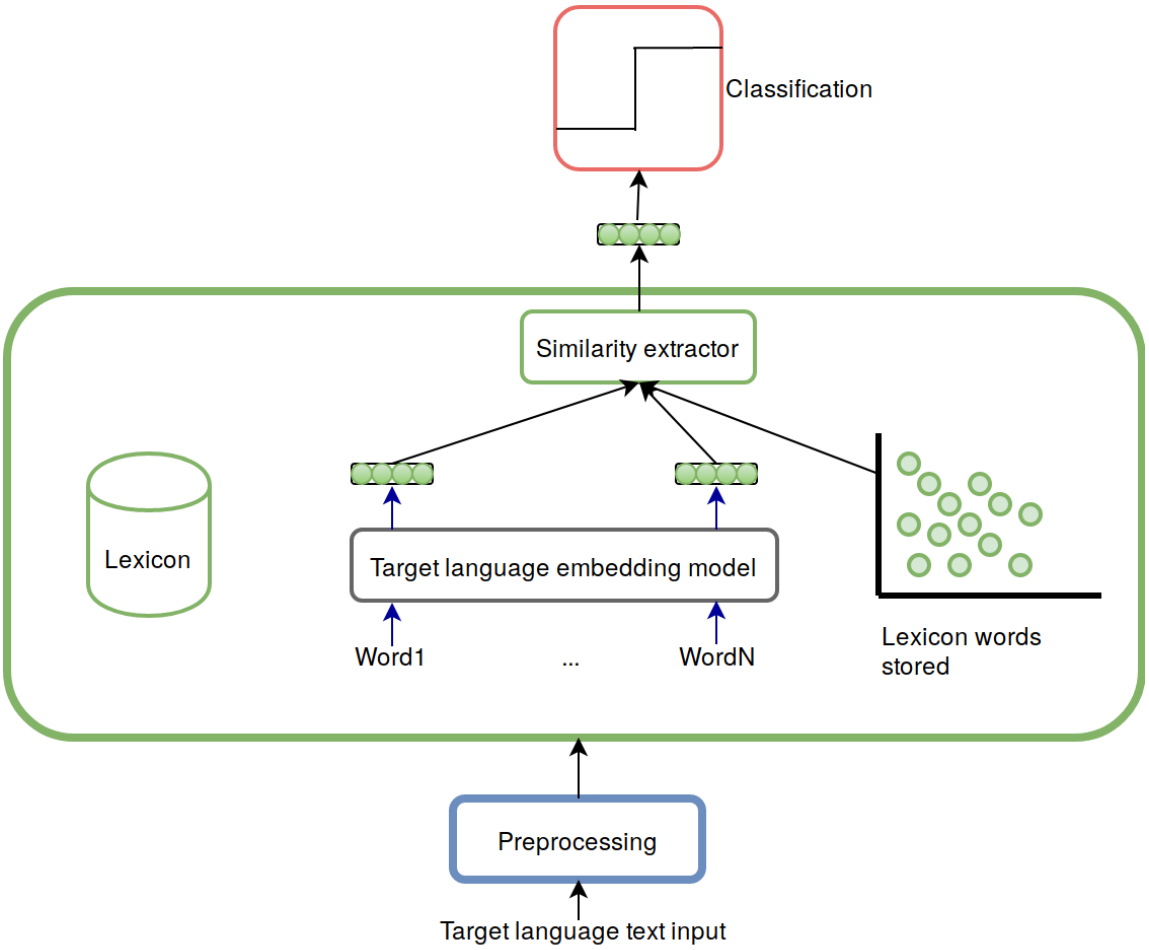

Figure 5.5: English trained system transformed to predict hate speech in the target language

Finally, further modification can be carried out to improve the transfer process. We refer to keep training the algorithm in the classification step with data from the target language. This is known as fine-tuning in terms of Transfer Learning and can also serve us to measure the transfer quality as seen in Section 5.1. Fine-tuning in our system is depicted in Figure 5.6.

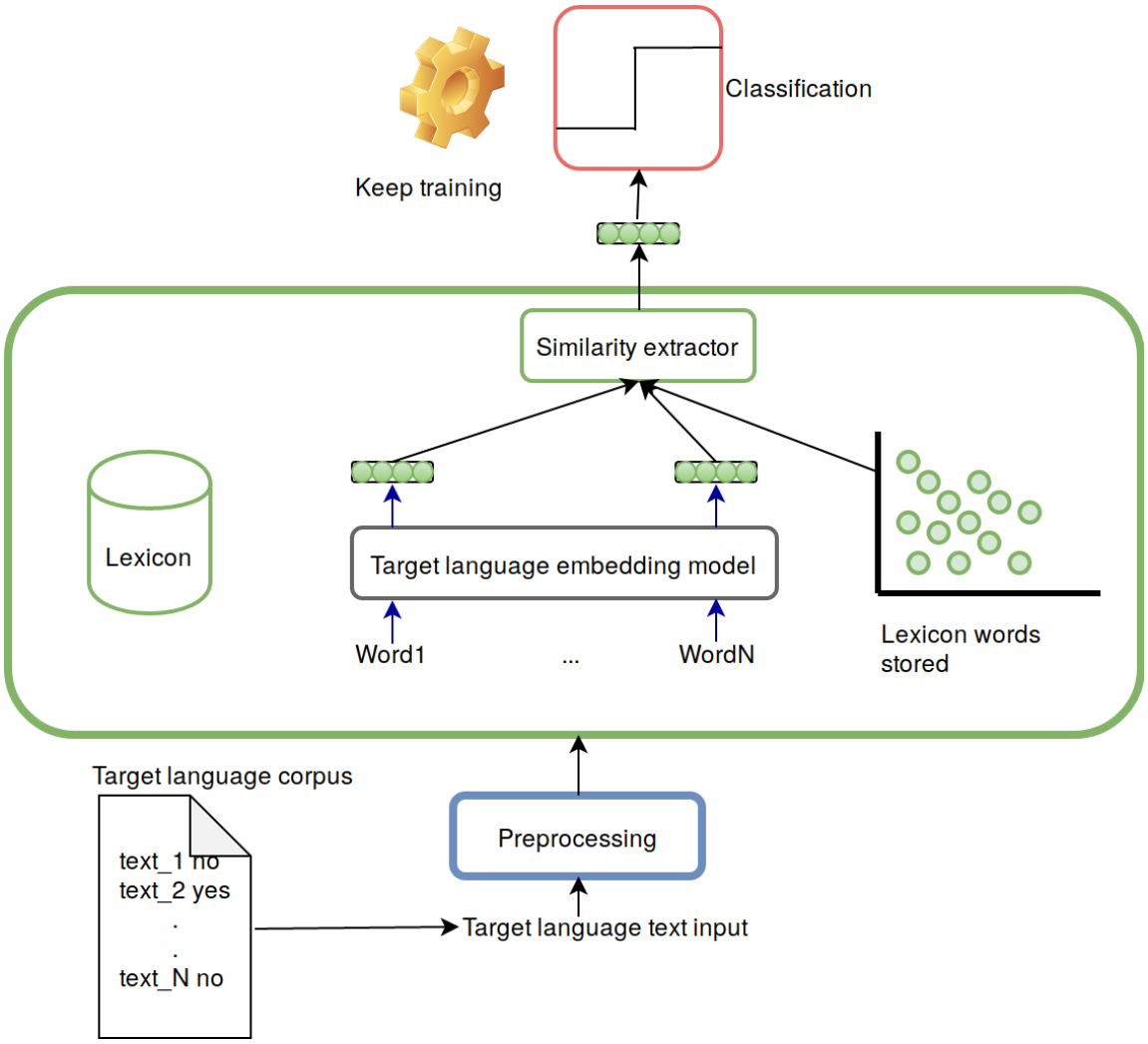

Figure 5.6: English trained system fine-tuned with target language data

5.3.2 Results

This section provides a set of results obtained using the Transfer Learning process explained in the previous section. For each language, the system with Transfer Learning is compared against what we call the native system. Native system is built in the same way than the English system was, but only considering target language data (in Figure 5.4 changing "English" by the target language).

The evaluation is made by means of learning curves as seen in Figure 5.2. With the exception of the Spanish case, F1 scores of the curve are obtained with a three-fold cross-validation approach. The same three algorithms used in the previous chapter will be evaluated. In the posterior graphics, green curve represents the system with Transfer Learning, while red line is represented by the native system in the language at hand.

5.3.2.1 Spanish

Spanish experiments represent a special case due to data splitting. Instead of using cross-validation we have taken advantage of the data separation provided in the competition. This way, we use the training data to fit the model and the development and test sets to compute the scores. Recycling data this way can serve us to make further comparison. Learning curves for both, development and test experiment can be found in Figures 5.7 and 5.8 respectively.

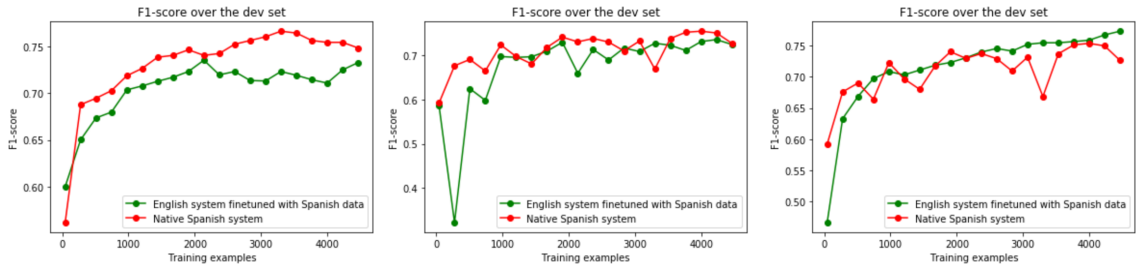

Figure 5.7: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish development set

With the development data, some promising results can be observed. First, Random Forest system with Transfer Learning achieves a better performance after training is complete. Nevertheless, such improvement is not significative, and we appreciate more the Logistic Regression results where English fine-tuned system starts better. This is a key result since if this becomes a trend we can assume that the transferred system can be used in languages where data is scarce. In this case, we have more than 4,000 training examples and normally the native system will surpass the English trained model, but next experiments where only exist hundreds of examples, the improvement in the beginning will be much more clear. Additionally, the fine-tuned model grows more linearly than the native system which grows in a noisy way.

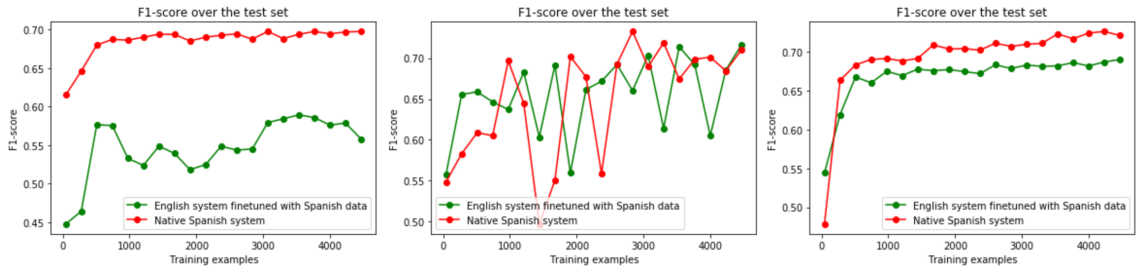

Figure 5.8: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set

On the other hand, test results are a bit worse. F-score metric is five point under the results from the previous graphic (from 75% to 70%) and transfer systems do not behave better than native systems, excepting Random Forest at the learning commence. In relation to the competition results, in the development set, we are far from the best feature combination (85%), while we are really near in the test set (72.5%), though it was easier since results are lower.

Another aspect to keep in mind is the noisy behaviour of Linear SVM. This fact will be repeated along with all the experiments. This noise may be introduced by the Stochastic Gradient Descent optimizer which divides training data into mini-batches and the optimization process into steps. The use of smaller mini-batches might produce a bad minimization of the cost function, but that is compensated by the fact that there will be much more steps, and taking the wrong direction sometimes means escaping local minimum.

5.3.2.2 German

German data was annotated separately by two annotators groups, so another pair of experiments were carried out. Unfortunately, authors in [81] did not develop any solution to classify texts as hateful or not, being unable to compare the results obtained. These results in terms of learning curves for experts group 1 and 2 are presented in Figures 5.9 and 5.10 respectively.

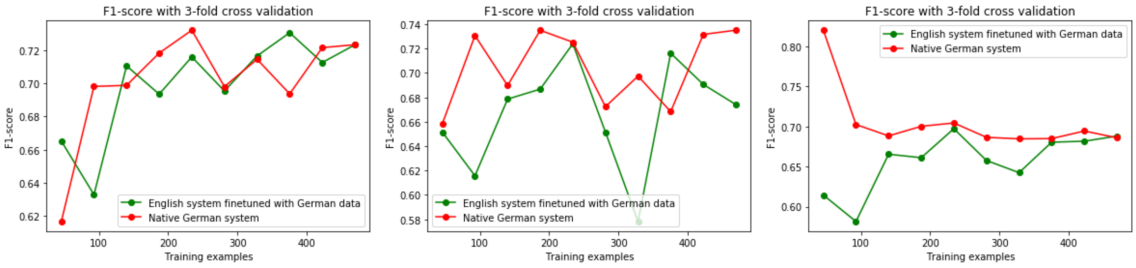

Figure 5.9: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for German annotation from experts group 1

Seeing the end of the curves, expert 2 seems to be easier to predict. With Logistic Regression a higher start is always achieved with the Transfer Learning system, while also the linear SVM has a higher start in the expert 2 annotation. Also notice the great results of native Random Forest at the beginning of expert 1, though when training finalizes Transfer Learning system matches the results. Although it is true that German data we have is very limited and the Transfer Learning curves should have a better shape, results are also affected by the imbalanced distribution of the data.

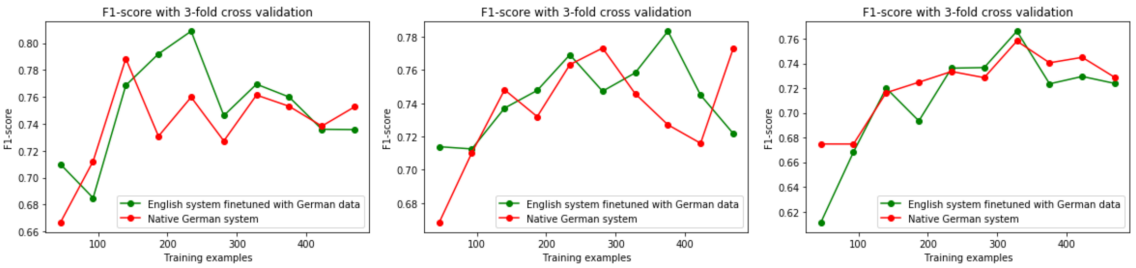

Figure 5.10: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for German annotation from experts group 2

5.3.2.3 Indonesian

Indonesian is one of those cases where low data is available, data size is not bigger than thousand samples. Therefore, this set of experiments can really test the Transfer Learning quality at early learning stages.

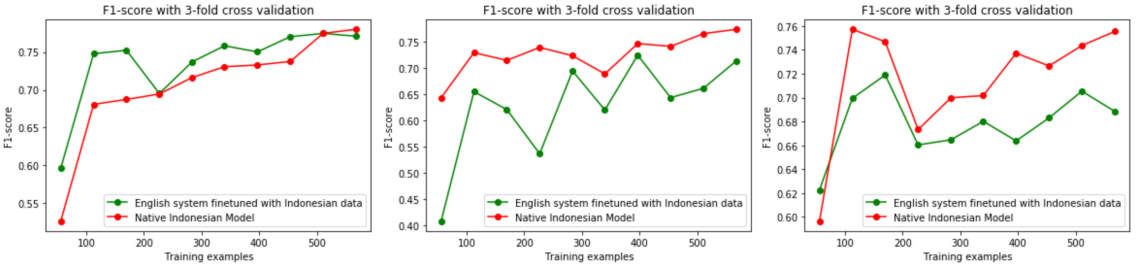

Figure 5.11: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Indonesian Instagram dataset

Graphics from Figure 5.11 represent the curves for the Instagram Indonesian dataset. In [77] the authors reported an F-measure of only 65.7%, consequently, our both approaches make a significative improvement of 8% approximately. Since we lack Indonesian linguistic understanding this is such a surprising thing. One more time, best results are achieved when using Logistic Regression, but in this case, the Transfer Learning system surpasses the native system during almost the whole training process.

Nevertheless, much more promising results are obtained in the Twitter data, as shown in Figure 5.12, where the difference in the start is much higher than in previous experiments. Also note that the three algorithms with Transfer Learning has a higher start. Besides, final F-scores obtained are really good, around 85%, though in the original research [6] they reached up to 93.5% of F-score [8].

[8]Results may not differ so much, but they evaluated their algorithm using a balanced distribution of the

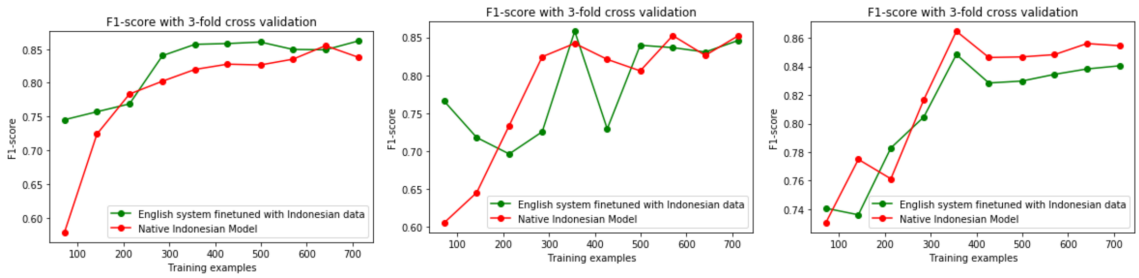

Figure 5.12: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Indonesian Twitter dataset

5.3.2.4 Portuguese

Portuguese experiments go in line with results already seen with other languages. Logistic Regression performs well, linear SVM behaves noisily, and Random Forest Transfer Learning system performs really bad. The three curves are presented in Figure 5.13. In relation to the data providers [34], both achieve similar results: around 77% of F-score.

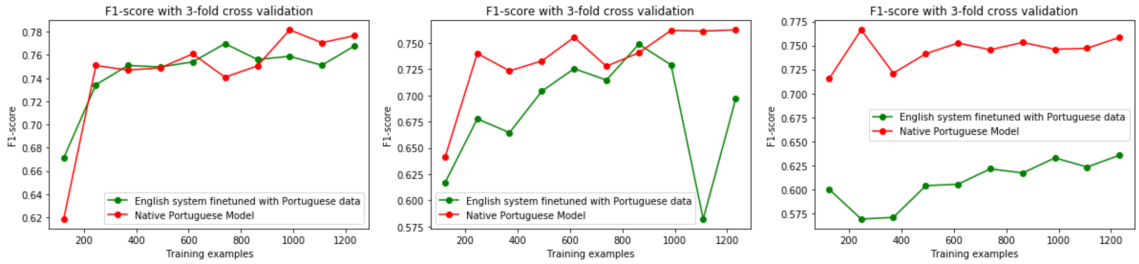

Figure 5.13: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Portuguese data

5.3.2.5 Italian

The final cross-language experiment is concerned with the Italian language. Data size is similar to that used in the Spanish case, however, some interesting results can be observed. As occurred with German data, we cannot establish a comparison, since we do not know the results from the Italian competition over the training data. Curves evaluated with Facebook data are included in Figure 5.14, while Twitter related evaluation can be found in Figure 5.15.

With Facebook data, the system with Transfer Learning that uses Logistic Regression outperforms native system throughout all the training process. In addition, this happens

dataset, so the comparison does not match exactly

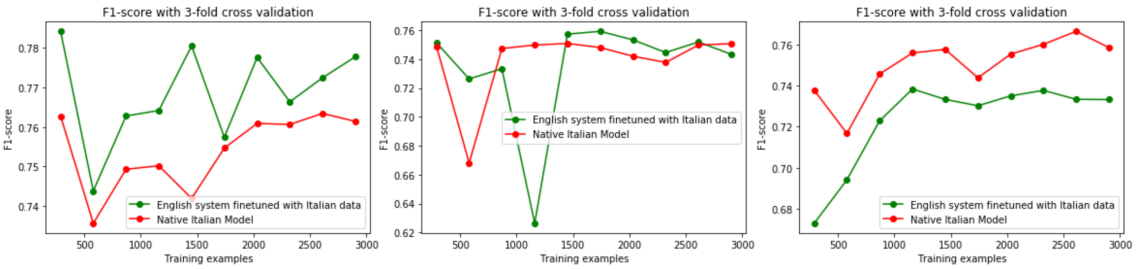

Figure 5.14: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Italian Facebook data

when working with 3,000 training examples, which excluding Spanish data is one more magnitude bigger than other languages data.

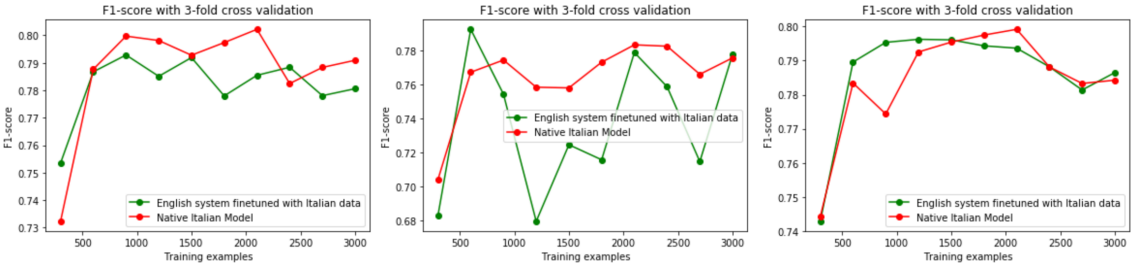

Figure 5.15: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Italian Twitter data

Twitter data results are better than Facebook data results regarding the F-score metric. This is an expected result because our text preprocessing is mainly oriented to the Twitter domain (the same thing occurred with Instagram and Twitter data in the Indonesian language). Again, Logistic Regression transferred system has a higher start.

Overall, results achieved during experimentation suggest that the performance of Logistic Regression is indeed positive in the studied scenario. Logistic Regression is a simple algorithm with also a simple training process what makes it few adapted to the data seen. Consequently, this makes a better adaptation in order to learn from new data than other algorithms that are much more adapted to the first training data. For this reason, Logistic Regression is a good algorithm to use in Transfer Learning problems.

5.4 Cross Domain Transfer Learning

The last set of experiments try to address a cross-domain Transfer Learning problem. Particularly, we are going to study aggressiveness and target goals of hateful tweets from those

provided in SemEval-2019 Task 5. The experiments consist of creating a system to detect aggressiveness and use it to detect the target and the same in opposite order. The workflow is easier than in previous experiments because a system trained to classify aggressive tweets can immediately be evaluated to predict the target, there is no word embedding behind.

We are going to test Spanish and English-speaking tweets, but the experiments will be mono-language, that is to say, the systems created will only be able to detect aggressiveness and target in one unique language. For better performance, the best systems developed during the competition will be used. Nevertheless, in order to make able the fine-tuning, it is necessary that features extracted for aggressive tweets and target ones must be the same. Therefore, we will use the features optimized to detect aggressiveness cause it was the label most difficult to predict and these results do not make target results to differ in a significative manner.

5.4.1 Results without Fine-tuning

In this section, we present the results when no fine-tuning process is applied. Systems trained in one domain are directly evaluated in the other domain, we also include the results for systems without Transfer Learning. As aforementioned, we are going to use SemEval data, training with the training set and testing over the test and development set. The combinatorial problem makes us computing 48 different values: 2 languages $\times$ 2 domains $\times$ 2 evaluation sets $\times$ 2 kind of systems (Transfer Learning and no Transfer Learning) $\times$ 3 algorithms = 48. These 48 values are separated into development and test evaluation in Tables 5.3 and 5.4 respectively.

The obtained results must be interpreted by only considering the only four results corresponding to an algorithm and can be observed in two directions, column-wise and row-wise. When looking column-wise we are comparing the Transfer Learning system with the native system for one label, while when looking row-wise we are seeing the performance of a system trained with one label in such label and the other label. Column-wise results are as expected, native system always achieves better results than systems with Transfer Learning.

On the other hand, row-wise results differ for each language. In the English case, the systems trained for a label always predict better that label than the other one. However, there are Spanish systems when the label used to train is not the best predicted. For example, in Table 5.4 algorithms trained with aggressiveness predict better the target. This may be due to that the target label is easier to predict, cause results achieved in this class are around 90% in all native cases.

—	Logistic Regression		Linear SVM		Random Forest	
Trained\Evaluated	AG	TR	AG	TR	AG	TR
English						
AG	0.705	0.44	0.707	0.55	0.681	0.40
TR	0.50	0.934	0.50	0.932	0.48	0.911
Spanish						
AG	0.813	0.23	0.830	0.7	0.779	0.81
TR	0.75	0.924	0.72	0.924	0.75	0.919

Table 5.3: Cross Domain Transfer Learning results in the development set

—	Logistic Regression		Linear SVM		Random Forest	
Trained\Evaluated	AG	TR	AG	TR	AG	TR
English						
AG	0.682	0.33	0.674	0.25	0.639	0.25
TR	0.40	0.892	0.39	0.895	0.39	0.896
Spanish						
AG	0.787	0.88	0.787	0.87	0.725	0.76
TR	0.70	0.902	0.78	0.900	0.76	0.869

Table 5.4: Cross Domain Transfer Learning results in the test set

5.4.2 Results with Fine-tuning

We now proceed to apply fine-tuning in the Transfer Learning process. When a system has been first built to detect one label, then training follows using the same features than before changing the label to learn. In this case, the number of graphics to plot is equal to 24 cause the Transfer Learning and native systems are grouped into one unique graphic. The

analysis is divided into languages and evaluation sets resulting in eight figures presented below. For each language there are four images, two evaluated with the development set and another two with the test one. Finally, each set is used in a Transfer Learning problem, aggressive model to detect target and target model to detect aggressive tweets.

We start by discussing English curves which are plotted in Figures 5.16, 5.17, 5.18, and 5.19. Generally, there are few successful cases where Transfer Learning advantages can be seen. These cases are concentrated in Figure 5.18, where Logistic Regression and Linear SVM Transfer Learning systems achieve better scores at early stages. In addition, native systems in the experiments with the test set have a notable step around 2,000 training examples.

Pearson correlation coefficient gives a measure of how similar are two variables, applying it to this case, we obtain a 0.04 in the development set and a 0.21 in the test set. This result suggests that the two tasks are quite different in the development set and therefore applying transfer learning as has been done does not serve to improve a system without Transfer Learning. In the test set correlation between variables is better, so we have been able to see better curves.

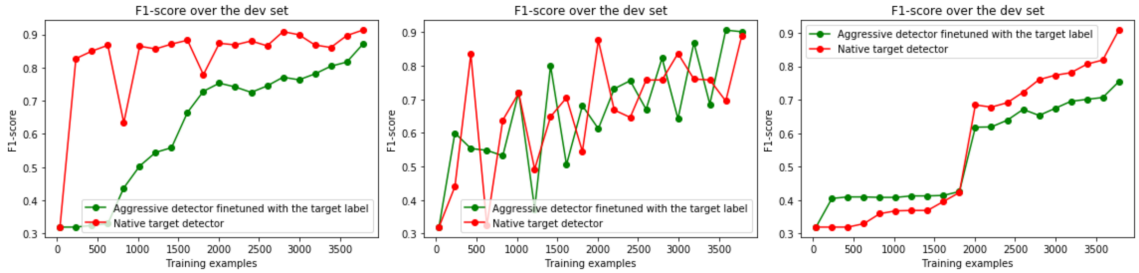

Figure 5.16: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English development set and transfer from aggressive to target learning

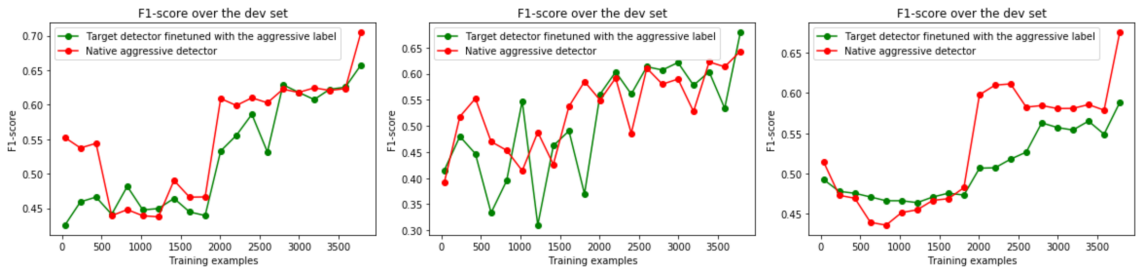

Figure 5.17: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English development set and transfer from target to aggressive learning

Better results can be observed in Spanish curves, represented in Figures 5.20, 5.21, 5.22,

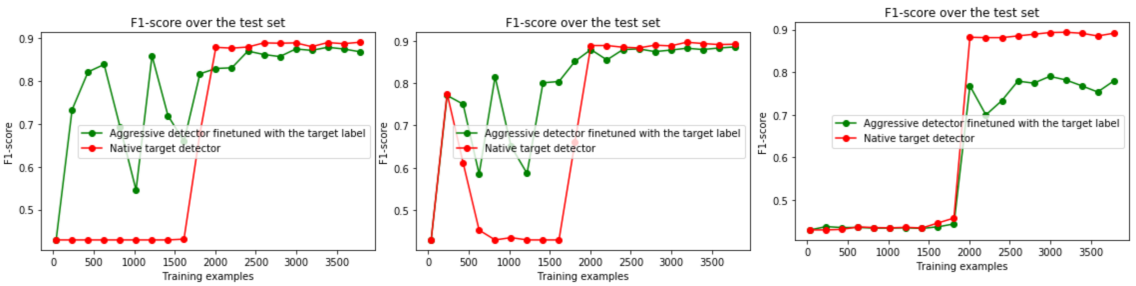

Figure 5.18: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English test set and transfer from aggressive to target learning

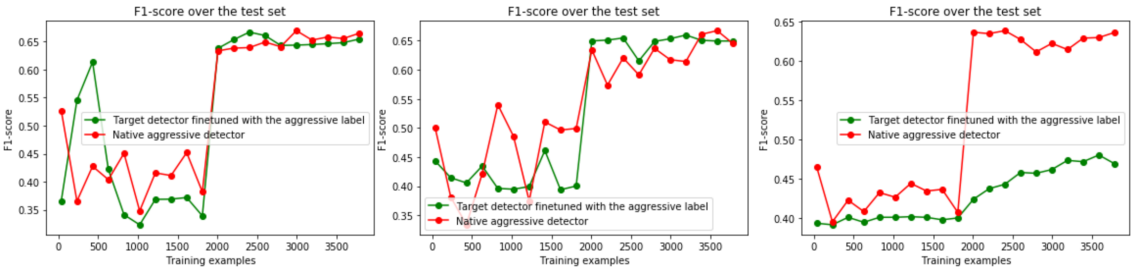

Figure 5.19: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for English test set and transfer from target to aggressive learning

and 5.23. First of all, unlike the previous case, it can be seen a better start of some systems with Transfer Learning and some of them are significant (see for example Linear SVM in Figure 5.23). Secondly, even there is no such better start, some Transfer Learning curves grow really fast surpassing the native ones in few iterations (see for example Logistic Regression and Linear SVM in Figure 5.23). Lastly, Random Forest with Transfer Learning is always above in the last experiment.

This improvement can also be seen as an improvement in the correlation between the variables for the Spanish language. In this case, the Pearson correlation coefficient is of 0.42 for the development set and 0.47 for the test set, which are really high correlation values and consistent with the results observed.

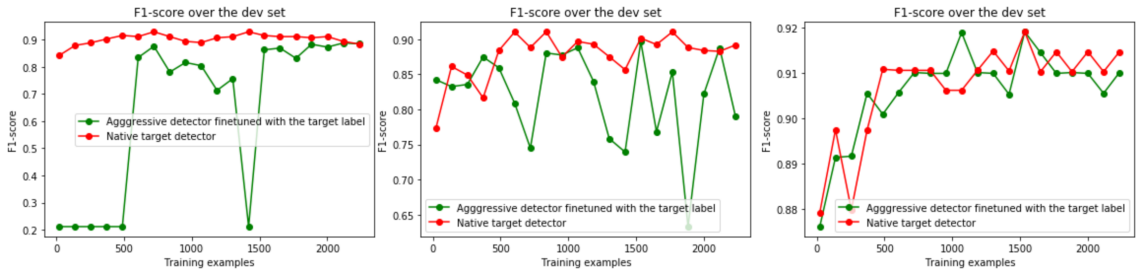

Figure 5.20: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish development set and transfer from aggressive to target learning

Figure 5.21: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish development set and transfer from target to aggressive learning

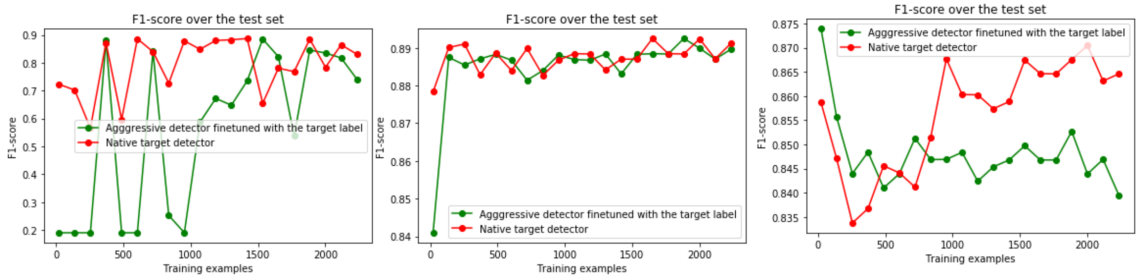

Figure 5.22: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set and transfer from aggressive to target learning

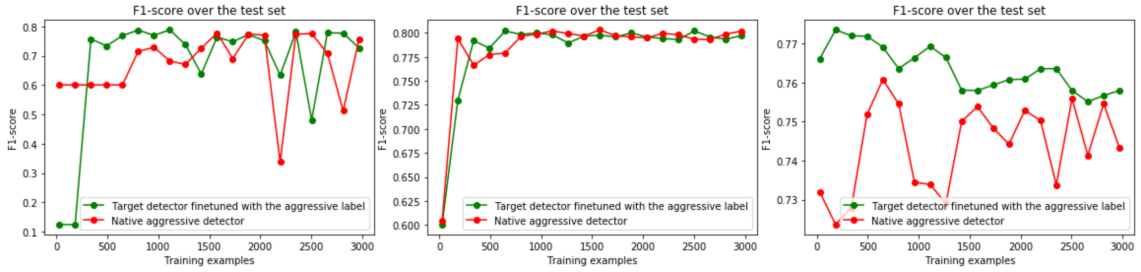

Figure 5.23: Logistic Regression (left), Linear SVM (center), and Random Forest (right) curves for Spanish test set and transfer from target to aggressive learning

CHAPTER 6

Conclusions

This final chapter presents the conclusions derived from this master thesis, as well as the achieved goals, problems faced and future work that can be initialized from this project. The conclusions offer some final thoughts about the development of this project. Then, with the achieved goals, the main results that were obtained are pointed out. Next, main problems encountered during the realization of this project are described. Finally, As for the future work, some lines of research are outlined, with special attention to the Transfer Learning and neural network fields, in which many of this master thesis work is inspired.

6.1 Conclusions

Social networks platforms contain huge amount of information which is really hard to process manually. The proliferation of hate speech within this information needs to be filtered and automated tools are required. The solutions developed mainly relies on NLP and Machine Learning tools in order to ease such treatment automatically. New emerged paradigms based on deep learning technologies have been explored, specially those concerned with word embeddings and semantic similarity. As seen throughout the previous reading, our project has been evaluated against two different study cases, the participation at SemEval-2019 Task 5 and the Transfer Learning for hate speech detection research.

Regarding the SemEval-2019 Task 5, we presented our system in the famous international competition of SemEval, which the task addressed [12] revolves around analyzing text messages from Twitter in order to detect hate speech against immigrants and women. The different features that feed this system have been thoroughly evaluated, considering its suitability in the field of hate speech detection. It has been seen that both novel and traditional approaches do not yield so promising when used separately. Nevertheless, properly combining several types of features, as well as with content analysis features (e.g., sentiments and subjectivity) can improve the system to the point of reaching a reasonably good performance.

Overall, results confirm that hate speech detection against women and immigrants in micro-blogging texts is challenging, with a large room of improvement. Hate speech concept has demonstrated its complexity, which significantly requires better definitions and guidelines in order to be annotated reliably. For the same reason, researchers might want to consider hate speech detection a regression problem, predicting, for example, the degree of hatefulness of a message, instead of a binary yes-or-no classification task. Language dependence on the context in which it is expressed makes it a rather complicated task to solve by simply looking at keywords. Swear words used in a humorous context and hate speech phrased in a slightly convoluted way are common error sources.

Task 5 has been one of the most popular tasks in SemEval-2019, which confirms the growing interest of the community around abusive language in social media and hate speech detection in particular. Besides, the multilingual perspective contributed to make the task accessible to a wider community of scholars.

Concerning the Transfer Learning problem, we described the principles of Transfer Learning and applied them by tackling out two transfer approaches, Transfer Learning across languages and across domains. Firstly, when working with Transfer Learning be-

tween languages, an English trained model was used to predict hate speech in different languages. Then, we showed how to afford this problem using a technique strongly based on multilingual aligned vectors. Compared to this, the second Transfer Learning approach is much more trivial, where a system trained to detect aggressive hateful tweets can be immediately used to detect target or not, hateful tweets.

Furthermore, with the aim of evaluating the proposed models, the corresponding native models without Transfer Learning were built for comparison. In order to do this, trained systems are fine-tuned with the target task data and learning curves of both systems are plotted in the same graphic with the aim of facilitating the visual comparison. Through this evaluation, we empirically verified some benefits of using Transfer Learning to solve some specific tasks.

6.2 Achieved Goals

The development of the project has successfully reached the proposed goals. Firstly, we have reached some degree of expertise with the hate speech domain. An extensive current state-of-the-art research has been carried out to know the fundamentals of this field.

Then we decided to participate at SemEval-2019 Task 5, where we got the first contact with practical hate speech detection, and we reached 5th place on the Spanish sub-task A, being only 0.5% apart from the best performing system. This is, undoubtedly, a promising result that highlights the capacity of the proposed method to obtain nearly state-of-the-art performance in this task. When comparing with the same sub-task in the English case, in which we scored lower, it is necessary to study further the applicability of the system to different languages.

For that reason, a multilingual experiment was carried out to see how to take advantage of the learning acquired in one language to apply it to another language from scratch, or with few data in the target language. Experiments addressed has shown the improvement of using Transfer Learning when target language data is scarce or even does not exist.

6.3 Problems Faced

During the development of this project, we had to face some problems. These problems are listed below:

- **Lack of knowledge:** the development of the project strongly depends on really com-

plex theoretical concepts related to neural networks in the NLP field. At early stages of the thesis, these concepts were not mastered at all, so it was necessary to include study phases to understand the theoretical fundamentals in which this project relies on. The knowledge acquired helped to improve the analysis, troubleshoot emerging errors efficiently, and make more accurate interpretations of the results obtained.

- **Out of vocabulary words:** This is a limitation of word embeddings models, which forces a word to be seen in the training data in order to have an embedding vector. But not all words are inside the training data, so these words do not have an embedding vector associated and are called out of vocabulary words. Handling out of vocabulary words can be done in different manners, specifically in this project out of vocabulary words are mapped to all null values vectors. A derived problem of out of vocabulary words appears when the input text is compound by only out of vocabulary words. These samples are assumed to be very rare and can be removed from the analysis.

- **Errors in user-generated content:** this is a frequent problem encountered by any project that works with social network data. People share their opinions on social networks committing grammatical errors and/or with an incorrect language structure. Order-less approaches such us BOW or TF-IDF do not care about language structure and are not affected by inconsistent text inputs. On the other hand, the effect of misspelling errors can be attenuated using features at the character level. Writing a word badly can cause it to become an out of vocabulary words. Besides, special characters, including non-alphanumeric characters need to be cleaned at the text preprocessing step.

- **Language ambiguity:** ambiguity is a characteristic and essential language requirement in order to afford world complexity and avoid having a word for each thing. Words can have several meanings and the associated meaning at each moment will depend on the human interpretation or the context in which it is being expressed. Ambiguity affects when working with aligned vectors, where we assumed to use the same lexicon embedding vectors extracted with the English embedding in the target languages. While it is true that this decision is correct due to alignment, when calculating similarity between words, the meaning considered is the one provided by the word embedding model at hand that could be making a wrong interpretation. For example, terms that refer to certain Spanish female animals usually have negative connotations. Therefore, should have a similar position in the vectorial space than slang terms, but instead, these words are near other animal words. The fine-tuning process in the target language can be seen as a partial solution to the ambiguity problem.

- **Imbalanced data:** the last problem consists on the predominance of harmless texts in the overall sets of data used. The amount of representation of each class can cause classifiers to predict more frequently the majority class. Thanks to the library Imbalanced-learn, some under-sampling techniques have bee tested in order to obtain the optimal performance of the algorithms.

6.4 Future Work

Results obtained in the English tasks of SemEval evidence that hate speech detection still represents an open challenge with great room of improvement. As future work, several lines of work could be addressed. Firstly, we plan to implement deep learning architectures which have shown to obtain better results in previous research [101, 102]. These architectures have the advantage of saving the feature extraction step because it is done automatically. It works by means of an embedding matrix at the input that feeds a Convolutional Neural Network which applies several filters to extract the features. Each row in the embedding matrix is the corresponding embedding vector of each word in the text input. Another benefit of using deep learning is that embedding vectors are seen as weights in the network and are actualized during the training process, so they are fine-tuned to solve the task addressed. Also, context aware approaches [36] could represent an improvement, since having general knowledge of hate speech (e.g., anti- LGBT or racism) may boost the performance of learning systems.

In addition, in order to afford imbalanced distributions, data augmentation [46] techniques could be explored. Data augmentation generates new samples of data instead of selecting a sub-sample of the original data. New hateful text samples can be generated using the original hateful texts and replacing synonyms on the basis of two criteria: the grammar category and the cosine similarity.

Finally, more refinement could be done in relation to the work carried out with word embeddings. Initial training of a word embedding model does not consider language ambiguity and hate speech problem, so fine-tuning word vectors as aforementioned can modify the vectorial space to represent properly the correct meaning of the words. But then, the alignment between language vectors are not consistent and same transformation must be applied to other models. In order to conclude, a more elegant way of treating out of vocabulary words could be implemented. Given an out of vocabulary word and the sentence it is in, language modelling is used to sequence words in the sentence and predict the meaning of the word by comparison with similar sentences.

APPENDIX A

Ethical, Economical, Social and Environmental Impact

A.1 Introduction

This appendix will show the Ethical, Economical, Social and Environmental impacts of this project. In order to clarify the identification of impacts and problems, the context in which the project has been developed should be defined beforehand. This thesis is framed within the sector of Information Technology (IT), specifically, there has been made a research work on the applicability of novel technologies, such as NLP and Machine Learning. As it has been described in the document, the project is mainly centered in the context of the social networks in order to study online hate speech. Some stakeholders may be affected by the realization of the project and the whole analysis will be focused around them.

The interested parties analyzed will be directly or indirectly affected by the development and implementation of the project in any of its phases, both positive and negative. We start from the idea that most of the institutions are positioned in favor of the elimination of hate speech content. The beneficiaries left by the development of this thesis are listed below.

- **Public authorities:** Govern and Administrations must promote a citizen coexistence based on respect and harmony.
- **Social networks:** they have to protect decent users because otherwise the will become unused platforms or the use they receive will be different from the one they intended. Besides, the European Union Comission has obliged them to do it (See Section 1.2).
- **Researchers:** the study carried out here can serve as initialization for further research.
- **Hate speech victims:** They are not willing to be discriminated.

On the flip side, hate speech authors, sites promoting hate speech including white supremacist forums and similar will be negatively affected cause their content could be possibly banned or blocked. Following, each of the dimensions is evaluated, but due to the nature of this project, the realization of this impact makes sense specially concerning social and ethical impacts, so they will be more deeply analyzed.

A.2 Ethical Impact

This project has clear ethical implications as it gathers data from public profiles on social networks and the Internet and tries to classify their content according to whether they contain any hate speech utterance or not. The automatic classification of online posts carried out in this project has several ethical issues: freedom of speech, information privacy, and human job dignity. The last of them also affects in the economic impact as seen in Section A.3.

Due to advances in technology, machines are automating human tasks, so they are being replaced in order to save costs. We believe that a system such as ours provokes jobs reducing and a transformation of jobs because although the system does work that until now has been manual, it requires people to maintain it and analyze the results it presents. The remaining two dimensions are more in-depth studied in the following sections.

A.2.1 Freedom of Speech

Freedom of speech is a right that supports the freedom of an individual or a community to articulate their opinions and ideas without fear of retaliation, censorship, or legal sanction.

Our system can cause some individuals and online hate communities are deprived of the right to freedom of expression because if their content is identified as hateful should be banned or blocked. In this case, the suppression of this right is justified since some limits on expression must be contemplated and hate speech must be treated as an exception. In fact, hate speech violates other human rights relative to justice, non-discrimination and equal treatment regardless of age, gender, sexual orientation, social class, race, ethnicity, religion, disability, etc. This right states that no one should be self-conscious or attacked for showing himself as he is. It is true that the system could wrong classify non-hateful content as hateful one and no blame users would be adversely affected. In that case, human intervention should be needed to solve the problem.

Another aspect to consider is hate crime, which is a crime motivated by hate speech. Therefore, hate speech has entered into the legislation and The International Covenant on Civil and Political Rights (ICCPR) states that "any advocacy of national, racial or religious hatred that constitutes incitement to discrimination, hostility or violence shall be prohibited by law". On top of this, many countries have developed their own laws to fight against hate speech, Spain has the following considerations [41] in relation to hate speech into its penal code:

- Threats to a group with an evil that constitutes a crime
- Provocation of discrimination, hatred or violence against groups or associations and dissemination of insulting information about groups or associations
- Unlawful association to promote discrimination, hatred or violence against persons, groups or associations
- Crimes against freedom of conscience and religious feelings
- Discrimination in the workplace

A.2.2 Information Privacy

The new General Data Protection Regulation (GDPR) [88], which limit of appliance ended in May 2018, has raised awareness around personal data and information privacy. Personal data is any information that serves to identify a natural person, which is a very broad definition and makes that almost any data can be considered as personal. For example, knowing if a person is a hate speech author is personal data. This project makes use of public data available via Internet and social networks which are also protected by the regulation. Users consent their data to be treated when publishing content in the platform

on the social network. There are exceptions because users can make their profile private and remove the content they want. Special attention requires what is considered sensitive data, which includes information related to health, political beliefs and also information relating to the commission of a crime. It can be considered that we are using sensible information because of hate crimes. The project only works with text information and the associated annotation information, so data is anonymized and anonymous data is not under GDPR scope. On the other hand, when using social network data, we must adhere to what is stated in the social network's privacy policy and be aware of the rights of users so that they do not conflict with our treatment. We have been careful about using only the data needed and no information is related to any real person.

We have seen that our treatment is legal, but also we have to analyze if is ethic. Maybe persons want their content to be public to interact with friends and/or relative, but not for data analytics and/or text mining. It is true that our system contributes to equality, but the goal does not justify the means. Social networks could help in a solution by offering the possibility of avoiding the automatic processing of user data (even if it is public) as they offer the possibility of making an account private.

A.3 Economic Impact

This section asses the possible economic impact that companies and public institutions may experience by using the ideas developed in this project. As told in the previous section, the process of machine automating tasks can lead to savings in personnel costs. On the other hand, acquisition of new equipment to deploy de system will cause new additional costs. It will then be discerned whether the automation is more cost-efficient.

The cost of manually doing all the tasks performed by the system would be extremely high with respect to the automated approach. Using some Twitter stats as example [1], 326 million of users should be monitored, which post around 500 millions tweets per day. On top of this, the amount of human resources to tackle the problem in a single social network is vastly huge. The transfer of this effort to the technology can reduce the people needed since a machine can do the work of several people at once. Assuming that a machine is cheaper than several people and that the number of machines can be updated depending on the load (specially in cloud environments), the developed system is cost-effective against other methodologies. Finally, despite all the above, human labour is always needed cause

[1] https://s22.q4cdn.com/826641620/files/doc_financials/2018/q3/TWTR-Q3_18_InvestorFactSheet.pdf

the automated system is not perfect and may fail, so companies will be helped considerably and humans shall intervene in a few cases.

A.4 Social Impact

The social impact in this project is also important cause we have worked with a social environment as social networks and Internet are. Due to the popularity reached by these social communities during the last few years, many studies have emerged in order to draw conclusions about human behaviour as the one presented in this thesis.

In our specific case, the study carried out around hate speech can serve the scientific community to make further analysis and research in this field. Specifically, the guidelines given in the future work section can be used as initialization for improvements in terms of deep learning and Transfer Learning applied to hate speech. Besides, hate speech is a really good candidate to test new discoveries to come in computer science and artificial intelligence because of the challenging nature of this phenomenon.

Another interesting social perspective of this project is the quest for equality of this project. The developed tool contributes to the fulfillment of human rights in terms of equality non-discrimination and culture diversity. This is achieved by making research for automating identifying non-respectful content in as many languages as possible. We have made analysis in six languages due to data availability, but thanks to word embeddings alignment, we have the potential to extend the system until 30 languages.

In addition, the thesis has also been developed within a social environment with the participation at SemEval-2019 Task 5. Different teams of different countries of different native language participated. All of them contributing to find the best solution to the automation of hate speech detection, some of them also sharing their findings to help in the investigation of this field.

A.5 Environmental Impact

The effect over the environment of this project is not clear, except for the computing power needed for running the described experiments. Computers and other information technology systems need electrical energy in order to work properly. Personal computers and shared

pool of resources in a cloud environment (specifically, Google Colab [2] and GSI cluster [3] have been used) have made possible the realization of this project. In addition to this consumption, the energy required for the cooling system associated with this equipment must also be added. The process of generating this electrical energy can result in the emission of greenhouse gases depending on the type of generation. In Spain, almost 40% (2018) of electricity consumption comes from technologies that emit CO2 [4]. Therefore, 2/5 parts of our consumption generate CO2, which according to an online calculator [5] are 148 kg of CO2 [6].

[2] https://colab.research.google.com

[3] https://hub.gsi.upm.es

[4] https://www.diariorenovables.com/2019/01/generacion-electrica-en-espana-2018_17.html

[5] https://www.ceroco2.org/calculadoras/electrico

[6] We have considered a total consumption of 400 kWh summing up the personal computer, cloud services, and cooling systems.

APPENDIX B

Economic Budget

B.1 Introduction

When carrying out a project, a series of costs are incurred which define the final budget of the project. In this appendix, we evaluate the possible costs that the development of this system could have. These costs come mainly from the salaries of the developers, although some costs have been provoked by hardware needs. Besides, the taxes involved will be outlined.

There are many types of costs faced by a project. It is important to clearly know what each of them consists of in order to analyze how they affect us correctly. The costs incurred in this project can be classified according to how they affect it:

- **Direct costs:** are costs attributable to the project at hand.
- **Indirect costs:** are costs not attributable to the project we are developing and are transversal to other projects.

They also can be classified depending on their behaviour:

- **Fixed costs:** are costs that do not change over time.

- **Variable costs:** are costs that depend on the production activity of the project. The higher the latest, the higher the variable cost

Following, the different costs of this project are described.

B.2 Physical Resources

This project requires a personal computer for its development and a server to run the experiments. The server is the most resource demanding, so the requisites of the computer can be relaxed. The computer used needs the following:

- **Hard disk:** 1 TB
- **RAM:** 8 GB
- **CPU:** Intel i5 processor, 3.20GHz $\times$ 4

On average, a machine with such capabilities costs around 600€ as of 2019. The requirements from the server are much stricter. The GSI cluster consists of two computers with the following capabilities.

- **Hard disk:** 3 TB
- **RAM:** 16 GB
- **CPU:** Intel Xeon E5-2430 v2 2,50GHz $\times$ 6

The price of each computer reach 2,000€, but the cluster is used by other people, and we can consider that we have used only one of those computers with an associated cost of 1,000€.

These machines are acquired at the beginning of the project, and they do not produce more costs during the development, so they are catalogued as fixed costs, which summing up reaches the amount of 1,600€.

B.3 Human Resources

Human resources are a variable cost which depends on time, so it is necessary to calculate the total time needed to develop the project. This Master thesis has assigned 30 ECTS

credits [1], which approximately compute as 30 work hours, so the amount of work in the development would be of 900 hours. Assuming that each month has 30 days, of which 22 are working days and each working day is 8 hours, then the total duration of the project are five months.

We suppose to have two employees, a software engineer for developing the whole solution and another engineer for configuring the server and maintaining it. No experience is needed, only some notions about programming and Machine Learning, so we suppose a gross monthly salary of 1,300€ for each one. The maintenance person could also be focused on similar tasks for other projects, so his associated cost in this project would be around 500€. The final cost due to human resources is: (1,300 + 500) × 5 = 9,000€.

B.4 Indirect Costs

Other costs considered are associated with the activity of the project. These are indirect costs and include Internet connection, electricity, water, and gas. These costs are not unique for this project, they are shared among other projects, so we will only consider a 10% of the total indirect costs.

The Internet connection used has a velocity of 100 Mbps which is valued with a price of 30€ monthly. Electricity price in Spain is 0.11854€/kWh, so the cost assumed can be calculated knowing the kWh consumed (In this case the 10% do not have to be applied, cause the direct cost associated to the project is calculated). Approximately half of the kWh spent (400kWh) in the project are generated by us and the total electricity cost is then: 200 kWh × 0.11854€/kWh = 23.71€.

The bill associated with the natural gas under consideration is 15€ per month, as well as the consideration for water, is 10€ per month. Finally, total indirect costs are: (1.5 + 1) × 5 + 23.71 = 36.21€.

Having shown all the costs incurred, we are now in good position to calculate the total cost: 1,600 + 9,000 + 36.21 = 10,636.21€, a part of this budget is caused by Spain imposed taxes, which are accounted in the next section.

[1] https://ec.europa.eu/education/resources-and-tools/european-credit-transfer-and-accumulati
en

B.5 Taxes

This final section assesses the taxes involved in the project. The entire set of costs outlined in the previous sections have an associated tax which is expressed as percentage of the total cost. Taxes to consider are:

- 21% of VAT in computer systems, electricity and gas. VAT in water is a 10%.
- Electricity tax additional to VAT of 5.113%.
- Tax on hydrocarburos (natural gas) of 2.34%.
- 23.6% of gross salary to social security.

On top of this, the total taxes are:

0.21 ·1,600 + 0,236·9,000 + 0.05113·23.71 + 0.21·(23.71 - 0.05113·23.71) + 0.0234·7.5 + 0.21·(7.5 - 0.0234·7.5) + 0.1·5 = 2,468.15€.

Bibliography

[1] Facebook community policy. https://www.facebook.com/communitystandards/hate_speech. Accessed October 17, 2018.

[2] Twitter hateful conduct policies. https://help.twitter.com/es/rules-and-policies/hateful-conduct-policy. Accessed October 17, 2018.

[3] Youtube hate speech policy. https://support.google.com/youtube/answer/2801939?hl$=$en. Accessed October 17, 2018.

[4] Swati Agarwal and Ashish Sureka. Characterizing linguistic attributes for automatic classification of intent based racist/radicalized posts on tumblr micro-blogging website. *arXiv preprint arXiv:1701.04931*, 2017.

[5] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45 – 65, 2003.

[6] I. Alfina, R. Mulia, M. I. Fanany, and Y. Ekanata. Hate speech detection in the indonesian language: A dataset and preliminary study. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 233–238, Oct 2017.

[7] Ika Alfina. Indonesian hate speech detection. https://github.com/ialfina/id-hatespeech-detection, 2018.

[8] James Allen. *Natural language understanding*. Pearson, 1995.

[9] Oscar Araque, Ignacio Corcuera-Platas, J. Fernando Sánchez-Rada, and Carlos A. Iglesias. Enhancing deep learning sentiment analysis with ensemble techniques in social applications. *Expert Systems with Applications*, 77:236 – 246, 2017.

[10] Oscar Araque, Ganggao Zhu, and Carlos A. Iglesias. A semantic similarity-based perspective of affect lexicons for sentiment analysis. *Knowledge-Based Systems*, 165:346 – 359, 2019.

[11] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee, 2017.

[12] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Rangel, Paolo Rosso, and Manuela Sanguinetti. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval-2019)*. Association for Computational Linguistics, 2019.

[13] Yoshua Bengio and Yann LeCun, editors. *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[14] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[15] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[16] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016.

[17] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.

[18] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[19] Mark Bridge. Facebook fails to delete hate speech and racism. `https://www.thetimes.co.uk/article/facebook-fails-to-delete-hate-speech-and-racism-hwrzw0qzn`. Accessed October 19, 2018.

[20] Pete Burnap and Matthew L Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242, 2015.

[21] Pete Burnap and Matthew L Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11, 2016.

[22] Peter Burnap and Matthew Leighton Williams. Hate speech, machine classification and statistical modelling of information flows on twitter: Interpretation and communication for policy decision making. 2014.

[23] Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, March 2016.

[24] Ying Chen. Detecting offensive language in social medias for protection of adolescent online safety. 2011.

[25] CLiPS. Hades. `https://github.com/clips/hades`, 2016.

[26] European Union Commission. Framework decision on cobating certain forms and expressions of racism and xenophobia by mean of criminal law. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM:l33178`. Accessed October 17, 2018.

[27] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. *CoRR*, abs/1710.04087, 2017.

[28] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[29] Crowdflower. Hate speech identification. https://data.world/crowdflower/hate-speech-identification, 2017.

[30] Thomas Davidson. Automated hate speech detection and the problem of offensive language. https://github.com/t-davidson/hate-speech-and-offensive-language, 2017.

[31] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language, 2017.

[32] Ona de Gibert, Naiara Pérez, Aitor García Pablos, and Montse Cuadros. Hate speech dataset from a white supremacy forum. *CoRR*, abs/1809.04444, 2018.

[33] Luiz Felipe de Melo et al. Hate speech. https://github.com/lffloyd/HateSpeech, 2018.

[34] Rogers Prates de Pelle and Viviane P. Moreira. Offensive comments in the brazilian web: a dataset and baseline results. In *6º Brazilian Workshop on Social Network Analysis and Mining (BraSNAM 2017)*, Porto Alegre, RS, Brasil, 2017. SBC.

[35] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. 2017.

[36] Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. Common sense reasoning for detection, prevention, and mitigation of cyberbullying. *ACM Trans. Interact. Intell. Syst.*, 2(3):18:1–18:30, September 2012.

[37] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM, 2015.

[38] Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth M. Belding. Hate lingo: A target-based linguistic analysis of hate speech in social media. *CoRR*, abs/1804.04257, 2018.

[39] Manuela Sanguinetti et al. Italian twitter corpus of hate speech. https://github.com/msang/hate-speech-corpus, 2018.

[40] C. Fellbaum and G.A. Miller. *WordNet: An Electronic Lexical Database*. Language, speech, and communication. MIT Press, 1998.

[41] C Güerri Ferrández. La especialización de la fiscalía en materia de delitos de odio y discriminación. *InDret*, 1:1–33, 2015.

[42] Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85, 2018.

[43] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015.

[44] Edel Greevy. *Automatic text categorisation of racist webpages.* PhD thesis, Dublin City University, 2004.

[45] Edel Greevy and Alan F Smeaton. Classifying racist texts using a support vector machine. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 468–469. ACM, 2004.

[46] Konstantin Hemker. Data augmentation and deep learning for hate speech detection. Master's thesis, Imperial College London, 2018.

[47] Alex Hern. Facebook, youtube, twitter and microsoft sign eu hate speech code. `https://www.theguardian.com/technology/2016/may/31/facebook-youtube-twitter-microsoft-eu-hate-speech-code`. Accessed October 16, 2018.

[48] Michael Herz and Peter Molnar. *The content and context of hate speech.* Cambridge University Press, 2012.

[49] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[50] ILGA-Europe. Hate crime & hate speech. `https://www.ilga-europe.org/what-we-do/our-advocacy-work/hate-crime-hate-speech`. Accessed October 17, 2018.

[51] Jigsaw. Perspective api. `https://www.perspectiveapi.com/`, 2017.

[52] Gustavo Ariel Kaufman. *Odium dicta. Libertad de expresión y protección de grupos discriminados en internet.* México, DF: CONAPRED, 2015.

[53] Hideto Kazawa, Tomonori Izumitani, Hirotoshi Taira, and Eisaku Maeda. Maximal margin labeling for multi-topic text categorization. In *Advances in neural information processing systems*, pages 649–656, 2005.

[54] P. Koehn. *Statistical Machine Translation.* Cambridge University Press, 2009.

[55] Ivana Kottasová. Europe says twitter is failing to remove hate speech. `https://money.cnn.com/2017/06/01/technology/twitter-facebook-hate-speech-europe/index.html`. Accessed October 19, 2018.

[56] Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In *AAAI*, 2013.

[57] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[58] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA, 2005. ACM.

[59] Shuhua Liu and Thomas Forss. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management-Volume 1*, pages 530–537. SCITEPRESS-Science and Technology Publications, Lda, 2014.

[60] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[61] Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*, 2014.

[62] C.D. Manning, C.D. Manning, H. Schütze, and H.A. SCHUTZE. *Foundations of Statistical Natural Language Processing*. Mit Press. MIT Press, 1999.

[63] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[64] W. McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 2012.

[65] Yashar Mehdad and Joel Tetreault. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, 2016.

[66] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Bengio and LeCun [13].

[67] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *CoRR*, abs/1712.09405, 2017.

[68] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.

[69] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee, 2016.

[70] Council of Europe. No hate speech movement. `https://www.coe.int/en/web/no-hate-campaign`. Accessed October 16, 2018.

[71] Aitor García Pablos. Hate speech dataset from a white supremacist forum. `https://github.com/aitor-garcia-p/hate-speech-dataset`, 2018.

[72] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[73] 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, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[74] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[75] Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, and Cristina Bosco. Hate speech annotation: Analysis of an italian twitter corpus. In *4th Italian Conference on Computational Linguistics, CLiC-it 2017*, volume 2006, pages 1–6. CEUR-WS, 2017.

[76] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[77] N. I. Pratiwi, I. Budi, and I. Alfina. Hate speech detection on indonesian instagram comments using fasttext approach. In *2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 447–450, Oct 2018.

[78] Nur Indah Pratiwi. Instagram comments dataset for hate speech detection. `https://github.com/wiwaaw/dataset-hate-speech-instagram`, 2018.

[79] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[80] E. Reiter and R. Dale. *Building Natural Language Generation Systems*. Building Natural Language Generation Systems. Cambridge University Press, 2000.

[81] Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *CoRR*, abs/1701.08118, 2017.

[82] Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. An italian twitter corpus of hate speech against immigrants. In *Proceedings of the 11th Conference on Language Resources and Evaluation (LREC2018), May 2018, Miyazaki, Japan*, pages 2798–2895, 2018.

[83] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, 2017.

[84] Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, Mar 1966.

[85] Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. Analyzing the targets of hate in online social media. In *ICWSM*, pages 687–690, 2016.

[86] Aaron Smith and Monica Anderson. Social media use in 2018. http://www.pewinternet.org/2018/03/01/social-media-use-in-2018/. Accessed October 16, 2018.

[87] D. Suarez, O. Araque, and C. A. Iglesias. How well do spaniards sleep? analysis of sleep disorders based on twitter mining. In *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, pages 11–18, Oct 2018.

[88] The European Parliament and of the Council. Regulation (EU) on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. http://data.europa.eu/eli/reg/2016/679/oj.

[89] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.

[90] Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. A dictionary-based approach to racism detection in dutch social media. *arXiv preprint arXiv:1608.08738*, 2016.

[91] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[92] UCSM. Iwg hatespeech public. https://github.com/UCSM-DUE/, 2016.

[93] S. van der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30, March 2011.

[94] A. H. Wang. Don't follow me: Spam detection in twitter. In *2010 International Conference on Security and Cryptography (SECRYPT)*, pages 1–10, July 2010.

[95] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics, 2012.

[96] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, LSM '12, pages 19–26, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[97] Zeerak waseem. Twitter annotations. https://github.com/ZeerakW/hatespeech, 2016.

[98] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics.

[99] Yahoo! Webscope datasets. https://webscope.sandbox.yahoo.com/, 2017.

[100] Shuhan Yuan, Xintao Wu, and Yang Xiang. A two phase deep learning model for identifying discrimination from tweets. In *EDBT*, pages 696–697, 2016.

[101] Ziqi Zhang and Lei Luo. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *CoRR*, abs/1803.03662, 2018.

[102] Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on twitter using a convolution-gru based deep neural network. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 745–760, Cham, 2018. Springer International Publishing.