FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Feature extraction and selection for automatic hate speech detection on Twitter

João Guilherme Routar de Sousa

Mestrado Integrado em Engenharia Informática

Supervisor: Sérgio Nunes

Co-Supervisor: Paula Fortuna

March 25, 2019

Feature extraction and selection for automatic hate speech detection on Twitter

João Guilherme Routar de Sousa

Mestrado Integrado em Engenharia Informática

March 25, 2019

Abstract

In recent decades, information technology went through an explosive evolution, revolutionizing the way communication takes place, on the one hand enabling the rapid, easy and almost costless digital interaction, but, on the other, easing the adoption of more aggressive communication styles. It is crucial to regulate and attenuate these behaviors, especially in the digital context, where these emerge at a fast and uncontrollable pace and often cause severe damage to the targets. Social networks and other entities tend to channel their efforts into minimizing hate speech, but the way each one handles the issue varies. Thus, in this thesis, we investigate the problem of hate speech detection in social networks, focusing directly on Twitter.

Our first goal was to conduct a systematic literature review of the topic, targeting mostly theoretical and practical approaches. We exhaustively collected and critically summarized mostly recent literature addressing the topic, highlighting popular definitions of hate, common targets and different manifestations of such behaviors. Most perspectives tackle the problem by adopting machine learning approaches, focusing mostly on text mining and natural language processing techniques, on Twitter. Other authors present novel features addressing the users themselves.

Although most recent approaches target Twitter, we noticed there were few tools available that would address this social network platform or tweets in particular, considering their informal and specific syntax. Thus, our second goal was to develop a tokenizer able to split tweets into their corresponding tokens, taking into account all their particularities. We performed two binary hate identification experiments, having achieved the best f-score in one of them using our tokenizer. We used our tool in the experiments conducted in the following chapters.

As our third goal, we proposed to assess which text-based features and preprocessing techniques would produce the best results in hate speech detection. During our literature review, we collected the most common preprocessing, sentiment and vectorization features and extracted the ones we found suitable for Twitter in particular. We concluded that preprocessing the data is crucial to reduce its dimensionality, which is often a problem in small datasets. Additionally, the f-score also improved. Furthermore, analyzing the tweets' semantics and extracting their character n-grams were the tested features that better improved the detection of hate, enhancing the f-score by 1.5% and the *hate* recall by almost 5% on unseen testing data. On the other hand, analyzing the tweets' sentiment didn't prove to be helpful.

Our final goal derived from a lack of user-based features in the literature. Thus, we investigated a set of features based on profiling Twitter users, focusing on several aspects, such as the gender of authors and mentioned users, their tendency towards hateful behaviors and other characteristics related to their accounts (e.g. number of friends and followers). For each user, we also generated an ego network, and computed graph-related statistics (e.g. centrality, homophily), achieving significant improvements - f-score and *hate* recall increased by 5.7% and 7%, respectively.

Resumo

Nas últimas décadas, a tecnologia da informação atravessou uma evolução explosiva, que revolucionou a forma de comunicar, permitindo uma interação digital mais fácil, rápida e quase sem custos. Por outro lado, também facilitou a adoção de estilos comunicativos mais agressivos. É crucial regular estes comportamentos, especialmente no contexto digital, dado que surgem a um ritmo rápido e incontrolável e muitas vezes causam danos sérios aos alvos. As redes sociais e outras entidades tendem a canalizar esforços para minimizar o discurso de ódio, mas as abordagens diferem. Assim, nesta tese, investigamos o problema da detecção do discurso de ódio no Twitter.

O primeiro objetivo consistiu em realizar uma revisão sistemática da literatura do tema, visando abordagens teóricas e práticas, com maior foco na segunda. Recolhemos e resumimos de forma crítica a literatura mais recente, abordando a detecção do discurso do ódio e destacando definições populares, alvos comuns e diferentes manifestações deste tipo de discurso. A maioria das perspectivas aborda o problema usando tarefas de classificação de aprendizagem computacional, usando metodologias de processamento de texto, com foco em plataformas de redes sociais, como o Twitter. Um número reduzido de abordagens foca-se também nos utilizadores e nos seus perfis.

Embora as abordagens mais recentes tenham como alvo o Twitter, poucas ferramentas abordam esta rede social ou os tweets em particular, dada a sua sintaxe informal e específica (hastags, menções, etc). Assim, desenvolvemos um *tokenizer* capaz de dividir os tweets nos *tokens* correspondentes, tendo em conta todas as suas particularidades. A nossa ferramenta demonstrou resultados melhores em alguns parâmetros, relativamente a outros *tokenizers*, numa tarefa de classifcação de ódio binária.

O nosso terceiro objetivo baseou-se em avaliar quais as *features* baseadas em texto e técnicas de pré-processamento que produzem melhores resultados para a deteção de ódio. Para isso, recolhemos um conjunto de *features* baseadas nas da literatura e extraímos as mais comuns de pré-processamento, sentimento, análise semântica e vetorização que melhor se adequavam ao nosso contexto (Twitter). Concluímos que a limpeza e o pré-processamento de dados são cruciais para reduzir a dimensionalidade dos dados, sendo um problema recorrente em conjuntos de dados de pequena dimensão. A análise semântica dos tweets em conjunto com os n-gramas de carateres (em contraste com os de palavras) e técnicas de pré-processamento, demonstraram ser as *features* que melhor otimizaram a deteção de ódio, ao contrário das de sentimento. A melhor combinação de *features* melhorou o f-score e o *hate* recall em 1.5% e 5%, respetivamente, nos dados de teste.

O nosso objetivo final derivou do número reduzido de abordagens baseadas em *features* relacionadas com os utilizadores. Desta forma, investigámos um conjunto de funcionalidades, focando em vários aspectos, como o género de autores e dos utilizadores mencionados nos tweets, a sua tendência relativamente a comportamentos de ódio e outras características relacionadas com os seus perfis de utilizador (por exemplo, número de amigos e seguidores). Para cada utilizador fizémos a geração da sua *ego network* e calculámos estatísticas relacionadas com a mesma (por exemplo centralidade, homofilia), tendo obtido melhorias significativcas - aumento do f-score e *hate* recall em 5.7% e 7%, respetivamente.

Acknowledgements

I would like to express my deepest appreciation to all those who fully supported me throughout this journey.

- To my family, for the 24/7 support and love
- To my supervisor, prof. Sérgio Nunes and co-supervisor, Paula Fortuna, for all the guidance and knowledge provided during the process
- To my friends, who were always free to grab a drink and talk thinks through
- To my colleagues, whom I constantly exchanged knowledge and good moments with
- To all the researchers who contributed to the topic

Guilherme Routar

"This is how you do it: you sit down at the keyboard and you put one word after another until its done. It's that easy and that hard."

Neil Gaiman

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

ML	Machine Learning
UP	User Profiling
DL	Deep Learning
PoS	Part of speech
TF-IDF	Term frequency - inverse document frequency
NER	Named Entity Recognition
SVM	Support Vector Machines
LR	Logistic Regression
NB	Naive Bayes
DT	Decision Tree
GB	Gradient Boosting
CNN	Convolutional neural networks
RNN	Recurrent neural networks
LSTM	Long short-term memory
GRU	Gated recurrent unit
BiLSTM	Bi long short-term memory
BiGRU	Bi gated recurrent unit
TP	True positives
TN	True negatives
FP	False positives
FN	False negatives
TPR	True positive rate
FPR	False positive rate
ROC	Receiver operating characteristic
AUC	Area under curve

Chapter 1

Introduction

In recent decades, information technology has been undergoing a huge evolution, with an expressive adoption of online social networks and social media platforms. Such progress revolutionized the way communication takes place by enabling a rapid, easy and almost costless digital interaction between its users. Although its numerous advantages, the anonymity associated with these interactions often leads to the adoption of more aggressive and hateful communication styles. These emerge at a fast and uncontrollable pace and usually cause severe damage to its targets, being crucial that governments and social network platforms are able to successfully detect and regulate aggressive and hateful behaviors occurring on a regular basis on multiple online platforms. The detection of this type of speech is far from being trivial due to the topic's abstractness. Therefore we propose to deliver and complement current solutions on the detection of hate speech online, focusing on social media (Twitter). To achieve this, we focus our efforts on extracting and selecting both textual and user-based features. The following section describes in detail the goals of our work.

It is important to mention that this work is a continuation of another *hate speech detection* dissertation. While we adopt a more technical approach, focusing mostly on feature extraction and selection, Fortuna (2017) looked to enrich research in Portuguese by collecting a dataset in the same language and investigating whether hierarchical classes would be helpful in optimizing the detection of hate speech. Although both dissertations summarize the state of the art of the area, ours targets more recent approaches, most of them conducted in 2018.

1.1 Goals

In order to complement the current solutions in hate speech detection we have two main goals, composed of different subgoals. Firstly, we propose to collect the literature that addresses and surrounds the topic in a methodology denoted as Systematic Literature Review. Hate speech is an area of study included in several fields such as social sciences and law, although our goal is to adopt a computer science and engineering methodology. Based on this approach, we include hate speech detection in text as a subclass of text mining, generally useful for exploratory analytics,

and natural language processing, convenient for understanding the semantic meaning conveyed in text (e.g. semantic analysis). In order to complement both methodologies (text mining and NLP), we will also focus on literature addressing user profiling (e.g. social network analysis).

Our second goal is divided into two different sub goals. We will conduct an exhaustive experimentation methodology, where we aim to extract and select the features that maximize the efficiency of machine learning algorithms, typically used to classify instances of text. In order to enrich the prediction process, the methodology will also contemplate a user profiling module, which can be defined as a digital representation of users. Understanding how they fit and interact in communities may be crucial knowledge in predicting their social behavior.

1.2 Outline

The dissertation is composed of five different chapters. Initially, in our first chapter, we briefly introduce our motivation, goals and intended approaches.

The second chapter summarizes the state of the art for hate speech detection. Initially, we present some basic statistics regarding the topic (e.g. most popular authors in 2018) and later provide a deeper look at the most common text mining and natural language processing techniques in hate speech detection, along with user profiling methodologies.

On the third chapter, we investigate which text-based features perform better, while on the fourth we investigate how user-based features can improve the detection of hate speech in Twitter.

In the last chapter, we provide an overview of what we have done in this thesis, whether our goals were concretized or not and future work.

1.3 Language concerns

This dissertation may contain a number of profane words used to describe specific examples of the topic addressed.

Chapter 2

Hate speech detection: state of the art

This chapter provides an overall view of hate speech detection in text, addressing not only approaches through natural language processing and text mining, but also through user profiling techniques. For this purpose, we conducted a Systematic Literature Review and tried to cover not only practical methodologies but also theoretical ones. By collecting the literature of both areas we aimed to link the two subjects and provide a merged approach to detect hate speech more efficiently.

Section 2.1 describes the methodology adopted to collect the documentation addressing the topics targeted on this thesis, also known as a systematic literature review.

Section 2.2 focuses on the theoretical definition of hate speech, it's different types and who are the most common targets of such aggressive communication.

Both on Section 2.3 and 2.4 we display the results of the systematic literature review. The first one mentioned presents computed general statistics of the topic (e.g. number of publications by year, average citations), while the latter groups qualitative results of feature extraction and selection, including natural language processing and user profiling techniques.

2.1 Systematic literature review

A Systematic Literature Review has been conducted in order to withdraw all the documentation and studies addressing hate speech detection in text with focus on feature selection and extraction, a traditional machine learning approach featuring natural language processing and text mining techniques. We generalize the term *document* as a synonym for articles, dissertations or any other sort of text document.

2.1.1 Methodology

The methodology adopted to collect the state of the art was inspired on the one used by Fortuna (2017). Figure 2.1 summarizes the whole process.

For each module described on Figure 2.1, a few nuances had to be taken into account. They are described in the paragraphs below.

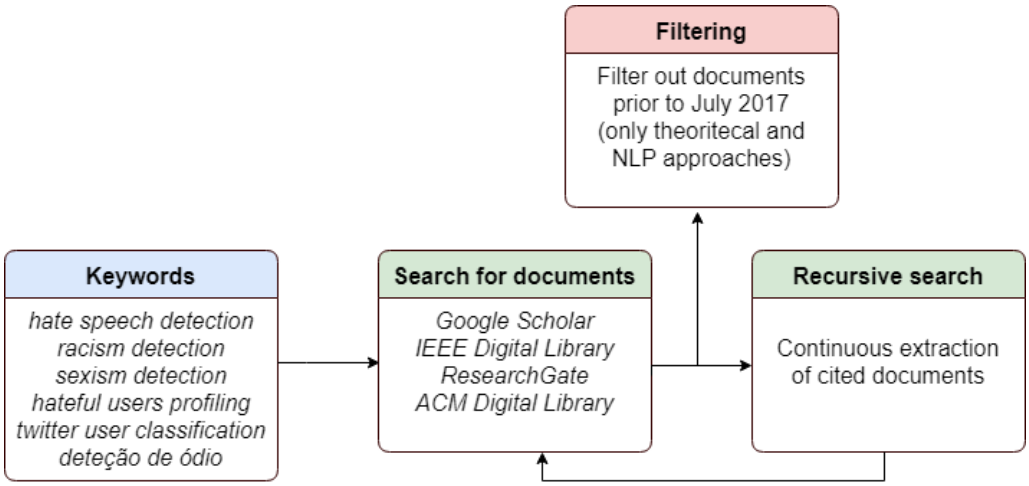


Figure 2.1: Methodology to conduct the systematic literature review

- **Keyword selection**: The definition of hate speech is not linear, as described in Section 2.2. Consequently, we looked up a number of keywords that generally fit the definition of hate. We also searched documentation in Portuguese "deteção de discurso de ódio" which translates to *hate speech detection*.

- **Search for documents**: We gathered documents from a number of different sources (detailed in Figure 2.1) that focused on the detection of hate speech through machine learning approaches.

- **Recursive search**: We selected and extracted useful documents cited in the ones searched through the keywords mentioned.

- **Filtering**: We only collected theoretical and text-based technical documents from July 2017 to October 2018, since the previous literature had already been summarized in Fortuna (2017). On the other hand, we collected all user profiling articles we found to be relevant, since this topic was not addressed on her dissertation.

2.1.2 Documents metadata

We gathered a total of 96 documents, mostly from July 2017 to October 2018. We also included older approaches which we found to be useful for our work, especially literature addressing user profiling. The following metrics were used to annotate and categorize the documents regardless of their category (e.g. practical, theoretical).

- **Category**: The category the document belongs to: machine learning, data annotation or analysis, theoretical and/or user profiling.

- **Title**

- **Authors**
- **Number of citations**
- **Keywords**

For the practical categories, in which we include all but theoretical one, extra metrics have been collected. They are listed and described on the list below.

- **Languages targeted**: Language of the classified text.
- **Approach**: Type of machine learning problem: classification or regression.
- **Social networks targeted**: Social networks from which the classified text was extracted.
- **Number of classes**: Number of classes used on classification approaches.
- **Datasets used**: Origin of the data used on the document.
- **Text pre-processing techniques**: Pre-processing techniques applied on the data.
- **Textual features extracted**: Textual features extracted from the data. Includes text mining and natural language processing techniques.
- **User features extracted**: User profiling techniques and features extracted from the data.
- **Algorithms**: Algorithms used to generate models.
- **Performance metrics**: Performance metrics used to classify the performance of the algorithms.
- **Results**: Results obtained according to the performance metrics used.

2.2 Hate speech overview

Fully understanding *hate speech* is the first step to be able to detect such discourse. Thus, in the following sub chapters, we address the concept itself by providing an overall view of it's origins, possible definitions and why we want to study it.

2.2.1 Origins

Hate speech is, approximately, a century old phenomenon. Although it occurs in various degrees and intensities, there is no exception when it comes to its presence in all societies. One of the first materializations of the ideology dates back to the end of the 19th century with the creation of the Ku Klux Klan Bullard (1998). This extremist clan regrouped 3 times for different reasons (e.g. Civil War), but their ideology always rested on strong foundations of hate.

Throughout time, several other manifestations of hate emerged globally. The Holocaust is a striking example of a nation (Nazi Germany) that spewed hatred against national minority groups (e.g. Jews). Later on, the introduction of the internet and social network platforms (e.g. Facebook, Twitter), on one hand, boosted the communication between people all over the world, but, on the other, also enabled the proliferation of hate discourses.

2.2.2 Definition

The definition of *hate speech* is not linear, even for humans Fortuna (2017). Different backgrounds and beliefs shape a set of distinct definitions and views, resulting in an abstract and volatile concept. For this reason, we collected a set of definitions from a number of different sources, in order to provide a global perspective of what *hate speech* is and what it encompasses nowadays. Table 2.1 shows the definitions contemplated and the sources they were extracted from.

Table 2.1: Definitions of hate speech from different sources. Part of these definitions (Twitter, Facebook, Code of conduct and ILGA) have been directly extracted from Fortuna (2017).

Source	**Definition**
Twitter	Hateful conduct: You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or disease. We also do not allow accounts whose primary purpose is inciting harm towards others on the basis of these categories." twi
Facebook	"Content that attacks people based on their actual or perceived race, ethnicity, national origin, religion, sex, gender or gender identity, sexual orientation, disability or disease is not allowed. We do, however, allow clear attempts at humor or satire that might otherwise be considered a possible threat or attack. This includes content that many people may find to be in bad taste (ex: jokes, stand-up comedy, popular song lyrics, etc.)." fac
Code of conduct, between EU and companies	"All conduct publicly inciting to violence or hatred directed against a group of persons or a member of such a group defined by reference to race, colour, religion, descent or national or ethnic" Eur
ILGA	"Hate speech is 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." ilg
Portuguese Constitution	"To be privileged, prejudiced or deprived of any right in consequence of descendence, gender, race, language, territory of origin, religion, political or ideological beliefs, instruction, economical status, social condition or sexual orientation."PTc
Paula	"Hate speech is language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity or other, and it can occur with different linguistic styles, even in subtle forms or when humour is used." Fortuna (2017)

Table 2.1 was partly extracted from Fortuna (2017). The first two entries are definitions proposed by the social networks Facebook and Twitter that, by being the most popular ones among web users, allow for a huge and continuous flow of information often carrying hateful content. It is crucial that their perception of *hate* is clear, since they actively try to prevent and filter it on a daily basis.

The third entry (Code of conduct) is the definition provided by the European Union Commission. Their role is to promote the general interest of the EU by proposing and enforcing legislation which means they have an important role, especially within the European Union.

ILGA, International Lesbian, Gay, Bisexual, Trans and Intersex Association, is an international group which gathers together sexual orientation and gender related minorities, aiming to protect and defend their rights. Since these minority groups are often targets of hate discourse, ILGA tries to protect and defend their rights.

The table entries listed above were extracted from Fortuna (2017). We opted to add two more:

- **Portuguese Constitution**: Since part of our work aims to enrich hate speech detection in Portuguese, we think it is preponderant to expose the Portuguese Constitution's definition of hate.
- **Definition proposed by** Fortuna (2017): In order to homogenize and merge the definitions together, Fortuna (2017) created a set of 4 boolean parameters ("Hate speech has specific targets", "Hate speech is to incite violence or hate", "Hate speech is to attack or diminish", "Humour has a specific status") to which the public definitions were evaluated to. This allowed to come up with a more general definition of *hate*, being the **definition we adopted**.

2.2.3 Common targets and examples

Hate comes in different shapes and formats, targeting several different groups and minorities. In Silva et al. (2016a), a systematic large scale measurement study of the main targets of hate speech was conducted on the social media platforms Twitter and Whisper, capturing not only common targets of hate but also their frequency on these platforms. The most common categories are listed below.

- **Race**: White and black people.
- **Behavior**: Insecure, sensitive people.
- **Sexual orientation**: People whose sexual orientation is not *straight*, i.e. gays, lesbians, bisexuals, etc.
- **Class**: Typically people belonging to lower classes, i.e. poorer.
- **Gender**: Commonly associated to sexism, discrimination towards women, transgenders, etc.
- **Ethnicity**: Indian, Pakistani, Jews.
- **Disability**: Retarded people.
- **Religion**: Muslims, Judaists.
- **Other**: Drunk, shallow people.

Silva et al. (2016a) collected 1% of English Twitter data (512 million tweets) from June 2014 to June 2015, of which 20,305 are considered to be hate tweets. Taking into account the categories listed above, the percentage of hate tweets was computed for each category. Table 2.2 lists the frequency for each category and an exemplifying tweet.

Table 2.2: Distribution of different types of hate found in the data.

Categories	**% Twitter posts**	**Example tweet**
Race	48.73	“Dad cuts up Water Melon for my lunch, looks at me and says ‘Here’s your nigger food’.”
Behavior	37.05	"I don’t even bother replying to wallflowers"
Physical	3.38	"Your nose nearly poked me through my screen"
Sexual orientation	1.86	"You gays are disgusting."
Class	1.08	"You left the ghetto but the ghetto never left you"
Ethnicity	0.57	“I think I hate cardio as much as I hate paki’s”
Gender	0.56	"Back to the kitchen, woman!"
Disability	0.19	"Fucking autistic."
Religion	0.07	"I just saw someone planting a time bomb, or as the Muslim women call it ’having a baby’"

2.2.4 Why study hate speech?

Hate speech highly impacts and undermines the right of the targeted person to equality and freedom. While this is enough motivation to go ahead and fight it, history has proven that it’s consequences are, in the long run, potentially catastrophic if no measures are taken against it. Hate speech promotes prejudice and hate and might shake the foundations of societies, creating gaps between social groups which might lead to deep fractures in the social cohesion. As mentioned in Subsection 2.2.1, the Holocaust is a striking example of such, where media and political parties spewed hatred against national minorities, escalating a conflict that led to a mass murder.

The popularity and continuous growth of online communities has also been contributing to the abundance of hateful behaviors. Being able to post and interact, mostly anonymously or without providing much personal information, acts as an incentive to give away unpopular and hostile opinions without many or any consequences at all.

Governments and especially social media have been trying to come up with efficient solutions to avoid hate speech Nobata et al. (2016), but the lack of studies and research on automatically identifying and detecting these behaviors makes it hard to accomplish significant results. Consequently, it is rather important to contribute with solutions for automatic hate speech detection in text.

2.3 Literature analytics

2.3.1 Category

We have grouped the documentation gathered regarding hate speech detection by the approach the authors conducted:

- **Machine learning** approach, *ML*, which includes feature extraction and selection and the application of supervised classification algorithms, evaluated by certain evaluation measures. Includes both text mining (TM) and user profiling techniques (UP).
- **Data analysis or annotation** approach, *DA*, where there is a deeper analysis of the data used or the methods used to create new datasets.
- **Theoretical** approach, *TH*, where the focus is mainly conceptual.

Although each category is well defined, part of the approaches focus on multiple elements, e.g. a document may address both machine learning techniques and dataset annotation. Figure 2.2 shows the frequency of the approaches discriminated from each other, while figure 2.3 shows the frequency of the composed approaches, present in the documents.

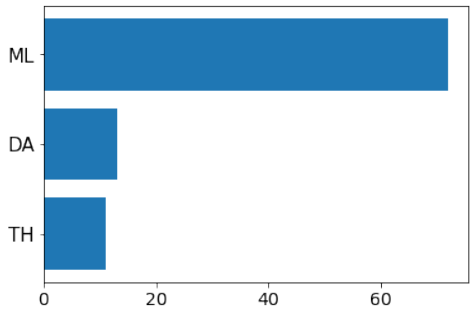


Figure 2.2: Frequency of approaches

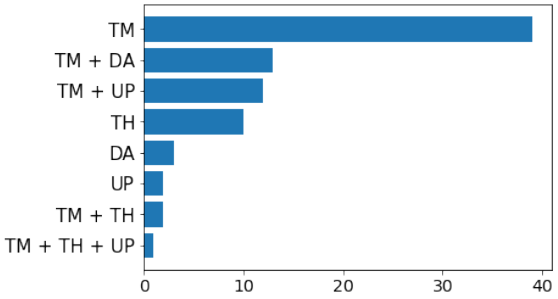


Figure 2.3: Frequency of combined approaches. TM: text mining, UP: user profiling

2.3.2 Documents yearly distribution

Although we only collected literature from the last couple of years, by merging the data collected in Fortuna (2017), it is easily concluded that hate speech detection is a highly growing topic in recent years, specially in 2018, as shown in Figure 2.4.

2.3.3 Authors frequency

The top five authors who contributed the most to hate speech detection in the last couple of years are summarized in table 2.3, alongside their adopted approaches.

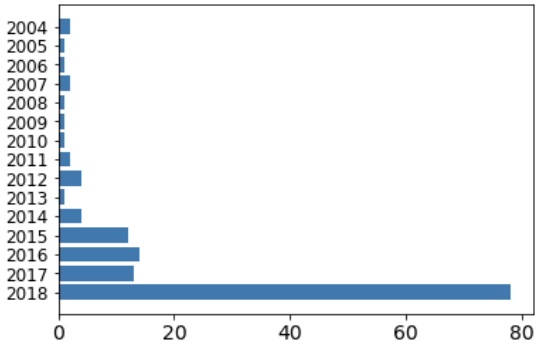


Figure 2.4: Number of publications by year

2.3.4 Citations frequency

In order to have an overview of the relevance of the documentation addressing hate speech detection, we also collected the citations for each document withdrawn. Considering we mainly focused on the last two recent years, the literature is mostly still under review and not very popular among researchers as denoted in figure 2.5.

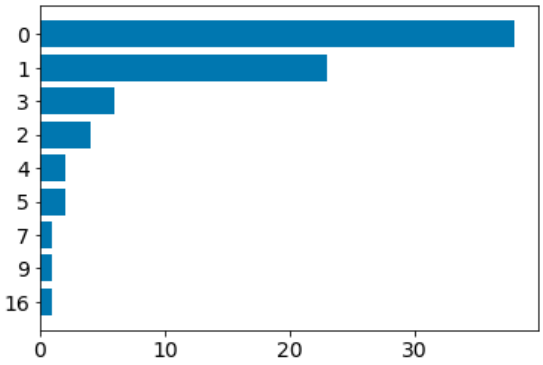


Figure 2.5: Frequency of citations for documents published on the second half of 2017 and most of 2018

2.3.5 Keywords distribution

The keywords highlighted in the documentation have been collected and grouped according to the area of knowledge they belong to. They are listed below:

- **Hateful Speech**: *Hate Speech*, *Cyberbullying*, *Cybercrime*, *Free Speech*, *Hate Crime*, *Terrorism*, *Cyberterrorism*, *Extremism*, *Cyberhate*, *Hate*, *Anger*, *Offensive Language*, *Aggressive Behavior*, *Violence*, *Abusive Language*, *Abusive Behavior*, *Aggression*, *Bullying*, *Harassment*, *Offensive Lexicon*, *Profane Word*, *Social Media Aggression*.

Author	# Publications	Approaches
Mai ElSherief	4	ML, UP, TH
Ziqi Zhang	3	ML, DA
William Yang Wang	3	ML, UP, TH
Elizabeth Belding	3	ML, UP, TH
Julian Risch	3	ML

Table 2.3: Number of publications and approaches used by the most frequent authors in hate speech detection.

- **Social Media**: *Twitter*, *Micro Blogging*, *Online Social Networks*, *Facebook*.
- **Machine Learning**: *Supervised Learning*, *Classification*, *Text Classification*, *Document Classification*, *Misogyny Detection*, *Automatic Misogyny Identification*, *Aggressiveness Detection*.
- **Natural Language Processing**: *Sentiment Analysis*, *Text Analytics*, *N-gram*, *TF-IDF*, *Linguistic Analysis*.
- **Deep Learning**: *Recurrent Neural Networks*, *Neural Network*, *CNN*, *GRU*, *Skipped CNN*, *Long Short-term memory*.
- **Feature Engineering**: *Feature Analysis*, *Feature Selection*, *Information Extraction*, *Text Mining*, *Opinion Mining*.
- **User Profiling**: *Credibility of Users*.

Figure 2.6 displays the frequency for each area of knowledge collected. Hate speech naturally predominates considering it is the keyword we searched for. Machine Learning, Natural Language Processing and Social media are also highly correlated with hate speech detection in text, while User Analysis is still an area under explored.

Although we may consider Deep Learning a sub section of Machine Learning, we think it's important to differentiate it, since it's usage has been increasing significantly.

2.3.6 Languages targeted

Although textual features are a limited set, they might differ from language to language. Being English one of the most spoken languages in the world, it is also the main focus for hateful speech detection in text as shown by figure 2.7. It is followed by Hindi and German, due to recent shared tasks introducing such languages.

2.3.7 Social networks targeted

An overwhelming majority of publications focus their approaches on Twitter, even though there are other social networks addressed. Figure 2.8 displays the frequency of social networks targeted on the hate speech detection topic.

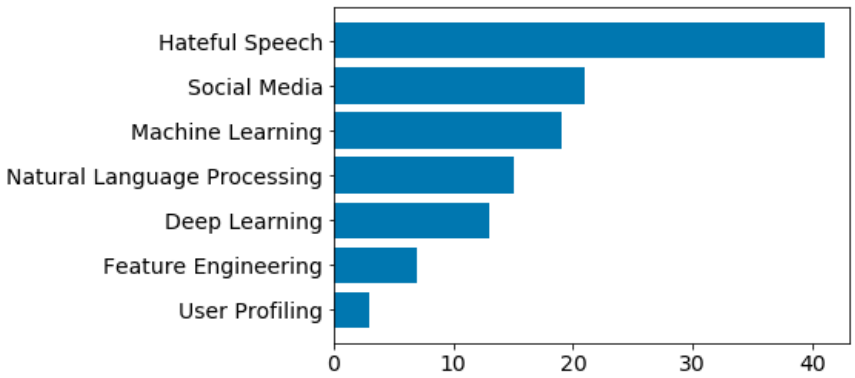


Figure 2.6: Keywords frequency

2.3.8 Machine learning approach

Machine learning branches into several subcategories of problems. If we consider the data, it may be divided into two main categories: supervised and unsupervised learning, being our focus the first one mentioned. Supervised learning is characterized by approximating a function using labeled data, while unsupervised learning focus is to patternize unlabeled data. Within supervised learning, we might also consider two different categories: classification, whose goal is to identify the category the instance belongs to, and regression, where one intends to predict a continuous quantity Bishop (2007).

Hate speech detection in text is addressed by the whole literature as a classification machine learning problem, although the number of classes considered varies between the approaches, as seen in figure 2.9.

The typical approach for hate speech detection in text focuses on distinguishing between hateful and non hateful instances of text. While such approach is quite common, the most frequent are 3-class based. For the most part, this is due to a recent shared task (TRAC-1) that classifies text as being *Overtly aggressive*, *Covertly aggressive* and *Non-aggressive* (e..g. Aroyehun and Gelbukh (2018), Nikhil et al. (2018)). Instead of classifying the intensity of hate, the remaining 3-class based literature distinguishes between different classes of hate: *Sexism*, *Racism*, *None* (e.g. Gröndahl et al. (2018), Zhang and Luo (2018)) and *Offensive*, *Abusive*, *None* (e.g. Ibrohim and Budi (2018), Gaydhani et al. (2018)).

Part of the literature also targets multiple classes of hate (above 3). Risch et al. (2018), Schefler et al. (2018), Stammbach et al. (2018), Bai et al. (2018) and Wiedemann et al. (2018) consider a 4-class based approach: *Profanity*, *Abuse*, *Insult*, *Other*, while Qian et al. (2018b) considers a wide range (40) of hierarchical classes.

2.3.9 Datasets

One of the issues in hate speech detection in text is the data available. Although there is a decent collection of data as described in table 2.4, a few problems arise:

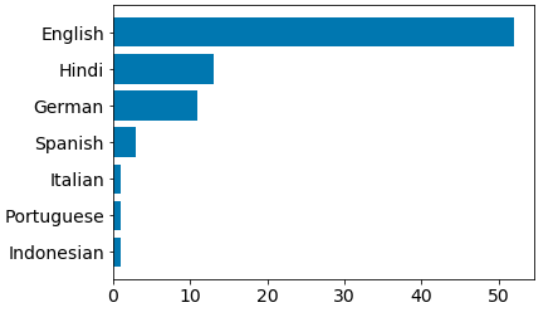


Figure 2.7: Frequency of addressed languages

- **Classes targeted**: part of the datasets address solely *hate* and *non hate* classes, although a big part target different classes (e.g. *sexism*, *racism*, *bullying*). Having an irregular data collection and annotation, makes it harder to standardize the identification of hate speech.

- **Annotation criteria**: being *hate* an abstract concept, as stated in section 2.2, classifying instances of text also makes for an abstract, and eventually biased, task. Different annotators may have a different definition of *hate*, thus an experiment on the same data, but differently annotated, is prone to produce different results as described in Robinson et al. (2018).

- **Availability**: not all datasets created for *hate* detection are available publicly.

Table 2.4 briefly describes the *hate* datasets collected for competitions and shared tasks, highlighting the classes, language, total instances, source from which the data was collected and whether the ID of the tweets is provided, useful to collect information regarding the authors of the tweets.

2.4 Data preprocessing and feature extraction

Extracting features consists of building a set of derived values from a collection of raw data, being a step often decisive in improving the performance of machine learning problems Jurafsky and Martin (2014).

On this subsection we highlight a set of techniques to extract features from both text (in which we include preprocessing approaches) and social media users, commonly used in the literature of hate speech detection . We also mention and briefly describe algorithms and performance metrics commonly used to classify text in this field of research.

2.4.1 Text processing techniques

Considering our target is social media, more specifically social networks such as Twitter, there's a big linguistic diversity in the content we may find in the platforms. Whether we focus on English,

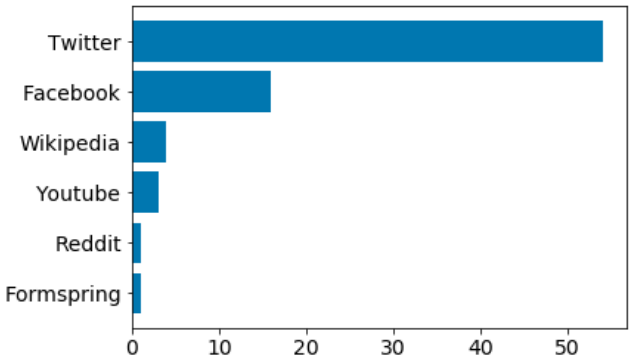


Figure 2.8: Social networks and their frequencies addressed on the approaches for hate speech detection

Portuguese or any other language, the amount of noise in the data is substantial due to the comments' shortness and informality, usually containing useless or unknown characters, emoticons, among other things. In any machine learning problem it is important to have clean data in order to maximize the efficiency of the algorithms used in the classification processes. Consequently, there's a set of techniques that can be applied in text mining that might be relevant for the goal:

- **Tokenization**: is defined as slicing a stream of text into pieces, denoted as tokens. The tokenization varies from language to language but lexical characteristics such as colloquialism (e.g. "u" instead of "you"), contractions (e.g. "aren't" instead of "are not") and others (e.g. "O'Neil) make the task harder. This is the most common preprocessing technique used by recent publications: Pitsilis et al. (2018), Watanabe et al. (2018), Zimmerman et al. (2018), Zhang et al. (2018), Biere and Bhulai (2018), Kumar et al. (2018), Scheffler et al. (2018), Pitsilis et al. (2018), Stammbach et al. (2018), Risch and Krestel (2018), Wiedemann et al. (2018), Schäfer (2018), Unsvåg (2018), Ahluwalia et al. (2018) and Sharma and Kshitiz (2018). Upon tokenizing, some authors (**?** Founta et al. (2018)) also remove the less frequent tokens of the data.

- **Lowercasing**: is the process of converting a stream of text to lowercase. Such technique may improve the performance of the classification since it reduces the dimensionality of the data. Not applying this technique may raise problems such as "tomorrow", "TOMORROW" and "ToMoRroW" being considered different words Aroyehun and Gelbukh (2018) (Biere and Bhulai, 2018) (Kumar et al., 2018) (Stammbach et al., 2018) (Mishra et al., 2018a) (Wiedemann et al., 2018) (Lee et al., 2018) (Samghabadi et al., 2018) (Unsvåg, 2018) (Sharma and Kshitiz, 2018) (Sahay et al., 2018).

- **Punctuation removal**: punctuation often is not relevant to text classification, so it was removed in Aroyehun and Gelbukh (2018), Bohra et al. (2018), Kapoor et al. (2018), Nikhil et al. (2018), Kumar et al. (2018), Stammbach et al. (2018), Roy et al. (2018), Mathur et al. (2018), Ibrohim and Budi (2018) and Sharma and Kshitiz (2018).

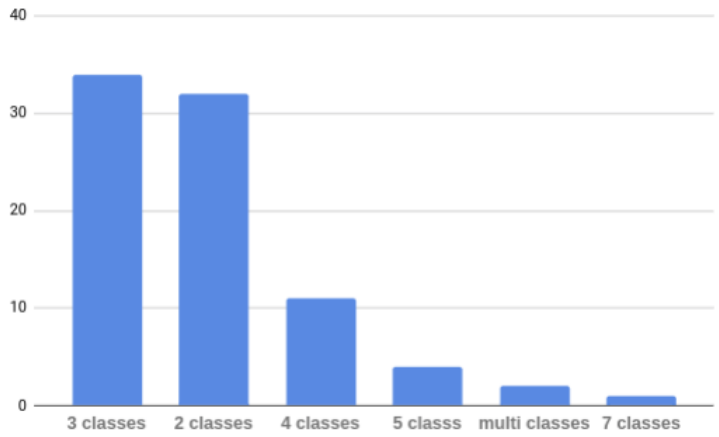


Figure 2.9: Frequency of the number of classes usually used in hate speech detection in text.

- **Irrelevant characters removal**: Irrelevant and/or invalid characters (e.g. "?|%&!"), in which we may also include punctuation, are also typically removed from the text since they do not contribute to the classification task as seen in Watanabe et al. (2018), Aroyehun and Gelbukh (2018), Zhang et al. (2018), Biere and Bhulai (2018), Sharma et al. (2018), Stammbach et al. (2018), Frenda and Somnath (2018), Schäfer (2018) and Sahay et al. (2018).

- **Stemming**: is the process of reducing inflected words to a common base form (e.g. "ponies" turns into "poni" and "cats" into "cat"). Stemming also improves performance by reducing the dimensionality of the data, since the words "fishing", "fished", and "fisher" are treated as the same word "fish". This technique is used in Zhang et al. (2018), Biere and Bhulai (2018), Scheffler et al. (2018), Maitra and Sarkhel (2018), Frenda et al. (2018), Samghabadi et al. (2018), Gaydhani et al. (2018), Sharma and Kshitiz (2018) and Sahay et al. (2018).

- **Lemmatization**: although very similar to stemming, lemmatization considers the morphological analysis of the words. While stemming would shorten the words "studies" to "studi" and "studying" to "study", lemmatization would shorten both to "study". It is used in Watanabe et al. (2018), Nikhil et al. (2018), Scheffler et al. (2018), Frenda et al. (2018) and Sharma and Kshitiz (2018).

- **Stopwords removal**: stop words are frequently used words that carry no useful meaning (e.g. "a", "and", "this"). Their commonness and lack of meaning makes them useless and eventually bad for classification problems in text. They are removed in Kapoor et al. (2018), Maitra and Sarkhel (2018), Mishra et al. (2018a), Mishra et al. (2018b), Unsvåg (2018), Gaydhani et al. (2018), Sharma and Kshitiz (2018).

- **PoS tagging**: Part of speech tagging, more commonly associated with feature extraction, is a technique to extract the part of speech associated with each word of the corpus, grammatically wise. Upon doing so, it might be common to remove words belonging to certain parts

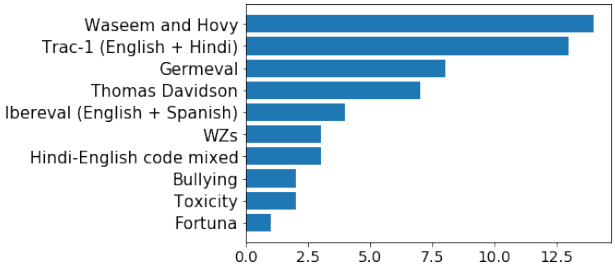


Figure 2.10: Most popular datasets addressed in the hate speech detection.

of speech that might end up not being so relevant (e.g. pronouns). This is done in Watanabe et al. (2018), Robinson et al. (2018), Scheffler et al. (2018) and Frenda and Somnath (2018).

- **Emojies**: are usually either **removed** Kapoor et al. (2018), Kumar et al. (2018), Stammbach et al. (2018), Frenda et al. (2018), Ibrohim and Budi (2018)) or **translated** to their correspoding description, e.g. ':)' is translated to 'smile' (Bohra et al. (2018), Mathur et al. (2018), Raiyani et al. (2018)). Keeping emojies might be useful since they usually denote a state of spirit or sentiment.

- **Spell checker**: misspelling is quite common specially in online platforms due to their informal nature. Having a spell checker (Nikhil et al. (2018)) might be an important preprocessing technique to avoid having unidentified or intentionally camouflaged words (e.g. "niggr", "fck").

2.4.1.1 Twitter preprocessing techniques

As described in section 2.3.7, Twitter is the most commonly addressed social network for automatic hate speech detection in text. Not because there is a higher incidence of hateful comments, but because its API is easily accessible, making it simpler to extract data. This extracted data has a specific format, containing usernames, URL's, hashtags, which may have to be removed or parsed. For this reason, there is a set of specific preprocessing techniques that are commonly applied on Twitter comments:

- **Hashtags**: hashtags are a common element of tweets (e.g. "#thisisahashtag"). In Ibrohim and Budi (2018) they are removed, but they often contain relevant information and keywords as stated by Maitra and Sarkhel (2018) and Stammbach et al. (2018), which is why most of the literature opts to extract their content. In Watanabe et al. (2018), Aroyehun and Gelbukh (2018), Zhang et al. (2018), Sharma et al. (2018), Maitra and Sarkhel (2018), Risch and Krestel (2018), Lee et al. (2018) and Mathur et al. (2018) the hashtags were decomposed into the words that compose them, e.g. "#thisisahashtag" becomes "this is a hashtag". In Stammbach et al. (2018), Roy et al. (2018), Schäfer (2018) and Unsvåg (2018) only the hash symbol ("#") is removed.

Table 2.4: Description of hate speech datasets. We mention the authors who created them, the distribution of classes across the data, number of examples, language used, platform from which the data was extracted and whether the tweets were encoded with their ID's (for those that apply).

Authors / Task	**Classes distribution**	**Instances**	**Language**	**Platform**	**Tweet ID**
Waseem Zeerak Dirk Hovy	Racism: 20% Sexism: 12% None: 68%	16k	English	Twitter	x
Thomas Davidson Dana Warmsley Michael Macy Ingmar Weber	Hate: 6% Non hate: 94%	25k	English	Twitter	
Paula Fortuna	Hate: 21% Non hate: 79%	6k	Portuguese	Twitter	x
Germeval	Offense: 35% Other: 65%	5k	German	Twitter	
TRAC-1	Overtly aggressive: 35% Covertly aggressive: 23% Non-aggressive: 42%	15k	English	Facebook	
Evalita (HaSpeeDe)	Hate: 47% No hate: 53%	4k	English	Twitter	
Antigoni-Maria Founta Constantinos Djouvas Despoina Chatzakou Ilias Leontiadis Jeremy Blackburn Gianluca Stringhini Athena Vakali Michael Sirivianos Nicolas Kourtellis	Abusive: 13% Spam 17% Normal 66%	80k	English	Twitter	x
IberEval (AMI)	Non-misogynous: 52% Misogynous: 48%	4k	English	Twitter	
SemEval (HatEval)	Hate: 38% No hate: 62%	6.5k	English	Twitter	

- **Usernames**, **mentions removal**: raw tweets always display the username associated with the account that tweeted and sometimes may contain mentions to other users. While they aren't relevant for text classification itself, thus being removed by most literature (Watanabe et al. (2018), Zimmerman et al. (2018), Scheffler et al. (2018), Stammbach et al. (2018), Mathur et al. (2018), Ahluwalia et al. (2018), Ibrohim and Budi (2018) and Gaydhani et al. (2018)), considering them may be helpful in generating the profiles of the users and see the interactions between them.

- **Fixed length comments**: Deep learning is often addressed in the literature as stated in Figure 2.14. Neural networks require a fixed size input, hence it is important to normalise the length of comments. The first step is to define a maximum length, which isn't usually too high considering Twitter comments are usually short. In Pitsilis et al. (2018), the defined maximum tweet size is 30 words, while in Founta et al. (2018) it is defined as the 95th percentile of length of tweets (token wise). In both approaches, the tweets shorter than the predefined maximum length are left padded with zeros, while the ones that overflow are truncated.

- **Emails**, **URL removal**: raw tweets contain a set of elements that are not useful for text classification. They are removed by all of the literature addressing twitter comments.

2.4.1.2 Text preprocessing summary

The amount of existing preprocessing techniques is quite vast and highly depends on the data being modeled. Figure 2.11 displays the frequency of all the preprocessing methods found in the literature for hate speech detection in text. In table 2.5, the methods used by each author are summarized. For simplicity purposes, some techniques were merged together (e.g. punctuation removal and irrelevant/invalid characters removal) and others were omitted since they were not relevant (e.g. Emails, URL removal).

	Tokeniz.	**Lowercase**	**Punct. / charact. removal**	**Stemming**	**Stopwords removal**	**PoS tagging**	**Lemmatiz.**	**Emoticons parsing**	**Fixed length comments**	**Hashtags parsing**
	Twitter									
Pitsilis et al. (2018)	X								X	
Watanabe et al. (2018)	X		X			X	X			X
Robinson et al. (2018)						X				
Zimmerman et al. (2018)	X									
Zhang et al. (2018)	X	X	X	X						X
Biere and Bhulai (2018)	X	X	X	X						
Kapoor et al. (2018)			X		X			X		
Scheffler et al. (2018)	X			X		X	X			
Sharma et al. (2018)			X							X
Stammbach et al. (2018)	X	X	X					X		X
Frenda and Somnath (2018)			X			X				
Mishra et al. (2018a)		X			X					
Founta et al. (2018)			X						X	
Mishra et al. (2018b)		X			X					
Wiedemann et al. (2018)	X	X								
Lee et al. (2018)		X								X
Frenda et al. (2018)				X			X	X		
Schäfer (2018)	X		X							X
Unsvåg (2018)	X	X			X					X
Mathur et al. (2018)			X					X		X
Ahluwalia et al. (2018)	X									
Ibrohim and Budi (2018)			X					X		X
Gaydhani et al. (2018)				X	X					
Sharma and Kshitiz (2018)	X	X	X	X	X					
	Other social network platforms									
Aroyehun and Gelbukh (2018)		X	X					X		X
Bohra et al. (2018)			X					X		
Nikhil et al. (2018)			X				X			
Kumar et al. (2018)	X	X	X					X		
Maitra and Sarkhel (2018)				X	X					X
Roy et al. (2018)			X							X
Risch and Krestel (2018)	X									X
Samghabadi et al. (2018)		X		X						
Raiyani et al. (2018)								X		
Sahay et al. (2018)		X	X	X						

Table 2.5: Subsets of preprocessing techniques used by recent literature on Twitter and other social networks

2.4.2 Feature extraction techniques

Feature extraction consists of collecting derived values (features) from the input data (text in this specific scenario) and generating distinctive properties, hopefully, informative and non-redundant, in order to improve the learning and generalization tasks of the machine learning algorithms. Upon their extraction there is usually a subset of features that will contain more relevant information.

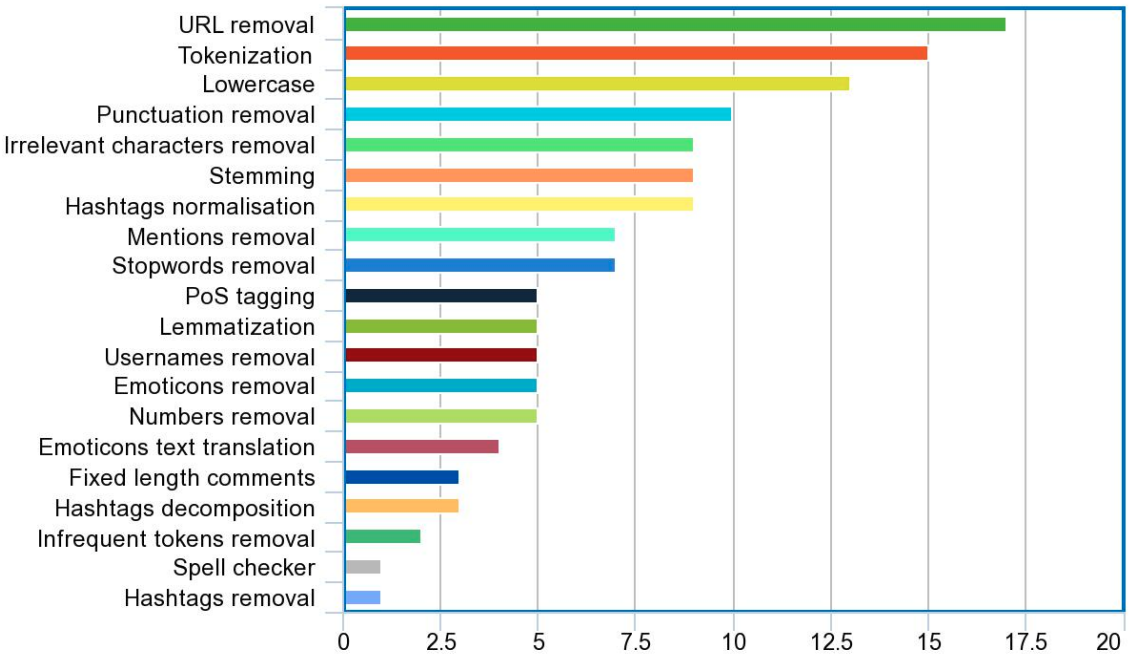


Figure 2.11: Frequency of preprocessing techniques used in hate speech detection approaches.

Selecting the right features is also a complex task but when done successfully improves the predictive model built by the algorithms. The following subsections describe generally most of the methodologies used to extract features from text and users.

2.4.2.1 General features

Textual features are the ones extracted from the text itself. They can be divided into different groups, according to the area they belong to (e.g. sentimental, semantic). On the other hand, user features represent the ones that target the user directly and the characteristics associated. Below are described the main feature extraction approaches used in text classification and hate speech detection in specific.

N-grams By definition, an n-gram is a contiguous sequence of (adjacent) words or letters of length "n" extracted from text Watanabe et al. (2018). They may be characterized as word n-grams if they group a sequence of "n" words Robinson et al. (2018), or character n-grams if they group a sequence of "n" characters Aroyehun and Gelbukh (2018).

There are three different ways to use N-grams as a feature: encode using tfidf (term frequency–inverse document frequency), as seen in Risch and Krestel (2018) and Klubicka and Fernández (2018), using a simple token counter (Sharma et al. (2018)) or encode using a hashing function.

Figure 2.12 displays how frequently n-grams (and respective encodings) are used in hate speech detection in text.

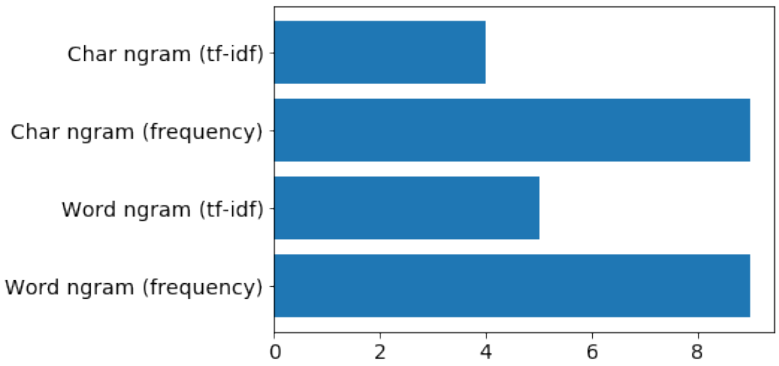


Figure 2.12: Frequency distribution of n-grams (and encodings)

TFIDF Term frequency-inverse document frequency is a numerical statistic that measures the importance of a certain word in a data corpus. This might be an important feature in understanding the importance of certain words to express specific types of speech (e.g. "hate") Sharma et al. (2018). Additionally, words that typically appear often and carry less meaning (e.g. stop words) are not as prominently considered using such encoding.

Bag of Words (BoW) Bag of words is a representation of words which disregards grammar and the order of the words in sentences, while keeping multiplicity. Similarly to n-grams, BoW can be encoded using tfidf Anagnostou et al. (2018), token counter or hashing function. Although it is typically used to group textual elements as tokens (Frenda and Somnath (2018), Pamungkas et al. (2018), de Gibert et al. (2018)), it can also group other representations such as parts of speech as used in Anzovino et al. (2018).

Sentiment Analysis The classification of text goes beyond analytical processing. It is rather important to grasp the sentiment behind the message, otherwise its true meaning will probably be misunderstood and/or misinterpreted (e.g. sarcasm). Users, mainly on social media, tend to formulate opinions on a diversity of topics, especially when they express an extremist attitude Bermingham et al. (2009), in which we include hate speech.

Regarding social media, sentiment analysis approaches usually focus on identifying the polarity (positive or negative connotation) of comments' tokens individually Watanabe et al. (2018) and sentences as a whole Scheffler et al. (2018), Risch and Krestel (2018), Schneider et al. (2018).

Table 2.6 describes the features related to sentiment analysis used in hate speech detection in text.

Named Entity Recognition (NER) Named Entity Recognition aims to extract and classify named entities in text, i.e. identify persons, locations or any other category that may be present. This might be particularly useful in hate speech detection since hate instigators tend to target people, groups or ethnicities (e.g. migrants), Gambäck and Sikdar (2017). Identifying entities is also

Sentiment feature	Used in
Score of positive words	Watanabe et al. (2018)
Score of negative words	Watanabe et al. (2018)
Number of positive slang words	Watanabe et al. (2018)
Number of negative slang words	Watanabe et al. (2018)
Number of positive emojies	Watanabe et al. (2018), Frenda and Somnath (2018), Risch and Krestel (2018)
Number of negative emojies	Watanabe et al. (2018), Frenda and Somnath (2018), Risch and Krestel (2018)
Number of positive hashtags	Watanabe et al. (2018)
Number of negative hashtags	Watanabe et al. (2018)
Number of words with positive sentiment	Nikhil et al. (2018), Salminen et al. (2018)
Number of words with negative sentiment	Nikhil et al. (2018), Risch and Krestel (2018)
Comment polarity score	Robinson et al. (2018), Schneider et al. (2018), Scheffler et al. (2018), Maitra and Sarkhel (2018), Risch and Krestel (2018), Orasan (2018)

Table 2.6: Sentiment analysis features used in hate speech detection

usually combined with sentiment analysis in hate speech detection. As an example, Risch et al. (2018) identifies the entities negatively mentioned.

Part of speech Part of speech tagging, also denominated as word-category disambiguation, consists of labeling words with their grammatical designation (e.g. verb, adjective). The main difficulty in tagging text is that each word doesn't necessarily belong to one category of speech (e.g. "fish" might refer to the animal, noun, or the act of fishing, verb). For this reason, the task of identifying the part of speech is not trivial and the context of the phrase must be considered.

PoS tagging is particularly helpful in detecting hate speech in text, since hate discourse often contains (offensive) adjectives (e.g. ugly fat boy) Scheffler et al. (2018). This feature is sometimes also combined with sentiment analysis. In Risch et al. (2018), verbs with negative polarity are identified.

Table 2.7 describes the features related to part of speech tagging combined with others used in hate speech detection in text.

PoS tagging	Combined feature	Used in
Presence of verbs with negative polarity	Sentiment	Risch et al. (2018)
Check if verb with negative polarity refers to entity	Sentiment + NER	Risch et al. (2018)
For each PoS that may contain sentiment, replace the token by it's PoS + polarity (e.g. 'coward' - adjective_negative)	Sentiment	Watanabe et al. (2018)
Number of adjectives	Semantic	Anzovino et al. (2018)

Table 2.7: PoS tagging features used in hate speech detection

Semantic Formal corpora obeys a set of both syntax and semantic rules, regardless of the language they are written on. Likewise, in text classification, it might be useful to inspect both in order to obtain good performance classifying text.

While n-grams target syntax, there are other features which focus on semantics. Punctuation may play an important role in identifying hate speech, as there is usually a bigger number of exclamation and interrogation marks Watanabe et al. (2018), Salminen et al. (2018). Other examples are the identification of capitalized words Scheffler et al. (2018), Frenda and Somnath (2018)], number of swear words Salminen et al. (2018), Pamungkas et al. (2018), the length of the comment Watanabe et al. (2018), Robinson et al. (2018), among many others.

Table 2.8 describes the features related to semantic analysis used in hate speech detection in text.

Semantic feature	**Used in**
Number of "!"	Watanabe et al. (2018), Salminen et al. (2018)
Number of "?"	Watanabe et al. (2018), Salminen et al. (2018)
Number of "."	Watanabe et al. (2018)
Number of punctuation	Robinson et al. (2018), Köffer et al. (2018), Bohra et al. (2018), Nikhil et al. (2018), Scheffler et al. (2018)
Number of all-capitalized words	Watanabe et al. (2018), Scheffler et al. (2018), Frenda and Somnath (2018)
Number of quotes	Watanabe et al. (2018), Salminen et al. (2018)
Number of interjections	Watanabe et al. (2018)
Number of laughing expressions	Watanabe et al. (2018)
Number of words in tweet	Watanabe et al. (2018), Robinson et al. (2018), Köffer et al. (2018), Nikhil et al. (2018), Salminen et al. (2018), Frenda and Somnath (2018), Risch and Krestel (2018)
Number of mentions	Robinson et al. (2018), Anzovino et al. (2018)
Number of hashtags	Robinson et al. (2018), Pamungkas et al. (2018)
Number of characters	Robinson et al. (2018), Salminen et al. (2018), Anzovino et al. (2018), Frenda and Somnath (2018)
Number of syllables	Robinson et al. (2018)
Ratio misspelled words / total words	Robinson et al. (2018)
Number of misspelled words	Salminen et al. (2018)
Number of emojies	Robinson et al. (2018), Salminen et al. (2018)
Percentage of capitalized characters	Robinson et al. (2018), Risch and Krestel (2018)
Number of capital letters	Salminen et al. (2018), Frenda and Somnath (2018)
Number of sentences	Köffer et al. (2018)
Number of URLs	Köffer et al. (2018), Salminen et al. (2018), Anzovino et al. (2018), Pamungkas et al. (2018)
Average word length	Köffer et al. (2018), Salminen et al. (2018)
Number of negation words	Bohra et al. (2018)
Number of special characters	Salminen et al. (2018)
Number of single character words	Salminen et al. (2018)
Number of modal verbs	Salminen et al. (2018)
Number of tokens with non-alphabetic characters in the middle	Salminen et al. (2018)
Number of swear words	Salminen et al. (2018), Scheffler et al. (2018), Frenda and Somnath (2018), Pamungkas et al. (2018)
Ratio swear words / all words	Salminen et al. (2018)
Number of adjectives	Anzovino et al. (2018)

Table 2.8: Semantic analysis features used in hate speech detection

Word embeddings Word embeddings are vector-based word representations which map related words to closer vectors, being the most common feature used in deep learning in natural language processing Augenstein et al. (2018). This is a feature that works well especially if the embeddings are trained on a corpus within the same domain Stammbach et al. (2018), e.g. word embeddings used to classify Twitter comments should be trained on Twitter corpora for better performance. There is a set of different techniques to embed words as described below.

- **Word2Vec**: The granularity of the embedding is word wise, generating a vector for each word of the corpus. There are 2 different possible models: CBOW (continuous bag of words), that learns to predict the word by the context, and skip-grams, which is designed to predict the context itself. According to Goldberg and Levy (2014), CBOW is faster to train and has slightly better accuracy for the frequent words. On the other hand, skip-grams work well with a small amount of training data and represent well even rare words or sentences. Most of the approaches that used Word2Vec (e.g. Zhang and Luo (2018) and Kapoor et al. (2018)) apply the skip-gram model Pennington et al. (2014).

- **GloVe**: This embedding model is quite frequent in the literature of hate speech detection in text as shown in Figure 2.13. It is an unsupervised learning algorithm for obtaining vector representation of words. Several corpora are provided to pre-train the models, including a Twitter-based one, used in van Aken et al. (2018) and Roy et al. (2018).

- **FastText**: This embedding is essentially an extension of Word2Vec, except each word is composed of character ngrams. The vector for each word is the sum of it's character ngrams. For example, for the word "hate" and considering an ngram with an n ranging from 3 to 5, the sub vectors are "<ha", "hat", "hate", "hate>", "ate", "ate>", "te>". Used in Schneider et al. (2018) and Aroyehun and Gelbukh (2018).

- **BERT**: BERT is a very recent word embedding that presented quite appealing results Devlin et al. (2018). The innovation behind BERT is that, unlike Word2Vec and GloVe, each word isn't restricted to a single vector, depending on their context and meaning - the same word may have different meanings. Introducing the first deeply bidirectional unsupervised language representation, this embedding might be a breakthrough in natural language processing tasks.

The usage of word embeddings might be prejudicial at some point. Although most existent word embeddings are trained on huge data collections (e.g. Google News), there is a chance they don't contain tokens used in the data. One common example is omitting or replacing characters in some words (e.g. 'fuck' becomes 'fck' or 'random' as 'r4ndom'). Using FastText may be advantageous because the embeddings are done character wise, instead of word wise like Word2Vec and GloVe. Figure 2.13 displays the frequency of word embedding models used in the literature of hate speech detection.

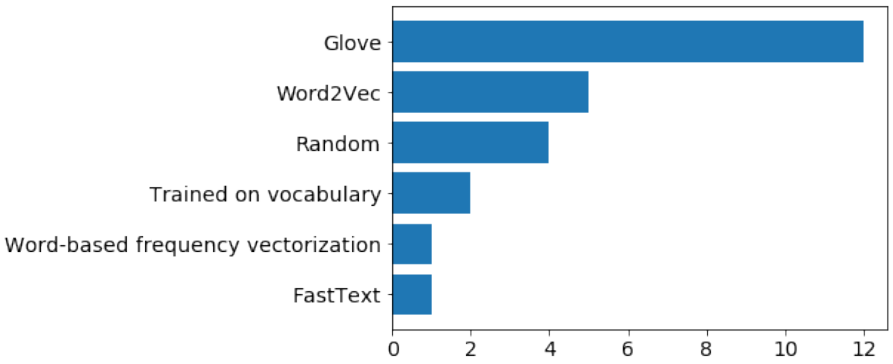


Figure 2.13: Frequency distribution of word embeddings

Dictionaries This feature consists of comparing the data corpus to a dictionary containing words of some kind, depending on the final objective. This might be a useful feature in hate speech detection, considering there's often an uncommon frequency of profane words Dadvar et al. (2012). A common approach is to collect a set of swear words and check for their presence in the tweets Salminen et al. (2018), Scheffler et al. (2018).

Topic extraction Topic extraction consists on extracting and classifying the underlying topic addressed by the document. This comes up as a useful feature in hate speech detection since certain topics are more commonly addressed than others in this type of discourse (e.g. Race, Religion, Politics) Agarwal and Sureka (2017). In Wiedemann et al. (2018), the users mentioned in the corpus are identified and their tweets are collected and the most probable topic is extracted.

User profiling Unlike all the features mentioned above, user profiling targets the users themselves instead of the actual text. This is a technique that hasn't been much explored (see figure 2.6), but might output interesting results. Different approaches can be conducted in order to model users, including users interactions within social networks, their behavior, etc. In Pitsilis et al. (2018), the users tendency towards a specific behavior (racism or sexism) is computed. In Qian et al. (2018a) the inter and intra user representation is also considered. Further detail is provided in the subsection 2.4.2.2.

2.4.2.2 User features

As mentioned before, user related features is an under explored area when it comes to text classification. Most of the approaches focus on text mining and processing, ignoring inter and intra user representation. The following paragraphs summarize the approaches used in the literature regarding user profiling.

User history More often than not, single tweets portray little information, especially due to their small length. This often results in classification errors. By considering users' Twitter history,

an user profile can be generated for each author which might be helpful in classifying single messages.

In Pitsilis et al. (2018), three parameters are created which evaluate users' tendency towards certain behaviors: *racism*, *sexism* and *none*. Such features are computed taking into account the tweets history of each user and the class they belong to, i.e. an user with mostly sexist tweets will have a high sexist tendency. It is not clear whether the considered tweets history regards only the training data or the actual user Twitter profile.

In Qian et al. (2018a) an intra-user representation is generated according to the users' history on Twitter. For each user, a collection of recent tweets (400) is extracted. These tweets are labeled as *hateful* or *not hateful* using a model pre-trained on the original data.

Gender information Identifying the gender of users is a feature commonly extracted in user profiling (Klubicka and Fernández (2018), Waseem and Hovy (2016), ElSherief et al. (2018b) and Schäfer (2018)). This might be particularly useful in detecting sexist messages, since the instigators are usually men who target women. Although Twitter doesn't provide gender information, it can be predicted in a number of ways. Schäfer (2018) predicts the gender by comparing the authors' user names with a dictionary of gender labeled names. There are quite some limitations with this approach since user names are frequently random or based on something else other than the author's actual name. Schneider et al. (2018) pre-trains a model based on the TwiSty corpus Van Hee et al. (2015) and uses it to label the training data.

Account characteristics Twitter provides a set of characteristics related to the accounts themselves, such as the presence of a profile image, location and timezone. Although these aren't directly related to the user, they may unveil certain behaviors (e.g. *fake* accounts, with no profile picture or personal information are more likely to engage in hateful discourse). Table 2.9 lists account based user features.

Tweet augmentation In order to suppress noise in the target tweets, a set of similar tweets, posted by other users, is gathered, in Qian et al. (2018a). These were selected from a large unlabeled corpus using Locality Sensitive Hashing, an hasher able to reduce the dimensionality of the data and, consequently, decrease the search space Indyk and Motwani (1998). This hasher enabled the identification of the n nearest neighbours of the target tweet. This feature proved to be helpful because some tweets may contain subtle additions that might not be easily detectable. Having a parallel comparable corpus aids the algorithm in detecting these subtleties.

Network analysis As a social network, Twitter enables the linkage between several million users. These connections can be expressed in the format of a graph, where the nodes represent the users and the edges the relations between them.

Account feature	**Source**
Presence of profile image	ElSherief et al. (2018b)
Presence of location	ElSherief et al. (2018b)
Presence of timezone	ElSherief et al. (2018b)
Enabled geo-location (to be posted alot with tweets)	ElSherief et al. (2018b)
Verified account	ElSherief et al. (2018b)
Length of profile description	ElSherief et al. (2018b)
Length of profile name	Klubicka and Fernández (2018)
Age of the account	Founta et al. (2018) Klubicka and Fernández (2018)
Number of lists	ElSherief et al. (2018b) Founta et al. (2018)
Number of tweets	ElSherief et al. (2018b) Founta et al. (2018) Klubicka and Fernández (2018)
Number of favourited tweets	Klubicka and Fernández (2018) Founta et al. (2018)
Number of retweets	ElSherief et al. (2018b)

Table 2.9: Account based user features

In Founta et al. (2018), simple computations are made concerning the amount of friends and followers (e.g. ratio between both of the measures). Table 2.10 lists network related features considering these measures.

Account feature	**Source**	**Description**
Number of friends	ElSherief et al. (2018b) Founta et al. (2018) Klubicka and Fernández (2018)	-
Number of followers	ElSherief et al. (2018b) Founta et al. (2018) Klubicka and Fernández (2018)	-
Followers/friends ratio	Founta et al. (2018)	Ratio between the number of followers and friends
Power difference between user and mentions	Founta et al. (2018)	Popularity (followers/friends) difference between user and mentioned users
Follower/friend reciprocity	Founta et al. (2018)	Extent to which user follows back potential new followers

Table 2.10: Network related features.

Besides the trivial computations listed in Table 2.10, there is a set of other measures which consider the graph itself. A user graph ug has n nodes and e edges, where n is the sum of friends and followers, plus the user himself and e is the number of connections. A graph can either be directed, if we distinctly consider friends and followers, or undirected, if friends and followers represent the same connection. Naturally, directed graphs enable the extraction of more insights regarding the user data. Below are listed a set of measures computable from a directed user graph. The descriptions have been adapted from Easley and Kleinberg (2010).

- **Clustering coefficient**: degree to which nodes in a graph tend to cluster together. May be computed as the graph's average or for single nodes. Used in Founta et al. (2018).

- **Node degree**: number of edges incident to a vertex. In directed graphs, the *indegree* is the number of edges incoming to a vertex, while the *outdegree* is the number of edges outgoing from a vertex.
- **Network diameter**: the diameter of a graph is the longest shortest path between any 2 nodes.
- **Centrality**: the most important nodes in the graph have usually high centrality measures. These include betweenness, eigenvector, closeness and page rank centrality. The user for whom the graph was generated is likely to have higher centrality values than the other ones in the network.

2.5 Algorithms and performance metrics

2.5.1 Algorithms

As mentioned on Section 2.3.8, hate speech detection in text is mostly a supervised classification machine learning problem. Although the range of algorithms for such problems is quite wide, there is a smaller subset that performs better on text classification, thus being used more often. Figure 2.14 summarizes the frequency of algorithms used on automatic hate speech detection in text. Since deep learning has a wide range of possible architectures, neural network approaches were merged together. Figure 2.14 shows that the usage of deep learning recently has been increasing significantly.

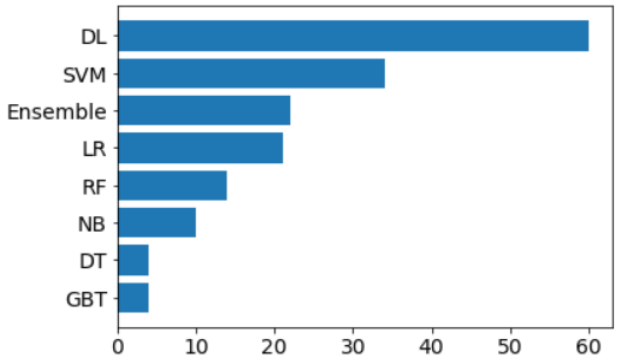


Figure 2.14: Algorithms' frequency for automatic hate speech detection in text.

The algorithms used in the literature are summarized in the paragraphs below:

- **Support Vector Machines**: SVM's are widely used in classification problems and the algorithm can be described as an hyperplane that categorizes input data (text in this case) Chapelle (2007), Bennett and Bredensteiner (2000). In 2017, SVM's held the best results for text classification tasks, but in 2018 deep learning took over, especially in hate speech detection as described here Watanabe et al. (2018).

- **Logistic Regression**: logistic regression is a (predictive) regression analysis which estimates the parameters of a logistic model, a statistical model that uses a logistic function to model a binary dependant variable Sperandei (2014).

- **Random Forest**: Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest Breiman (2001). This model requires almost no input preparation, performs implicit feature selection and is very quick to train, performing well overall.

- **Naive Bayes**: This is an algorithm based on the Bayes' theorem with strong naive independence assumptions between the feature of the data. it generally assumes that a particular feature in a class is unrelated to any other feature. Naive Bayes is a model useful for large datasets and does well despite being a simple method Kaur and Oberai (2014).

- **Decision Tree**: This is an algorithm that provides support for decision making, providing a tree-like model of decisions and their possible consequences and other measures (e.g. resource cost, utility). They are often used since their output is usually readable, being simple to understand and interpret by humans. They are also fast and perform well on large datasets, but they are prone to overfiting Topîrceanu and Grosseck (2017).

- **Gradient Boosting** is a predicition model consisting of an ensemble of weak prediction models, typically decision trees (that's why it may also be called gradient boosted trees), in which the predictions are not made independently (as in Bagging), but sequentially. The sequential modeling allows for each model to learn from the mistakes made by the previous one Natekin and Knoll (2013).

2.5.1.1 Deep learning

Deep learning popularity has been growing significantly over the recent years, especially in text classification as seen in Figure 2.14. This is partly due to the disclosure of artificial neural networks' architecture, which made it possible and easier to tune the parameters and, consequently, model the behavior of such algorithms. This produced better results, outperforming baseline algorithms as described in Section 2.5.3. The main artificial neural networks' architectures are described below:

- **CNN**, convolutional neural networks are a class of deep feed-forward artifical neural networks. A CNN consists of an input and output layer and multiple hidden layers which consist of convolutional layers, pooling layers and fully connected layers Yin et al. (2017).

- **RNN**, recurrent neural networks, another class of artificial neural networks that, unlike CNN's, are able to handle sequential data, allowing to produce temporal dynamic behaviors according to a time sequence. The connections between nodes form a directed graph. RNN's

have feedback loops in the recurrent layer, which act as a memory mechanism. Despite this fact, long-term temporal dependencies are hard to grasp by the standard architecture, because the gradient of the loss function decays exponentially with time (vanishing gradient problem, Hochreiter (1998)). For this reason, new architectures have been introduced:

- **LSTM**, long short-term memory neural networks, are a type of RNN that use special units in addition to standard units, by including a memory cell able to keep information in memory for long periods of time. A set of gates is used to control when information enters the memory, when it's output, and when it's forgotten enabling this architecture to learn longer-term dependencies as detailed in Chung et al. (2014) and Yin et al. (2017).
- **GRU**, gated recurrent unit neural networks, are similar to LSTM's, but their structure is slightly simpler. Although they also use a set of gates to control the flow of information, these are fewer when compared to LSTM's Yin et al. (2017) Chung et al. (2014).

Although both single directional LSTM and GRU's neural network architectures are able to handle sequential data, they fail to consider the contextual information from the future tokens. Thus, bidirectional architectures, **BiLSTM** and **BiGRU**, were introduced, where the algorithm is fed with the original data from the beggining to the end and vice-versa. Processing the sequence on two opposite directions allows the neural network to consider both the previous and future context of a single token as described in Liu et al. (2016).

According to the descriptions provided above, one would conclude that RNNs would perform better on text classification versus CNNs, considering words are usually connected within a context. Figure 2.15 shows the usage of CNNs, RNNs and mixed architectures in hate speech detection in text. Figure 2.16 descriminates which RNN architectures are the most used.

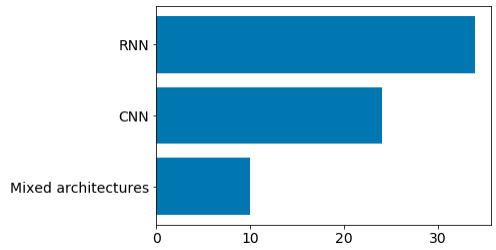


Figure 2.15: Frequency of DL architectures

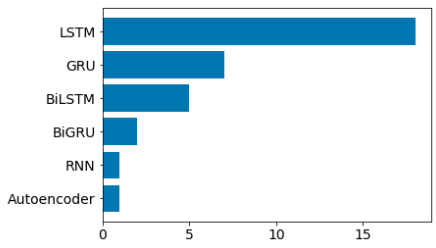


Figure 2.16: Frequency of RNN architectures

2.5.2 Performance metrics

A wide range of performance metrics are used to evaluate machine learning algorithms and models. These measures are originally built from a confusion matrix (see Table 2.11) that, despite not being a performance measure by itself, serves as the basis for a number of other methods. The

confusion matrix records which samples of the data have been correctly and incorrectly predicted for each class Sokolova et al. (2006).

Table 2.11: Confusion matrix extracted from Sokolova et al. (2006)

Class \Recognized as	**Positive**	**Negative**
Positive	True Positives (TP)	False Negatives (FN)
Negative	False Positives (FP)	True Negatives (TN)

Accuracy is a generic performance measure that assesses the overall effectiveness of the algorithm, by computing the number of correct predictions over all the predictions made. Although it is commonly used (see Figure 2.17), accuracy doesn't distinguish between different classes Sokolova et al. (2006). Consequently, this performance metric may be misleading, especially when the classes of the data are unbalanced Harrell (2017).

There is a subset of performance metrics that consider classes (e.g. Recall, Precision and Specificity). These are usually more useful in sets of data that contain unbalanced classes, since the performance of the algorithm can be assessed class wise. This is quite often in hate speech datasets as described in Section 2.3.9. The most used, class wise, performance measures in hate speech detection (Figure 2.17) are:

- **Recall**, also known as **Sensitivity** or **True Positive Rate**, is defined as the proportion of real positives that are correctly predicted as positive Powers (2008).
- **Precision** denotes the proportion of predicted positive cases that are actually positive Powers (2008).

The most used performance metric in hate speech detection in text is the **F1 score** (Figure 2.17). It is defined as the harmonic mean of Precision and Recall, and considers class imbalance, unlike accuracy Sasaki (2007), hence it's wide usage in hate speech detection. Table 2.12 summarizes the most common performance metrics used in hate speech detection, including their formula and most frequent usage.

Using the above mentioned performance metrics, a graphical visualization of the algorithm's predictions can be computed, known as **ROC**, receiver operating characteristic. It shows the relation between the sensitivity and the specificity of the algorithm and is created by plotting the true positive rate (TPR) against the false positive rate (FPR) Sokolova et al. (2006). The higher the TPR, the higher the area under ROC, also known as **AUC** (area under curve), as described in Powers (2008).

2.5.3 Results

The results obtained by the state of the art experiments for hate speech detection in text are quite wide and rely on several different parameters. The tools chosen, libraries used and, mainly, the distribution and characteristics of the data used in the experiments contribute to produce a variety

Table 2.12: Description of performance metrics and their formulas. TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives.

	Formula	**Common usage** (Shung (March, 2018))
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$	Balance between Precision and Recall
Precision	$\frac{TP}{TP+FP}$	High FP cost
Recall	$\frac{TP}{TP+FN}$	High FN cost
F1 score	$\frac{2*Precision*Recall}{Precision+Recall}$	Balance between Precision and Recall for unbalanced classes

of results, which are often not comparable. Thus, and since we can not guarantee full uniformity in the procedures used in each experiment, we can at least make sure the datasets used are the same for each comparison we perform, ensuring a certain degree of viability.

In the following subsections we summarize the best available approaches for 3 different hate speech datasets: Waseem & Hovy, TRAC-1 and Germeval. These have been chosen because their popularity is quite high in the current state of the art as described in Figure 2.10, granting a wider variety of approaches. The approaches are sorted in ascending order according to their F1-score and for each we highlight the:

- Article
- Preprocessing techniques applied
- Features extracted
- Algorithms
- F1-score

2.5.3.1 Waseem & Hoovy

The dataset collected by Waseem & Hovy Waseem and Hovy (2016) consists of a set of 136,052 tweets, collected during a 2 months period, from which 16,914 were annotated as "Racist", "Sexist" or "None", with a distribution of 12%, 20% and 68%, respectively. The class unbalance was maintained in order to keep the realism.

The inter-annotator agreement is k = 0.84, having 85% of all disagreements occurred in annotations of "Sexism".

Considering the length of the dataset and the feasibility of the annotations, this dataset has been broadly used in hate speech detection.

Table 2.13 summarizes the approaches and techniques used on detecting hate speech in Waseem & Hoovy's dataset.

Article	Preprocessing	Features	Algorithms	F1-score
Pitsilis et al. (2018)	Tokenization Fixed tweets' length	- Word-based frequency vectorization - Users tendency to sexism/racism/none	LSTM	0.9320
Founta et al. (2018)	Removed infrequent tokens Fixed tweets' length	- GloVe embedding - Random embedding for new words - Number of hashtags - Number of mentions - Number of emojies - Number of capitalized words - Number of URLs - Tweet polarity score - Users' popularity: - Number of friends - Number of followers - Ratio between friends and followers - Users' position in the network: - Clustering coefficient - Authority - Eigenvector - Closeness centrality	GRU	0.8900
Watanabe et al. (2018)	Removed URLs Removed mentions Removed invalid characters Decomposed hashtags Tokenization Lemmatization	- Tweets polarity score - Score of pos/neg words - Ratio positive/negative words - Number of pos/neg slang words - Number of pos/neg emojies - Number of pos/neg hashtags - Number of exclamation/question marks - Number of full stops - Number of all-capitalized words - Number of quotes - Number of interjections - Number of laughing expressions - Number of words in tweet - Word 1-grams - Replaced words with polarity + PoS	C4.5	0.8780
Mishra et al. (2018a)	Lowercase Removed stopwords Fixed tweets' length	- Community graph (connections between authors) - Node2vec embedding - Char n-grams	GRU	0.8757
Mishra et al. (2018b)	Lowercase Removed stopwords	- Char n-grams - Augmented word-sum	LSTM	0.7980
Zimmerman et al. (2018)	Removed URLs Removed mentions Removed numbers Tokenization Fixed tweets' length	- Unknown word embedding	CNN ensemble	0.7862
Qian et al. (2018a)	Fixed tweets' length	- Intra-User representation (tweets history) - Inter-User representation (similar tweets collected)	BiLSTM	0.7740
Agrawal and Awekar (2018)	Removed stopwords Removed punctuation Lowercase	- Word 1-grams - SSWE embedding	SVM	0.77

Table 2.13: Approaches used in hate speech detection in Waseem & Hoovy's dataset.

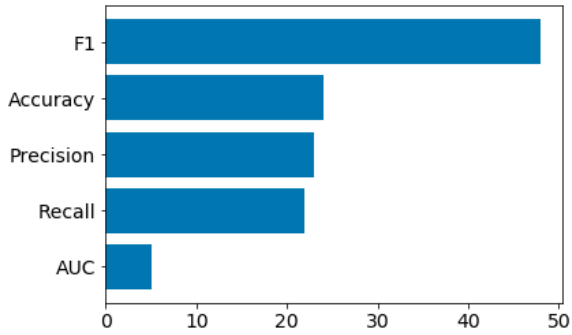


Figure 2.17: Frequency of performance measures used in the literature.

2.5.3.2 TRAC-1

TRAC-1 2018 was the first workshop on Trolling, Aggression and Cyberbullying, consisting of a shared task on aggression identification. The data consists of a collection of 4 different datasets from 2 different languages and 2 different platforms:

- English Facebook
- English Twitter
- Hindi Facebook
- Hindi Twitter

Our focus is the English Facebook (EF) dataset since English is a language we are more familiar with and the Twitter one has a significantly smaller amount of data, made available for testing purposes.

The data has 15,000 instances and each comment is annotated as "Overtly Aggressive", "Covertly Aggressive" or "Non-aggressive" with an unbalanced unknown distribution.

Table 2.14 summarizes the approaches and techniques used on detecting hate speech in TRAC-1's English Facebook dataset.

2.5.3.3 Germeval

Germeval is a shared task based on offensive language identification, created in order to initiate and foster research on the identification of offensive content in German language microposts, namely Twitter.

The task itself consists of 2 sub tasks:

- Binary classification - 2 classes: "Offensive" and "Other".
- Fine-grained classification - 4 classes: "Profanity", "Insult", "Abuse" and "Other".

Article	Preprocessing	Features	Algorithms	F1-score
Aroyehun and Gelbukh (2018)	Lowercase Removed punctuation Removed numbers Removed URLs Removed invalid characters Decomposed hashtags Translated emojies	- FastText embedding - Character n-grams	LSTM	0.6425
Modha et al. (2018)	Fixed tweets' length	- FastText embeddings	LSTM	0.6178
Golem et al. (2018)	Removed non alpha tokens Tokenization Lowercase Fixed tweets' length	- GloVe embeddings - Presence of swear word - Number of PoS tags - Text length - Number of capitalized words - Number of named entities - Text sentiment polarity	BiLSTM	0.616
Risch and Krestel (2018)	Tokenization Decomposed hashtags	- Number of characters - Relative number of uppercase characters - Relative number of non-alpha characters - Relative number of exclamation marks - FastText embeddings	GB	0.6060
Samghabadi et al. (2018)	Lowercase Removed URLs Removed numbers Stemming	- Word n-grams - Char n-grams - K-skip n-grams - TF-IDF - Word2Vec embeddings - Sentiment mean - Sentiment std deviation - Gender probability	LR	0.5921
Orasan (2018)	Tokenization Emojies translation	- GloVe embeddings - Text sentiment score - Emojies sentiment score	RF	0.5830
Nikhil et al. (2018)	Removed punctuation Spell checked Lemmatization Tokenization	- Number of words with pos/neg sentiment - Number of punctuations - Total number of words - Unknown embedding	LSTM	0.5746
Maitra and Sarkhel (2018)	Removed stopwords Stemming Decomposed hashtag Fixed tweets' length	- Word-count vectorization - Sentiment score	Autoencoder	0.5694
Roy et al. (2018)	Removed URLs Normalized hashtags Removed punctuation Fixed tweets' length	- 1-grams - TF-IDF - GloVe embeddings	CNN	0.5151

Table 2.14: Approaches used in hate speech detection in TRAC-1's dataset.

Article	Preprocessing	Features	Algorithms	F1-score
Rother and Rettberg (2018)	Tokenization (freq >5) Removed invalid characters Fixed tweets' length	- Presence of URLs - Presence of mentions - Random embedding	LSTM	0.8000
Schneider et al. (2018)	Fixed tweets' length	- Gender information - Retrieved profile description of mentioned users - Tweets sentiment score - Generated new training set based on users' friends labeled automatically - FastText embeddings	CNN	0.7790
Scheffler et al. (2018)	Tokenization Stemming Replaced URLs Replaced mentions Lemmatization Removed stopwords	- Word2Vec embeddings - Character n-grams - TF-IDF - Bag of words - Number of words all capitalized - Number of swear words - Number of punctuations - Tweets sentiment score	SVM	0.7700
Wiedemann et al. (2018)	Tokenization Lowercase	- FastText embeddings - Topic extraction of mentioned users cluster	BiLSTM-CNN	0.7749
Stammbach et al. (2018)	Tokenization Removed punctuation Removed invalid characters Removed emojies Lowercase Normalized hashtags Removed mentions Fixed tweets' length	- Heidelberg embeddings - N-grams	CNN	0.7590
Bai et al. (2018)	Upsampling Fixed tweets' length	- Word2Vec embeddings - Tweets length - Number of swear words	SVM-CNN	0.7445
Schäfer (2018)	Tokenization Removed mentions Removed invalid characters Normalized hashtags	- Number of swear words - Tweets length - Number of words starting with uppercase - Number of mentions - Number of mentions on the first half og the tweet - Number of mentions on the second half of the tweet - Number of hashtags - Number of punctuation marks - Number of reduplications of punctuation marks - Number of special characters (mostly emojies) - Number of words with uppercase letters - Sentiment features	CNN	0.7369

Table 2.15: Approaches used in hate speech detection in Germeval's dataset.

Targeting the binary classification task, the data consists of an unbalanced set of 5009 tweets, where 34% are "Offensive" and 66% are "Other".

Table 2.15 summarizes the approaches and techniques used on detecting hate speech in Germeval's binary dataset.

Chapter 3

Extraction and selection of textual features

Our goals for this dissertation have been defined upon conducting a systematic literature review on the topic of hate speech detection in text. We collected and grouped the literature we found to be the most relevant in recent years (2018 and late 2017) and found some limitations surrounding the topic that we think should be addressed in order to improve research in this area.

As mentioned in the previous section, hate speech detection was, for many years, a topic poorly addressed, taking into account the lack of detection approaches conducted until 2017. By then, and mostly in 2018, the topic's attention grew significantly (see Figure 2.4). Although we consider such growth a major breakthrough (also due to a bigger awareness from social media platforms and society in general), the rise of that many approaches, in a short period of time, advocated the conduction of parallel methodologies. In an attempt to optimize them, we extracted, grouped and compared different textual features frequently used in hate speech detection, combined with preprocessing techniques:

- **Preprocessing**
- **Sentiment**
- **Semantic**
- **Vectorization**

Besides, we also noticed that most of the natural language processing methodologies and tools focus on text processing as a whole, instead of grouping it according to its source. Naturally, formal text would have different nuances when compared to informal corpora, such as comments from social network platforms. Applying the same tools on different types of text could affect the results on classifying text or, in this case, identifying hate speech. In order to fill this gap, we also developed a tool able to tokenize tweets specifically. We provide the results obtained compared against state of the art tokenizers.

3.1 Dataset

The experiments described in this chapter were conducted using a subset of a Twitter dataset originally created by Waseem and Hovy (2016). The original dataset consists of a total of 136,052 tweets, reduced to 16,914 annotated examples of three different classes: *Racist*, *Sexist* and *None*. We chose this collection of tweets, since this is the most commonly used dataset in recent hate speech detection approaches (Figure 2.10), having a high inter-annotator agreement (k = 0.84). It was labeled by 3 people - the authors themselves and a 25 year old woman studying gender studies and non activist feminist. Since the tweets were encoded with their respective id's, our first step was to extract the original content of the data, resulting in a total of 11,223 tweets, from which 25 are racist, 2,990 are sexist and 8,208 contain no type of hate. We missed out on almost 5,000 tweets since their authors had been banned by the time we performed the extraction. Considering our goal is to perform binary classification, we merged both *sexist* and *racist* classes into a single *hate* class, ending up with a total of 3,015 hateful tweets and 8,208 non hateful.

***Hate* class oversampling** Despite having a considerable number of hateful instances (3,015), most of these (2,990) are sexist. Conducting experiments on a dataset with such distribution would probably lead to the creation of a model able to detect sexism but not hate in general. Thus, in order to complement the *hate* class, we extracted a collection of hateful tweets from another dataset Founta et al. (2018). We discarded any tweet that was explicitly sexist and added 210 extra examples of hate to our dataset, resulting in a final distribution of 3,225 *hate* (28%) and 8,208 *non hate* (72%) tweets, with a total of 11,433 examples.

3.1.1 Methodology

Since our goal in this thesis is mostly based on comparing results using different combinations of features, we created a methodology to conduct all the necessary testing by following a traditional train, validation and test setting commonly used in most machine learning tasks Bishop (2007). To accomplish this, our first step consisted on splitting our data into a training set, containing 75% of the total instances, and a testing set, containing the remaining 25%. Due to the unbalanced distribution of classes, typical in hate speech detection datasets (Table 2.4), the split was done by using a stratified approach, i.e. we kept a balanced number of positive (*hate*) and negative (*no hate*) instances in each train and test split. The training data resulted in a total of 8,575 tweets, with 2,401 *hate* and 6174 *non hate* examples, whereas the testing data resulted in a total of 2,858 tweets, with 800 *hate* and 2058 *non hate* examples.

Train, validation and test We used the training set to train and validate the features we extracted throughout the experiments. For that matter, we performed 5 fold cross validation and computed the average (arithmetic mean) f-score, *hate* class precision and recall to assess the performance of features both individually and within each feature group, e.g. for the sentiment features we computed the average measures obtained by the 5 different cross validation results, for each feature

individually and for the best combination of sentiment features. After extracting and evaluating all feature groups, we tested different combinations of feature groups using the testing set to obtain the final results. Pipeline 3.1 summarizes the methodology followed to assess the experiments conducted.

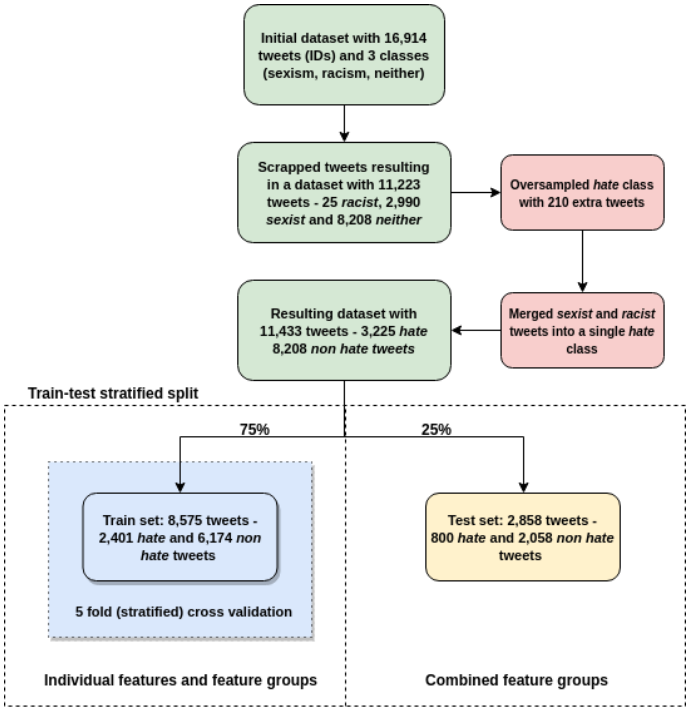


Figure 3.1: Pipeline followed to test different textual features.

3.2 Tweets tokenization

Machine learning algorithms cannot directly parse the data used in text classification problems. Consequently, this text has to be tokenized, i.e. split into its tokens, whether the goal is to count their frequency or group them according to some criteria (e.g. n-grams). This process is typically simple if we address formal and structured text, such as the example shown in Table 3.1, but tweets have nuances that make the task harder. Twitter comments tend to be unstructured, informal and often contain hashtags, mentions, (sometimes purposefully) misspelled words (e.g. *f*ck*), non-alphanumeric characters mixed in-between, abbreviations and so on. In order to be able to successfully tokenize tweets it is important to understand the concept and structure behind both hashtags and mentions, as described on the following paragraphs.

Mentions (similar to replies): When referring to another particular twitter user or account, authors can use mentions. These are composed of an initial character "@", followed by an user name (e.g. @user123). The characters following the "@" must be alphanumeric, except for "_", which is considered to be valid. A mention containing non-alphanumeric characters is either fully

Table 3.1: Example of a sentence tokenization

Sentence	*This is a (simple) example of tokenization.*
Tokens	*"This", "is", "a", "(", "simple", ")", "example", "of", "tokenization", "."*

invalid, if such character comes immediately after the "@" (e.g. @?user123), or partially valid, if such character doesn't come right after the "@" (e.g. @user?123). In the last case, the mention is only valid until the occurrence of the non-alphanumeric character. Table 3.2 shows some examples of valid, partially valid and invalid mentions and how they should be splitted accordingly.

Table 3.2: Examples of hashtags/mentions and their respective proper tokenization (how it should be done).Note that mentions and hashtags suffer exactly the same tokenization process.

Hashtag\Mention	Tokenizer split	Fully valid	Partially valid	Invalid
@Donald_Trump	"@Donald_Trump"	x		
#Donald_Trump	"#Donald_Trump"	x		
@Donald-Trump	"@Donald", "-", "Trump"		x	
#Donald-Trump	"#Donald", "-", "Trump"		x	
@-Donald_Trump	"@", "-", "Donald", "_", "Trump"			x
#-Donald_Trump	"#", "-", "Donald", "_", "Trump"			x
@123Trump	"@123Trump"	x		
#123Trump	"#123Trump"	x		
@_Trump	"@_Trump"	x		
#_Trump	"#_Trump"	x		

Hashtags: Hashtags are user-generated metadata tags which allow users to group related messages within a specific topic or content. For example, the hashtag *#Trump* contains tweets and other content mostly related to the President of the United States, Donald Trump. Hashtags are semantically similar to mentions. Although the initial character is an hash, "#", the following characters must be alphanumeric with the exception of "_". Table 3.2 shows some examples of valid, partially valid and invalid hashtags and how they should be splitted accordingly.

3.2.1 Common Python tokenizers

Being *Python* the programming language under which we conducted our experiments, we highlighted three common tokenizers developed in the same language: **Spacy** [ExplosionAI (2015)] and **NLTK**'s both word and tweet tokenizers Steven Bird (2001b). All three tokenizers use a different algorithm to split the tokens, hence we felt the need to understand how they behave individually. We generated a sentence by combining different tweets containing as many tweet-related particularities as possible (e.g. hashtags, mentions, hidden profanity) , but removed the real user names mentioned in order to avoid identifying real users. We used the three tokenizers aforementioned to split the sentence and compare the various approaches, as denoted by Table 3.3.

Table 3.3: Tokenization done by common Python sentence tokenizers.

Generated tweet	*I f*cking hate when @random_user tw33ts $hitty non_relevant g@y content!! #WakeUp #hate-you @friend-12*
NLTK word	'I', 'f*cking', 'hate', 'when', '@', 'random_user', 'tw33ts', '$', 'hitty', 'non_relevant', 'g', '@', 'y', 'content', '!', '!', '#', 'WakeUp', '#', 'hate-you', '@', 'friend-12'
Spacy	'I', 'f*cking', 'hate', 'when', '@random_user', 'tw33ts', '$', 'hitty', 'non_relevant', 'g@y', 'content', '!', '!', '#', 'WakeUp', '#', 'hate', '-', 'you', '@friend-12'
NLTK tweet	'I', 'f', '*', 'cking', 'hate', 'when', '@random_user', 'tw33ts', '$', 'hitty', 'non_relevant', 'g', '@y', 'content', '!', '!', '#WakeUp', '#hate-you', '@friend', '-', '12'

Obviously, different tokenizers have diferent ways of splitting the tweet. Although the splits are mostly well-defined we think that they fail in a few cases:

- **NLTK word** considers non-alphanumeric characters (except "_") to be individual tokens. This poses a problem when identifying mentions and hashtags. In the example, in Table 3.3, the mention *@random_user* is splitted into the tokens @ and *random_user*, while it should preserve the mention. The same happens for hashtags (e.g. *#WakeUp* is tokenized into *#* and *WakeUp*). It also splits, on one hand, *g@y* into *g*, @ and *y*, *$hitty* into *$* and *hitty* but, on the other, doesn't split the token *non_relevant* into *non*, "_" and *relevant*.

- **Spacy**, on one hand, preserves the mentions but, on the other, ignores hashtags (e.g. *#WakeUp* is splitted into *#* and *WakeUp*). The word *$hitty* is also tokenized into *$* and *hitty* and *non_relevant* kept as a whole token.

- **NLTK tweet**, as the name suggests, was created to specifically tokenize tweets. As expected, it tokenizes correctly all the valid hashtags and mentions of the tweet, except for partially valid hashtags: *#hate-you* is tokenized as is, while it should be *#hate*, "-" and *you*. NLTK tweet tokenizer also fails to consider words with non alphanumeric characters: *$shitty* is tokenized into *$* and *hitty* but the token *non_relevant* is kept, when it shouldn't.

It is clear that the tokenizers originally available in Python fail in a few cases when splitting tweets, being the *NLTK tweet* tokenizer the most accurate one for these circumstances. Thus, and since Twitter is the social network platform most commonly addressed for hate speech detection, we realized it would be useful to create a twitter tokenizer similar to NLTK's that would surpass the aforementioned limitations. We consider these limitations as such, since hateful discourse often contains subtle hidden profanity that is unlikely to be acknowledged otherwise. Profanity is not necessarily a synonym of hate, but they are usually associated Fortuna (2017).

3.2.2 Twikenizer: our tweet tokenizer

Taking into account the limitations mentioned before, we developed a tokenizer for Twitter, named *Twikenizer*, that operates similarly to NLTK's (tweet) but considers more cases. It's main features are:

Table 3.4: Tokenization example for *Twikenizer* and NLTK's tweet tokenizer. The differences between both are highlighted in bold for the *Twikenizer* tokens

Generated tweet	*I f*cking hate when @random_user tw33ts $hitty non_relevant g@y content!! #WakeUp #hate-you @friend-12*
Twikenizer	'I', **'f*cking'**, 'hate', 'when', '@random_user', 'tw33ts', '**$hitty**', '**non**', '_', '**relevant**', '**g@y**', 'content', '!', '!', '#WakeUp', '**#hate**', **'-**', '**you**', '@friend', '-', '12'
NLTK tweet	'I', 'f', '*', 'cking', 'hate', 'when', '@random_user', 'tw33ts', '$', 'hitty', 'non_relevant', 'g', '@y', 'content', '!', '!', '#WakeUp', '#hate-you', '@friend', '-', '12'

- **Hashtags tokens**: *Twikenizer* considers hashtags as tokens if they are fully valid (e.g. *#hashtag*). For partially valid hashtags (e.g. *#hash?tag*), it considers the valid hashtag and splits the rest of the tokens (e.g. *#hash*, "?", *tag*), unlike NLTK's that considers the full partially valid hashtag as a token. Invalid hashtags, i.e. words that start with a hash followed by a non-alphanumeric character (except "_"), are splitted as any other token: *#-hash* is tokenized into "#", "-" and *hash*.

- **Mentions tokens**: Mentions are handled similarly to hashtags. Fully valid mentions are tokenized as a whole token (e.g. *@mention*), while partially valid ones are splitted into the valid mention plus the rest of the tokens: *@mention-1* is splitted into *@mention*, "-" and "1". This feature is shared with NLTK's tweet tokenizer.

- **Underscore separation**: Words separated by "_" are splitted accordingly, except for hashtags and mentions. The word *non_relevant* is splitted into *non*, "_" and *relevant*. NLTK's tweet tokenizer considers underscores as part of the token, so the same word would be considered a whole token.

- **Non-alphabetic words**: *Twikenizer* doesn't split words that contain both alphanumeric and non alphanumeric characters (except for punctuation). This is the way we chose to handle hidden profanity. The word *$hitty* remains as is, so does *g@y*. On the other hand, a word that also contains punctuation marks is splitted accordingly: *$h!t* is splitted into "$", "h", "!" and "t". This feature might be particularly useful when combined with the **levenshtein distance** for profanity identication. The word *f*cking*, tokenized as is, has a levenshtein distance of 1 when compared to *fucking*. Using NLTK tweek tokenizer the same word splits into "f", "*" and *ucking*. In the best case scenario (using the last token, *ucking*), the levenshtein distance would be 2.

Finally, Table 3.4 shows how *Twikenizer* splits the tweet in comparison to NLTK's tweek tokenizer. Under certain circumstances, *Twikenizer* can possibly be a better solution to tokenize tweets, when compared to NLTK's. It is expected to output the best results combined with the levenshtein distance to profane words feature, in a dataset containing hidden profanity. The following section presents the results using both *Twikenizer* and NLTK's tweet tokenizer.

3.2.3 Results comparison

In order to evaluate *Twikenizer*'s splitting methodology, we conducted a set of experiments and compared the results with the ones obtained by the other tokenizers (Spacy and NLTK's). To achieve this, we used the dataset described in Section 3.1 and performed a stratified 5-fold cross validation, keeping an equivalent number of instances for both positive (hateful) and negative (non hateful) classes. We used a term frequency bag of words and a linear SVM algorithm to obtain the average f-score for each tokenizer, as displayed in Table 3.5.

Since *Twikenizer* considers words containing non-alphanumeric characters, eventually masking profanity, we extracted an extra feature to test whether tokenizing such words could be useful in improving the detection of hate speech. To accomplish this, for each tweet, we computed and stored the minimum levenshtein distance of each token compared to 2 dictionaries of slang words and profanity Tan (2017), Friedland (2013). Results show that using this feature with *Twikenizer* can improve the detection of hate speech.

Table 3.5: Running time and average f-score of the 5 folds cross validation for each tokenizer. The running time regards the time taken to tokenize the whole corpus (11,223 tweets). Results in bold are the best for each group of features.

Tokenizer		**F-score (%)**		**Running time**
		TF BoW	**TF BoW + MinProfDist**	
NLTK	Word	68.063	68.363	6.85s
	Tweet	69.612	70.000	4.34s
Spacy		**70.921**	71.014	2.54s
Twikenizer		70.307	**72.178**	10.12s

3.3 Data cleaning

Considering tweets' peculiarities and data cleaning techniques used in hate speech detection, we used a set of tweets' preprocessing techniques to clean the data before we transformed it further or extracted any other feature, displayed in Table 3.6. Noise brings no predictive advantage to any classification task and often skews the results obtained, but it is crucial to clearly identify what may be considered noise in the data, especially in tweets. URL's themselves are an example, since Twitter has it's own service of URL shortening, converting any to a prefix (*http(s)://t.co/*) followed by a set of random characters (e.g. *eXyu2FGR*). Thus, there is no added value in considering an URL to classify tweets, other than acknowledging their existance (as done in the semantic analysis - Section 3.4.2). On the other hand, mentions and other user identifiers (e.g. emails, retweet tags) may be relevant in improving the detection of hate speech, since there may be users who are more frequently targeted by hate instigators than others.

Different data cleaning methods were combined together and tested how they would affect the detection of hate speech. The results in Table 3.7 show that URL's and HTML codes indeed bring no prediction advantage, hence removing them is beneficial, as long as long as their presence is considered in the semantic analysis. On the other hand, removing users' identifiers significantly

Table 3.6: Data cleaning techniques used on tweets.

Data cleaning method	**Example**
Removed HTML codes	&amp, &lt
Removed retweet tags	RT @mention
Removed URL's	http://t.co/KGT0swVfCp
Removed mentions	@mention
Removed emails	user@gmail.com

drops the performance of all measures in the detection of hate speech, indicating that certain users may be more prone to being targets of hate. Thus, for the upcoming experiments, we only cleaned the URL's and HTML codes, keeping all the other content.

Table 3.7: Performance using different data cleaning methods.

Data cleaning feature	**F-score (%)**	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
None	70.344	82.289	61.463
Removed URL's and HTML codes	**70.428**	**82.371**	**61.551**
Removed users identifiers (mentions, retweet tags, emails)	67.250	79.772	58.151
All data cleaning features	67.737	80.231	58.631

3.3.1 Preprocessing

Besides the basic data cleaning techniques applied in the previous section, we collected and extracted a set of other common preprocessing procedures, based on the ones listed in Table 2.5. Most of these are common techniques used in natural language processing (lowercase, reduction to words' root form and removal of words and characters), with the addition of a couple that specifically address tweets (emojies and hashtags treatment). The groups of preprocessing features tested are briefly described in the paragraphs below, along with their potential value in detecting hate speech in tweets.

Lowercase All characters were converted to lowercase. This is helpful in reducing the dimensionality of the data, since capitalized words are interpreted equally to non capitalized words. In a model without characters converted to lowercase, *Case*, *case* and *CASE* would be interpreted (eventually tokenized) as 3 different words. Despite it's usefulness, hate is often correlated with the usage of capital words and characters ElSherief et al. (2018b), hence acknowledging these may be helpful in detecting hate speech (addressed later in the semantic analysis).

Reduction to words' root form Reducing words to their root form consists on removing their suffixes, by reducing the words' inflections to their root forms. This is yet another feature usually beneficial to the reduction of the data's dimensionality, since words with similar meaning are converted to the same stem Van Rijsbergen et al. (1980), e.g. the words *affects*, *affection*, *affected* and *affecting* are converted to the stem *affect* using the Porter's stemming algorithm Maitra and

Sarkhel (2018). Although this is useful in most cases, there are exceptions in which the meaning of the word might be altered. An example is the stemming of *plane* and *planned*, which is *plane*. *Planned* is the past tense of *to plan* and *plane* (in this case) refers to an airplane. Although both words share the same stem, their meaning is quite different. Aiming to reduce words to their root form, we used 2 different techniques:

- **Stemming**: tokens were reduced using Porter's stemming algorithm Maitra and Sarkhel (2018). The resulting root form may not be a word.
- **Lemmatization**: the process is similar to stemming's, although it makes sure that the root form generated belongs to the language, using a dictionary. For that purpose, we used WordNet's Frenda et al. (2018).

Word and characters removal Removing certain words and characters may also be beneficial in reducing the sparsity of the data. **Stop words** are words that occur frequently in text data and typically bring no meaning to the message being passed, hence removing them is a technique quite common in natural language processing and hate speech detection (Table 2.5). Common approaches tend to use pre-compiled dictionaries or other methods for their dynamic identification, however the technique used to identify them may be crucial in truly separating noise from useful data [Saif et al. (2014)]. In our experiments we used NLTK's stop words pre-compiled dictionary Steven Bird (2001a). Other authors also remove characters such as **non-alphanumeric** ones and **punctuation**. We also considered these preprocessing features in our experiments to assess how would their removal affect the results.

Emojies treatment Emojies are pictures that accompany the text that usually symbolize emotions and sentiments, whether they are true or not (e.g. sarcasm). They also tend to be useful in identifying the message and sentiment conveyed by the text, hence most literature considers them in their textual analysis. Aiming to assess their influence in hate speech detection, we tested three different approaches. We **removed** them from the tweets, **ignored** their occurrence and **translated** them to their description. To accomplish the latter, we used the Python library *emoji* Kim and Wurster (2017) and did some further parsing. As an example. for the translated emoji *:thumbs_up:*, we removed the colons both in the beginning and end of the string and split the keywords. For the example provided, the final string would be *thumbs up*.

Hashtags treatment As mentioned before, hashtags are user-generated metadata that group related messages with a specific topic. These are usually helpful in identifying the topic being addressed in that specific tweet, hence their presence may be relevant in detecting hate speech. Aiming to test their influence in automatic hate speech detection, we considered 4 different features. We **removed** them from the tweets, **ignored** their occurrence, **simplified** them by removing their hash (e.g. *#hashtag* was converted to *hashtag*) and **decomposed** them. To do so, we removed

the hash and split the compound words into their constituents using Python's *compound-word-splitter* library Kampik (2017), e.g. #hateyou is converted to hate you. This approach poses some limitations, since splitting streams of text into the corresponding words is an ambiguous task. For the same hashtag, there may be multiple splitting possibilities.

Using the features described above, we conducted a set of tests to gauge how they would influence, individually, the performance of text classification tasks. The results, displayed in Table 3.8, show that lowercasing and stemming turned out to be the preprocessing features that improved the results the most, with an f-score increase of around 1% and 2%, respectively. While the overall performance decreased using lemmatization, there was a slightly higher number of *hate* tweets being correctly identified. On one hand, removing stopwords and non-alphanumeric characters also contributed to slightly improve the overall results (f-score) of the model, but, on the other, the identification of hateful tweets (recall) deteriorated by almost 1% in both cases. Removing punctuation increased the *hate* class precision, however this was due to a substantial decrease of the model's f-score (almost -2%) and recall (almost -3%). Thus, we discarded this feature for future experiments. Finally, regarding Twitter-related features, ignoring emojies and the decomposition of hashtags into their compound words produced the best results out of the features considered to handle them.

Table 3.8: Results obtained on individual preprocessing features tested against the baseline computed on Table 3.7. Results were obtained using a term frequency bag of words and a Logistic Regression algorithm. The ones in bold improved the baseline. Precision and Recall concern the class *hate*.

Preprocessing feature		**F-score (%)**	**Precision ((%)) (Hate class)**	**Recall (%) (Hate class)**
Baseline		70.428	82.371	61.551
Lowercase		**71.596**	82.203	**63.468**
Stemming		**72.403**	81.992	**64.863**
Lemmatization		70.406	81.759	**61.855**
Removed stopwords		**70.429**	**83.752**	60.809
Removed non-alphanumeric		**70.553**	**84.009**	60.853
Removed punctuation		68.601	**82.431**	58.804
Emojies	Removed	70.272	82.178	61.420
	Translated	70.379	82.229	61.551
Hashtags	Removed	68.153	80.358	59.197
	Simplified	70.415	82.229	61.551
	Decomposed	**71.196**	81.783	**63.076**

Upon testing the extracted preprocessing features individually, we investigated further which combinations would work out the best in our model. With that in mind, each experiment conducted consisted of a set of static features, i.e. that we kept as a base, combined with others that produced results close to the baseline (either above or below), which we considered to be inconclusive. **Lowercasing**, **stemming**, **decomposing hashtags** and **ignoring emojies** were the features we considered as our static ones, whereas **lemmatization** and the **removal of stop words** the inconclusive ones. Table 3.9 shows that lemmatization slightly decreased the f-score of the

model for both experiments *Xp2* and *Xp4*, when compared to *Xp1*. On the other hand, removing stop words improved the overall performance, increasing the f-score by 0.2%, despite lowering the *hate* class recall by around 0.1%. For this reason, we considered the preprocessing features used in *Xp3* to be the best for our task. The improvements over the baseline (without any preprocessing features) were 2.7%, 1.4% and 3.6% for the f-score, *hate* precision and *hate* recall, respectively.

Table 3.9: Results obtained combining different preprocessing features with a term frequency bag of words and tested on a Logistic Regression algorithm. Results in bold are the best for each performance measure. Base features are lowercasing, stemming, ignoring emojies and hashtags decomposition.

Experience	**F-score (%)**	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
Baseline (no preprocessing features)	70.428	82.371	61.551
Xp1: Base features	72.889	82.551	**65.299**
Xp2: Base features + lemmatization	72.833	82.409	**65.299**
Xp3: Base features + stop words removal	**73.123**	83.484	65.181
Xp4: Base features + lemmatiz. + stop w. removal	73.122	**83.775**	64.907
Improvement over baseline (*Xp3*)	2.7%	1.4%	3.6%

3.3.2 Data dimensionality analysis

High dimensionality is usually a problem in classification tasks, mainly when the number of examples in the data is small compared to the number of dimensions. This translates into an exponential raise of the problem's complexity (increasing computing time to generate the models), leads to lack of data to efficiently train a model Bishop (2007), eventually resulting in overfit Verleysen and François (2005). Cleaning and preprocessing data is one efficient method to reduce it's dimensionality Tang et al. (2005), as demonstrated by the pipeline in Figure 3.2. We encoded the tweets using a term frequency bag of words and, for a total of 11,384 raw tweets, 21,946 different features were generated - almost the double. Upon cleaning the data (removing URL's and HTML codes), around 1,300 features were removed and, after using the preprocessing techniques mentioned in the previous subsection, another 7,194 dimensions were discarded, resulting in a feature reduction of almost 40% when compared to the original raw tweets' dimensions.

3.4 Feature extraction and selection

Aiming to tackle the lack of uniformity in the extraction and selection of features for hate speech detection, we identified a set of features commonly used in the field (highlighted in Section 2.4), and extracted and selected the ones that performed better in the detection of hate in Twitter comments (tweets), under the specific environment we setup: dataset chosen, number of classes considered, etc.

We grouped the sets of features according to their nature:

- **Sentiment**: features related to the corpus' sentiment (e.g. tweets' sentiment score).

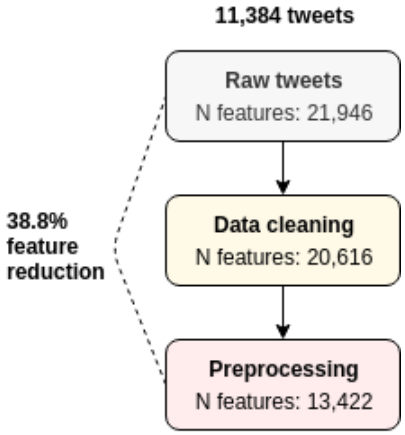


Figure 3.2: Data cleaning and preprocessing pipeline

- **Semantic**: includes all the features related to the semantic of the corpora, such as the number of words per comment, average word length, among others.
  - Punctuation: considers punctuation-related features (e.g. number of exclamation marks).
  - Word: features related to the words individually (e.g. average word length).
  - Characters: features related to the characters individually (e.g. number of capital letters).
  - Twitter: twitter-related features, such as number of mentions and hashtags.
- **Vectorization**: includes all the features able to vectorize the tokens and characters of the tweets (e.g. bag of words, n-grams).

3.4.1 Sentiment analysis

More often than not, and especially on social network platforms, people tend to formulate opinions on a diversity of topics: reviews, ratings, recommendations, among other forms of online expression. Identifying the sentiment(s) behind these opinions usually turns out to be useful in extracting insights from the data. The main idea behind sentiment analysis is to detect any positive and/or negative words or expressions, relying heavily on the direct meaning of words Watanabe et al. (2018). Explicit, extremist, hate discourse is, typically, easily identified and often conveys a negative message, e.g. *I fucking hate Muslims*, but this isn't always the case. Unfortunately, mainly on the perspective of detection strategies, hate is often subtle and not easily noticeable Fortuna (2017). Furthermore, regular text may contain negative words, hate synonyms or the word (hate) itself, but the context associated may not be related to hate discourse. A good example is the tweet: *I freaking hate waking up early! My life sucks!!!*. The word *hate* has been employed on this generic example and the text itself has an overall negative connotation, although this is clearly not an example of hate speech. This is why the usage of sentiment analysis techniques is arguable

for hate speech detection in text. To strengthen this claim, we manually selected 2 different tweets from the dataset used in the experiments (described in Section 3.1).

Tweet 1, *t1*: *@Forbes U.S. government gives them away FOR FREE... arming funding training terrorists is what the US does best!!!!*

Tweet 2, *t2*: *Feel sorry for Annie; Lloyd. Such a bad choice. I do have soft spot for Lloyd. #mkr*

According to the annotation done, the tweet *t1* was labeled as *hate*, while *t2* was labeled as *non hate*. For each tweet, *t1* and *t2*, we computed the sentiment score using a common Python library, *TextBlob* Loria (2013), where the sentiment polarity score threshold goes from -1 (most negative sentiment) to 1 (most positive sentiment), being 0 neutral. The results obtained, displayed in Table 3.10, show that, although *t1* is considered to be a hateful tweet, it's sentiment polarity score is quite close to 1, while *t2* is considered to be a negative tweet, especially due to the usage of the words *bad* and *sorry*, with an individual sentiment polarity score of -0.5 and -0.699, respectively.

Table 3.10: Sentiment score and label of example tweets.

	Polarity score	**Label**
t1	0.7	*Hate*
t2	-0.274	*Non hate*

Considering the most common sentiment-related features implemented in hate speech detection (see Table 2.6), we selected, extracted and tested a set of features individually on our data, described below:

Overall tweets sentiment score We computed 3 different mutually exclusive features to extract the overall sentiment of tweets:

- **Sentiment score**: computed using *TextBlob* Loria (2013), this feature is the single overall sentiment score of the tweet.
- **Sentiment subjectivity score**: also computed using *TextBlob*, the sentiment subjectivity score outputs both the sentiment score of the tweet and the subjectivity of the classification. The subjectivity introduces the concept of ambiguity when scoring the words' sentiment.
- **Multiple sentiment score**: this feature was extracted using *VaderSentiment* Gilbert (2014). Although it also computes the sentiment of text, it provides a wider range of outputs. For each classification, the positive, negative, neutral and compound sentiment score is calculated, providing a more encompassing approach.

Words sentiment score The sentiment for each word was computed using *TextBlob* and the results combined differently, generating a set of new features:

- **Positive words score**: for each word with a sentiment score higher than 0.2 (the threshold goes from -1 to 1), the positive words score is incremented with the value.
- **Negative words score**: similar to the positive words score, but with negative words (sentiment score lower than -0.2).
- **Positive words count**: number of words with a sentiment score above 0.2.
- **Negative words count:** number of words with a sentiment score below -0.2.
- **Slang words score**: sum of the sentiment score for each slang word in the sentence.
- **Positive hashtags count**: number of hashtags with positive sentiment (above 0.2).
- **Negative hashtags count**: number of hashtags with negative sentiment (below -0.2).
- **Negative verbs count**: number of verbs with negative sentiment (below -0.2).

The results displayed in Table 3.11 show that sentiment-based features barely had an impact (individually) on the classification task, where only the *multiple sentiment score* and *slang words score* features yielded positive results in detecting hateful tweets (*hate* class recall). Thus, we merged these 2 features and computed their combined performance and improvement over the baseline. These combined features improved the baseline's f-score, *hate* precision and recall by 0.3%, 0.3% and 0.2%, respectively, still not granting a significant refinement of the results as seen in Table 3.11.

Table 3.11: Results obtained on on both individual and combined sentiment features tested against a baseline with no (sentiment) features, all encoded with a term frequeny bag of words and tested on a Logistic Regression algorithm. Results in bold improved the baseline. Precision and recall concern the *hate* class.

Sentiment features	**F-score (%)**	**Precision (%) (*Hate* class)**	**Recal (%)l (*Hate* class)**
Baseline	73.123	83.484	65.181
f1: Sentiment score	**73.202**	**83.545**	65.168
f2: Sentiment subjectivity score	**73.219**	**83.599**	65.168
f3: Multiple sentiment score	**73.368**	**83.769**	**65.299**
f4: Positive words score	**73.141**	**83.529**	65.081
f5: Negative words score	73.114	**83.610**	64.994
f6: Positive words count	73.092	83.469	65.037
f7: Negative words count	**73.128**	83.570	65.037
f8: Slang words score	**73.269**	**83.653**	**65.212**
f9: Positive hashtags count	73.023	81.594	65.120
f10: Negative hashtags count	73.088	83.404	65.081
f11: Negative verbs count	73.044	83.121	65.077
f3 + *f8*	**73.416**	**83.827**	**65.342**
Improvement (*f3* + *f8*)	**0.3%**	**0.3%**	**0.2%**

3.4.2 Semantic analysis

Unlike sentiment analysis, that considers only the sentiment conveyed by text, semantic analysis actually considers every aspect of it, in an attempt to capture the real meaning of text, by identifying the elements and assigning them to their logical and grammatical role. Although cleaning

and preprocessing the data is mostly beneficial, it's inevitable to lose some, eventually valuable, information during the process (e.g. URL's). This is one of the reasons that makes semantic analysis useful in text classification. It is common to see the use of punctuation or employment of capitalization associated with segregation and aggressive or even hateful discourse Watanabe et al. (2018). A pratical example is the following tweet:

If we want the opinion of a WOMAN, we'll ask you dear... For now keep quiet.

The usage of the capitalized word in this tweet clearly aims to emphasize the discrimination of gender towards women. Lowercasing characters is a preprocessing feature that would ignore this subtle occurrence. This is a single example of how keeping track of punctuation, capitalized words, etc, might be useful in detecting hate speech online. There are a lot of possible ways to conduct a semantic analysis. The most common features to do so, used in hate speech detection, are summarized in Table 2.8. We have extracted and grouped them according to their category on the following paragraphs.

Punctuation features In order to track down the occurrence of punctuation throughout the text, we extracted 4 different features: The overall **number of punctuation marks**, the **number of exclamation marks**, **question marks** and **full stops**. We tested each feature individually and results, displayed in Table 3.12, show that they bring no advantage in detecting hate speech for this dataset in particular.

Table 3.12: Results obtained on individual semantic (punctuation) features tested against the baseline with no (semantic) features, all encoded with a term frequency bag of words and tested on a Logistic Regression algorithm. No results have improved the baseline. Precision and recall concern the *hate* class.

Punctuation features	**F-score (%)**	**Precision (%) (*Hate* class)**	**Recall (%) (*Hate* class)**
Baseline	73.123	83.484	65.181
Number of exclamation marks	73.123	83.484	65.081
Number of question marks	73.123	83.484	65.081
Number of full stops	72.477	80.991	65.605
Number of punctuation marks	73.123	83.484	65.081

Word features Word-based semantic features may be relevant in classifying text, especially considering some subtleties are discarded upon cleaning and preprocessing the data. As mentioned before, lowercasing characters will automatically ignore any possible capital letters or words, hence acknowledging them may be relevant. We consider a word to be a set of characters, with a size higher than 1, containing at least 1 alphanumeric character, e.g. *sh#t* is considered to be a word. Words may be 1 character long if that character is alphabetical. For each, we extracted a set of features described below.

- **Number of all-capitalized words**

- **Ratio between all-capitalized words and total number of words**
- **Number of words**
- **Number of (only) alphanumeric words**
- **Ratio between the number of alphanumeric words and total number of words**
- **Number of syllables (only alphanumeric words are considered for this task)**
- **Average word length**
- **Ratio between the number of slang words and total number of words**
- **Number of adjectives**
- **Number of interjections**

For each feature listed above, we conducted individual experiments using a term frequency bag of words and a Logistic Regression algorithm. Results, displayed in Table 3.13, show that acknowledging capitalized words and their relative frequency in each tweet has a positive impact on the overall performance of the model, however the identification of hateful tweets is worse when compared to the baseline (slightly decreases by 0.02%). Thus, we considered the number of words, ratio of slang words and number of adjectives the best semantic features assessed, being the latter the one with higher improvements - increase of 0.4% and 0.2% of f-score and *hate* recall, respectively. We conducted a final experiment using the 3 features mentioned and obtained an overall improvement of 0.5% (f-score), and almost a 1% increase in the detection of hate, as displayed in Table 3.13.

Table 3.13: Results obtained on both individual and combined semantic (word) features tested against the baseline with no (semantic) features, all encoded with a term frequency bag of words and tested on a Logistic Regression algorithm.. Results in bold improved the baseline. Precision and recall concern the *hate* class.

Word features	**F-score (%)**	**Precision (%) (*Hate* class)**	**Recall (%) (*Hate* class)**
Baseline	73.123	83.484	65.181
f1 Number of all capitalized words	73.060	83.455	64.994
f2 Ratio of all capitalized words	**73.153**	83.419	65.168
f3 Number of words	**73.485**	83.428	**65.691**
f4 Number of alphanumeric words	72.899	83.411	64.775
f5 Number of syllables	72.885	**83.795**	64.602
f6 Average word length	73.105	**83.570**	64.645
f7 Ratio slang words	**73.215**	82.522	**65.264**
f8 Number of adjectives	**73.577**	**83.673**	**65.691**
f9 Number of interjections	73.076	83.510	64.994
f3 + *f7* + *f8*	**73.597**	**83.224**	**65.997**
Improvement (*f3* + *f7* + *f8*)	**0.5%**	**-0.3%**	**0.8%**

Character features Character features target characters individually instead of sets or words. For these, we extracted the **number of capital letters**, **characters** and **special characters** and computed the results output by each feature individually as seen in Figure 3.14. Both the number of capital letters (*f1*) and special characters (*f3*) features increased the overall performance of the

model (both f-score and hate detection). On the other hand, while acknowledging the number of characters slightly improved the detection of hateful tweets, the f-score also decreased. Thus, for our final experiment, we combined *f1* and *f3*, having obtained a f-score and *hate* class recall improvement of 0.5% and 0.7%, respectively. However, combining these features actually output poorer results than *f3* alone, with an improvement of almost 1% for both f-score and *hate* recall.

Table 3.14: Results obtained on both individual and combined semantic (character) features tested against the baseline with no (semantic) features, all encoded with a term frequency bag of words and tested on a Logistic Regression algorithm. Results in bold improved the baseline. Precision and recall concern the *hate* class.

Character features	**F-score (%)**	**Precision (%) (*Hate* class)**	**Recall (%) (*Hate* class)**
Baseline	0.73123	0.83484	0.65181
f1: Number of capital letters	**73.258**	**83.607**	**65.212**
f2: Number of characters	73.015	82.325	**65.648**
f3: Number of special characters	**73.897**	**83.920**	**66.040**
f1 + *f3*	**73.613**	**83.456**	**65.866**
Improvement (*f3*)	**0.8%**	**0.4%**	**0.8%**

Twitter features Finally, Twitter-based semantic features contemplate all the peculiarities related to the social network platform, for tweets in particular. The features extracted are:

- **Number of mentions**; as mentioned before, hate discourse is often targeted at some entity or person in particular. Mentions are a way to address these targets.
- **Number of hashtags**
- **Number of URL's**
- **Number of emojies**; hate and other extremist opinions are often complemented by emojies Salminen et al. (2018).

The number of mentions (*f1*) and emojies (*f4*) were the only Twitter-based features that improved the baseline results as seen in Table 3.15. These 2 combined resulted in a improvement of 0.5% and 0.6% of the f-score and *hate* recall, respectively. Note that, despite *f4* output a worse *hate* recall than the baseline, it boosted the identification of hate tweets when combined with *f1*.

3.4.2.1 Combined semantic features

Finally, using all the best punctuation, word, character and Twitter semantic features identified in the previous paragraphs, we tested all the possible different combinations for each feature subgroup (punctuation included). Although punctuation-based features had no positive impact on the model individually, when combined with other features, results may improve. For this feature subgroup we only considered the total number of exclamation and question marks, since they may be the most representative kind of punctuation used in expressive discourses. Table 3.16 summarizes the results obtained by each feature group in the experiments conducted in the previous subsections and the results obtained by the feature groups combined. Punctuation still didn't contribute

Table 3.15: Results obtained on both individual and combined semantic (tweets) features tested against the baseline with no (semantic) features, all encoded with a term frequency bag of words and tested on a Logistic Regression algorithm. Results in bold improved the baseline. Precision and recall concern the *hate* class.

Twitter features	**F-score (%)**	**Precision (%) (*Hate* class)**	**Recall (%) (*Hate* class)**
Baseline	73.123	83.484	65.181
f1: Number of mentions	**73.485**	**83.552**	**65.605**
f2: Number of hashtags	73.077	83.430	65.037
f3: Number of URL's	73.042	83.417	64.994
f4: Number of emojies	**73.155**	**83.491**	65.125
f1 + f4	**73.630**	**83.723**	**65.735**
Improvements (*f1 + f4*)	**0.5%**	**0.2%**	**0.6%**

to improving the results. On the other hand, the combination of semantic word (*bwf*) and character (*bcf*) features produced the best overall results with an f-score and *hate* recall increase of 0.9% and 1.4%, respectively.

Table 3.16: Results obtained on semantic feature groups individually and combined tested against the baseline with no (semantic) features, all encoded with a term frequency bag of words and tested on a Logistic Regression algorithm. Results in bold improved the baseline. Precision and recall concern the *hate* class.

	Feature subgroups	**F-score (%)**	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
	Baseline	73.123	83.484	65.181
Individual semantic subgroups	Punctuation feats. (pctf)	73.123	83.484	65.081
	Best word feats. (bwf)	73.597	83.224	65.997
	Best char. feats. (bcf)	73.897	**83.920**	66.040
	Best Twitter feats. (btf)	73.630	83.723	65.735
Combined semantic subgroups	*pctf + bwf*	73.061	82.950	65.342
	pctf + bcf	73.880	83.876	66.040
	pctf + btf	73.630	83.723	65.735
	bwf + bcf	**74.009**	83.327	**66.607**
	bwf + btf	73.904	83.811	66.127
	bcf + btf	73.830	83.746	66.040
	pctf + bwf + bcf	73.576	83.885	65.561
	pctf + bwf + btf	73.839	83.901	65.953
	bwf + bcf + btf	73.765	83.543	66.084
	pctf + bwf + bcf + bft	73.938	83.806	66.171
	Improvement (bwf + bcf)	0.9%	-0.2%	1.4%

3.4.3 Vectorization

Most machine learning algorithms only take numeric inputs. Thus, in natural language processing, it is required to vectorize the data (text), i.e. represent each token numerically, whether it is a word, set of words, character or set of characters. Thus far, in all experiments conducted, the data was encoded using a term frequency bag of words, a vectorization technique that collects the set of tokens present in the corpus, disregarding grammar and the order in which they appear, while

keeping multiplicity Jurafsky and Martin (2014). In this subsection, we consider other approaches using TFIDF and n-grams.

TFIDF The term frequency method (TF) accounts for the absolute frequency of the tokens in the corpus. This was the base feature used in all the experiments previously described. The TFIDF considers the frequency of each token according to it's inverse frequency in the corpus. This means that tokens with less occurrences have a weighted frequency value higher than those with high occurrences.

N-grams The tokens bagged by bag of words depend on the n-grams used (contiguous sequence of tokens). Word n-grams consider words, or sets of words (if n is higher than 1), as tokens, while character n-grams consider characters, or sets of characters. In the previous experiments we used 1-grams (single tokens). In this subsection we try different combinations of n.

We conducted a significant number of experiments, where we tested different combinations of n, with a lower bound ranging from 1 to 4 and an upper bound ranging from 1 to 6, i.e. n: [1,1], ..., [1,6], [2,3], ..., [3,4], ..., [4,6]), along with both TF and TFIDF encodings. Similarly to the other experiments, we used bag of words (BoW) with cleaned and preprocessed tweets and tested with a Logistic Regression algorithm. The features used are summarized below.

- TF Word n-grams BoW
- TF Character n-grams BoW
- TFIDF Word n-grams BoW
- TFIDF Character n-grams BoW

For each experiment, we picked the best setup (lower and upper n bound) that outperformed the others in all the performance measures listed. N-grams with significant discrepancies between the n's lower and upper bound (e.g. (1-6)-grams) generate a large number of additional features, increasing exponentially the dimensionality of the training data. Thus, we limited the number of maximum features generated by the bag of words model to the 12,000 (a value close to the number of tweets in our dataset) most relevant by excluding tokens with both high and low frequency that belonged to both classes. This is a process automatically done by the Python library *scikit-learn* Pedregosa et al. (2011) using the *CountVectorizer* model and limiting the amount of features by assigning the pre-defined value to the *max_features* parameter.

Results in Table 3.17 show that TF character [3,4]-grams are the features that improve the most the detection of hate, with an increase of around 3% of the class' recall versus the baseline. As a consequence, the identification of non hate tweets slightly drops, with an overall decrease of the model's performance, with f-score dropping by 0.3%. Additionally, TFIDF encoding generally produces worse results for all the performance measures considered versus TF.

Table 3.17: Best results for each different combination of n-grams and encodings (TF and TFIDF) tested on a Logistic Regression algorithm.Results in bold improved the baseline. Precision and recall concern the *hate* class.

		F-score (%)	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
	Baseline	73.123	83.484	65.181
xp1	TF Word [1,6]-grams BoW	**73.181**	82.566	**65.735**
xp2	TF Character [3,4]-grams BoW	72.783	77.722	**68.438**
xp3	TFIDF Word 1-grams BoW	65.168	**89.893**	51.132
xp4	TFIDF Character [1,2]-grams BoW	67.515	**88.273**	54.707
	Improvements (*xp2*)	-0.3%	-5.8%	3.2%

3.4.3.1 N-grams analysis

We plotted the results for both word and character n-grams for each n lower and upper bound considered in the experiments, as seen in Figure 3.3 and 3.4. We observed that word n-grams obtain the best results for small values of n's lower bound, likely due to tweet's typically short length, informality and lack of structure, along with the presence of typos and diminutives, making it harder to find sets of words often occurring together. Additionally, single words are more likely to portray hate than their combinations. We also observed that the performance of the model using word n-grams drops significantly for each increment of n's lower bound. On the contrary, character n-grams achieved better results for higher values of n, outperforming word n-grams in all performance measures and for most combinations of n. This is the case because single or pairs of characters aren't long enough to portray relevant information. Overall, character n-grams (encoded with TF) are a better choice in hate speech detection in tweets.

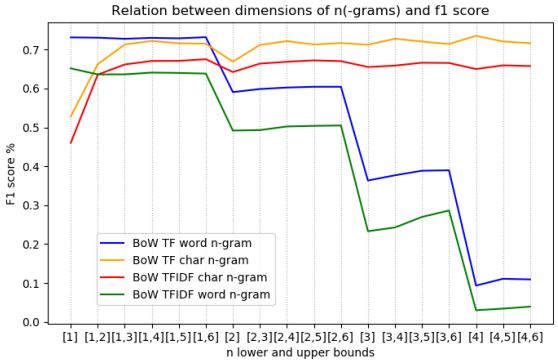


Figure 3.3: Variation of f-score with different n lower and upper bounds

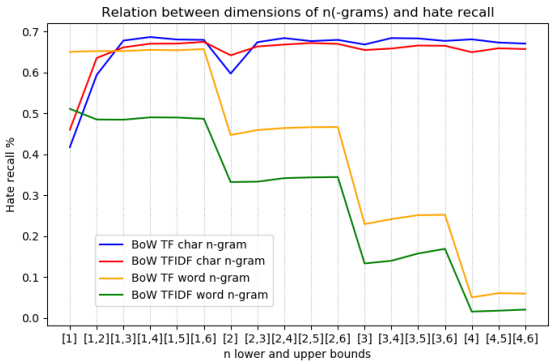


Figure 3.4: Variation of hate recall with different n lower and upper bounds

3.4.4 Features combination and analysis

We conducted a set of final experiments on the testing set, using different combinations of the feature groups (with the best individual features) extracted in the previous subsections. These experiments are relevant since we can observe how these features perform when combined together

on unseen data. We computed a baseline without any features, except for a term frequency bag of words to compare the results obtained by the different combinations. For all experiments we used a TF encoding, since it outperformed TFIDF in all previous tests. The combinations tested are summarized below:

- **Baseline**: No features, except for a TF bag of words
- ***c1***: Data cleaning and best preprocessing features (DCPP) , Table 3.9.
- ***c2***: DCPP + best sentiment features (Sent), Table 3.11.
- ***c3***: DCPP + best semantic features (Sem) , Table 3.16.
- ***c4***: DCPP + character [3,4]-grams (CG34), Table 3.17.
- ***c5***: DCPP + Sem + CG34.
- ***c6***: DCPP + Sent + CG34.
- ***c7***: All features (DCPP + Sent + Sem + CG34).

Table 3.18 shows the results achieved by the different combinations of features. Despite improving baseline's performance measures, sentiment features slightly decreased the overall performance of the model when combined with semantic and character n-grams. Experiment *c5* achieved the best results with an f-score and *hate* recall improvement of 1.5% and 4.6%, respectively, in detriment of the class' precision, with a 3% drop. Results show that cleaning and preprocessing tweets, analyzing their semantic and extracting common sets of characters (n-grams) tend to be relevant techniques to improve the detection of hate speech on Twitter. While sentiment analysis techniques may also contribute to such improvements, their use prejudiced the predictions of the model, strengthening the claim that these features are arguable in hate detection tasks Watanabe et al. (2018).

Table 3.18: Results obtained by the different combinations of feature groups using a TF bag of words and tested on a Logistic Regression algorithm. Results in bold concern the experiment with the best results. Precision and recall address the *hate* class.

	Features	**F-score (%)**	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
	Baseline	73.989	84.202	65.985
c1	DCPP	74.512	84.580	66.586
c2	DCPP + Sent	74.680	84.627	66.826
c3	DCPP + Sem	75.921	83.866	69.350
c4	DCPP + CG34	75.243	81.895	69.591
c5	**PP + Sem + CG34**	**75.498**	**81.189**	**70.552**
c6	PP + Sent + CG34	75.242	81.398	69.951
c7	All features	75.532	81.589	70.312
	Improvement (c4)	1.5%	-3%	4.6%

To better understand which words and sets of characters were prominent for the *hate* class, we generated the 10 most common n-grams in our data for this class, displayed in Table 3.19. We chose the *n*'s lower and upper bound that output the best results on previous experiments (Table 3.17), character [3,4]-grams and word [1,2]-grams. It is clear that sexism is quite incident in this

dataset with n-grams such as *girl*, *sexi*, *xist*, *girl cook* and *sexist*. On the other hand, n-grams such as *idiot*, *idio*, *hate*, *fuc* point out to a more generalized type of hate, including profanity and insults.

Table 3.19: Most common character and word n-grams for the *hate* class.

Char [3,4]-grams	**Word [1,2]-grams**
cook	cook
hick	girl cook
hole	whole
girl	idiot
sexi	half bird
chic	mkr
idio	chicken
xist	im not
mkr	hate
fuc	sexist

Chapter 4

User Profiling

On the previous chapter we provided an overview of which text-based features generally perform better in hate speech detection on Twitter. Although the processing of natural language is the most common technique used to classify text, Twitter allows for the introduction of a new set of features, related to the authors of the text, or tweets in this case. As mentioned before, user profiling is an under-explored area in hate speech detection, mostly due to the limitations associated with the extraction of users' characteristics, which are further explained in Section 4.2.6. Thus, in Chapter 4, we aim to enrich research on the detection of hate speech on Twitter, by exploring a set of user profiling methodologies that have been poorly or not addressed at all before. For this matter, we explored a set of features briefly described below:

- **Gender information**: We identified the gender of the authors who posted the tweets and the users mentioned in those. The first feature mentioned proved to achieve good results in the detection of hate speech Waseem and Hovy (2016). We investigated further whether considering the gender of the mentioned users is also relevant for the task.

- **Users' Twitter history**: Tweets are considerably short which makes it hard to identify relevant nuances that can be interpreted as hateful. Having users' tweets history analyzed eases the process of identifying the nature of a certain tweet. Using their posts history, we generated a tendency towards hateful behaviors score.

- **Users' Twitter account characteristics**: The characteristics of the account belonging to the user who posted the tweet may portray relevant information about the user himself (e.g. number of friends, followers).

- **Users' Twitter network**: Similarly to the user account's characteristics, generating a user's network of interactions between him and other users (friends and followers) may help predicting behavior and interaction patterns of users with a tendency towards hate.

In the following subsections we provide more in-depth details of each feature extracted and the results they achieved, along with the methodology used to conduct such experiments.

4.1 Dataset

For the experiments described in the following sections, we used a subset of the data used in the previous chapter (more details provided on Section 3.1). Twitter enforces rate limits for the usage of it's API, making it unviable (time wise) to extract the networks of almost 8,000 different users. In order to extract an ego network of an user with 1,000 friends and 1,000 followers (see Section 4.2.5.1 for further detail on ego networks), we need to extract each friend and follower's list of connections, resulting in a total of 2,001 API calls required to generate the network; 1 for the main user, 1 for each of his friends and another for each of his followers. Twitter limits such request to 120 calls per hour, thus, in order to reduce the time consumed extracting content from the social network platform, we only extracted tweets (from the previous dataset - Section 3.1), whose authors had a total sum of friends and followers lower than 400. This resulted in a dataset with a total of 1,318 tweets and 760 different users (around two tweets per author), with 920 non hateful instances, 234 examples of sexism, 160 hateful tweets and 3 racist ones. The last 3 classes mentioned were merged together and converted into a single *hateful* one, resulting in a total of 920 negative (*non-hateful*) classes and 397 positive (*hateful*) classes.

Data augmentation In order to increase the size of the data used in the experiments, we scrapped, for each user, the 5 most recent tweets posted on their timeline, resulting in a total of 3,740 scrapped tweets. These were automatically annotated using a pre-trained model created in Waseem and Hovy (2016). We chose this approach, since this was a highly cited (138 citations) hate speech detection article, where the authors collected a dataset of tweets and predicted the hateful ones with good results (f-score=0.7389), using character n-grams and gender information. Besides, the model was trained on a larger corpus. Despite being a 3-class classification task (*sexism*, *racism* and *neither*), we merged *sexism* and *racism* into a single *hateful* class, resulting in a total of 428 *hateful* and 3,312 *non hateful* automatically annotated tweets. We complemented our dataset with all the *hateful* tweets and, due to the highly unbalanced distribution of the automatically labeled classes, we only included 1,000, randomly sampled, *non hateful* scrapped tweets. The final dataset resulted in 825 examples of *hate* and 1,920 examples of *non hate*, with a total of 2,745 instances.

4.1.1 Methodology

We followed a similar testing methodology as the one described on Section 3.1.1, by splitting the data (using a stratified approach) into a training and testing set, containing 75% and 25% of the original data, respectively. The training set resulted in a total of 2,059 tweets with 618 examples of hate, whereas the testing set resulted in a total of 686 tweets with 206 examples of hate.

Train, validation and test We used the training set to train and validate the features we extracted throughout the experiments. For that matter, we performed 4 fold cross validation and computed the average (arithmetic mean) f-score, *hate* class precision and recall for each fold, to assess the performance of each feature. We chose a lower number of folds (compared to Chapter 3.1.1) due

to the lower number of instances. In the end, we tested different combinations of features using the test set to obtain the final results. Pipeline 4.1 summarizes the methodology followed to assess the experiments conducted.

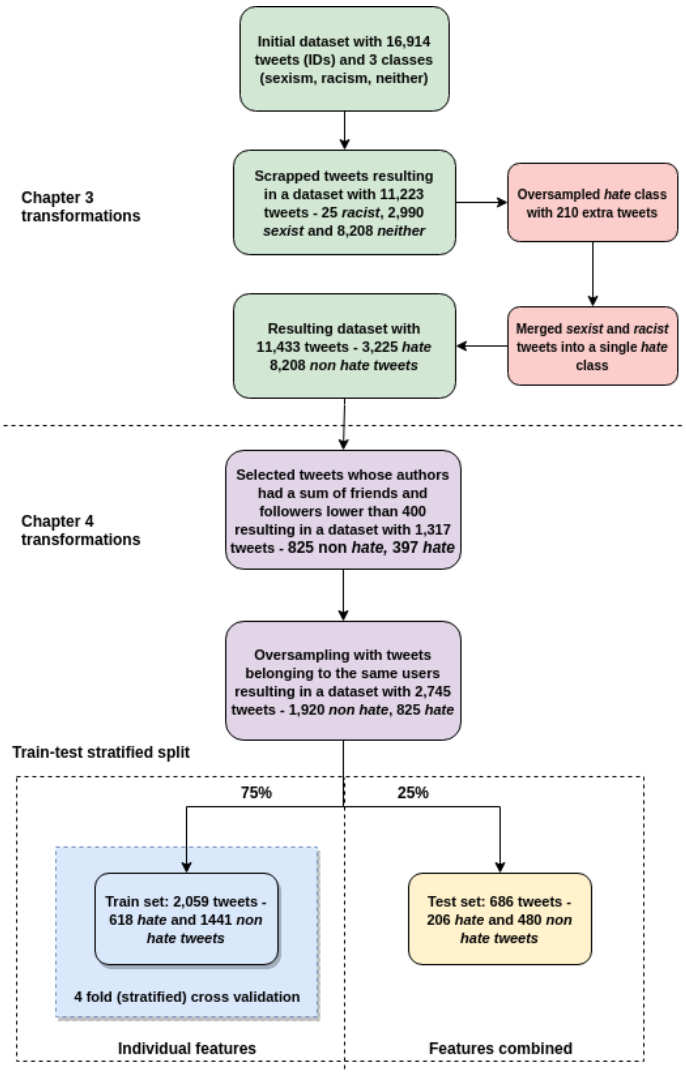


Figure 4.1: Pipeline followed to test different user profiling features.

4.2 Profiling features

4.2.1 Baseline

In order to test the different user profiling features extracted, we set up a baseline experiment to be able to compare the results obtained. For this task, we used similar features as the ones described on the previous section, i.e. data cleaning, preprocessing and TF vectorization with character n-grams. Since the data used on these experiments is quite different from the one used on the previous chapter, we tested other combinations of features:

- **Preprocessing**: we included all the data cleaning techniques described on Section 3.3, but, regarding the preprocessing features, we converted the tweets to **lowercase**, **stemmed** and **lemmatized** them, **removed stopwords** and **decomposed hashtags**.
- **Vectorization**: we used a **TF bag of words** model with a **character [3,4]-grams** model.
- **Algorithm**: **SVM** with a radial basis fuction (*rbf*) kernel.

The results obtained using the features described above are detailed in Table 4.1.

Table 4.1: Results obtained for the baseline.

	F-score (%)	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
Baseline	67.724	71.910	64.000

4.2.2 Gender information

Bullies typically address targets based on the perceived or actual ethnicity, behavior, physical characteristics, sexual orientation, class or gender Silva et al. (2016b). Thus, identifying these traits can be helpful in detecting hate speech as observed by Waseem and Hovy (2016) and [Klubicka and Fernández (2018), where gender information, combined with word n-grams outputs better results than just word n-grams. The main limitation of using gender-based features is that Twitter doesn't provide gender information of it's users, hence it needs to be somehow inferred. There are two main techniques to identify the gender of Twitter users, described below:

- **User names**: most approaches compare the authors' user names with a dictionary of gender-labeled users Waseem and Hovy (2016), ElSherief et al. (2018a), Schäfer (2018). Although this is quite effective in determining gender, user names don't often represent an actual name, especially on social networks such as Twitter. In Waseem and Hovy (2016), only about 53% of the users have their gender identified, not granting full precision either on those labelings. This is mainly due to ambiguous names, which may suit male or female individuals. Besides, it is not guaranteed that a certain user will have a name matching his gender.
- **Text data**: this approach is less frequent and doesn't grant much effectiveness either, since tweets are short and, more often than not, portray few information about gender. This is a machine learning classification approach that aims to identify the users' genders based on the content of their tweets, using gender-labeled text. There are some data sources that may be used for this task:
  - ***TwiSty corpus*** is a *multilingual Twitter Stylometry corpus for gender and personality profiling* Verhoeven et al. (2016). It contains information regarding 18,168 authors (including gender information), for which a set of tweets (id's) are provided. Using

this data, Schneider et al. (2018) created a model to identify the gender of tweets' authors for a dataset of German tweets. There is no English version of the data.

- The ***Twitter14k Dataset***, created by Bamman et al. (2014), contains gender information of 14,464 Twitter users, with the most frequent gender-labeled n-grams found in a 9,212,118 tweets corpus.
- The ***blog-gender*** dataset is a collection of 3,100 posts from blog hosting sites and blog search engines, labeled by 2 groups of students that manually checked the profiles of the authors. The labeling resulted in a set of 1588 (51.2%) blog posts written by men and 1512 (48.8%) written by women. Each post has an average of 250 words for men and 330 words for women Mukherjee and Liu (2010).

In this subsection we focus on how users' gender can influence the detection of hate speech, by identifying the gender of both tweets' authors and users mentioned by them. The next sub-sections described how we implemented these features and the results they achieved.

4.2.2.1 Gender identification approach

Since both approaches have limitations in determining users' gender, we opted to merge both in order to hopefully obtain a better classification.

Our gender-identification pipeline consists of, initially, comparing user names to *name-gender* dictionaries. This was done using 3 different Python libraries: *gender-guesser*, *Genderize* and *NamSor*. We computed the label for each Twitter user name and compared the results obtained by each library, keeping the label output by the majority, i.e. if at least 2 libraries labeled a user as *male*, then the final label is *male* (the same for *female*). The results for the *name-gender* approach are listed in Table 4.2.

Table 4.2: Distribution of users by gender using the *name-gender* approach.

	Female	**Male**	**Unknown**
Number of users	290	222	240
Percentage	39%	30%	31%

This first step managed to successfully label 512 users (out of 760) with their corresponding gender, while 248 others remained unknown. For these, a text-based approach was conducted. We trained a TF bag of words model on 70% of the blog data and tested it on the remaining 30%, keeping a balanced distribution of classes in both train and test sets. We achieved an accuracy of 73% for this task. For each user whose gender had been labeled as *unknown*, we extracted and merged a set of 20 tweets and applied the model trained on the blog data. Since this approach is arguable, as mentioned on the previous sub-section, and a set of users had a low number of tweets posted, we manually checked the predictions made by the model and compared them against their profile. The ones whose prediction was clearly wrong (e.g. user identified as female but profile picture of a man) were changed to what we thought to be their real gender. Users, according to their gender, are evenly distributed as displayed in Table 4.3. Note that we also considered users

with a neutral gender; these include entities such as *Fox News*, *Channel 7*, among others. Our gender identification pipeline is summarized in Figure 4.3.

Table 4.3: Distribution of users by gender.

	Female	**Male**	**Neutral**
Number of users	389	351	20
Percentage	51%	46%	5%

Figure 4.2

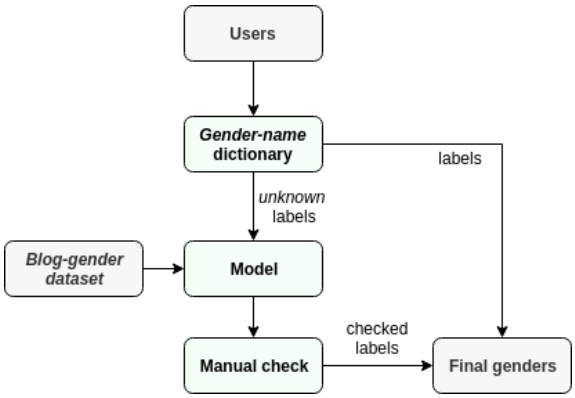


Figure 4.3: Gender identification pipeline.

4.2.2.2 Gender information features

Following the gender identification pipeline summarized in Figure 4.3, we extracted 2 different sets of features:

- ***Feat1***: Tweets authors' gender.
- ***Feat2***: Mentioned users' gender - for each tweet we extracted the gender of the mentioned users.
- ***Feat3***: Includes *feat1* and *feat2*, and the sum of number of friends of followers, individually, for each mentioned user.

Despite being different features, ***Feat1*** is included in ***Feat2***. The latter was extracted in order to understand whether there was a correlation between the gender of an author and the users' he addressed. As mentioned before, instigators and targets are typically characterized by opposing characteristics, whether it is race, religion or gender, so we think identifying authors and mentioned users may be helpful in detecting hate speech online. The results for these features are displayed in Table 4.4.

Table 4.4: Results obtained for the gender information feat

	F-score (%)	Precision (%) (Hate class)	Recall (%) (Hate class)
Baseline	67.724	71.910	64.000
Feat1	**68.421**	**72.222**	**65.000**
Feat2	**70.121**	**73.530**	**67.322**
Feat3	**68.001**	70.992	**64.320**
Improvements (*Feat2*)	**2.4%**	**1.6%**	**3.3%**

All the gender-related features improved the baseline results for hate speech detection, being *feat2* the one with the best results, with an increase of 2.4% of the f-score, 1.6% of the *hate* class precision and 3.3% of the *hate* class recall.

4.2.3 Data augmentation

Tweets are quite short per se. This makes it hard to identify relevant features that may portray hate or other related sentiment conveyed by the text. Besides, hate may often be subtle and ambiguous, making it hard to detect hate speech online.

Data augmentation consists of adding new data points to the original data. It is a common technique used in machine learning, but also in hate speech in particular. Increasing the dimensionality of the data (up to a certain point) can be beneficial to the prediction task. This was done in Qian et al. (2018a), denoted as *inter-user representation*, where tweets with similar content (from different users) were extracted and added to the data available. Fortuna et al. (2018) also merged different datasets for better results. Another data augmentation technique, that relies on users themselves, is the users' history extraction. Although this doesn't directly inject data into the original set, analyzing past tweets may help modeling users' behavior. In Pitsilis et al. (2018), a collection of 400 tweets is extracted for each user, aiming to measure their tendency towards a specific behavior: sexism, racism or neither. This approach produced very good results.

4.2.3.1 User history

We considered an approach similar to the one used in Pitsilis et al. (2018), by extracting a collection of tweets per user and using them to model their behavior on the social network. We collected the 200 most recent tweets per user, for all the 760 in our data. We ended up with a dataset with 92,783 scrapped tweets, considering not all users had made at least 200 posts, and automatically labeled them using the same model as Waseem and Hovy (2016). Again, we converted the scrapped tweets, automatically labeled as *sexist* and *racist*, into a single *hate* class, similarly to the methodology conducted in Section 4.1, resulting in a total of 75,353 non hateful and 17,430 hateful tweets. Using the scrapped data, we computed a user score reflecting their tendency towards hateful behaviors, by considering the number of hateful and total number of tweets posted on each

account. The *user hate score*, ranging from 0 to 1, where 0 is a non hateful user and 1 a hateful user, is described by Equation 4.1. The results obtained using this feature are listed in Table .

$$User\ hate\ score = \frac{Number\ of\ hate\ tweets}{Number\ of\ tweets} \tag{4.1}$$

Table 4.5: Results obtained using the user hate score feature

	F-score (%)	Precision (%) (Hate class)	Recall (%) (Hate class)
Baseline	67.724	71.910	64.000
Tendency	**68.421**	**72.222**	**65.000**
Improvement	**0.7%**	**0.3%**	**1%**

4.2.4 User account activity

Tracking the activity of Twitter users' accounts may be helpful in identifying certain patterns and characteristics. For example, it is more likely for an extremist, hateful user to have less personal information publicly available than any other regular user, whether it is a profile picture, description, etc. Twitter provides some basic metadata about acounts, even for protected users, as described in Table 2.9. Part of these features, such as the number of friends and followers, may not produce the best results possible, since users with a sum of friends and followers higher than 400 were filtered out from the data. The features considered were the following:

***Feat1*: Profile settings**

- Using default image
- Has location
- Has time zone
- Is geo enabled
- Is verified
- Description length
- Name length
- Age (time elapsed since tweet posted and account creation date)
- Has enabled contributors, i.e. tweets may be co-authored

***Feat2*: User activity**

- Number of lists
- Number of statuses (tweets and retweets)
- Number of favourites
- Number of friends
- Number of followers
- Ratio between friends and followers

The results using the features described above are described in Table 4.6. The results obtained by the second group of features, ***Feat2***, don't have an impact on the model, probably due to the filtering of users with a high number of friends and followers. Overall, the best group of features (***Feat1***) only slightly improve the baseline performance.

Table 4.6: Results obtained by the features related to users' accounts. The ones in bold have improved the baseline results.

	F-score (%)	Precision (%) (Hate class)	Recall (%) (Hate class)
Baseline	67.724	71.910	64.000
Feat1	**68.102**	**72.000**	**64.610**
Feat2	66.843	**72.199**	63.891
Feat1 + Feat2	**67.912**	71.876	64.142
Improvement (Feat1)	**0.4%**	**0.1%**	**0.6%**

4.2.5 User network

Social networks, and Twitter in particular, play an important role in propagating information, ideas, opinions and rumors whether they are positive or negative. This cascading phenomenon causes the adoption of more related behaviors and opinions by users who are positioned closer within the whole social network Jin et al. (2013). For this reason, analyzing the networks users belong to, and their interactions between each other, is an important contribution to improve the detection of hate speech online. Founta et al. (2018), Ribeiro et al. (2018) and Chatzakou et al. (2017) consider the users' Twitter network to improve the detection of hate speech, having achieved better results when using these features. Chatzakou et al. (2017) claims that network-based features are the most effective for detecting aggressive user behavior, allegating that bullies post less, participate in fewer online communities, and are less popular than regular users. Despite being a potentially unveiling and novel feature, it is still under explored within the field of hate speech detection. This is mostly due to limitations, imposed by Twitter itself, regarding the extraction of such content from the online platform. The extraction of friends and followers from a big number of users, and their consequent list of interactions, is a rather costly, (mostly) time wise, operation. For this reason, the data used to extract these features was significantly reduced, as mentioned in Section 4.1.

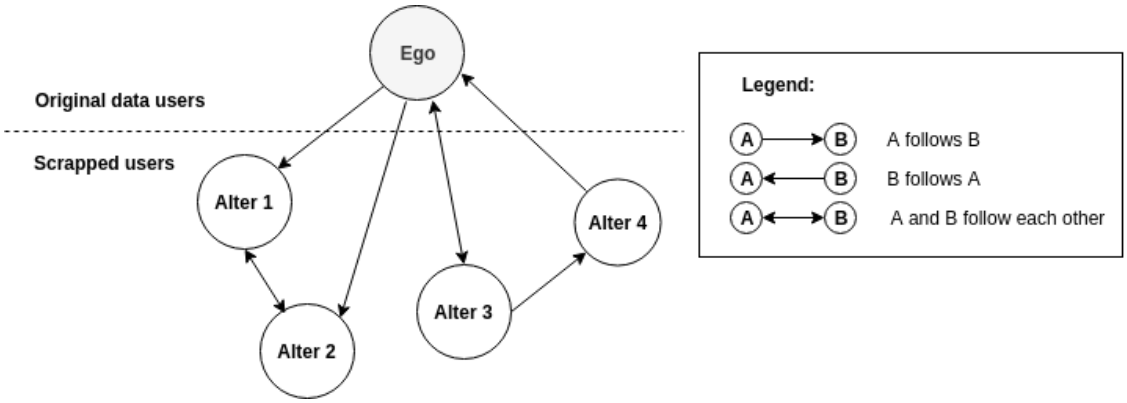
4.2.5.1 Ego networks

The approaches mentioned on the previous section don't provide much in-depth detail regarding the structure of the network, how it was generated or the interactions modeled. In our approach we focused on a user-based network analysis, also known as an **ego network** Chatzakou et al. (2017). These networks are characterized by having a main, central node - the user - connected to a set of alters, or secondary nodes - his friends and followers - and are typically good at inferring the potential for diffusion within the network Easley and Kleinberg (2010). In hate speech detection this might be useful in detecting the propagation and adoption of hateful behaviors.

A total of 760 different ego networks were generated (one per user), considering, for each, the interactions between the user and his connections (friends and followers) and whether the latter were linked with each other or not. The friend/follower connections were differentiated,

generating a directed graph. Figure 4.4 provides an example of an ego network for an user with 2 followers and 2 friends, who are connected between themselves.

Figure 4.4: Example of how users' interactions are modeled in networks. Original users are the authors of the tweets that are originally part of our data. Scrapped users are the ones extracted from the originals' lists of friends and followers.



In order to eventually differentiate hateful and normal users' ego networks, we computed a set of statistics associated with the generated directed graphs. We grouped these features according to the category they belong to: **centrality**, **connectivity**, **homophily** and other subgroups:

Centrality measures Centrality indicators roughly highlight the most important and influential nodes in a network or graph. Since each graph is generated from specific users belonging to the data, it is expected that these measures are heavily skewed towards them. The centrality measures considered for this task are detailed below:

- **Betweenness centrality** quantifies the number of times a node serves as a bridge along the shortest path between two other nodes. Introduced by Freeman (1977), it initially served the purpose of quantifying the control of a human on the communication between other humans in a social network.
- **Closeness centrality** measures the distance of a node to the other nodes of the social network, i.e. the extent to which a user is close to each other user in the network Chatzakou et al. (2017). It indicates how fast information may be spread or a behavior may propagate from a node to the others, sequentially.
- **Degree centrality** measures the popularity of a node in a network by counting the number of connections it is linked to. It may be divided into two sub-measures:
  - **In degree**: amount of incoming connections (user followers).
  - **Out degree**: amount of outgoing connections (user friends).

- **Eigenvector centrality** measures the influence of a user in his network Chatzakou et al. (2017), which is assigned a score based on their neighbours'. A node with a high eigenvector score means that most of it's neighbors also have high eigenvalues.

Homophily From a general perspective, groups of friends are typically similar when it comes to their preferences, hobbies, ethnicity or even physical characteristics. This tendency for similar users to group together is perceptible in social networks, where links tend to connect people who are similar to one another Easley and Kleinberg (2010). Homophily is the principle behind these interactions, and may be visible in platforms such as Twitter, where connected peers typically have similar preferences, e.g. hateful users are more likely to be linked with other hateful users, whereas regular users are connected with other regular ones Chatzakou et al. (2017). Ego networks perhaps don't carry enough information to thoroughly assess similarity between Twitter users, but there are some statistics that may bring out useful information:

- **Average neighbor degree** is defined as the probability that a node of degree k is connected to a node of degree n. This is useful in determining degree-based relations between users and their neighbors in the network Yao et al. (2017).
- **Degree pearson correlation coefficient** is a linear correlation measure of the similarity of connections in the graph with respect to the node degree Foster et al. (2010).
- **Average degree connectivity** or **k-nearest neighbors** is a measure that computes the overall average degree connectivity of the social network Barrat et al. (2004).

Connectivity A graph is considered to be connected if, for every pair of nodes, there is a path between them Easley and Kleinberg (2010). This isn't always the case in ego networks, especially for directed ones, hence analyzing their connectivity might be relevant. Regarding connectivity, we computed a set of features:

- **Connections**
  - Is connected
  - Number of connected components
- **Strong connections**: a directed graph is strongly connected if and only if every node in the graph is reachable from every other node.
  - Is strongly connected
  - Number of strong connections
- **Weak connections**: A directed graph is weakly connected if and only if the graph is connected when the direction of the edge between nodes is ignored, i.e. if replacing all of its directed edges with undirected ones produces a connected undirected graph. Note that a graph may be strongly and weakly connected.

- Is weakly connected
- Number of weak connections

Finally, we computed a set of additional features: **Hubs** and **Authority**, also considered in Chatzakou et al. (2017), a couple of measures originally created to rate web pages. The hub score computes, for a node, the sum of the authority score of the nodes pointing to it. On the other hand, authority reveals how many different hubs a node is connected to Kleinberg (1999). Lastly, we computed the **average clustering** of each ego network. It is defined as the mean of local clusterings, a measure of the degree to which nodes in a graph tend to cluster together.

Using the features described above, we conducted a set of experiments combining different sub-groups of features. Table 4.7 summarizes the results obtained and improvements over the baseline. Centrality measures obtained the best results individually when compared to the other groups of features, having achieved an improvement of almost 2% of the f-score and about 3% of the *hate* class recall. All the other feature groups have also increased the performance measures of the baseline, whereas the combined features achieved the highest performance, with an overall improvement of 4.6% and 6% for the f-score and *hate* class recall, respectively.

Table 4.7: Results obtained by different user network features. All results (in bold) have improved the baseline performance of the model. Precision and Recall concern the class *hate*.

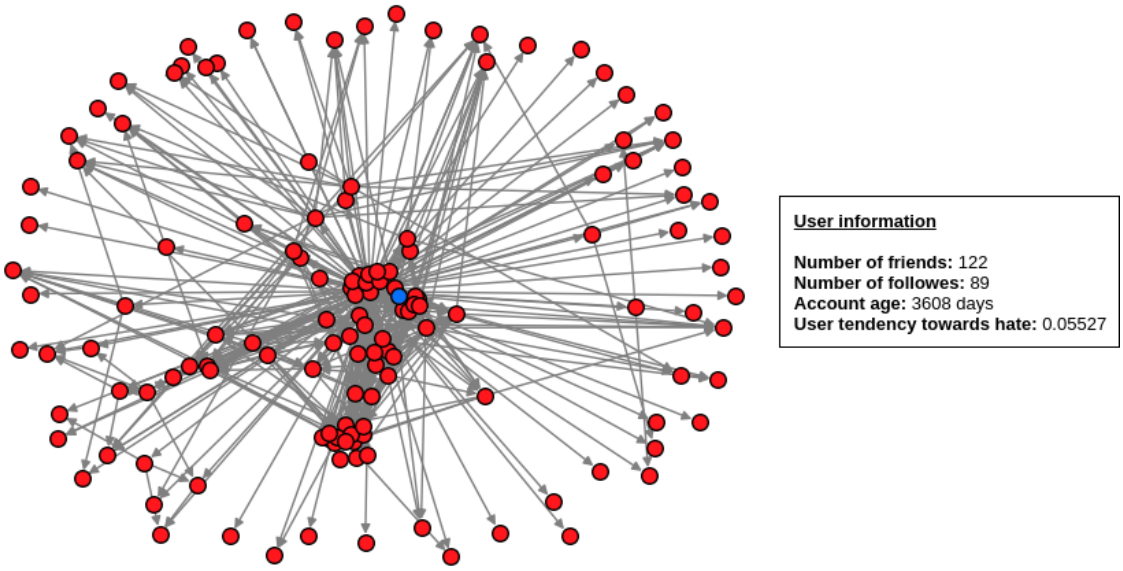
	F-score (%)	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
Baseline	67.724	71.910	64.000
Centrality	**69.012**	**72.331**	**67.121**
Homophily	**67.987**	**73.015**	**65.326**
Connectivity	**68.012**	**72.899**	**65.107**
Others (Hubs, Auth, Clustering)	**68.530**	**72.646**	**66.123**
All features combined	**72.333**	**73.102**	**70.003**
Improvements (all features)	**4.6%**	**1.2%**	**6%**

Figure 4.5 shows an example of a real ego network, computed using the measures of a user belonging to our data.

4.2.5.2 Network analysis

In order to fully understand how network measures may have improved the detection of hate speech, we also computed relations between these statistics and the users' hate score (ranging from 0 to 1, where 1 corresponds to a hateful user and 0 to a non-hateful user). It can be inferred that all centrality measures (eigenvectors, in and out degree, closeness and betweenness) decrease, for most users, as the hate score increases (Figure 4.6). This means that users with a hateful history may not be as central in their network, when compared to other social network users. Similarly, the Hubs and Authority scores also decrease with the increase of the user hate score (Figure 4.7). On the other hand, the homophily scores (degree pearson correlation coefficient, average neighbor degree and average degree connectivity) aren't conclusive (Figure 4.7), since

Figure 4.5: Example of a user ego network and some descriptive statistics associated with the account. The blue node, on the center of the network, is the ego (main user) and the red ones are the alters (secondary users).



they fluctuate similarly regardless of the users' hate score. Finally, as Figure 4.8 shows, users with an increasing hate score tend to not have regular or weakly connected ego networks.

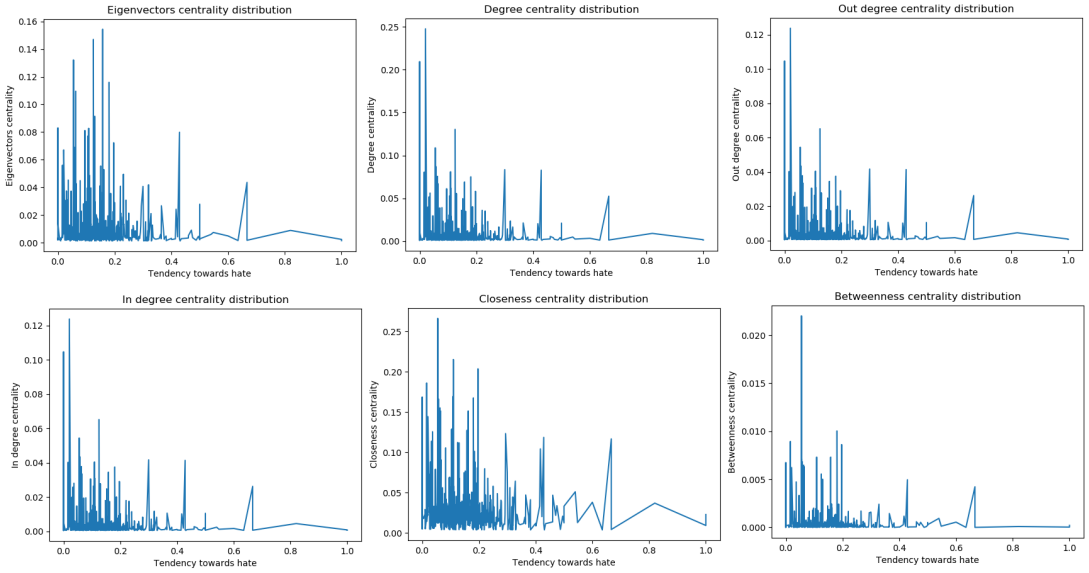
The observations described above converge with the ones mentioned in Chatzakou et al. (2017), who claim that hateful users are less participative in online communities and less popular than regular users, which results in a lower popularity (less central) in their networks.

4.2.6 Features combination and analysis

For each group of user profiling features described in the previous sections, we selected the ones that performed the best in the validation phase and combined them together to test their performance on unseen data (testing set). We computed a baseline using the same features as the ones described in Subsection 4.2.1, to compare the results obtained by the different combinations. We started off by testing each feature individually on the testing data and then each possible combination using a gridsearch approach. This is typically used to optimize hyperparameters in machine learning tasks by exhaustively searching through a specified subset of hyperparameters Wang et al. (2015). In this case, we tested which combinations of features output the best results and presented the 3 best in Table 4.8, along with the results for the features individually.

The overall improvements using user profiling features aren't directly comparable with the ones obtained on the previous chapter, considering the data used on the experiments conducted on this chapter was undoubtedly shorter and probably less informative. Despite this fact, it is unequivocal that user profiling features provide the model deeper understanding of the data (hence

Figure 4.6: Variation of centrality measures according to user tendency score. The measures on top, from left to right, are *eigenvectors*, *degree* and *out degree* centrality measures. The measure on the bottom, from left to right, are *in degree*, *closeness* and *betweenness* centrality. For all centrality measures it is perceptible that, as users' tendency score increases, the centrality scores decrease.



the overall improvements being higher), since they focus on a much wider range of insights regarding the authors and not the tweets themselves, which are often short and ambiguous.

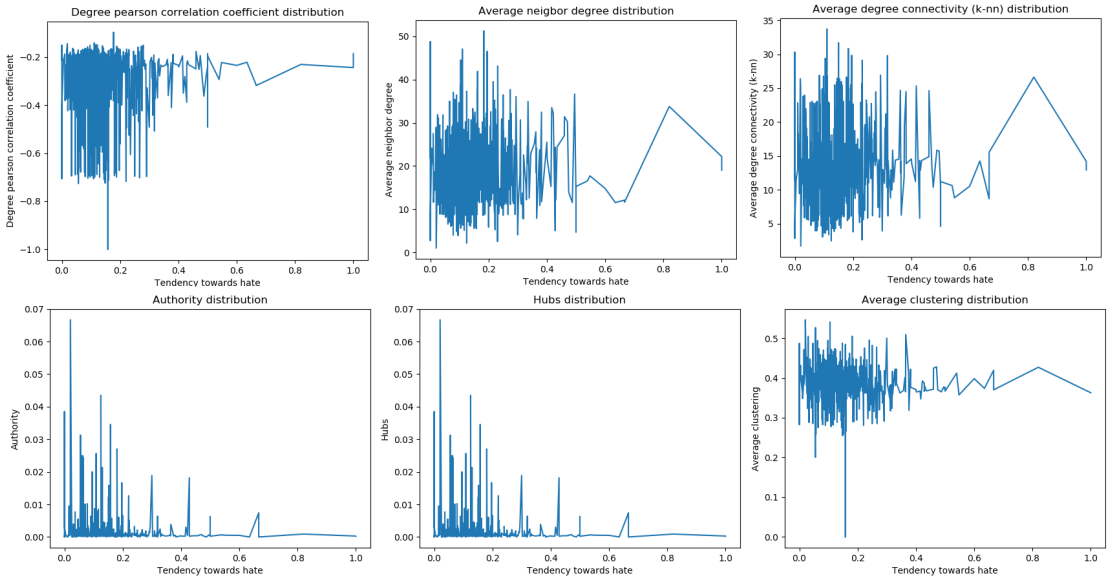
4.2.6.1 Limitations

Although all user profiling features improved the baseline performance, they also had some limitations, that eventually inhibited their full potential:

- Gender information may not be exactly precise, since identifying the gender of users is far from being a trivial task. Our identification approach consisted of 3 different steps (see Figure 4.3) and, after manually checking a portion of the gender predictions made, we still had to correct a good number of examples. This is doable for a small set of users, such as ours, but, for tasks with higher complexity, i.e. with a considerably big number of users, the effectiveness of the gender identification is arguable. Besides, some profiles aren't possible to label, even manually, e.g. profiles with few tweets, no profile picture and uncommon user names or protected users, not accessible to the public eye.

- Assessing a user's tendency towards a certain typical behavior isn't a trivial task either. This is mostly done through the analysis of their tweets history and whether they may be considered hateful or not. This classification is done using a pre-trained model which is likely to be inaccurate or at the very least not fully accurate. Besides, some users don't have

Figure 4.7: Variation of homophily scores according to users' tendency score on top. From left to right, *degree pearson correlation coefficient*, *average neighbor degree* and *average degree connectivity*. On the bottom, from left to right, variation of the authority, *hubs* and *average clustering coefficient* according to users' tendency score. For these, it is perceptible that, as users' tendency score increases, the measures drop.



enough content posted on their accounts to fully cover what their typical behaviors might be. Protected users can't also have their history tracked.

- The main limitation regarding ego networks is accessing the friends' and followers' lists of protected users. Although we guaranteed that every user in the data had a public profile, some of their friends and followers were private. Out of the, approximately, 42,000 users included in the networks, nearly 3,000 had protected accounts, for which we couldn't model potential interactions with others. This is a small number compared to the total of users, but may influence final results and is likely to escalate for more data.

Although using user profiling techniques are mostly beneficial to improve the detection of hate speech, extracting these features is not linear. The main limitation is that most datasets don't provide the ID's of the users who posted the tweets listed, as shown on figure 2.4. Furthermore, the ones that do, often have a part of their users banned as a consequence of posting offensive comments, which is a portion of what each dataset addresses - hate. Furthermore, the retrieval of Twitter content is also limited - by Twitter itself. It forces limited access rates (as most public API's), complicating mostly the creation of ego networks (and other kind of networks).

Figure 4.8: Distribution of the network connectivity type according to the users' tendency score. Connected networks on right and weakly connected networks on left. As users' tendency score increases, networks tend to be unconnected.

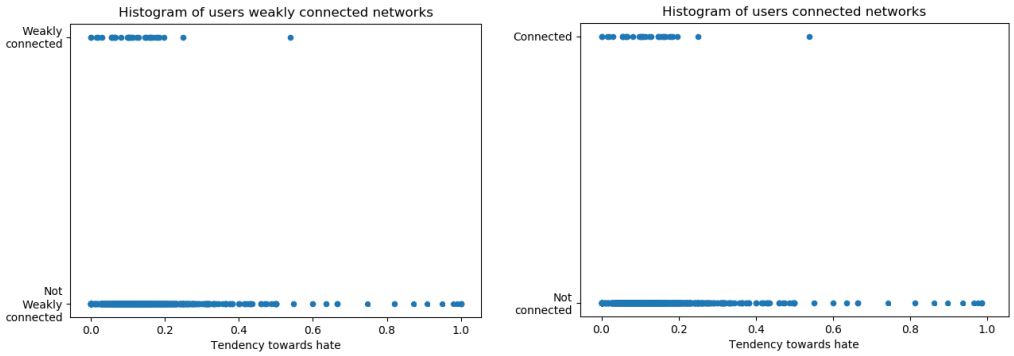


Table 4.8: Results of individual and combined user profiling features.

	Features	**F-score (%)**	**Precision (%) (Hate class)**	**Recall (%) (Hate class)**
	Baseline	67.913	72.012	64.101
Individual features	GI	68.799	73.412	65.252
	UH	69.011	73.274	65.731
	UA	68.211	74.384	64.600
	UN	70.322	71.081	68.342
Combined features	GI + UN	71.448	72.333	68.412
	UH + UN	72.321	73.844	69.133
	GI + UH + UN	73.634	73.451	71.002
	Improvement (GI + UH + UN)	**5.7%**	**1.4%**	**7%**

Chapter 5

Conclusions and Future work

In this chapter we provide an overview of our goals for this thesis, how we accomplished them and the results obtained. Besides, we also address limitations concerning the detection of hate speech and possible improvements to enrich research in this area.

5.1 Goals of our work

Our main goal for this thesis was to improve the field of automatic hate speech detection in text, with a particular focus on online social networks, specially Twitter. This primary objective branched out into 2 different sub-goals, being our first one the exploration of the field's state of the art, by conducting a systematic literature review. Our first task consisted on fully understanding what hate speech is, common targets and what are it's consequences. Only then, one is able to successfully identify and detect it.

We concluded that hate speech is an old topic, dating back to at least 1 century ago, that has been growing significantly over the last couple of decades, with an heavy affluence of people using social networks, and online means in general, to communicate and interact with each other. Several sources, including social networks and other entities, proposed a number of definitions which weren't always convergent or, at least, missed some points, mainly due to the ambiguity of the topic. We adopted the definition proposed in [Fortuna (2017)], which successfully wrapped up all the possible subtleties of hate.

The field of automatic hate speech detection in text was poorly addressed up until mid-2017 and 2018, and the existing approaches were not so popular, with the number of citations averaging around 5. Furthermore, most authors would collect their own data and classify the content collected, without actually making it publicly available. As any machine learning problem, data availability is crucial to be able to enrich research by comparing results and approaches. Mostly in 2018, this topic went through quite an expressive boom with the growing awareness from social entities and people in general. The number of available papers and articles increased by 5 times (in 2018), with most approaches focusing on Twitter and the English language. The amount of datasets available also increased largely, creating a pool of at least 9 different collections of data,

targeting, not only English, but also other languages such as German, Spanish, Hindi, etc. Part of the literature used basic natural language processing techniques to clean and extract features from the data (e.g. sentiment analysis, semantic features), whereas most recent ones focused on deep learning approaches, using pre-trained embeddings such as GloVe and Word2Vec.

Conducting an exhaustive review on the topic lead us to formulate a second goal for our thesis, which we split into 3 other sub-goals. The first one consisted of creating a tokenizer for tweets. Any text classification problem requires the text constituents to be split. The guidelines to separate sentences into their tokens are quite vast and vary according to the implementation. We found that tweets in particular, mostly due to their short length and nature (informal text), required a special kind of tokenization that would consider hashtags, mentions, hidden profanity (e.g. sh#t) and other subtleties. The most common Python tokenizers couldn't offer such a personalized tokenization, hence we developed our own tweet tokenizer, made available online as a Python package, which outperformed others in some parameters. We then used this tokenizer in our experiments.

Our second sub-goal consisted of extracting and selecting the text-based features that would improve the detection of hate speech in text, with special attention to tweets. Thus, we gathered a collection of the most commonly extracted features used in the literature and grouped them according to their category. Our approach started off with the data cleaning and preprocessing techniques. We identified a set of mandatory tweets' cleaning methods, such as the removal of URL's, mentions and HTML codes that we later considered in the semantic analysis. Secondly, we tested a set of tweet preprocessing features and observed that lowercasing, reducing words to their root form (stemming and lemmatization), the removal of stopwords and decomposing hashtags generally contributes to the improvement of the classification task. We also extracted a set of both sentiment and semantic features and observed that sentiment isn't a powerful indicator of hate. On the other hand, analyzing the tweets' semantics highly contributed to the enhancement of results. Words and tweet-based features were the ones that contributed the most to improving the detection of hate speech in text, such as the number of all capitalized words, letters, number of mentions and URL's, among others. We also tested different word vectorization techniques and encodings, and observed that character n-grams generally produce better results than word n-grams, probably due to the short length of tweets and often occurrence of typos, abbreviations and other hard-to-identify words. TF encoding also produced better results than TFIDF.

As most of the literature (around 90%) addressed text-based features to detect hate speech, we noticed a lack of approaches targeting the users themselves. In order to fill this gap, we proposed, as our final goal, to identify and compare different user profiling features. We started off by identifying the gender of the users who posted the original tweets of our data, and progressed to also identify the gender of the mentioned ones. Creating a relationship between the authors' and mentioned users' genders improved the results of our baseline. Our second user profiling feature consisted of a user (hate) tendency score, which was computed based on users' Twitter history. For each author, we labeled the tweets posted using a third party pre-trained model and computed a score ranging from 0 to 1, where 1 was considered to be *hateful* (all tweets in his history were labeled as *hateful*). As a third feature, we also considered settings and characteristics

of each Twitter account, such as the number of friends and followers, presence of profile picture, etc. Finally, we generated an ego network for each user, contemplating his connections to friends and followers but also the interactions between them. We generated a set of statistics, such as centrality, connectivity and homophily and obtained significant improvements over the baseline (around 10% on both f score and *hate* class recall for all user profiling features combined).

After conducting all the experiments we were able to conclude that both text and user-based features must be considered in order to improve the overall detection of hate speech in text. On the other hand, we reckon that profiling users isn't such a trivial task. Twitter poses a set of limitations, such as users with private accounts, from which we can't extract relevant information, coupled with rate limits regarding the API. The generation of ego networks is highly dependant on both of these limitations, which inhibits the full potential of what we consider and proved to be the most valuable feature in hate speech detection.

5.2 Future work

Hate speech detection is definitely a research field with a lot of progress to make. We believe that the starting point is to uniformize, globally, the definition of hate, since a model won't be able to generalize something we, humans, aren't fully aware of. Providing stricter rules and more uniform guidelines might be the step forward to homogenize the concept, which is mostly unclear and diverse from country to country.

Regarding the feature extraction and selection, we think that common text-based features are over explored. These have proven to obtain decent results, but should definitely be combined with user profiling features. As mentioned before, tweets are often short, ambiguous and often contain typos and abbreviations which sometimes makes it hard to extract relevant patterns from tweets. For this reason, we think user profiling techniques should be explored thorough, with special attention to network-based features which have obtained the best results out of the features tested.

Lastly, deep learning approaches have demonstrated that results can be significantly improved, hence their usage may be a step forward in hate speech detection. Since these focus solely on text, combining them with user profiling features may largely improve results.

References

European comission code of conduct. https://ec.europa.eu/newsroom/just/document.cfm?doc_id=42985. Accessed: 2018-12-23.

Portuguese constitution. http://bdjur.almedina.net/citem.php?field=item_id&value=1172842. Accessed: 2018-07-05.

Facebook hate speech definition. https://www.facebook.com/communitystandards/hate_speech. Accessed: 2018-10-12.

Ilga - international lesbian, gay, bisexual, trans and intersex association. https://ilga.org. Accessed: 2018-12-15.

Twitter hate speech definition. https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy. Accessed: 2018-07-05.

Swati Agarwal and Ashish Sureka. Characterizing linguistic attributes for automatic classification of intent based racist/radicalized posts on tumblr micro-blogging website. *Computng Research Repository*, abs/1701.04931, 2017.

Sweta Agrawal and Amit Awekar. Deep learning for detecting cyberbullying across multiple social media platforms. In *European Conference on Information Retrieval 2018*, pages 141–153. Springer, 2018.

Resham Ahluwalia, Evgeniia Shcherbinina, Edward Callow, Anderson Nascimento, and Martine De Cock. Detecting misogynous tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain*, volume 4, 2018.

Antonios Anagnostou, Ioannis Mollas, and Grigorios Tsoumakas. Hatebusters: A web application for actively reporting youtube hate speech. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5796–5798. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/841.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems 2018*, pages 57–64. Springer, 2018.

Segun Taofeek Aroyehun and Alexander Gelbukh. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 90–97. Association for Computational Linguistics, 2018.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. *Computing Research Repository*, abs/1802.09913, 2018.

Xiaoyu Bai, Flavio Merenda, Claudia Zaghi, Tommaso Caselli, and Malvina Nissim. Rug at germeval: Detecting offensive speech in german social media. In *Proceedings of the GermEval Workshop 2018*, 2018.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160, 2014.

Alain Barrat, Marc Barthelemy, Romualdo Pastor-Satorras, and Alessandro Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101:3747–52, 04 2004. doi: 10.1073/pnas.0400087101.

Kristin P. Bennett and Erin J. Bredensteiner. Duality and geometry in svm classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 57–64, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2.

Adam Bermingham, Maura Conway, Lisa McInerney, Neil O'Hare, and Alan Smeaton. Combining social network analysis and sentiment analysis to explore the potential for online radicalisation. *In Advances in Social Networks Analysis and Mining, 20-22 July, 2009, Athens, Greece.*, 07 2009. doi: 10.1109/ASONAM.2009.31.

Shanita Biere and Sandjai Bhulai. *Hate Speech Detection Using Natural Language Processing Techniques*. Master's dissertation, Vrije Universiteit Amsterdam, 2018.

Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN 9780387310732.

Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. A dataset of hindi-english code-mixed social media text for hate speech detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media*, pages 36–41. Association for Computational Linguistics, 2018.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

Sara Bullard. *The Ku Klux Klan: A History of Racism & Violence*. Diane Publishing, 1998.

Olivier Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5): 1155–1178, 2007.

Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Athena Vakali. Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 ACM on web science conference*, pages 13–22. ACM, 2017.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *Computing Research Repository*, abs/1412.3555, 2014.

Maral Dadvar, de FMG Jong, Roeland Ordelman, and Dolf Trieschnigg. Improved cyberbullying detection using gender information. In *Proceedings of the Twelfth Dutch-Belgian Information Retrieval Workshop (DIR 2012)*. University of Ghent, 2012.

Ona de Gibert, Naiara Pérez, Aitor García Pablos, and Montse Cuadros. Hate speech dataset from a white supremacy forum. *Computing Research Repository*, abs/1809.04444, 2018.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository*, abs/1810.04805, 2018.

David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

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. *Computing Research Repository*, abs/1804.04257, 2018a.

Mai ElSherief, Shirin Nilizadeh, Dana Nguyen, Giovanni Vigna, and Elizabeth M. Belding. Peer to peer hate: Hate speech instigators and their targets. *Computing Research Repository*, abs/1804.04649, 2018b.

ExplosionAI. Spacy tokenizer, 2015. URL https://spacy.io/api/tokenizer. Online; accessed 23-11-2018.

Paula Fortuna. *Automatic detection of hate speech in text: an overview of the topic and dataset annotation with hierarchical classes*. Master's dissertation, Faculdade de Engenharia da Universidade do Porto, 2017.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. Merging datasets for aggressive text identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 128–139, 2018.

Jacob G Foster, David V Foster, Peter Grassberger, and Maya Paczuski. Edge direction and the structure of networks. *Proceedings of the National Academy of Sciences 2010*, 107(24):10815–10820, 2010.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. A unified deep learning architecture for abuse detection. *Computing Research Repository*, abs/1802.00385, 2018.

Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

Simona Frenda and Banerjee Somnath. Deep analysis in aggressive mexican tweets. In *Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, volume 2150, pages 108–113. Ceur Workshop Proceedings, 2018.

Simona Frenda, Ghanem Bilal, et al. Exploration of misogyny in spanish and english tweets. In *Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, volume 2150, pages 260–267. Ceur Workshop Proceedings, 2018.

Ben Friedland. Profanity dictionary, 2013. URL https://pypi.org/project/profanity/. Online; accessed 18-12-2018.

Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, 2017.

Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. Detecting hate speech and offensive language on twitter using machine learning: An n-gram and TFIDF based approach. *Computing Research Repository*, abs/1809.08651, 2018.

CJ Hutto Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, 2014.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *Computing Research Repository*, abs/1402.3722, 2014.

Viktor Golem, Mladen Karan, and Jan Šnajder. Combining shallow and deep learning for aggressive text detection. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 188–198, 2018.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. All you need is "love": Evading hate-speech detection. *Computing Research Repository*, abs/1808.09115, 2018.

Frank Harrell. Damage caused by classification accuracy and other discontinuous improper accuracy scoring rules, Statistical Thinking, 2017. URL http://www.fharrell.com/post/class-damage/. Online; accessed 20-12-2018.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

Muhammad Okky Ibrohim and Indra Budi. A dataset and preliminaries study for abusive language detection in indonesian social media. *Procedia Computer Science*, 135:222–229, 2018.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Fang Jin, Edward Dougherty, Parang Saraf, Yang Cao, and Naren Ramakrishnan. Epidemiological modeling of news and rumors on twitter. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, page 8. ACM, 2013.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

Timotheus Kampik. Pypi: compound word splitter, 2017. URL https://pypi.org/project/compound-word-splitter/. Online; accessed 18-10-2018.

Raghav Kapoor, Yaman Kumar, Kshitij Rajput, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. Mind your language: Abuse and offense detection for code-switched languages. *Computing Research Repository*, abs/1809.08652, 2018.

Gurneet Kaur and Er Neelam Oberai. A review article on naive bayes classifier with various smoothing techniques. *International Journal of Computer Science and Mobile Computing*, 3 (10):864–868, 2014.

Taehoon Kim and Kevin Wurster. Pypi: emoji, 2017. URL https://pypi.org/project/emoji/. Online; accessed 12-11-2018.

Jon M Kleinberg. Hubs, authorities, and communities. *ACM computing surveys (CSUR)*, 31(4es): 5, 1999.

Filip Klubicka and Raquel Fernández. Examining a hate speech corpus for hate speech detection and popularity prediction. *Computing Research Repository*, abs/1805.04661, 2018.

Sebastian Köffer, Dennis M Riehle, Steffen Höhenberger, and Jörg Becker. Discussing the value of automatic hate speech detection in online debates. *Multikonferenz Wirtschaftsinformatik (MKWI 2018): Data Driven X-Turning Data in Value, Leuphana, Germany*, 2018.

Ritesh Kumar, Guggilla Bhanodai, Rajendra Pamula, and Maheshwar Reddy Chennuru. Trac-1 shared task on aggression identification: Iit (ism) @ coling'18. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 58–65, 2018.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. Comparative studies of detecting abusive language on twitter. *Computing Research Repository*, abs/1808.10245, 2018.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. Learning natural language inference using bidirectional LSTM model and inner-attention. *Computing Research Repository*, abs/1605.09090, 2016.

Steven Loria. Textblob, 2013. URL https://textblob.readthedocs.io/en/dev/. Online; accessed 15-11-2018.

Promita Maitra and Ritesh Sarkhel. A k-competitive autoencoder for aggression detection in social media text. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 80–89. Association for Computational Linguistics, 2018.

Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26, 2018.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098. Association for Computational Linguistics, 2018a.

Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. Neural character-based composition models for abuse detection. *Computing Research Repository*, abs/1809.00378, 2018b.

Sandip Modha, Prasenjit Majumder, and Thomas Mandl. Filtering aggression from the multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 199–207. Association for Computational Linguistics, 2018.

Arjun Mukherjee and Bing Liu. Improving gender classification of blog authors. In *Proceedings of the 2010 conference on Empirical Methods in natural Language Processing*, pages 207–217. Association for Computational Linguistics, 2010.

Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

Nishant Nikhil, Ramit Pahwa, Mehul Kumar Nirala, and Rohan Khilnani. Lstms with attention for aggression detection. *Computing Research Repository*, abs/1807.06151, 2018.

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.

Constantin Orasan. Aggressive language identification using word embeddings and sentiment features. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 113–119. Association for Computational Linguistics, 2018.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, and Viviana Patti. 14-exlab@ unito for ami at ibereval2018: Exploiting lexical knowledge for detecting misogyny in english and spanish tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain*, 2018.

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.

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.

Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. Detecting offensive language in tweets using deep learning. *Computing Research Repository*, abs/1801.04433, 2018.

David Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation. *Maching Learning Technologies*, 2, 01 2008.

Jing Qian, Mai ElSherief, Elizabeth Belding, and William Yang Wang. Leveraging intra-user and inter-user representation learning for automated hate speech detection. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 118–123. Association for Computational Linguistics, 2018a.

Jing Qian, Mai ElSherief, Elizabeth M. Belding, and William Yang Wang. Hierarchical CVAE for fine-grained hate speech classification. *Computing Research Repository*, abs/1809.00088, 2018b.

Kashyap Raiyani, Teresa Gonçalves, Paulo Quaresma, and Vitor Beires Nogueira. Fully connected neural network with advance preprocessor to identify aggression over facebook and twitter. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 28–41, 2018.

Manoel Horta Ribeiro, Pedro H. Calais, Yuri A. Santos, Virgílio A. F. Almeida, and Wagner Meira Jr. Characterizing and detecting hateful users on Twitter. *Computing Research Repository*, abs/1803.08977, 2018.

Julian Risch and Ralf Krestel. Aggression identification using deep learning and data augmentation. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 150–158. Association for Computational Linguistics, 2018.

Julian Risch, Eva Krebs, Alexander Löser, Alexander Riese, and Ralf Krestel. Fine-grained classification of offensive language. In *Proceedings of the GermEval Workshop 2018*, 2018.

David Robinson, Ziqi Zhang, and Jonathan Tepper. Hate speech detection on Twitter: Feature engineering vs feature selection. In *15th European Semantic Web Conference 2018*, pages 46–49. Springer, 2018.

Kristian Rother and Achim Rettberg. Ulmfit at germeval-2018: A deep neural language model for the classification of hate speech in german tweets. In *Proceedings of the GermEval Workshop 2018*, 09 2018.

Arjun Roy, Prashant Kapil, Kingshuk Basak, and Asif Ekbal. An ensemble approach for aggression identification in english and hindi text. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 66–73, 2018.

Kshitiz Sahay, Harsimran Singh Khaira, Prince Kukreja, and Nishchay Shukla. Detecting cyberbullying and aggression in social commentary using nlp and machine learning. *International Journal of Engineering Technology Science and Research*, 2018.

Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. *Ninth International Conference on Language Resources and Evaluation*, 2014.

Joni Salminen, Hind Almerekhi, Milica Milenkovic, Soon-Gyo Jung, Jisun An, Haewoon Kwak, and Bernard J. Jansen. Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media. In *The International Conference on Weblogs and Social Media 2018*, 2018.

Niloofar Safi Samghabadi, Deepthi Mave, Sudipta Kar, and Thamar Solorio. Ritual-uh at trac 2018 shared task: Aggression identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, 2018.

Yutaka Sasaki. The truth of the f-measure. *Teaching, Tutorial materials*, Version: 26th October, 2007.

Johannes Schäfer. Hiiwistjs at germeval-2018: Integrating linguistic features in a neural network for the identification of offensive language in microposts. In *Proceedings of the GermEval Workshop 2018*, 2018.

Tatjana Scheffler, Erik Haegert, Santichai Pornavalai, and Mino Lee Sasse. Feature explorations for hate speech classification. In *Proceedings of the GermEval Workshop 2018*, volume 6, page 8, 2018.

Julian Moreno Schneider, Roland Roller, Peter Bourgonje, Stefanie Hegele, and Georg Rehm. Towards the automatic classification of offensive language and related phenomena in german tweets. In *Proceedings of the GermEval Workshop 2018*, 2018.

Hitesh Kumar Sharma and Shailendra Kshitiz. Nlp and machine learning techniques for detecting insulting comments on social networking platforms. In *2018 International Conference on*

Advances in Computing and Communication Engineering (ICACCE), pages 265–272. IEEE, 2018.

Sanjana Sharma, Saksham Agrawal, and Manish Shrivastava. Degree based classification of harmful speech using twitter data. *Computing Research Repository*, abs/1806.04197, 2018.

Koo Ping Shung. Accuracy, precision, recall of f1?, Towards Data Science, March, 2018. URL https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. Online; accessed 10-12-2018.

Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabricio Benevenuto, and Ingmar Weber. Analyzing the targets of hate in online social media. *Computing Research Repository*, abs/1603.07709, 2016a.

Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. Analyzing the targets of hate in online social media. In *The International Conference on Weblogs and Social Media 2016*, pages 687–690, 2016b.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Sattar A., Kang B. (eds) AI 2006: Advances in Artificial Intelligence. AI 2006*, pages 1015–1021. Springer, 2006.

Sandro Sperandei. Understanding logistic regression analysis. *Biochemia medica*, 24(1):12–18, 2014.

Dominik Stammbach, Azin Zahraei, Polina Stadnikova, and Dietrich Klakow. Offensive language detection with neural networks for germeval task 2018. In *Proceedings of the GermEval Workshop 2018*, 2018.

Edward Loper Steven Bird. Nltk: stopwords, 2001a. URL http://www.nltk.org/_modules/nltk/corpus.html. Online; accessed 14-10-2018.

Edward Loper Steven Bird. Nltk: tokenizer, 2001b. URL https://www.nltk.org/api/nltk.tokenize.html. Online; accessed 10-10-2018.

Liling Tan. Expletives dictionary, 2017. URL https://pypi.org/project/expletives/. Online; accessed 18-12-2018.

Bin Tang, Michael Shepherd, Evangelos Milios, and Malcolm I Heywood. Comparing and combining dimension reduction techniques for efficient text clustering. In *Proceeding of SIAM international workshop on feature selection for data mining*, pages 17–26. Citeseer, 2005.

Alexandru Topîrceanu and Gabriela Grosseck. Decision tree learning used for the classification of student archetypes in online courses. *Procedia Computer Science*, 112:51–60, 2017.

Elise Fehn Unsvåg. Investigating the effects of user features in hate speech detection on Twitter. Master's thesis, Norwegian University of Science and Technology, 2018.

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. Challenges for toxic comment classification: An in-depth error analysis. *Computing Research Repository*, abs/1809.07572, 2018.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. Detection and fine-grained classification of cyberbullying events. In *International Conference Recent Advances in Natural Language Processing (RANLP) 2015*, pages 672–680, 2015.

Cornelis J Van Rijsbergen, Stephen Edward Robertson, and Martin F Porter. *New models in probabilistic information retrieval*. British Library Research and Development Department London, 1980.

Ben Verhoeven, Walter Daelemans, and Barbara Plank. Twisty: a multilingual twitter stylometry corpus for gender and personality profiling. In *Proceedings of the 10th Annual Conference on Language Resources and Evaluation (LREC 2016)*, pages 1–6, 2016.

Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *15th International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.

Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. Efficient hyperparameter optimization for nlp applications. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2112–2117, 2015.

Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the North American Chapter of the Association for Computational Linguistics student research workshop*, pages 88–93, 2016.

Hajime Watanabe, Mondher Bouazizi, and Tomoaki Ohtsuki. Hate speech on twitter a pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*, 2 2018. ISSN 2169-3536.

Gregor Wiedemann, Eugen Ruppert, Raghav Jindal, and Chris Biemann. Transfer learning from lda to bilstm-cnn for offensive language detection in twitter. In *Proceedings of the GermEval Workshop 2018*, 2018.

Dong Yao, Pim van der Hoorn, and Nelly Litvak. Average nearest neighbor degrees in scale-free networks. *Internet Mathematics*, 2018, 04 2017.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of CNN and RNN for natural language processing. *Computing Research Repository*, abs/1702.01923, 2017.

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

Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *15th European Semantic Web Conference 2018*, pages 745–760. Springer, 2018.

Steven Zimmerman, Udo Kruschwitz, and Chris Fox. Improving hate speech detection with deep learning ensembles. In *Language Resources and Evaluation Conference 2018*, 2018.