NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

MSc THESIS

Hate Speech Detection using different text representations in online user comments

Chrysoula K. Themeli

Supervisor: **Stamatopoulos Panagiotis**, Assistant Professor, UOA

ATHENS

October 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός ρητορικής μίσους σε σχόλια χρηστών στο διαδίκτυο με χρήση διαφορετικών αναπαραστάσεων

Χρυσούλα Κ. Θεμελή

Επιβλέπων: **Σταματόπουλος Παναγιώτης**, Επίκουρος Καθηγητής, ΕΚΠΑ

ΑΘΗΝΑ

Οκτώβριος 2018

MSc THESIS

Hate Speech Detection using different text representations in online user comments

Chrysoula K. Themeli
S.N.: 1423

SUPERVISOR: **Stamatopoulos Panagiotis**, Assistant Professor, UOA

EXAMINATION COMMITTEE: **Stamatopoulos Panagiotis**, Assistant Professor, UOA
Giannakopoulos George, Postdoc Researcher, NCSR-D

Examination Date: October 02 2018

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός ρητορικής μίσους σε σχόλια χρηστών στο διαδίκτυο με χρήση διαφορετικών αναπαραστάσεων

Χρυσούλα Κ. Θεμελή
Α.Μ.: 1423

ΕΠΙΒΛΕΠΩΝ: **Σταματόπουλος Παναγιώτης**, Επίκουρος Καθηγητής, ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:
Σταματόπουλος Παναγιώτης, Επίκουρος Καθηγητής, ΕΚΠΑ
Γιαννακόπουλος Γεώργιος, Μεταδιδ. Ερευνητής, ΕΚΕΦΕ-Δ

Ημερομηνία Εξέτασης: 02 Οκτωβρίου 2018

ABSTRACT

Hate Speech is abusive or stereotyping speech against a group of people, based on characteristics such as race, religion, sexual orientation and gender. It is illegal based on the current legislation in the USA and the EU, however the Internet and social media made it possible to spread hatred easily, fast and anonymously. The large scale of data produced through social media platforms requires the development of an effective automatic model to detect such content. We study the performance of several text representation techniques and classification algorithms, aiming to efficiently handle the online abusive language discrimination task. We examine various representation techniques such as Bag of Words (BoW), word and character Bag of n-grams, sentiment, syntax and grammar analysis, word embeddings and n-gram graphs. In addition, we test multiple classification algorithms: Naive Bayes, Logistic Regression, Random Forests, K-Nearest Neighbors and Aritificial Neural Networks. Our goal is to evaluate representation and classification algorithms with respect to their contribution to performance in the Hate Speech detection task. Moreover, we highlight the utility of n-gram graphs (NGGs) as an efficient, low-dimensional text representation that constructs similarity vectors which appear to constitute deep features with significant contribution to the classification results. Apart from the binary classification experiments, we additionally test our method in multi-class classification experiments on abusive language discrimination tasks. Our results showe that NGGs are informative and rich features - despite being represented by vectors with dimensions equal to the number of possible classes - performing slightly worse than the Bag of Words and word embeddings, which are in contrast constitute by high-dimensional representations. We furthermore execute statistical tests, to examine whether NGGs have significant contribution to the results. The tests not only showe that NGGs are significant features with respect to the classification result, but also that the combination of the three best performing features (BoW, NGGs and word embeddings) achieves the best classification performance, with the use of the remaining text representations yielding deteriorated results. Finally, the classification algorithm selection seems to be less important, since statistical results for all the tested algorithms are similar.

SUBJECT AREA: Natural Language Processing

KEYWORDS: Hate Speech, classification, Natural Language Processing

ΠΕΡΙΛΗΨΗ

Η ρητορική μίσους αφορά την διατύπωση προσβολών, απειλών ή στερεοτυπικών απόψεων απέναντι σε μια ομάδα ανθρώπων εξαιτίας κάποιου χαρακτηριστικού όπως η καταγωγή, το φύλο, η θρησκεία, οι σεξουαλικές προτιμήσεις κλπ. Τέτοιου είδους επιθέσεις είναι εκτός νόμου σε όλες τις σύγχονες και ανεπτυγμένες κοινωνίες, πχ ΗΠΑ, ΕΕ. Παρόλα αυτά το Διαδίκτυο και ιδιαίτερα οι πλατφόρμες κοινωνικής δικτύωσης δίνουν τη δυνατότητα διάδοσης τέτοιου είδους περιεχομένου εύκολα, γρήγορα και ανώνυμα. Έτσι, σε συνδυασμό με τη σημερινή οικονομική κρίση που ευνοεί την ανάπτυξη τέτοιων απόψεων, παρατηρούμε μια έξαρση του φαινομένου που δίνει τη δυνατότητα ο λόγος αυτός να φτάσει ένα πολύ μεγαλύτερο αριθμό ανθρώπων απ' ότι στο παρελθόν. Ο τεράστιος αριθμός των δεδομένων που παράγονται στις παραπάνω πλατφόρμες καθιστά αδύνατο τον εντοπισμό αναρτήσεων από κάποιον διαχειριστή σελίδας ή από αναφορές χρηστών, κάνοντας αναγκαία τη χρήση αυτόματων εργαλείων εντοπισμού ρητορικής μίσους. Στα πλαίσια αυτής της εφαρμογής, στην παρούσα εργασία μελετάμε πολλαπλές τεχνικές αναπαραστάσεων κειμένου (Bag of Words, Bag of word/character n-grams, sentiment, syntax and grammar analysis features, word embeddings και n-gram graphs), καθώς και πληθώρα ενώ οι αλγορίθμων ταξινόμησης (Naive Bayes, Logistic Regression, Random Forests, K-Nearest Neighbors και Artificial Neural Networks). Υλοποιήσαμε πειράματα τόσο για δυαδική ταξινόμηση, όπου ο σκοπός του μοντέλου είναι η απόφαση εάν το κείμενο εισόδου περιέχει ρητορική μίσους ή όχι, όσο και για κατηγοριοποίηση πολλαπλών κλάσεων, όπου ο ταξινομητής προσπαθεί να διαχωρίσει μεταξύ διαφορετικών ειδών ρητορικής μίσους (π.χ. σεξισμός, ρατσισμός, κ.α.). Στόχος μας είναι να εξετάσουμε την απόδοση της κάθε τεχνικής αναπαράστασης και ταξινόμησης και να αναδείξουμε τις μεθόδους με την καλύτερη απόδοση.

Επιπλέον, εξετάσαμε κατά πόσο οι συνδυασμοί διαφόρων τεχνικών αναπαράστασης κειμένων επιτυγχάνουν καλύτερα αποτελέσματα από τη μεμονωμένη χρήση τους. Τέλος, δείχνουμε ότι η χρήση των n-gram graphs, που αναπαρίστανται από ένα διάνυσμα μικρών διαστάσεων, μπορεί να συμβάλει σημαντικά στον εντοπισμό της ρητορικής μίσους. Τα πειράματα έδειξαν ότι τα πιο αποδοτικά features είναι τα BoW, word embeddings με τους NGGs να ακολουθούν με ελαφρώς χειρότερη απόδοση. Επιπρόσθετα, ο συνδυασμός των προαναφερθέντων μεθόδων αναπαράστασης έχει την καλύτερη απόδοση σε σύγκριση με όλα τα υπόλοιπα features, είτε αυτά εξετάστηκαν μεμονωμένα είτε σε συνδυασμό με άλλα features. Τέλος, οι αλγόριθμοι ταξινόμησης φαίνεται να μην έχουν ιδιαίτερη στατιστική σημασία, μιας και τα αποτελέσματα στα στατιστικα τεστ είναι παρόμοια για όλους τους αλγορίθμους που χρησιμοποιήσαμε.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επεξεργασία φυσικής γλώσσας

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ρητορική μίσους, ταξινόμηση, επεξεργασία φυσικής γλώσσας

ACKNOWLEDGEMENTS

I would like to thank my supervisors Assistant Professor Panagiotis Stamatopoulos and Dr. George Giannakopoulos for their consistent guidance and support, for always being available to answer my questions and advise me during the implementation of my thesis. I also want to thank Nikiforos Pittaras and Nikoletta Bika for helping me improve my writing skills through their advice and feedback. And of course, I would like to thank my family, especially my parents, for all the love and support they offered me all those years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This document is an investigation of natural language processing methods, that attempt to efficiently tackle the Hate Speech detection task in online user comments. For example, detect tweets with problematic content and remove them. The purpose of an automatic Hate Speech detection system is to be able to correctly produce such output, given a user's comment as input, with no human intervention after its construction. In this study, a set of approaches are investigated that build such model by taking advantage of multiple different features (text representations) such as sentiment analysis to detect negative sentiments - hate is usually associated with negative polarity, graph representations to keep words position in the text as information and relevant lexicons with hate keywords to transfer knowledge of this task in the used features. This software is mainly implemented in Java using Weka framework to transform the user texts into a form that is understandable by a computer and also run some experiments using multiple classifiers, such as Naive Bayes, Logistic Regression, K-Nearest Neighbors and Random Forests. Additionally, we have also implemented the software in Python, using scikit-learn and keras frameworks. In Python, we have implemented the aforementioned classifiers using scikit-learn, where we also used MLP classifier. Finally, related to Keras implementation, we have implemented a Neural Network with 5 hidden layers. All experiments are implemented in machines with 16GB RAM due to the time and space complexity in the features' creation and processing.

Our model was trained through supervised learning tasks, both binary and multi-class classification. We have used 10-fold cross-validation to our dataset and the results are in F_{micro} and F_{macro} metrics, calculating the average results of a specific feature or classifier. In addition to these experiments, we have evaluated our results through statistical tests, to determine the statistical significance of all the features and classifiers used. We reviewed several related papers on Hate Speech detection task in Social Media platforms and studied the proposed models, the text representations used and their performance. We examined their findings and we used two already public dataset with Tweets in English to also test our model. This thesis was implemented towards the fulfillment of the graduation requirements for the Information and Communications Technologies postgraduate program of the Department of Informatics and Telecommunications, of the University of Athens in Greece.

1. INTRODUCTION

The advance of the technology during the recent years affected modern life and society globally. The access to education, medicine, industry, transportation etc. has been simplified due to modern day technology. Due to the convenience and efficiency provided by technology, our lives have improved significantly. Communications are also affected from the advance of the technology. Nowadays, social media and social networking seem to play an inseparable part of peoples lives, making communication easier than ever before. According to a research in 2010 by [5], only in USA, 35% of adults keep an online profile, while this percentage was only 8% in 2005. Teenager users have an even higher percentage, also logging in their profiles in daily basis.

The increase of social media use in all modern societies has dramatically changed the way people interact with each other. Teenagers and adults have the opportunity to communicate with other people from all over the world, share ideas and thoughts very easily using their online profile. Moreover, anonymity make things even easier, since people can express themselves without being tracked. Of course, apart from the great deal of advantages that this evolution brings to people lives, it comes along with some disadvantages as well. We will focus on a specific disadvantage in this work; the spread of Hate Speech through the Internet and social media platforms.

Social media platforms are not to be blamed for this increase, since hatred is observed in many aspects of modern life. However, they give the opportunity to speed Hate Speech easily and anonymously. In an effort to control Hate Speech, modern societies, such as USA and EU, have voted against such kind of public speech making it illegal. As a result, social media platforms focus their efforts in complying with this legislation and effectively eliminate comments that promote hatred. These efforts will not solve the problem of the hatred against minorities, since this role belongs to the education system; however, they can reduce the amount of Hate Speech that people come across when they use the Internet or their online profiles.

This work aims to deal with Hate Speech spread through the Internet and more precisely, social media platforms. As we will discuss in the following sections, Hate Speech is a common affliction in modern society. The massive increase of the use of social media platforms, websites and forums containing user-created content made it possible to easily spread hateful content and reach a number of people larger than ever before, as we have already mentioned. Legislation and policies of social media platforms try to reduce such content by removing comments and accounts that promote hatred against minorities. In order to effectively eliminate Hate Speech from the platform, they should either use humans to detect hate content or automatic tools. Our goal is to implement an automatic tool that detects Hate Speech in Twitter posts. In this chapter, we present in more detail the reasons to deal with this phenomenon (i.e. Hate Speech) in Section 1.1, why hatred seems to increase in modern societies in Subsection 1.1.1 and analyze the role of Internet and social media in Hate Speech spread in Subsection 1.1.2.

1.1 Reasons to deal with Hate Speech

Previously, we described how human lives are affected by the increase of Internet and social media use globally. Today it is possible to interact with people from all over the world that you never met and share ideas, thoughts, opinions etc. More and more people keep online profiles in many social media platforms such as Facebook, Twitter, YouTube, Instagram and more. Therefore, large-scale data are shared every day through social media platforms with enormous speed and reach an incredibly huge number of people.

Social media can be used in many ways: communicate with family and friends, interact and change ideas with numerous people under a post, read the news, play games etc. Also, many people use forums to discuss about several issues anonymously (or even create anonymous accounts in social media such as Facebook and Twitter). Anonymity gives the possibility to write offensive posts or attack other users. Social media platforms need to improve user experience and protect users that belong to minorities to not come across abusive and hate content. For this reason, in many platforms and forums, it is usual to have administrators to detect offensiveness and attacks or check posts reported by users.

Below we discuss two reasons explaining why relying on administrators is not an effective method and automatic tools are required. The first reason is related to the increase of hatred in general in modern societies, by also underlying the increase of violence against minorities. Since the number of people expressing hate against other group of people is increased in our societies, the amount of online posts will also be increased. The second reason is already mentioned and has to do with the massive use of Internet and social media.

1.1.1 Economic Crisis

During the last decade, the most severe economic crisis is taking place globally. In Europe and USA, each year more and more people live below the poverty line due to unemployment and salary decrease combined with taxation increase. While in the advanced societies, the economic crisis is getting worse and worse, in Middle East (e.g. Syria or Palestine) people are facing endless war and death, which leads them to migrate from their countries to seek a better and safer living in EU or USA.

Due to the above situation, EU and USA face a worrying increase of extreme right movements (e.g. civil war in Ukraine, Donald Trump in US, Marie Le Pen in France, Golden Dawn in Greece etc). Apart from the political results of this crisis, there are several researches that shows an increase of violence against minorities such as refugees and immigrants [1], homosexuals [2] etc. Additionally, economic crisis has led to the rise of domestic violence against women (e.g. Renzetti et al. [22], Gavrilova et al. [11]).

[1] http://www.globalization101.org/the-financial-crisis-and-xenophobia-2/

[2] https://www.bbc.com/news/world-europe-22563843

The increase of racism and sexism in the society leads to an increase of the use of such language through social media, even with an extreme form of promoting violence against minorities. Economic stress or poverty can lead more easily to hatred or the belief that minorities are responsible for the current crisis. Although, social media platforms cannot deal with the causes of this phenomenon, they are interested in improving user experience but also to comply with anti Hate Speech legislation. The only possible way to deal with this phenomenon, is to block hatred messages and notify users to compromise with the platform's policies.

1.1.2 Internet and Social Media massive use

In modern societies, during the last decades, an exponential increase of Internet and social media use takes place. Especially, young people follow the news and communicate through those platforms. In this way, news, rumors, opinions etc can be spread almost instantly to all over the world and affect an enormous number of people.

This phenomenon combined with the increase of hate and violence mentioned above, gave the opportunity to spread Hate Speech anonymously and easily through popular social media such as Facebook or Twitter. However, these platforms have specific policies in order to deal with Hate Speech and offensive posts against other members of the platform. As already mentioned, the large amount of data distributed through social media platforms makes it impossible for a human (i.e. administrator) to track problematic and offensive content. One usual solution, although not very reliable, is to review only the reported posts by other platform users. This is also ineffective, since it relies on users' subjectivity and trustworthiness, as well as depending on their ability to thoroughly track and flag such content. For all the above reasons, an automation tool to detect Hate Speech is necessary to all social media platforms which wish to comply with global and EU laws against hatred (anti-racist laws etc) but also to protect their users that belong to a minority group and make their experience in the platform more pleasant.

In this chapter we reviewed the reasons to deal with Hate Speech and the inefficiency of humans in the task of detecting this content. In our work, we aim to develop an automatic tool to detect such kind of content. The process of creating such tool, which the goal of this study, includes the text transformation of user comments into a form that is understandable by a machine and, by extension, a computer program and the training of a classifier with an annotated dataset to effectively classify a post as Hate Speech or non Hate Speech. In chapter 2, we describe the Hate Speech detection task, we provide our definition on what constitutes Hate Speech and finally we list some fundamental terms for better understudying. Moreover, in chapter 3 we review the literature on Hate Speech detection and other similar tasks based on the relevant survey by Schmidt and Wiegand, 2017 [26]. In chapter 4 we provide a detailed description of our model and the techniques we have reviewed, while in chapter 5 we present the results of our experiments and statistical tests, concluding the study in chapter 6 by summarizing the findings and proposing future work.

2. PROBLEM DEFINITION

In Chapter 1, we reviewed some main reasons on why Hate Speech is a critical phenomenon to deal with in modern societies. During the last years where the most severe economic crisis takes place, exacerbating violence against immigrants, refugees, women or LGBTQ community members. Since violence is the most extreme expression of this phenomenon, a general increase of Hate Speech and abusive language usage in social media platforms is also noticed. In the previous chapter, we explained why these platforms need to detect such content and why the human factor in this task is ineffective. All the aforementioned reasons, lead to the demanding need to develop an automatic tool for Hate Speech detection.

Before presenting relevant works on this task, we will provide a formal definition of the automatic Hate Speech detection task in Section 2.1. As will discuss, one important factor is the annotation process. The classifier, which will detect problematic content, needs to be trained and tested on an already annotated dataset. Since the annotation process involves the human factor (experts or crowd-sourcing), it is important to provide a clear and concise definition of what constitutes Hate Speech. Therefore, in Section 2.2 we provide a formal definition by the European Committee of Ministers and we extend it to include two more aspects on Hate Speech. Finally, in Section 2.3 we list some important terms used in this study for better understudying.

2.1 Hate Speech Detection

The goal of a Hate Speech Detection model is, given an input text T, to output `True`, if T contains Hate Speech and `False` otherwise. Modeling the task as a binary classification problem, the detector is built by learning from a supplied training set and is subsequently evaluated on unseen data.

More specifically, the input text is transformed to a machine-readable format via a text representation method, which ideally captures and retains informative characteristics in the input text. The representation data is fed to a machine learning algorithm that assigns the input to one of the two classes, with a certain confidence. During the training phase, this discrimination information is used to construct the classifier. The classifier is then applied on data not encountered during training, in order to measure its generalization ability.

The aforementioned process requires an existing annotated dataset of Twitter post examples in order to be used as input to the classifier. Therefore, the first step is to use human annotators to label all the examples in the dataset. The absence of a formal and widely accepted definition of Hate Speech, allows the annotators to interpret the Tweets content based on their educational and cultural background and therefore conclude in a low inter-annotators agreement.

The authors of Waseem, 2016 [31] are investigating the different results using annotated data by expert and amateur annotators. They have used 6,909 tweets annotated by amateurs in CrowdFlower. As experts, they have recruited feminist and anti-racist activists, which annotated tweets that failed a test and are also given the possibility to skip a tweet or to annotate it as noise. The results show a low percentage of agreement between the two annotators group. If we consider only the data with high level agreement, it is highly possible to obtain good annotation from amateurs. Their system performs worse compare to the one used by [32], fact that is caused by the high number of false positives. What it is surprising about this, is the fact that this number is high even in the dataset annotated by experts.

2.2 Hate Speech Definition

In order to train an automatic model to detect Hate Speech, we need to feed it with annotated positive or negative examples. The annotation process involves humans (experts on the subject or amateurs) that read the collected data and annotate the examples using the provided labels. However, this task hides a significant difficulty. The absence of a formal definition for an issue such as Hate Speech concludes in relying on annotator's cultural background or opinion of what constitutes Hate Speech. According to the European Committee of Ministers (Brown, 2017 [4]), "it covers all forms of expressions that spread, incite, promote or justify racial hatred, xenophobia, antisemitism or other forms of hatred based on intolerance". Moreover, it can be "insulting, degrading, defaming, negatively stereotyping or inciting hatred, discrimination or violence against people in virtue of their race, ethnicity, nationality, religion, sexual orientation, disability, gender identity".

However, we cannot disregard that Hate Speech can be also expressed by statements promoting superiority of one group of people against another, or by expressing stereotypes against a group of people that do not correspond to reality. In their work, the authors of [14] have asked three students of different race and same age and gender to annotate whether a tweet contained Hate Speech or not, as well as the degree of its offensiveness. The agreement was only 33%, showing that Hate Speech detection can be highly subjective and based on the educational and/or cultural background of the annotator. Thus, an unambiguous definition is necessary to eliminate any such personal bias in the annotation process.

Although in this work we use already annotated public datasets on Hate Speech, we propose a definition that includes the aforementioned description by the European Committee of Ministers, but we extended it to include negate stereotyping speech (e.g. all immigrants are thieves) and also speech that promotes the superiority of one group of people against another. Therefore, online posts do not necessarily need to use offensive language or slurs so as to be characterized as Hate Speech. Usually, racist or sexist content does not contain abusive language, but uses negative stereotypes or argues on the superiority of one group (e.g. nationality).

2.3 Basic terms

Before proceeding further, we will provide some basic definitions of the terms that will be used in this study. Our model is trained via Supervised Learning techniques, using Binary Classifiers which are fed with features generated using Natural Language Processing methods. In this section, we define those terms so as to be more clear to the reader.

Supervised Learning is a machine learning process for training a model/function to map an unseen input to a category based on labeled example data.

Classification is a subset of Supervised Learning methods and constitutes an automatic process to identify which label corresponds to a new observation based on an already known labeled training dataset.

Natural Language Processing (NLP) is a scientific area of Artificial Intelligence which provides a computerized approach on understanding and manipulating natural language text or speech.

Binary Classification is a classification task which trains a model with positive and negative examples with only two possible labels (e.g. Hate Speech or non Hate Speech).

Apart from the binary classification task, one other problem usually addressed in related literature is **multi-class classification**. This means that now the possible classes are more than two, but the classifier still maps each instance with a single label.

3. BACKGROUND AND RELATED WORK

In this chapter, we provide a short review of the related work, not only for Hate Speech detection (presented in chapter 2), but for similar tasks as well. Examples of such tasks can be found in [18] where the authors aim to identify which users express Hate Speech more often, while [34] detect and delete hateful content in a comment, making sure what is left has correct syntax. The latter is a demanding task which requires the precise identification of grammatical relations and typed dependencies among words of a sentence. Their proposed method results have 90.94% agreement with the manual filtering results.

Automatic Hate Speech detection is usually modeled as a Binary Classification. However, multi-class classification can be applied to identify the specific kind of Hate Speech (e.g. racism, sexism etc) [1]. One other useful task is the detection of the specific words or phrases that are offensive or promote hatred, investigated in [30].

In the following sections, we present relevant studies on these tasks, categorized based on the selected features in Section 3.1, on the classification algorithms in Section 3.2 and on the datasets and annotation methods used in Section 3.3. Finally, in Section 3.4 we present two implementations of multi-class classification tasks, whose datasets are used in this study.

3.1 Text representation methods

In this section, we outline the representation methods used in bibliography to represent text. In this work we focus on representations, since the representation step transforms written human language into a form that is understandable by a computer and, by extension, a computer program such as a Hate Speech Detection model. Below we overview a number of different representations used within this domain.

A very popular representation approach is the Bag of Words (BoW) [14, 3, 1] model, a Vector Space Model extensively used in Natural Language Processing and document classification. In BoW, the text is segmented to words, followed by the construction of a histogram of (possible weighted) word frequencies. Since BoW discards word order, syntactic, semantic and grammatical information, it is commonly used as a baseline in NLP tasks.

An extension of the BoW is the Bag of N-grams [19, 14, 18, 7, 31], which replaces the unit of interest in BoW from words to n contiguous tokens. A token is usually a word or a character in the text, giving rise to word n-gram and character n-gram models. Due to the contiguity consideration, n-gram bags retain local spacial and order information.

The authors in [7] claim that lexicon detection methods alone are inadequate in distinguishing between Hate Speech and Offensive Language, counter-proposing n-gram bags with TF-IDF weighting along with a sentiment lexicon, classified with L2 regularized Logistic Regression [16]. On the other hand, [1] use character n-grams, BoW and TF-IDF

features as a baseline, proposing word embeddings from GloVe [1].

There is also a variety of other features used such as word or paragraph embeddings ([9], [30], [1]), LDA and Brown Clustering ([25], [33], [30], [31]), sentiment analysis([12], [7]), lexicons and dictionaries ([12], [27], [8] etc) and POS tags([19], [34], [25] etc).

A related work summarization on features used is provided in the table 1:

Table 1: Features used for Hate Speech Detection in related work

Features	Paper
Bag of words	Kwok and Wang, 2013 [14], Bourgonje et al., 2017 [3], Badjatiya et al., 2017 [1]
n-grams	Nobata et al., 2016 [19], Kwok and Wang, 2013 [14], Mubarak et al., 2017 [18], Davidson et al., 2017 [7], Waseem, 2016 [31], Badjatiya et al., 2017 [1], Del Vigna12 et al., 2017 [8]
character n-grams	Nobata et al., 2016 [19], Waseem, 2016 [31], Waseem and Hovy, 2016 [32], Del Vigna12 et al., 2017 [8]
LDA	Saleem et al., 2017 [25], Xiang et al., 2012 [33]
Brown clustering	Warner and Hirschberg, 2012 [30], Waseem, 2016 [31]
word embeddings	Badjatiya et al., 2017 [1], Del Vigna12 et al., 2017 [8]
paragraph embeddings	Djuric et al., 2015 [9], Warner and Hirschberg, 2012 [30]
sentiment analysis	Gitari et al., 2015 [12], Davidson et al., 2017 [7], Del Vigna12 et al., 2017 [8]
lexicons	Gitari et al., 2015 [12], Xu and Zhu, 2010 [34], Mubarak et al., 2017 [18], Davidson et al., 2017[7], Waseem, 2016 [31], Xiang et al., 2012 [33], Del Vigna12 et al., 2017 [8]
dictionaries	Gitari et al., 2015 [12], Silva et al., 2016 [27], Razavi et al., 2010 [21]
POS	Nobata et al., 2016 [19], Xu and Zhu, 2010 [34], Saleem et al., 2017 [25], Silva et al., 2016 [27], Davidson et al., 2017 [7], Waseem, 2016 [31], Waseem and Hovy, 2016 [32], Del Vigna12 et al., 2017 [8]
typed dependency relationships	Xu and Zhu, 2010 [34], Saleem et al., 2017 [25]

[1] https://nlp.stanford.edu/projects/glove/

3.1.1 Bag of Words and n-grams models

The most common feature, usually used as baseline in related work, is Bag of Words. In short, *Bag of Words or Vector Space Model*, extensively used in Natural Language Processing and document classification, is a representation model where each text is decomposed to its words, without keeping any information on text's grammar or syntax. Bag of Words is an histogram of how many times each keyword appears in a text. They are highly predictive features, which are often used in combination with other features. One disadvantage is that they ignore word semantics or word order in a document.

Other popular features in this category are Bag of word or character-ngrams. *Word n-grams* are sequences of n contiguous words in a text, keeping in this way spacial and order information. Similarly, *character n-grams* are contiguous characters in a document, having the advantage that are less sensitive to noise and misspellings when applied in text classification. In both models, after having extracting the n-grams from a text, the goal is to create an histogram of how many times an n-gram appears in given texts. However, in this way, the only order information that we keep is the one related to the character order inside the n-gram and not their spacial information compared to the other n-grams of the text. In this way, similar words with different semantics have the same n-grams and thus, this information is ignored.

The aforementioned models are the most popular and common features used in bibliography. There are also other features used in combination with n-grams and Bag of Words such as URLs appearing in a text, count of non-alphanumeric characters or punctuation, capitalization etc.

3.1.2 Word Generalization

N-grams models, especially when applied in small texts, face data sparsity problem. This happens due to the large number of possible n-grams compare to a small percentage appearing in a small text (such as a tweet). Similar issue could be faced in Bag of Words features, when counting the appearance of all possible words in a text. Since each instance should have the same vector dimensions, it is necessary to pre-produce all possible words or n-grams and then count how many times they appear in each text.

Apart from the sparsity problem, one other issue faced when using these feature is that word semantics are ignored completely and thus word generalization models are required. One known algorithm for this case is *Brown Clustering*, which produces word clusters and then represents each word to one specific cluster associating the word with the cluster id. On the other hand, *Latent Dirichlet Allocation (LDA)* algorithm represents each word as a topic distribution calculating the possibility that a word corresponds to each topic.

Finally, word embeddings are used widely, mainly in Neural Networks. *Word Embeddings* is a language model in Natural Language Processing that represents each word of a text as a vector of real numbers. They represent each word as a vector, having the advantage that semantically similar words would be represented by similar vectors. While vector

word representations may work well in Natural Language Processing, in Hate Speech Detection problem we are more interested in classifying sentences or paragraphs and not just words. In order to create paragraph vectors, two different methods are used. The first and most ineffective, is to average all word vectors while the second method uses paragraph embeddings directly.

3.1.3 Other features

A variety set of features are also used to improve classification performance. To begin with, sentiment analysis is a useful tool. Hate Speech usually expresses negative sentiments and hate against other people and therefore can be used as additional feature. There are multiple approaches in sentiment analysis either by using a parser for this task to extract sentiment polarity, to count positive or negative overwhelmed words or to pre-process the text to remove objective sentences (using a syntax parser) and then define the sentiment expressed on the specific text.

Moreover, lexical or syntactic/grammar features can be used. For lexical features, a common method is the use of lexicons/dictionaries to detect specific keywords in sentences to extract potential comments expressing hate speech, already mentioned as feature in Bag of Words. Linguistic or syntactic features may be POS tagging or typed dependency relationships. Additionally, there is a set of meta-information features that can be used. One common example is the knowledge if a comments is part of a conversation since previous answers contribute in understanding the meaning of text's content. Apart from the conversation, meta-information on the comment's author is also useful since users that express hatred more often are more likely to write Hate Speech comments again.

Finally, apart from textual analysis of user comments, it is important to detect Hate Speech in comments that include images, audio or video content to promote hatred. Although such content may be more violent and severe than simple text in Hate Speech case, little contributions are currently available in this area.

3.2 Classification Methods

Apart from the text representation, one other important issue is the classification algorithm to be used for the Hate Speech detection task. In this section, we provide a short summary of all the algorithms used in related literature in Table 2. The methods implemented so far in related studies can be divided into two categories: (i) supervised learning and (ii) semi-supervised learning. The algorithms most widely used belong to the supervised learning category and more specifically these are: SVM [6], Logistic Regression (LR) and Naive Bayes (NB) (e.g. [30, 25, 7, 9] etc).

Other algorithms used are Decision Trees and Random Forests (RF) ([7, 3, 33]), while [1] and [8] have used Deep Learning approaches via LSTM networks. Specifically, [1] use CNN, LSTM and FastText, i.e. a model that is represented by average word vectors similar

to BoW, which are updated through backpropagation. The LSTM model achieved the best performance with 0.93 F-Measure, used to train a GBDT (Gradient Boosted Decision Trees) classifier. In [7], the authors use several classification algorithms such as LR with L1 and L2 regularization, NB, Decision Trees, RF and Linear SVM, with L2-regularized LR outperforming all others with 0.91 precision, 0.90 recall and 0.90 F-score.

Table 2: Classification Methods in Hate Speech Detection related work

Classification	Paper no
SVM	Warner and Hirschberg, 2012 [30], Saleem et al., 2017 [25], Davidson et al., 2017 [7], Badjativa et al., 2017 [1], Xiang et al., 2012[33], Del Vigna12 et al., 2017 [8]
Logistic Regression	Djuric et al., 2015 [9], Saleem et al., 2017 [25], Davidson et al., 2017 [7], Waseem and Hovy, 2016 [32], Bourgonje et al., 2017 [3], Badjatiya et al., 2017 [1], Xiang et al., 2012 [33]
Naive Bayes	Saleem et al., 2017 [25], Kwok and Wang, 2013 [14], Davidson et al., 2017 [7], Bourgonje et al., 2017 [3], Razavi et al., 2010 [21]
Decision Trees	Davidson et al., 2017 [7], Bourgonje et al., 2017 [3]
Random Forest	Davidson et al., 2017 [7], Xiang et al., 2012 [33]
Deep Learning(LSTM)	Badjatiya et al., 2017 [1], Del Vigna12 et al., 2017 [8]
Bootstrapping	Waseem and Hovy, 2016 [32], Xiang et al., 2012 [33] Gitari et al., 2015 [12]

3.2.1 Supervised Learning

As we have already mentioned, the most widely used classification techniques belong to the Supervised Learning category. In Table 2, all the listed algorithms, exept Bootstrapping method, are Supervised techniques. In this section, we will provide a short definition of each algorithm and some performance results in works that outperformed other classifiers.

Support Vector Machines (SVM) are Supervised Learning techniques that can be employed for both classification and regression tasks, although they are most commonly used in classification. The goal of this kind of algorithms is to find a hyperplane (i.e. a line that linearly separates and classifies a set of data) that best divides a dataset into two classes (e.g. Hate Speech or non Hate Speech). Support vectors are the data points nearest to the hyperplane, which, if removed, they would alter the position of the dividing hyperplane. Warner and Hirschberg, 2012 [30] have used SVM^{light} algorithm achieving 94% accuracy, 68% precision, 60% recall and 0.63 F1 measure, while Badjativa et al., 2017 [1] have used SVM as baseline to compare it with Deep Neural Networks.

Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured

with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between dependent and independent variables of a problem. Logistic regression generates the coefficients of a formula to predict a logit transformation of the probability of presence of the characteristic of interest. One important consideration is the number of independent variables, since the increase of those variables may result in overfitting. Davidson et al., 2017 [7] have used Logistic Regression with L2 regularization which had an overall precision 0.91, recall of 0.90, and F1 score of 0.90.

Naive Bayes is a simple probabilistic algorithm based on Bayes' Theorem to classify objects making strong (naive) independence assumptions on the features used. The key insight of Bayes' theorem is that the probability of an event can be adjusted as new data is introduced. A naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms that make uses of statistical independence. Popular uses of naive Bayes classifiers include spam filters, text analysis and medical diagnosis. Due to their simplicity, these algorithms are usually used as baseline in the related work. One interesting implementation is in the work of Razavi et al., 2010 [21] where the authors use a three-level classification in order to train their model. In the first level they use the Complement Naïve Bayes classifier to select the most discriminative features as the new training feature space and pass them to the next level of classification. Then, in the second level, they use Multinomial Updatable Naïve Bayes classifier to update their model and extract the features for the last level. Apart from those features, in the last level, they use their Insulting and Abusing Language Dictionary, containing 2700 flame words, phrases, and expressions. Finally, they run the last classification level using a rule-based classifier named DTNB (Decision Table/Naive Bayes hybrid classifier). After preprocessing and before performing the feature selection, they ran the Complement Naïve Bayes classifier on the whole feature space, achieving accuracy 16% better than the baseline.

Decision Trees is a map of the possible outcomes of a series of related choices that uses a tree-like graph. A decision tree consists of three types of nodes: (i) decision, (ii) chance and (iii) end nodes. The decision tree can be linearized into decision rules, but this may conclude in deep paths. This algorithm is tested, among other in Bourgonje et al., 2017 [3] achieving 76.17 f-score when tested in English Tweets.

Random Forests are an ensemble learning method for classification, regression and other tasks that grow many classification trees. In order to classify a new object from an input vector, they feed the input vector down each of the trees in the forest. Each tree gives a classification output by labeling the input or a ranking result by sorting the most possible labels. The forest chooses the classification having the most votes (over all the trees in the forest). To avoid overfitting the *bootstrap aggregating* technique is used, which is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. This method is used in Davidson et al., 2017 [7] and Xiang et al., 2012 [33] with no significant results, since in both works the model that outperformed all others was Logistic Regression.

Long Short Term Memory networks (LSTMs) are a special kind of Recurrent Neural

Network(RNN), capable of learning long-term dependencies (Hochreiter & Schmidhuber (1997) [13]). LSTMs are explicitly designed to avoid the long-term dependency problem and therefore are well-suited to classifying, processing and making predictions based on time series data. All recurrent neural networks have the form of a chain of repeating modules of neural network such as a single tanh layer. LSTMs instead of having a single neural network layer, they are having four. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. This method is used in Badjatiya et al., 2017 [1] and Del Vigna12 et al., 2017 [8]. The first work is analyzed in Section 3.4. In the latter work, the authors use both SVM and LSTM classifiers implemented with Keras framework. Their results are similar with LSTM achieving 79.81 accuracy while SVM performs slightly better with 80.60 accuracy.

3.2.2 Semi-Supervised Learning

Bootstrapping approach is a semi-supervised learning process used either to generate additional data automatically or to create hatred lexical resources. It not widely used in related bibliography, however it is applied by Waseem and Hovy, 2016 [32], Xiang et al., 2012 [33] and Gitari et al., 2015 [12].

The authors of Waseem and Hovy, 2016 [32] used a bootstrapping method to automatically collect tweets by using common slurs associated with Hate Speech. Moreover, the authors of Xiang et al., 2012 [33] use the Map-Reduce framework in Hadoop to collect tweets automatically from users that are known to use offensive language, and a bootstrapping method to extract topics from tweets. On the other hand, bootstrapping can also be applied in order to build lexical resources used as part of the detection process. Gitari et al., 2015 [12] in their work implement a bootstrapping method in order to populate their hate verb lexicon, starting with a small seed verb list, and iteratively expanding it based on WordNet, by adding all synonyms and hypernyms of those seed verbs.

3.3 Datasets and Annotation Methods

In previous sections, we have reviewed the relevant literature related to features and classifiers used in bibliography. Other important steps in the Hate Speech detection task are the annotation process and the dataset to be used (i.e. the kind of comments based on the social media platform which will be used to extract data).

In this section, a review on datasets and annotation methods is included. In table 3 shows a summary of all available datasets in related work while 4 provides information on the annotation methods used on those datasets.

The most popular social media used to extract data is Twitter followed by Yahoo! News or Finance. Related to the annotation methods, there are multiple examples using expert annotators or crowdsourcing methods while Saleem et al., 2017 [25] and Warner and

Hirschberg, 2012 [30] annotated their data in an automated way by targeting communities that promote hatred and by automatically exporting posts from users in these selected communities.

Table 3: Datasets used in related work

Datasets	Paper no
Twitter	Silva et al., 2016 [27], Kwok and Wang, 2013 [14], Mubarak et al., 2017 [18], Davidson et al., 2017 [7], Waseem, 2016 [31], Waseem and Hovy, 2016 [32], Bourgonje et al., 2017 [3], Badjatiya et al., 2017 [1], Xiang et al., 2012 [33], Ross et al., 2017 [23]
Facebook	Ben-David and Matamoros-Fernandez, 2016 [2]
Reddit	Saleem et al., 2017 [25]
Yahoo!	Djuric et al., 2015 [9], Nobata et al., 2016 [19], Xiang et al., 2012 [33]
YoutTube	[34]
Voat	Saleem et al., 2017 [25]
Forums	Saleem et al., 2017 [25]
Whisper	Silva et al., 2016 [27]
Other	Gitari et al., 2015 [12], Nobata et al., 2016 [19], Xiang et al., 2012 [33], Bourgonje et al., 2017 [3], Razavi et al., 2010 [21]

Table 4: Annotation methods used in related work

Annotation	Paper no
Expert	Warner and Hirschberg, 2012 [30], Mubarak et al., 2017 [18], Waseem, 2016 [31], Waseem and Hovy, 2016 [32]
Crowdsourcing	Kwok and Wang, 2013 [14], Davidson et al., 2017 [7], Waseem, 2016 [31], Ben-David and Matamoros-Fernandez, 2016 [2]
User Reports	Nobata et al., 2016 [19], Warner and Hirschberg, 2012 [30], Mubarak et al., 2017 [18]
Automatic	Saleem et al., 2017 [25], Warner and Hirschberg, 2012 [30]
Hatebase	Silva et al., 2016 [27], Davidson et al., 2017 [7]
SentiWordNet	Gitari et al., 2015 [12]

For more information, the survey of [26], on which we were based to perform the above categorization, provides a detailed analysis of detector components used for Hate Speech detection and similar tasks.

3.4 Related Work Presentation

In this section, two works are presented (Davidson et al., 2017 [7] and Badjatiya et al., 2017 [1]). A short description or their methods is provided, including the features and the classification method used, as well as their results. Both works define a multi-class classification problem. Davidson et al., 2017 [7] [2] model is trained to recognize comments expressing hatred against user comments just using offensive language, while Badjatiya et al., 2017 [1] have trained their model to separate different kinds of hate speech (racism and sexism) by using the dataset provided by Waseem, 2016 [31] [3]. In our study, we have used both dataset provided by the authors of those two works in their public GitHub repositories.

3.4.1 Automated Hate Speech Detection and the Problem of Offensive Language

The first work to be presented is Davidson et al., 2017 [7]. In this work, the authors aim to distinguish user comments using offensive language from those expressing severe hate speech. Lexical detection and supervised learning have both failed in this task, based on previous relevant work findings. Bag-of-words features result in high recall but poor precision, since there is presence of a high number of false positives. Generally, keyword-based methods have similar results, since they cannot distinguish when those keywords are used for hate speech.

The authors create a more strict hate speech definition and use a multi-class classifier to label data as hate speech, offensive language or clean. In order to create an annotated dataset, they collected tweets using the Twitter API and searched hate speech keywords from the Hatebase.org and then they have used CrowdFlower to annotate them.

During the preprocessing phase, they have lowercased and stemmed the text using Porter stemmer and created unigrams, bigrams and trigrams with TF-IDF as features. In addition, they have used a sentiment lexicon for social media to include sentiment analysis as feature. In classification, they tested several algorithms such as logistic regression with L1 and L2 regularization, naive bayes, decision trees, random forests and linear SVM.

The final model they have chosen is logistic regression with L2 regularization and tested it with 5-fold cross validation. The best performing model has an overall precision 0.91, recall of 0.90, and F1 score of 0.90. However, 40% of hate speech is classified as less hateful than the manual annotation but only 5% of our true offensive language was labeled as hate.

As future work, the authors want to distinguish different uses of hate speech (to what target group is addressed, part of conversation etc) and also study the characteristics of people expressing hate speech.

[2] https://github.com/t-davidson/hate-speech-and-offensive-language

[3] https://github.com/zeerakw/hatespeech

3.4.2 Deep Learning for Hate Speech Detection in Tweets

In Badjatiya et al., 2017 [1], the authors use a dataset of 16K annotated tweets provided by Waseem and Hovy, 2016 [32] and classify them as racist, sexist or neither. They use as baseline methods char n-grams, TF-IDF and Bag of Words vectors (BoWV).They experiment with a number of different classification methods such as Logistic Regression, Random Forest, SVMs, Gradient Boosted Decision Trees (GBDTs) and Deep Neural Networks(DNNs).

The authors propose three neural network architectures initializing word embeddings with random or GloVe embeddings and are trained (fine-tuned) using labeled data with back-propagation. These methods are: CNN, LSTM(RNN) and FastText with word vectors similar to BoW, with the difference of getting updated through back-propagation. In addition, they have experimented with classifiers such as SVMs and GBDTs. Their source code is publicly available on their GitHub page. In order to evaluate their results they use 10-Fold Cross Validation and weighted macro precision, recall and F1-scores.

Their best method is “LSTM + Random Embedding + GBDT” which was initialized to random vectors, LSTM was trained using back-propagation, and then learned embeddings were used to train a GBDT classifier. As future goal, they want to test the importance of the user network features for hate speech detection.

4. PROPOSED METHOD

4.1 Goals

In this chapter, we present our implementation of an automatic tool for Hate Speech detection task. We focus on user-generated texts from social media platforms — specifically Twitter posts. Our goal is to effectively classify these tweets as Hate Speech or non Hate Speech. As we already mentioned, we focus on the text representation, since it is an important step to transform human written text in a format understandable by a machine without dropping useful information. Therefore, we have evaluated the performance of several established text representations (e.g. Bag of words, word embeddings) and classification algorithms, investigating the contribution of "n-gram graph"-based features to the Hate Speech classification process. Moreover, we examine whether a combination of deep features (such as n-gram graphs) and shallow features (such as Bag of Words) can provide top performance in the Hate Speech detection task.

In this work, we aim to show that, among the tested text representations, n-gram graphs can have a significant contribution in detecting Hate Speech content in user-generated text, handling inherent noise effectively without preprocessing. The N-gram graphs are a text model which associates all pairs of n-grams with edges, to capture local co-occurrence information of n-grams in the text. The nodes of the graph represent each n-gram of the text, while the edges correspond to the frequency of co-occurrence of the node n-grams within a given (parameter) text window.

Usually n-grams features are used as part of a bag of n-grams model, which is a histogram counting the number of times a n-gram appears in a text. However, similar words with different semantics will have the same n-grams although their meaning is completely different. In n-gram graphs, it is possible to capture the difference between morphologically similar but semantically different words, since the information kept is not only the specific n-gram but also its context (neighboring n-grams). Furthermore, the representation inherently holds good noise handling properties, as has been previously demonstrated in the literature [20].

4.2 Data Preprocessing

Prior to text transformation into attributes understandable by machines, we need to discard noise and useless artifacts. Twitter post, although being short texts, usually have a lot of noise such as hashtags(#), retweets (RT), URLs and mentions of other Twitter users (@username). All these characters should be removed since they are irrelevant to our task.

Additionally, we decided to lowercase the text, remove punctuation and common English stop words. **Stop words**, in Natural Language Processing, are words that are filtered out

before data processing. Though ”stop words” usually refer to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools. We have found a CSV [1] listing common English words irrelevant to our detection task. We note that, despite this preprocessing, we expect noisy content, due to e.g. spelling mistakes, jargon, emoticons, etc.

4.3 Text representation

As already mentioned, the representation step transforms written human language into a form that is understandable by a machine or program. This data representation stage is a crucial step in a machine learning pipeline, since:

1. The representation has to be informative; useful attributes and characteristics (with respect to the machine learning task, i.e. classification, in our case) present in the data has to be retained
2. Irrelevant / redundant input properties and noise has to be discarded from the representation output
3. It is the first step in the machine learning pipeline; thus, any errors, omissions and shortcomings of the resulting features, will persist and propagate through the pipeline components

In this section, we will present in more detail all the text representation techniques used for our task. In subsection 4.3.1 we review the Bag of Words model, enriched with knowledge on hate keywords, while in subsection 4.3.2 we present our Bag of n-grams (token and character) techniques and in subsection 4.3.3 the word embeddings representation using pre-trained vectors from GloVe [2]. In subsections 4.3.4, 4.3.5 and 4.3.6 we review all syntactic and grammatic features as well as the sentiment analysis used on the dataset. Finally, in subsection 4.3.7 we present the NGGs method, which we want to show that it is a rich text representation with significant performance.

Apart from the aforementioned features, we have also search for relevant features used in similar areas such as Authorship Attribution. After reading a survey on the methods used for this task in [28], we came to the conclusion that we have experimented with the majority of the features mentioned (i.e. bag of words, word and character ngrams, spell checking, syntax analysis, embeddings). One of the methods mentioned, i.e. Profile-Based Approach, has a very similar logic to n-gram graphs. This method creates sets of texts of the same author and each text is compared to all texts of a set computing the similarity. The difference between n-gram graphs and Profile-Based Approach is that in the first method the graph created is the average from all graphs under the same label while in the latter the set contains the whole initial documents.

[1] CSV list here: `https://github.com/igorbrigadir/stopwords`
[2] https://nlp.stanford.edu/projects/glove/

4.3.1 Bag of words

The first feature included in this work is Bag of Words (vector space model) which is, as already mentioned, a representation model where each text is decomposed to its words, without keeping any information on text's grammar or syntax. In order to extract those features, each tweet was split to each words and then an histogram was created showing the number of times each word appears in each text.

One straight forward way to create bag of words features is to store all available words among all tweet instances in a data structure and count the number of times each word appears in a text, which is an expensive approach in memory and time, since it leads to very large feature vectors, which encumbers latter components in the machine learning pipeline in which they are used.

Below figure presents the BoW technique with two document example, listing also the relevant stop words found in the two texts.

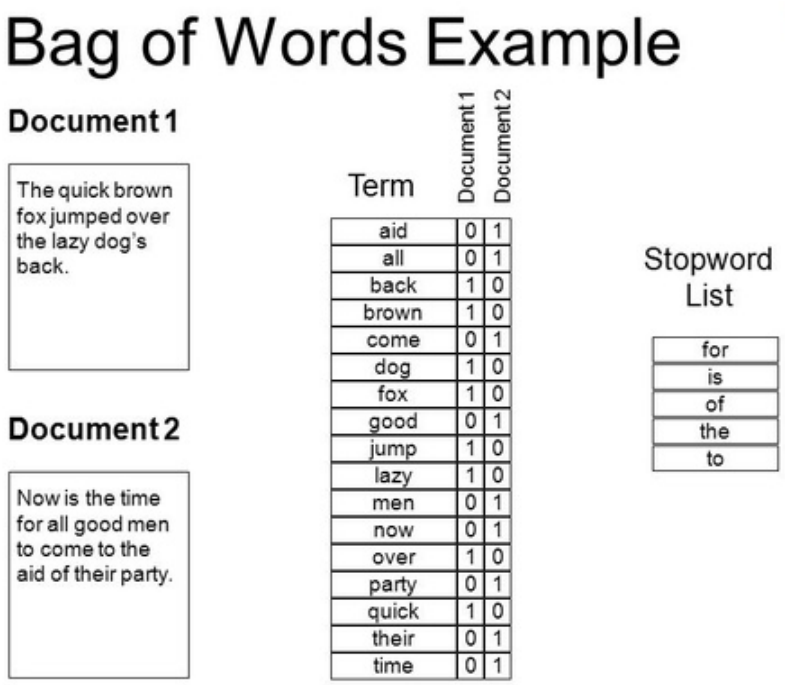

Figure 1: Bag of Words example

Several research studies on Hate Speech detection used lexicons and dictionaries in order to detect hatred in user comments, as already presented in chapter 3. These lexicons contain common words or slurs associated with Hate Speech and are used to help automatic detection. Apparently, these features should not be used separately, since hate

keywords may also be used by defenders of minority groups who do not promote hatred, as stated in Saleem et al., 2017 [25]. One popular list containg hate keywords is HateBase [3].

In order to improve system's performance, we decided to use a csv list with HateBase keywords provided by Davidson et al., 2017 [7] [4]. Therefore, instead of using all available words from all tweets, we only used the hate keywords from HateBase and generated the histogram only for those words. This has reduced the vector space making possible to keep all data in memory and improve the time needed to run the experiments.

4.3.2 Word and character n-grams

Apart from BoW, we have used additional bag models, with respect to word and character n-grams. Word n-grams are sequences of words in a text while character n-grams are sequences of characters in a document keeping character spacial and order information of the characters inside the n-grams.

One useful advantage of character n-grams compared to token n-grams is that they might provide a way to overcome the spelling variation problem that is usually faced when working with user generated comments. For example, in order to avoid detection from automatic tools, users usually omit or replace some characters with symbols (e.g. ki11 yrslef a$$hole). The aforementioned problem result in very rare or unknown tokens in the training data when working with token n-grams. On the other hand, character n-grams are more likely to capture the similarity to the initial spelling of those words.

Similarly to BoW features, in order to guarantee a common bag feature vector dimension across texts, we pre-compute all n-grams that appear in the dataset, resulting in a sparse and high-dimensional vector. To generate these features for all texts, Weka's tokenizer is used to tokenize each text. In order to handle this high-dimensional vector and to improve time and space complexity, it is necessary to reduce the vector space. Therefore, we keep only the 100 most frequent n-grams features, discarding the rest.

Unfortunately, as we will illustrate in the experiments, this decision resulted in highly sparse vectors and, thus, reduced the efficiency of those features.

4.3.3 Word embeddings

Another usual text representation are word embeddings, which is an NLP technique that maps each word in a language's vocabulary to a vector of real numbers. This method allows to represent similar words with similar vectors and take into account word semantics. The fact that word embeddings consider word semantics and are able to have similar representations for similar words makes them much more effective than n-gram or character n-gram features.

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

[4] https://github.com/t-davidson/hate-speech-and-offensive-language

We have used the pre-trained GloVe [5] word embeddings to represent the words of each tweet, mapping each word to a 50-dimensional real vector. If a word did not exist into the GloVe embeddings, we did not include the word into the vector. Although, the file contained several words, even informal ones used in social media, a small amount of words in tweets did not exist in the file and therefore were excluded. To arrive to a tweet-level representation, we compute the average vector from all words in the tweet.

In order to create features based on embeddings, we have loaded a serialized object containing english words with the relevant vector, acquired from the aforementioned site. The next step is to split each text to its words and replace those words with the corresponding vectors if available. Finally, we have aggregated all vectors for a text calculating the average for each dimension (we also provide the possibility to aggregate the vectors by keeping the maximum value in each dimension).

4.3.4 Sentiment Analysis

Sentiment analysis process uses NLP and text analysis to extract user's sentiment polarity and subjective information. We have experimented with sentiment analysis as feature to determine if a tweet promotes hatred or not. Usually, when someone expresses hatred against another group of people, he also expresses negative sentiments.

In order to extract sentiment polarity, we have used Stanford NLP parser [6], which creates a Tree for each sentence of the text. Then, it annotates the sentences based on sentiment polarity from which we have kept the sentiment of the longest phrase.

Our experiments have shown that sentiment analysis cannot be used alone as feature to distinguish Hate Speech content. This is normal since negative sentiment can be expressed for many other reasons and classifier will not have any other feature to help in determining the difference between negative sentiment promoting hatred and negative sentiment in non Hate Speech content. Therefore, sentiment analysis is a useful tool, however needs to be combined with other NLP techniques in order to have valuable results.

4.3.5 Linguistic Features

Apart from Sentiment Analysis, linguistic and grammatical features were extracted to examine whether Hate Speech is correlated to the user's proficiency in writing. Usually, people that express hatred against minorities are not well educated and this results in messages containing multiple misspellings or grammar errors.

We have used an English dictionary [7] to collect all English words with correct spelling and, then, for each word in a tweet, we have calculated its edit distance from each word in the

[5] https://nlp.stanford.edu/projects/glove/
[6] https://nlp.stanford.edu/software/lex-parser.html
[7] http://www.bragitoff.com/2016/03/english-dictionary-in-csv-format/

dictionary, keeping the smallest value (i.e. the distance from the best match). The final feature kept was the average edit distance for the entire post. If the majority of words were spelled correctly the edit distance would be close to 0.

As we will indicate in our experiment results, the grammatical or syntax errors are not directly associated with users expressing Hate Speech. Although from our experience in Greek social media, this is a common pattern, it appears that in our dataset we cannot have valuable results, at least when those features are tested separately. However, we have decided to include them in our study and review them through significance testing (ANOVA and Tukey's tests).

4.3.6 Syntax Analysis

In a similar logic, as we did for the linguistic and grammatical features (subsection 4.3.5), we decided to also examine user's syntax in Twitter post and check if there is a common pattern for users that promote hatred. Similarly to Sentiment Analysis, we have used the Stanford NLP parser for syntax analysis. The parser tokenizes the text and creates syntax trees. As feature we keep the best score of the parser.

In case of wrong syntax of a message, the parser will have low score. Based on our hypothesis that it is likely uneducated people to express hatred, we expect that Hate Speech comments will have problematic syntax. Unfortunately, as we will show in the experiments, we can extract little information from syntax analysis, since there is not a clear pattern between wrong syntax and Hate Speech comments.

4.3.7 N-gram graphs

Expanding the use of n-grams, we employ n-gram graphs (NGGs) as a text representation method. N-gram graphs is a text representation method using edges and nodes to create a graph of the document either with token n-grams or with character n-grams. This method keeps information about the words order connecting neighboring n-grams with edges.

This approach requires to first create a representative graph for each class (i.e. Hate Speech or Clean). For each category, we use a 90% of the available *training instances* to create the class graph. Then each instance is represented based on its (graph-based) similarity to the representative graph of each class. Thus, each instance is represented as a vector of similarities.

We note that the use of only 90% of the training instances for the representative class graph is required in order to avoid overfitting of our model. In short, if we used all training instances to create the representative graph would result in very high instance-class similarity during the training phase. Since we use the instance-class similarities as a classifier input feature in the next step, the above approach would introduce extreme overfit to the classifier, biasing it towards expecting perfect similarity scores in cases of an instance belonging to a class, a scenario which of course rarely – if ever – happens with real world

data.

As indicated above, after creating the graph for each category and instances, we calculate the graph similarity of each tweet graph to each category graph. Thus, in our case, we end up with a vector in $\mathbb{R}^n$ where n is the number of possible classes.

4.4 Classification

In this section, we will describe the classification process and the algorithms used to detect Hate Speech content in our dataset. As a reminder, classification is the task to train a function f to map input variables to a label (output variable). The classifier is based on already seen instances (training phase) to predict the category/label of any unseen instance with the same features.

As we have already mentioned, we have created three datasets based on the datasets provided by [1] and [7]. First of all, we have unified the two datasets in one, to use it for Binary Classification task. The labels in this dataset are `Hate Speech` or `Clean`. The classifier here is a binary classifier, since each text can be assigned to one of the two available labels (Hate Speech or non Hate Speech).

We also tested our model in multi-class classification tasks using the initial datasets. In the first dataset [31], the possible labels are `Racism`, `Sexism` or `Clean` while in the second [7], the categories are `Hate Speech`, `Offensive Language` and `Clean`.

4.4.1 Instances creation

Prior to classifiers analysis, we will illustrate some implementation details related to the text transformation in order to feed the classifier with training examples.

Firstly, we have implemented the feature extraction in Java, while for classification algorithms we have used Weka, Keras and scikit-learn frameworks. The two Python frameworks were used for Artificial Neural Networks (ANNs) implementation. The rest classifiers were implemented in Weka and scikit-learn. The code of our model can be found in our GitHub repository [8].

So far, we have created one HashMap per Twitter post, containing all the available features and the relevant values. In order to feed the classifiers with these features, we decided to use Weka Framework in order to generated .arff files containing Instances objects per feature kind for all the three datasets. Therefore, before starting to train our classifiers, we have created and exported .arff files with Weka Instances objects. The dataset is split in 10 folds (we have used 10-fold cross validation) and each fold contains training and testing instances.

These Instances are also created for each dataset. Therefore, we concluded with three

[8] https://github.com/cthem/hate-speech-detection

dataset folders containing separate instances for all features using 10-fold cross validation. Apart from testing our features separately, we have merged these instances in three different ways: all available instances (we will refer to this combination as `all`), all instances apart from n-gram graphs (we will refer to this combination as `vector`) and all instances containing our best performing features (i.e. NGGs, BoW and word embeddings - which will be refered as `best`).

Finally, we will provide a short description of the Instances data structure. Each Instances object (train and test) contains multiple Instance objects. In order to create an Instance (in our problem an Instance represents a specific Twitter post), we used the Attribute object in Weka Framework, which represents a specific feature described by its name and its value. Therefore, each Instance is associated with a list of Attribute objects which will be used in order to train and test a classifier.

This format was used with Weka classifiers. However, we have implemented all the classifiers with scikit-learn and Keras (for ANNs) frameworks as well. Since .arff format is not compatible with Python, we have used the arff2pandas library [9] in order to convert all the Instances in the .arff files into pandas dataframes.

4.4.2 Algorithms

Having described the text transformation process to features and the features into Instances, we will now describe the classification algorithms that we have used for Hate Speech detection. As previously mentioned, classifiers are implemented both in Java and Python. Specifically, Naive Bayes, Logistic Regression, Random Forest and K-Nearest Neighbors were implemented with Weka and scikit-learn libraries while for ANNs we have used Keras and Scikit-Learn frameworks.

4.4.2.1 Naive Bayes

Naive Bayes (NB) [24] is a simple probabilistic classifier, based on Bayesian statistics. It is one of the simplest classifiers with limited performance due to the independence assumption related to the features, ignoring any possible correlations between them. The classifier assigns one label (or multiple in case of multi-label classification) to an instance which is represented by feature vectors as already described. We used this classifier as baseline to test our model.

For each available class, NB calculates the possibility to assign this class to an Instance described by a vector X of independent features $x_1, x_2, \ldots, x_n$, i.e. $p(C_k|x_1, x_2, \ldots, x_n)$ where C_k is a possible class out of k available labels. To deal with large number of features, the possibility can be calculated based on Bayes theorem:

[9] https://github.com/garicchi/arff2pandas

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

NB usually performs rather well compared to other more complicated classifiers, serving as a common baseline. Additionally, this assumption simplifies the learning process, which is reduced to the model learning the attributes separately, vastly reducing time complexity on large datasets.

4.4.2.2 Logistic Regression

Logistic Regression (LR) [17] is a statistical model applied in a dataset described by a binary dependent variable (e.g. Hate Speech or non Hate Speech). The probability of an output is calculated by the linear combination of the independent or predictor variables. This model can be generalized to be used for multi-class classification having more than two dependent variables, named multinomial logistic regression.

LR is named after the core function used, the logistic or sigmoid function, which is represented by an S-shaped curve that assigns values between 0 and 1 to all real numbers. The equation used in LR to represent the model is very similar to linear regression model. The input feature values are combined linearly with relevant weights and coefficient values to calculate the output.

The output of a LR classifier is the possibility that an input vector can be assigned with an available class/label. The next step is to transform the calculated probabilities into binary values (0 or 1). This is achieved by using the logistic function which limits the range of the output values between 0 and 1 as already mentioned.

The training of this model consists in learning the coefficient values from the training instances using maximum-likelihood estimation, a common algorithm used in machine learning applications. The result should be a high probability (close to 1) for the correct class and a low probability close to zero for the wrong class.

Similarly to NB classifier, the model used both from Weka and Scikit-Learn needed only a few minutes to train and test the model, however LR has performs better than NB.

4.4.2.3 Random Forest

This algorithm is used for supervised learning for both classification and regression tasks. The main idea is that Random Forest (RF) [15] creates an ensemble of decision trees during the training phase and each tree predicts the label of the input instance. Usually, deep trees tend to overfit during training. One way to overcome this issue is the bagging or bootstrap aggregating technique, which prevents from creating correlated trees and reduces the value of the variance of the system. Correlated trees are usually responsible for the model's overfitting to the training dataset.

RF measures each features importance on the prediction result by looking how much prediction error increases to each node that uses the specific feature. One other important measure provided by RF is the proximity measure, i.e. a proximity matrix showing a fraction of trees where two elements fall in the same terminal node. [15]

In our work, we have used Random Forest algorithm implementations in Weka and Scikit-learn library. In both cases, the classifier provided better results than NB classifier without being expensive in time.

4.4.2.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) [10] is a simple and commonly used non-parametric and instance-based learning algorithm, used both for classification and regression. Non parametric means that that the algorithm does not have knowledge and does not make any assumptions related to the underlying data distribution, while instance-based means that during the training phase the algorithm does not use the training instances to do any generalization. The training phase is really fast and the training instances are used in the testing phase to classify an unseen instance.

While KNN is pretty simple algorithm, with no assumptions on the data and applies to both classification and regression tasks, it is very expensive in memory since there is need to store all training data and also in time to compare each new instance to all the training instances and calculate the similarity.

The k is a pre-defined integer number which creates ”boundaries” between the possible classes/labels. After having decided the value of k, we load all training data and keep them in memory. During the testing phase, for each new instance the algorithm calculates the difference of this instance to each training instance using some distance metric (e.g. Euclidean distance) and sorts the results in ascending order. The algorithm retrieves the top k distances and collects the most frequent label among these results.

Similarly to the above algorithms, Scikit Learn and Weka were used for implementing KNN. While the precision, recall and f-measure in both libraries were similar, Weka implementation was much more expensive in time when it comes to our largest dataset (for binary classification). In more detail, Scikit-Learn KNN algorithm needed only a couple of minutes during the testing phase, Weka needed more than three hours to complete the training and the testing of a fold.

4.4.2.5 Neural Networks

Artificial Neural Networks (ANNs) are inspired by biological nervous systems (e.g. human brain) on how they are structured and how they process information. ANNs are not a new idea, they first appeared in the 1940s. However, the research on this area was freezed due to limited computers capacity back then. Nowadays, that computer machines have larger processing power, neural networks are again in the foreground.

Since ANNs are imitating biological neural networks, their learning process is also similar. They are not programmed with task-specific rules but they learn by example just as humans learn. ANNs are composed by a large number of highly interconnected neurones in multiple layers where each connection transmits a signal to the next layer of neurons.

The edges that connect neurons are characterized by their weight and each neuron has a specific threshold determining if the neuron is activated. The weight and bias (threshold) values are tuned through the learning process. The method used by the system in order to learn is called Backpropagation.

Before explaining how Backpropagation works, it is important to describe the concept of a neuron and the architecture of an ANN. An artificial neuron is characterized by two phases, the training and using phases. During the training phase, based on the input patterns learns whether to fire or not (activated or not). For unseen inputs, in the using phase, the neuron will decide the weight and bias values based on the rule learned during the training phase.

ANNs are composed by multiple layers. The first layer constitutes the input feature vector and each neuron of this layer represents a specific feature value. On the other hand, the last layer has only one neuron which gives the output of the system for a given input. Between these two layers, it is possible to have any number of hidden layers which also consist of artificial neurons. The neurons of each layer are connected to all the neurons of the next layer with weighted edges. As we have already mentioned, each neuron is characterized by a threshold value. The function that determines if a neuron will be fired or not is constituted by the weights of the edges that connect this neuron with the neurons of the previous level and the threshold (bias) value.

Backpropagation algorithm calculates the gradient of the loss function required for the tuning of weights and biases. Based on the output, the algorithm calculates the error (the difference from the expected output) and transmits it back to all layers through the network. All weights and biases are now re-calculated and this process is repeated. The goal is reduce the value of the loss function, but without overfitting to the training dataset. We remind the reader that, in this study, we have used Scikit-Learn and Keras frameworks in order to build a neural network model. From scikit-learn library we have used MLP classifier with the default parameters, therefore we will not describe it further. On the other hand, we have created a Sequential model with Keras library using three hidden layers between the input and output layers. The two out of the three hidden layers where used in order to reduce overfitting to the training dataset. The first one was Dropout layer while the second was Gaussian Noise.

5. EXPERIMENTS AND RESULTS

In this chapter, we present the experimental setting used to answer the following questions:

- Which features have the best performance?
- Does features combination achieve a better performance than using them separately?
- Do NGGs have significant / comparable performance to BoW or word embeddings despite being represented by low dimensional vectors?
- Are there classifiers performing statistically better? Features or classifiers are more significant?

We also elaborate on the the datasets utilized, experimental and statistical significance results, as well as a discussion of our findings.

5.1 Datasets and Experimental Setup

We use the datasets provided by [32] [1] and [7] [2]. We will refer to the first dataset as RS (racism and sexism detection) and to the second as HSOL (distinguish Hate Speech from Offensive Language).

In both works, the authors perform a multi-class classification task against the corpora. In [32], their goal is to distinguish different kinds of Hate Speech, i.e. racism and sexism, and therefore the possible classes in RS are `Racist`, `Sexist` or `None`. In [7], the annotated classes are `Hate Speech`, `Offensive Language` or `Clean`.

Given the multi-class nature of these datasets, we combined RS and HSOL into a single dataset, keeping only instances labeled `Hate Speech` and `Clean` in the original.

We use the combined (RS + HSOL) dataset to evaluate our model implementations on the binary classification task. Furthermore, we run multi-class experiments on the original datasets for completeness.

We perform three stages of experiments. First, we run a preliminary evaluation on each feature separately, to assess its performance. Secondly, we evaluate the performance of concatenated feature vectors, in three different combinations: 1) the top individually performing features by a significant margin (`best`), 2) all features `all` and 3) vector-based features (`vector`), i.e. excluding NGGs. Via the latter two scenarios, we investigate whether NGGs can achieve comparable performance to vector-based features of much higher dimensionality.

[1] `https://github.com/ZeerakW/hatespeech`

[2] `https://github.com/t-davidson/hate-speech-and-offensive-language`

Given the imbalanced dataset used (24463 Hate Speech and 14548 clean samples), we report performance in both macro and micro F-measure. Finally, we have evaluate the statistical significance of the run components by a series of ANOVA and Tukey HSD test evaluations.

5.2 Results

In this section we will provide the results of our experiments both for the binary and the multi-class classification tasks. We have separated our tables based on the features used. We provide separate table for feature combinations (i.e. `best`, `all` and `vector`. As we have already mentioned, the `best` combination includes our best performing features (NGGs, BoW and word embeddings), the `all` combines all the features we tested and finally the `vector` combination includes all vector features exept the NGGs. Respectivelly, we have separate tables for our text represenations. NGGs, BoW and word vectors are in the same table, followed by a table with sentiment, syntax and grammar analysis and finally we present the results of bag of n-gram models (word and character).

5.2.1 Binary Classification

In this subsection, we will present the results of our binary classification task. Firstly, to answer the question on the value of different on different feature types, we perform individual runs which designate BoW, word2vec embeddings and NGG as the top performers, with the remaining features (namely sentiment, spelling / syntax analysis and n-grams) performing significantly worse. All approaches however surpass a baseline performance in terms of a naive majority-class classifier (scoring $0.382/0.473$, in terms of macro and micro F-measure respectively) and are described below.

The results of the top individually performing features, in terms of micro / macro average F-Measure, are presented in table 6. **Bold** values represent column-wise maxima, while underlined ones depict maxima in the left column category (e.g. feature type, in this case). The best performer is BoW with either LR or NNs, followed by word embeddings with NN classification. NGGs have a slightly worse performance, which can be attributed to the severely shorter (2D) feature vector it utilizes. On the other hand, BoW features are 1000-dimensional vectors. Compared to NGGs, this corresponds to a 500-fold dimension increase, with a 9.0% micro F-measure performance gain.

Subsequently, we test the question on whether the combination of features achieve a better performance than individual features. The results are illustrated in Table 5. First, the `best` combination that involves NGG, BoW and Word2Vec features is, not surprisingly, the top performer, with LR and scikit-learn NNs obtaining the best performance. The `all` configuration follows with NB achieving macro/micro F-scores of 0.795 and 0.792 respectively. This shows that the additional features introduced significant amounts of noise, enough to reduce performance by canceling out any potential information the extra features might

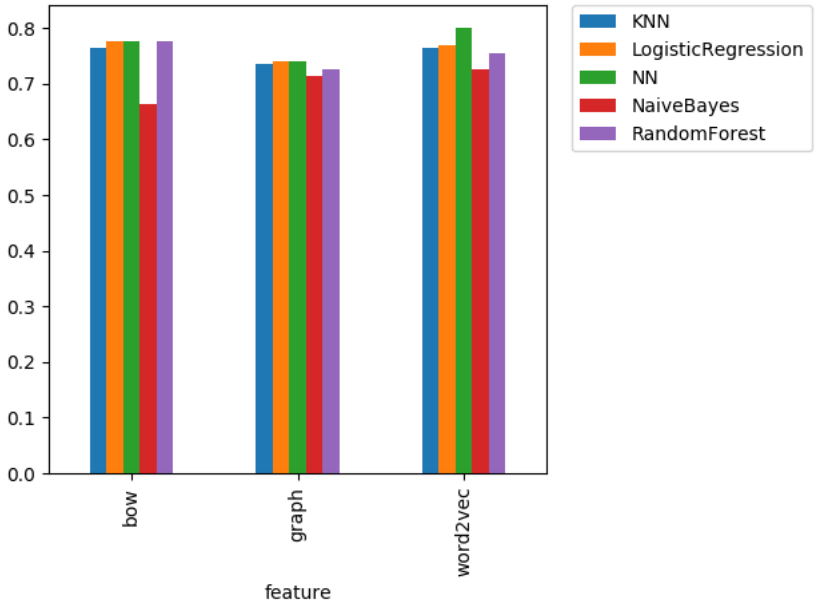

Figure 2: Micro F-Measure results for our best performing features, tested separately

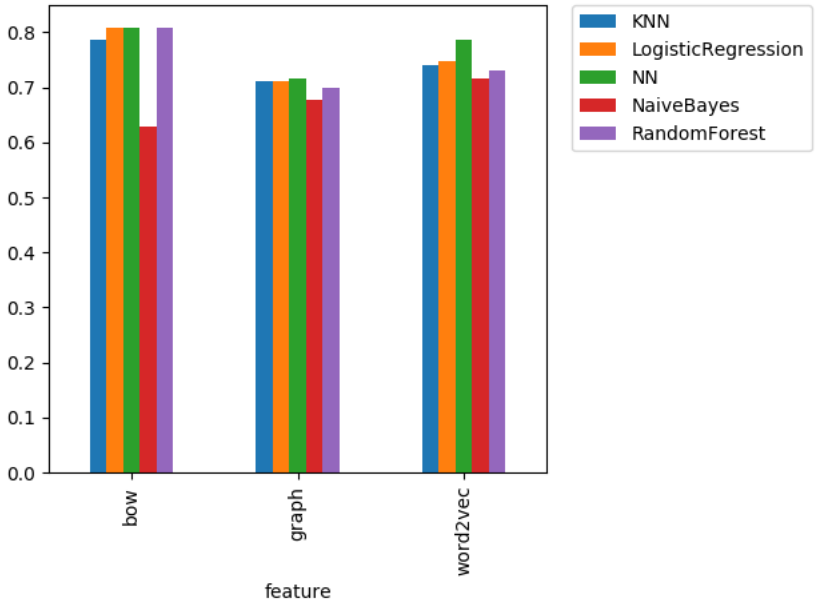

Figure 3: Macro F-Measure results for our best performing features, tested separately

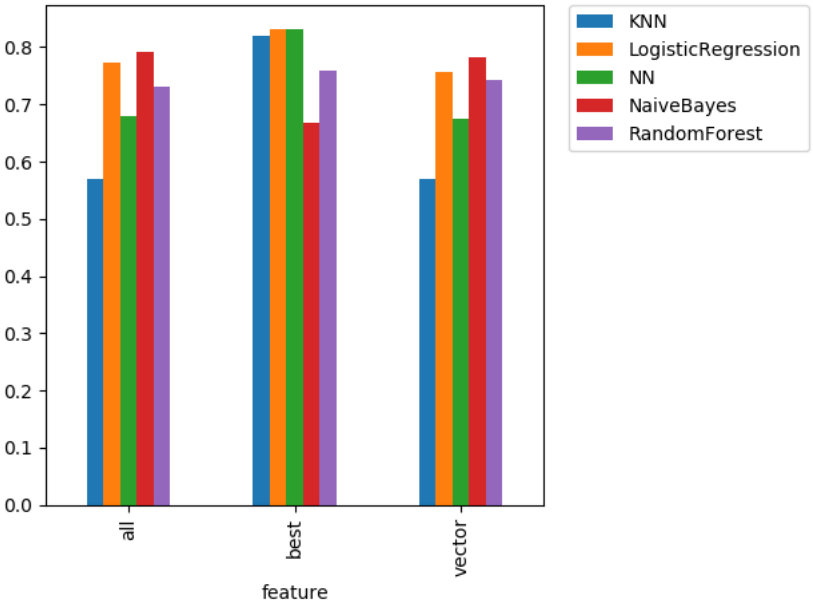

Figure 4: Micro F-Measure results for feature combinations

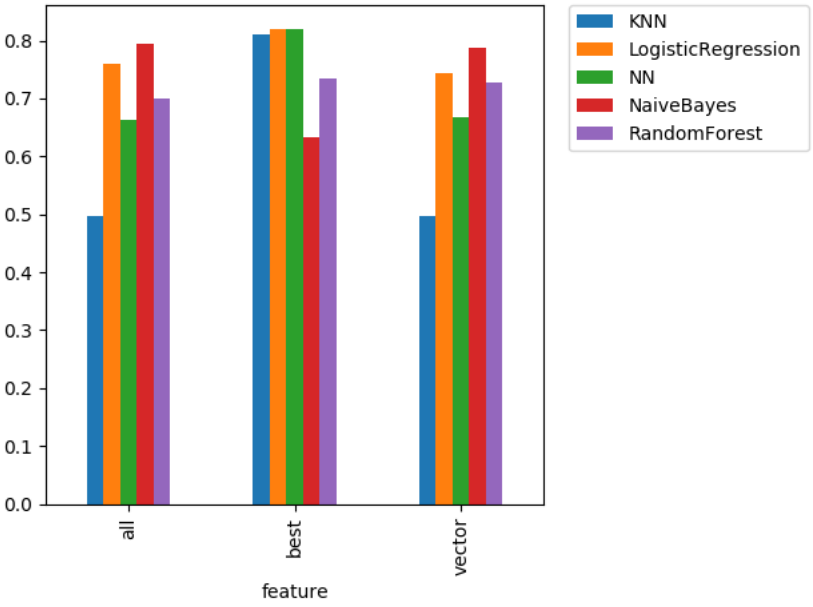

Figure 5: Macro F-Measure results for feature combinations

Table 5: Average micro & macro F-Measure per feature combination setting.

Features	Classifiers	Macro F	Micro F
best	KNN	0.810	0.820
	LR	**0.819**	**0.831**
	NB	0.632	0.667
	NN_keras	0.807	0.819
	NN_sklearn	**0.819**	**0.831**
	RF	0.734	0.759
all	KNN	0.497	0.569
	LR	0.760	0.772
	NB	0.795	0.792
	NN_keras	0.537	0.629
	NN_sklearn	0.664	0.678
	RF	0.700	0.731
vector	KNN	0.497	0.569
	LR	0.745	0.756
	NB	0.787	0.783
	NN_keras	0.592	0.640
	NN_sklearn	0.669	0.675
	RF	0.727	0.742

have provided. Finally, the `vector` combination achieves the worst performance: 0.787 and 0.783 in macro/micro F-measure. This is testament to the added value NGGs contribute to the feature pool, reinforced by the individual scores of the other vector-based approaches.

Sentiment, spelling and syntax features proved to be insufficient information sources to the Hate Speech detection classifiers when used separately – not surprisingly, since they produce one-dimensional features. The results are presented in table 7. The best performers are syntax with NNs in terms of micro F-measure (0.633) and spelling with NNs in terms of macro F-measure (0.566) . In contrast n-gram graph similarity-based features perform close to the best performing BoW configuration (cf. Table 6), having just one additional dimension. This implies that appropriate, deep / rich features can still offer significant information, despite the low dimensionality. NGG-based features appear to have this quality, as illustrated by the results. Finally, N-grams were severely affected by the top-100 token truncation. The best character n-gram model achieves macro/micro F-Measure scores of $0.507/0.603$ with NN classification and the best word n-gram model $0.493/0.627$ with KNN and NN classifiers.

Table 6: Average micro & macro F-Measure for NGG, BoW and word2vec features.

Features	Algorithms	Macro F	Micro F
NGG	KNN	0.712	0.736
	LR	0.712	0.739
	NB	0.678	0.713
	NN_keras	0.718	0.727
	NN_sklearn	0.716	0.740
	RF	0.699	0.726
BoW	KNN	0.787	0.763
	LR	**0.808**	0.776
	NB	0.629	0.665
	NN_keras	**0.808**	0.776
	NN_sklearn	**0.808**	0.776
	RF	0.807	0.776
word2vec	KNN	0.741	0.765
	LR	0.749	0.769
	NB	0.715	0.726
	NN_keras	0.774	0.788
	NN_sklearn	0.786	**0.800**
	RF	0.731	0.755

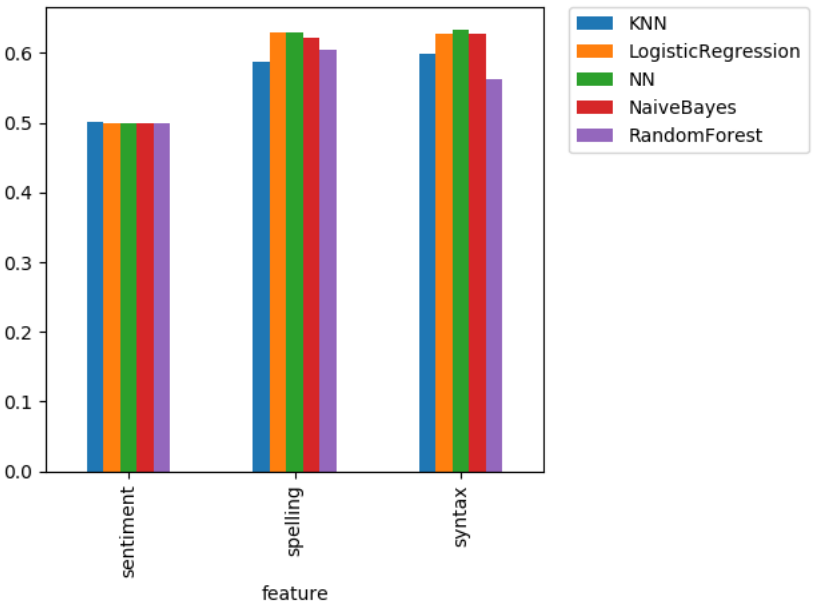

Figure 6: Micro F-Measure results for syntax, sentiment and grammar analysis

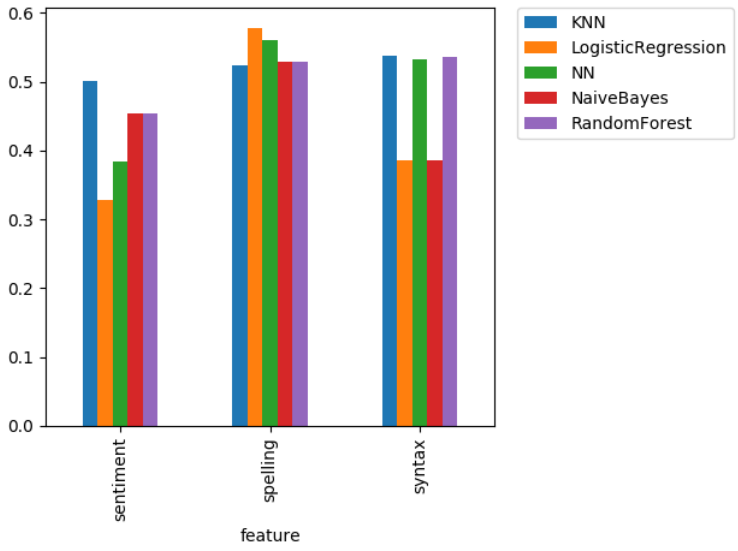

Figure 7: Macro F-Measure results for syntax, sentiment and grammar analysis

Table 7: Average micro and macro F-Measure for sentiment, grammar and syntax analysis in Binary Classification

Features	Algorithms	Macro F	Micro F
sentiment	KNN	0.501	0.501
	LR	0.328	0.500
	NB	0.453	0.500
	NN_keras	0.429	0.500
	NN_sklearn	0.384	0.500
	RF	0.453	0.500
spelling	KNN	0.523	0.587
	LR	0.578	0.629
	NB	0.529	0.621
	NN_keras	0.566	0.630
	NN_sklearn	0.561	0.630
	RF	0.530	0.604
syntax	KNN	0.538	0.599
	LR	0.385	0.627
	NB	0.385	0.627
	NN_keras	0.527	0.633
	NN_sklearn	0.533	0.633
	RF	0.536	0.562

Table 8: Average micro and macro F-Measure for word and character n-grams in Binary Classification

Features	Algorithms	Macro F	Micro F
char n-grams	KNN	0.494	0.567
	LR	0.450	0.596
	NB	0.442	0.591
	NN_keras	0.409	0.603
	NN_sklearn	0.507	0.526
	RF	0.494	0.563
word n-grams	KNN	0.493	0.561
	LR	0.461	0.594
	NB	0.440	0.589
	NN_keras	0.385	0.627
	NN_sklearn	0.488	0.563
	RF	0.490	0.499

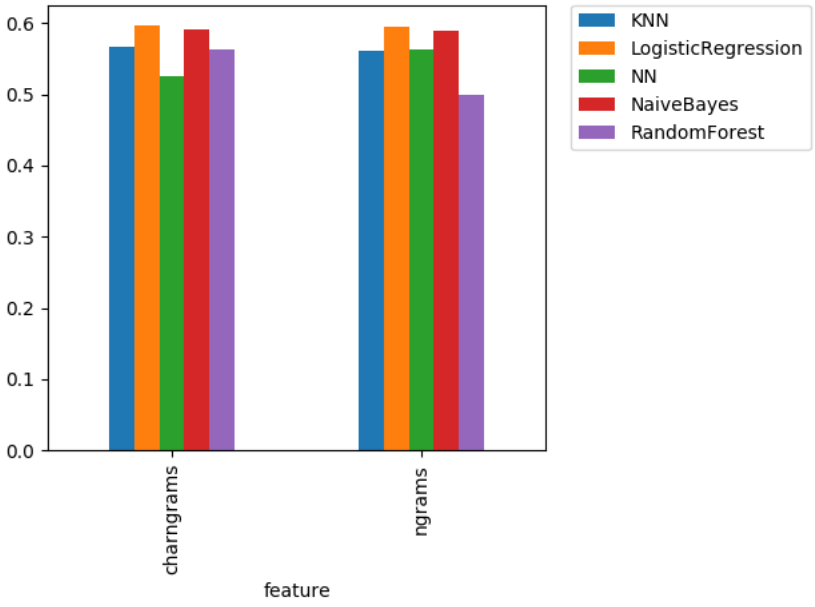

Figure 8: Micro F-Measure results for token and character n-grams

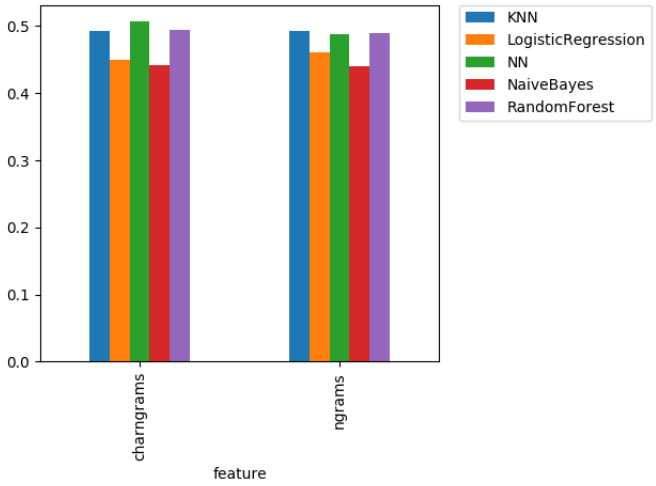

Figure 9: Macro F-Measure results for token and character n-grams

5.2.2 Multi-class classification

Apart from experiments in the binary Hate Speech classification on the combined dataset, we have tested our classification models in multi-class classification, using the original RS and HSOL datasets (results tables available in the appendix). In RS, our best score was achieved with the `all` combination and the RF classifier with a micro F-Measure of 0.696. For the HSOL dataset, we achieved a micro F-Measure of 0.855, using the `best` feature combination and the LR classifier. The experiments that we run have exactly the same setting as in the binary classification task. We tests all our text representations and classification algorithms, described in the Proposed Method chapter. The results in the multi-class classification task are poor, especially with features such as syntax, grammar or sentiment analysis, which are represented by one-dimensional vector each and therefore provide little information to the classifier related to which category an instance belongs to.

5.2.2.1 Results - RS Dataset

In tables 9, 11, 12 and 13, we present results of our experiments using the RS dataset of [32] on the multi-class classification task, for all features, the features in `best`, syntax and grammar - based features and n-grams, respectively. The distinguished classes are `racism`, `sexism` or `clean`.

In this dataset, due to the highly unbalanced dataset (Racist: 1910, Sexist: 3035 and Clean: 10543) the classifier has high accuracy in the majority class, but for the remaining classes cannot predict correctly the relevant instances. Especially, with sentiment, syntax and spelling analysis. Although micro F-Measure has around 0.60 score, the macro F-

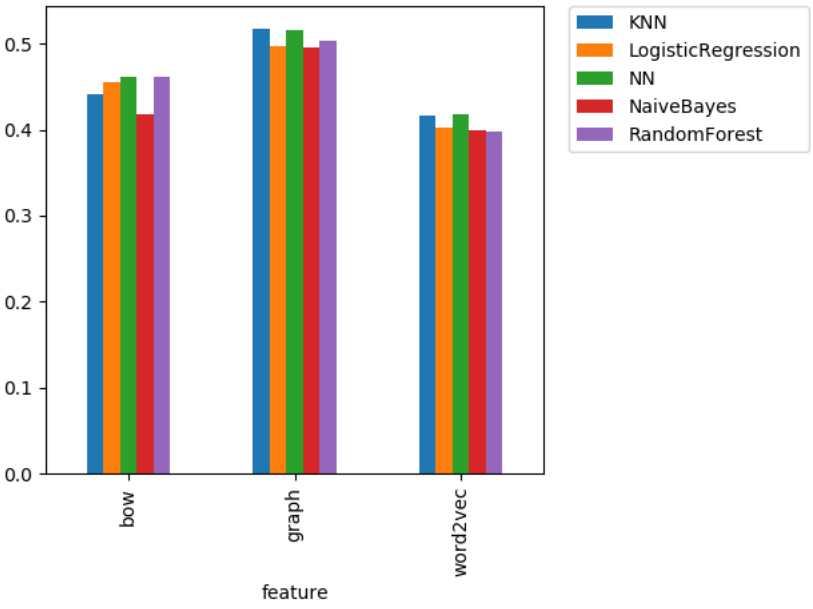

Figure 11: Macro F-Measure results for our best performing features, tested separately

Measure of these feature varies between 0.27 and 0.37.

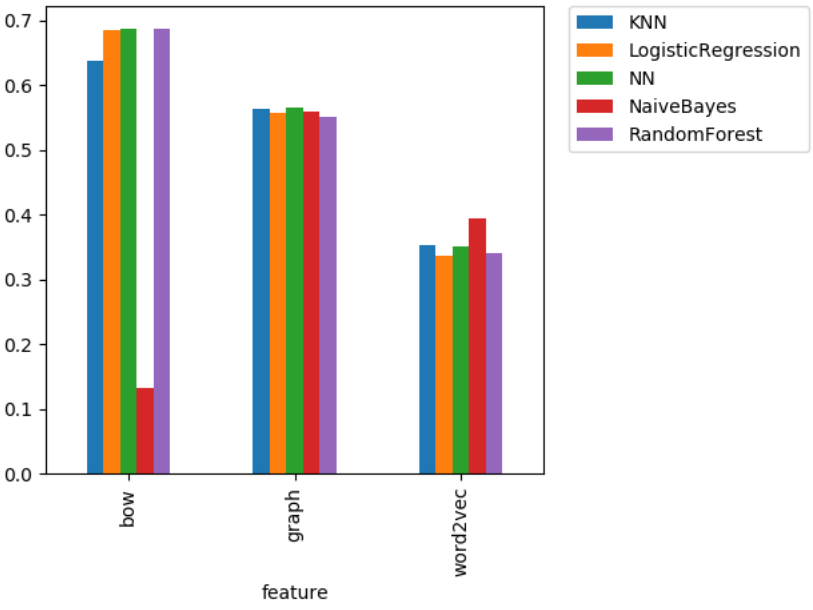

Figure 10: Micro F-Measure results for our best performing features, tested separately

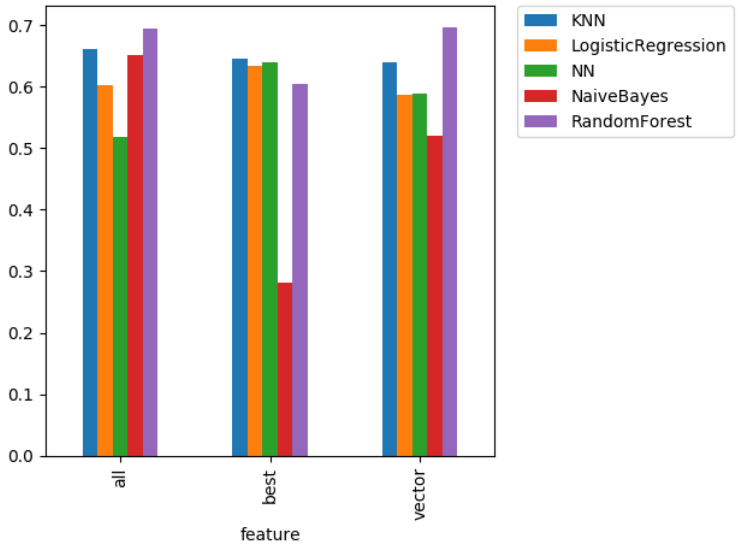

Figure 12: Micro F-Measure results for feature combinations

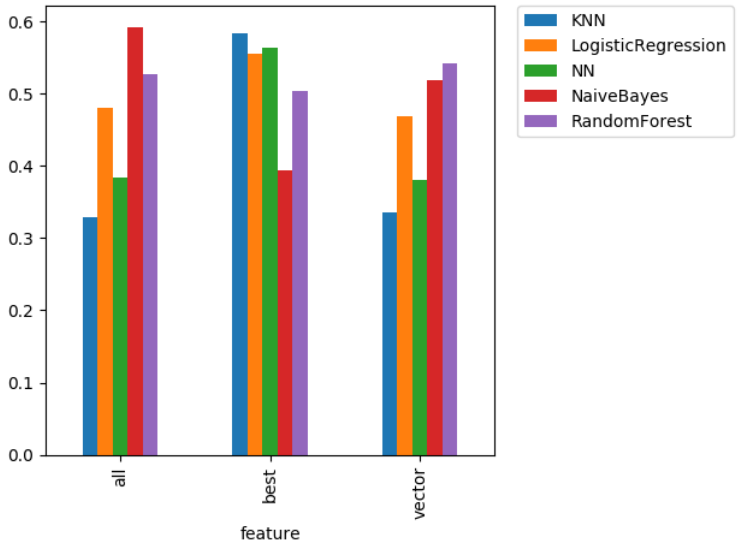

Figure 13: Macro F-Measure results for feature combinations

Table 9: RS dataset: Average micro & macro F-Measure for combination of features

Features	Algorithms	Macro F	Micro F
best	KNN	0.583	**0.647**
	LR	0.555	0.634
	NB	0.394	0.281
	NN_keras	0.546	0.623
	NN_sklearn	0.564	0.640
	RF	0.504	0.605
all	KNN	0.328	0.661
	LR	0.480	0.603
	NB	0.591	0.652
	NN_keras	0.323	0.641
	NN_sklearn	0.383	0.519
	RF	0.526	**0.696**
vector	KNN	0.336	0.639
	LR	0.469	0.587
	NB	0.519	0.520
	NN_keras	0.296	0.680
	NN_sklearn	0.381	0.590
	RF	0.542	**0.696**

In this dataset, BoW seems to perform better than the other features with 0.68 micro F-Measure. Although, sentiment, syntax and grammar analysis also achieve a 0.68 micro F-Measure, the results in macro F-Measure are significantly worse, i.e. 0.28. On the other hand, NGGs micro F-Measure performance may be 0.56, however they scored a 0.52 macro F-Measure.

The above means that sentiment, syntax and grammar features always predicted the majority class as shown in the below example:

Table 10: Confusion matrices: sentiment analysis on the left & NGGs to the right

Confusion Matrices					
1057	0	0	952	84	18
191	0	0	202	101	0
308	0	0	141	2	48

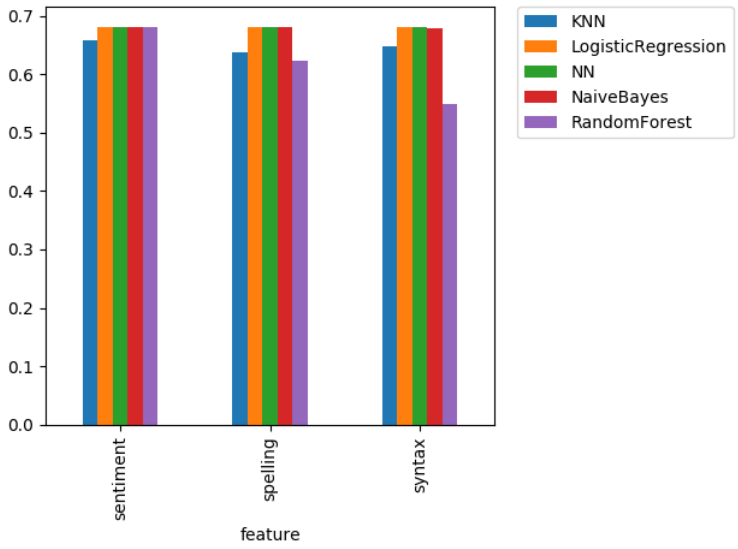

Figure 14: Micro F-Measure results for syntax, sentiment and grammar analysis

Table 11: RS dataset: Average micro & macro F-Measure for our best performing features

Features	Algorithms	Macro F	Micro F
graph	KNN	0.517	0.564
	LR	0.498	0.557
	NB	0.496	0.559
	NN_keras	**0.529**	0.569
	NN_sklearn	0.515	**0.565**
	RF	0.503	0.551
BoW	KNN	0.441	0.637
	LR	0.455	0.686
	NaiveBayes	0.418	0.131
	NN_keras	0.446	0.686
	NN_sklearn	0.461	**0.687**
	RF	0.461	0.686
word2vec	KNN	0.417	0.353
	LR	0.402	0.337
	NB	0.399	0.394
	NN_keras	0.408	0.351
	NN_sklearn	0.418	0.350
	RF	0.398	0.340

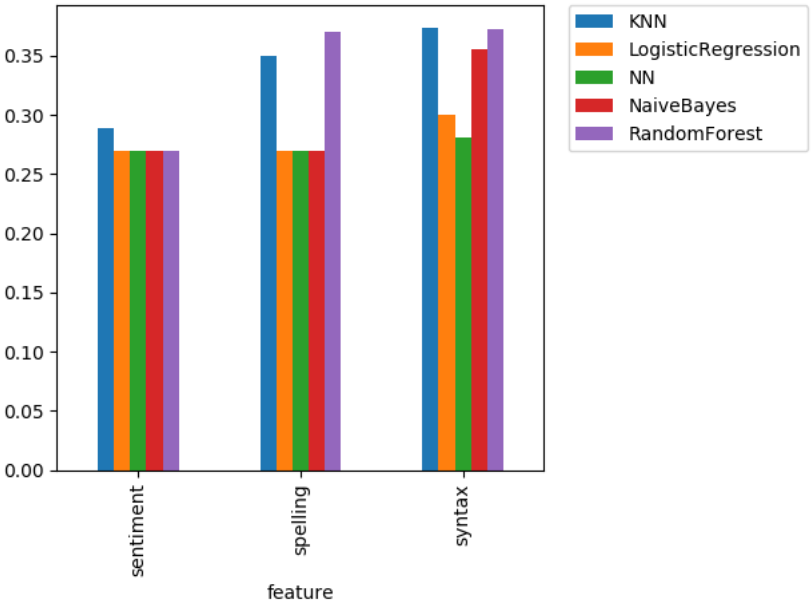

Figure 15: Macro F-Measure results for syntax, sentiment and grammar analysis

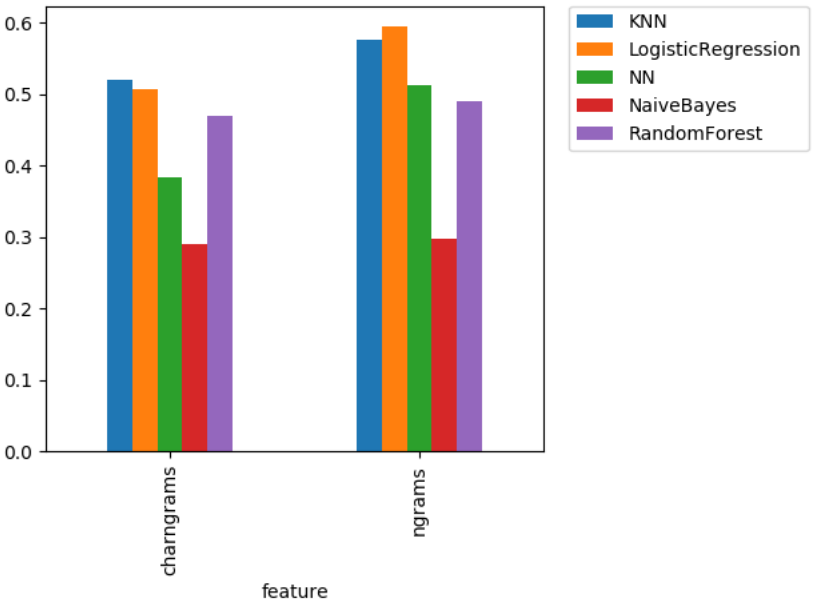

Figure 16: Micro F-Measure results for token and character n-grams

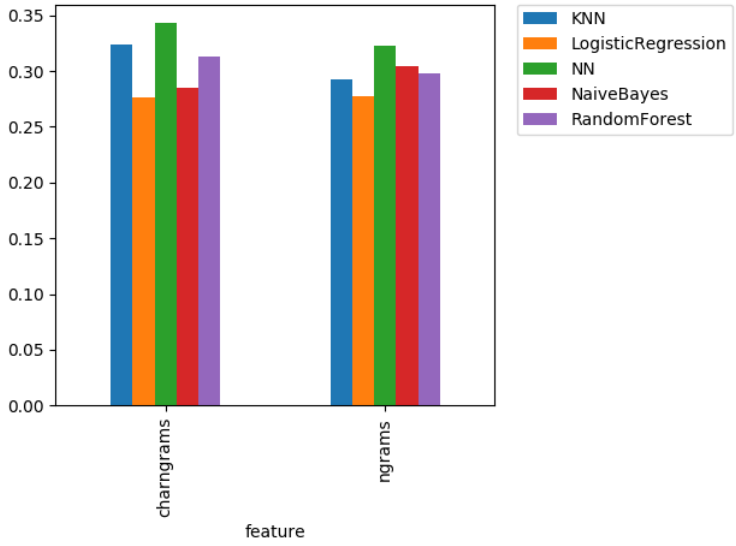

Figure 17: Macro F-Measure results for token and character n-grams

Table 12: RS dataset: Average micro & macro F-Measure for syntax, grammar and sentiment analysis

Features	Algorithms	Macro F	Micro F
sentiment	KNN	0.289	0.659
	LR	0.270	0.681
	NB	0.270	0.681
	NN_keras	0.270	0.681
	NN_sklearn	0.270	0.681
	RF	0.270	0.681
spelling	KNN	0.350	0.638
	LR	0.270	0.681
	NB	0.270	0.681
	NN_keras	0.270	0.681
	NN_sklearn	0.270	0.681
	RF	0.370	0.623
syntax	KNN	0.374	0.648
	LR	0.300	0.681
	NB	0.355	0.679
	NN_keras	0.270	0.681
	NN_sklearn	0.281	0.681
	RF	0.372	0.549

Table 13: RS dataset: Average micro & macro F-Measure for character and word n-grams

Features	Algorithms	Macro F	Micro F
char n-grams	KNN	0.323	0.520
	LR	0.277	0.506
	NB	0.285	0.290
	NN_keras	0.222	0.535
	NN_sklearn	0.343	0.384
	RF	0.313	0.470
word n-grams	KNN	0.293	0.577
	LR	0.277	0.594
	NB	0.304	0.298
	NN_keras	0.285	0.652
	NN_sklearn	0.323	0.513
	RF	0.298	0.490

5.2.2.2 Results - HSOL Dataset

In tables 14, 15, 16 and 17 we present the results of our experiments using the HSOL dataset of [7] for multi-class classification between `Hate Speech`, `Offensive Language` and `Clean`.

In Table 14, the best performance is achieved by the combination of our top performing features, i.e. NGGs, word2vec and BoW with 0.85 micro F-Measure. When combining all features the performance is worse, i.e. 0.80 micro F-Measure, while in the case where we have excluded NGGs the performance is getting even worse with 0.77 micro F-Measure.

In Table 15, we present the results of our experiments when testing our best performing features separately. The best result was achieved by word2vec features with 0.82 micro F-Measure, while BoW and NGGs had 0.81 and 0.80 respectively.

In table 16, we present the results when testing sentiment, syntax and grammar features separately. We notice that the results are similar to the RS dataset results. The low macro F-Measure is explained due to predicting only True Positives and False Negatives for the majority class and therefore such kind of features are not reliable.

This result is normal, since these feature are represented with an one-dimensional vector. In the case of sentiment analysis, the classifier relies only in this one value to classify a tweet. However, having negative sentiment in a post does not mean that it also contains Hate Speech and therefore additional information is required in order to classify the tweet.

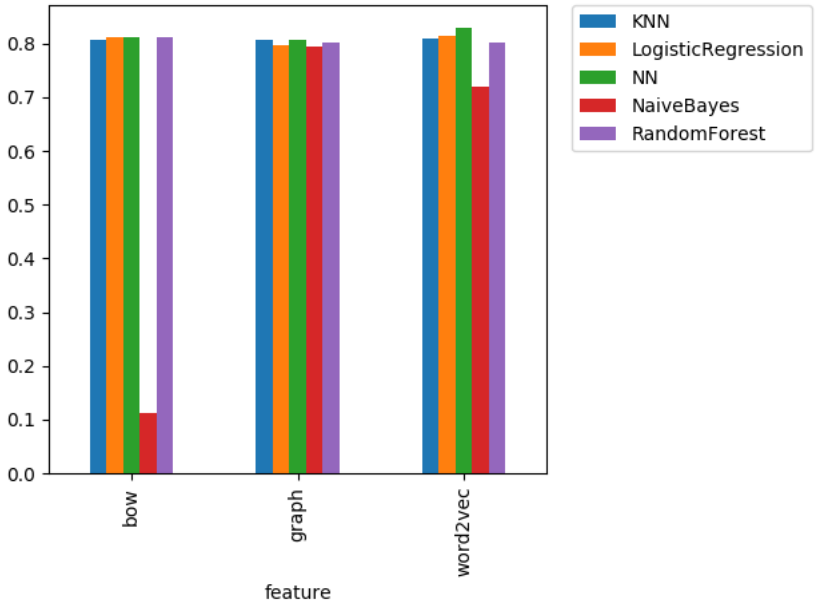

Figure 18: Micro F-Measure results for our best performing features, tested separately

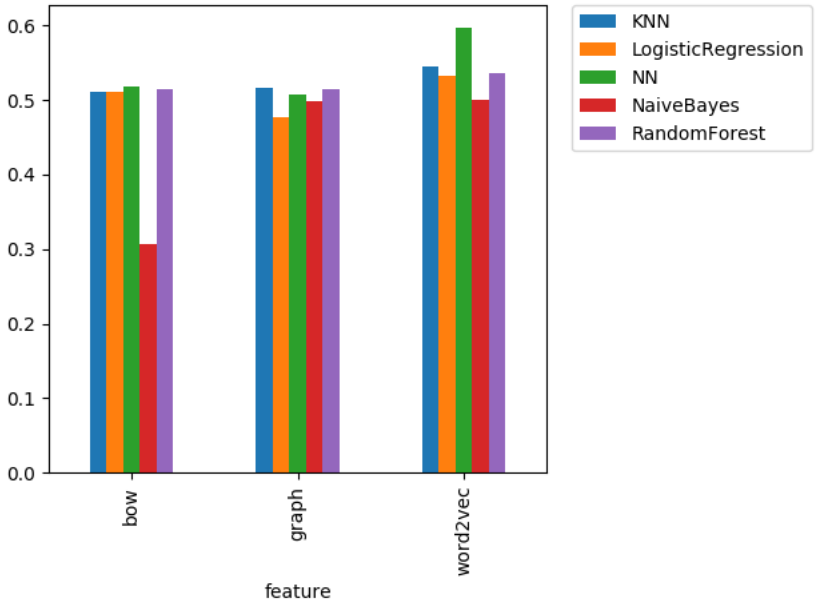

Figure 19: Macro F-Measure results for our best performing features, tested separately

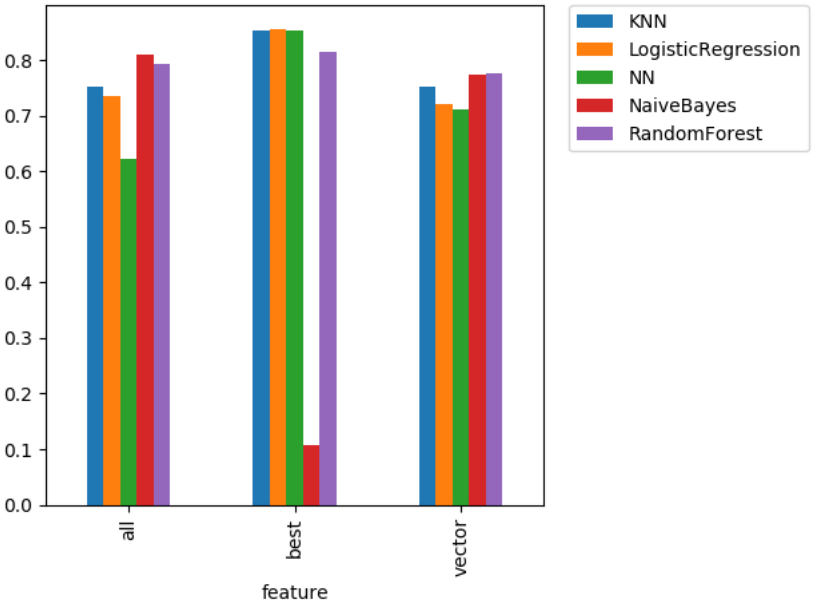

Figure 20: Micro F-Measure results for feature combinations

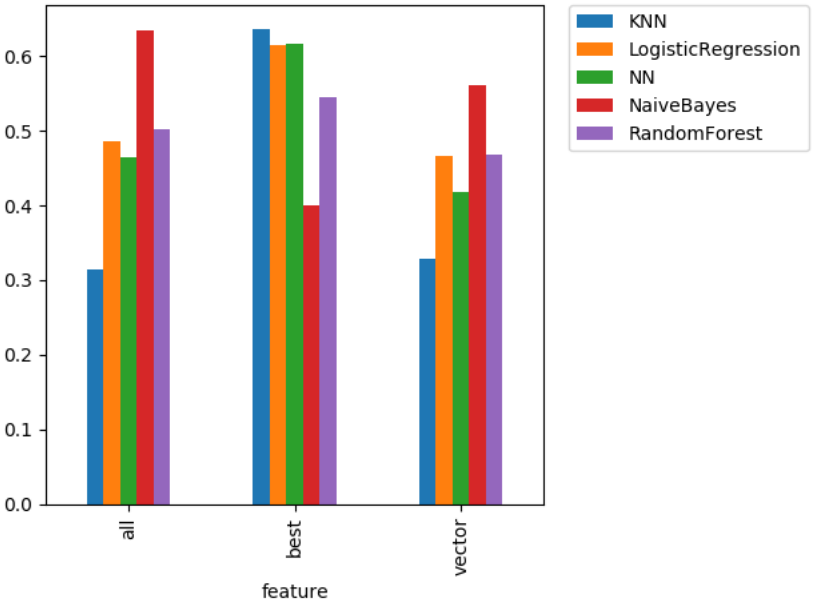

Figure 21: Macro F-Measure results for feature combinations

Table 14: HSOL dataset: Average micro & macro F-Measure for combination of features

Features	Algorithms	Macro F	Micro F
best	KNN	0.636	0.854
	LR	0.615	**0.855**
	NB	0.400	0.107
	NN_keras	0.617	0.848
	NN_sklearn	0.617	0.853
	RF	0.546	0.814
all	KNN	0.314	0.752
	LR	0.486	0.734
	NB	0.634	**0.808**
	NN_keras	0.362	0.771
	NN_sklearn	0.464	0.623
	RF	0.503	0.792
vector	KNN	0.328	0.752
	LR	0.466	0.721
	NB	0.562	0.774
	NN_keras	0.342	0.771
	NN_sklearn	0.419	0.712
	RF	0.468	**0.776**

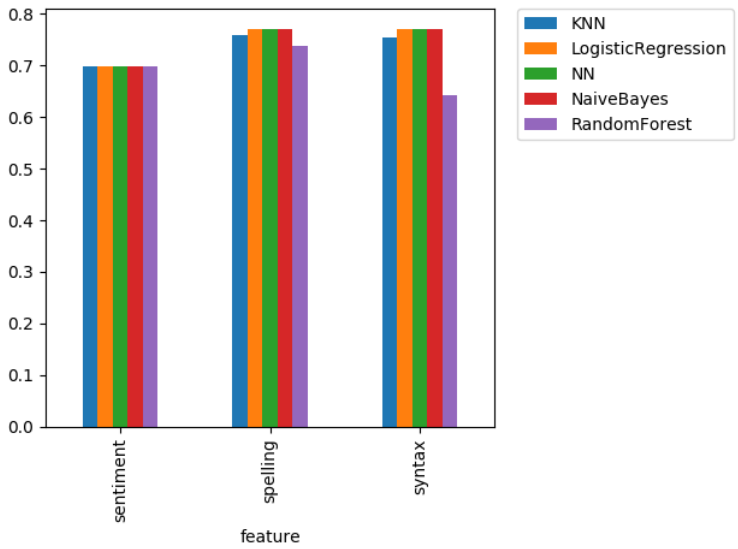

Figure 22: Micro F-Measure results for syntax, sentiment and grammar analysis

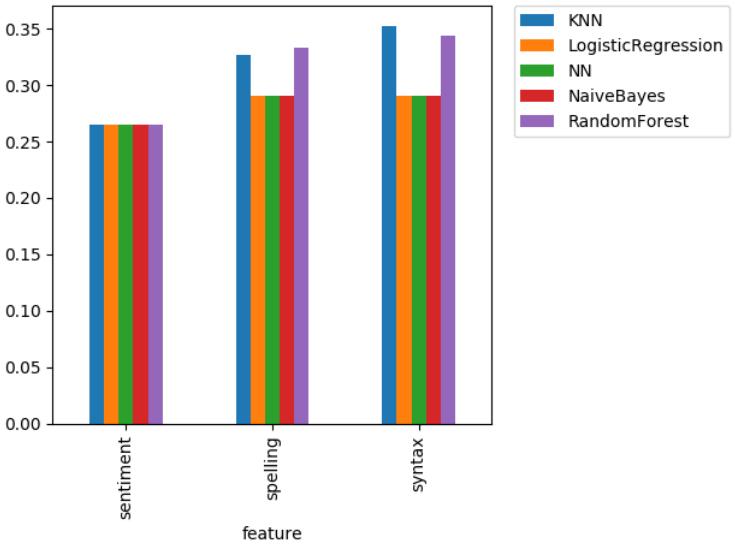

Figure 23: Macro F-Measure results for syntax, sentiment and grammar analysis

Table 15: HSOL dataset: Average micro & macro F-Measure for our best performing features

Features	Algorithms	Macro F	Micro F
graph	KNN	**0.516**	**0.806**
	LR	0.477	0.797
	NB	0.499	0.794
	NN_keras	0.510	0.802
	NN_sklearn	0.507	**0.806**
	RF	0.515	0.802
BoW	KNN	0.511	0.807
	LR	0.512	**0.813**
	NB	0.308	0.112
	NN_keras	0.515	0.811
	NN_sklearn	**0.519**	0.812
	RF	0.515	0.812
word2vec	KNN	0.545	0.810
	LR	0.532	0.813
	NB	0.500	0.720
	NN_keras	0.590	0.819
	NN_sklearn	**0.597**	**0.829**
	RF	0.535	0.801

Table 16: HSOL dataset: Average micro & macro F-Measure for sentiment, syntax and grammar analysis

Features	Algorithms	Macro F	Micro F
sentiment	KNN	0.265	0.699
	LR	0.265	0.699
	NB	0.265	0.699
	NN_keras	0.265	0.699
	NN_sklearn	0.265	0.699
	RF	0.265	0.699
spelling	KNN	0.327	0.760
	LR	0.290	0.771
	NB	0.290	0.771
	NN_keras	0.290	0.771
	NN_sklearn	0.290	0.771
	RF	0.334	0.738
syntax	KNN	0.352	0.755
	LR	0.290	0.771
	NB	0.290	0.771
	NN_keras	0.290	0.771
	NN_sklearn	0.290	0.771
	RF	0.344	0.642

Table 17: HSOL dataset: Average micro & macro F-Measure for character and word n-grams

Features	Algorithms	Macro F	Micro F
char n-grams	KNN	0.317	0.753
	LR	0.290	0.733
	NB	0.300	0.266
	NN_keras	0.290	0.771
	NN_sklearn	0.342	0.518
	RF	0.309	0.648
word n-grams	KNN	0.299	0.696
	LR	0.296	0.709
	NB	0.298	0.264
	NN_keras	0.290	0.771
	NN_sklearn	0.321	0.533
	RF	0.300	0.626

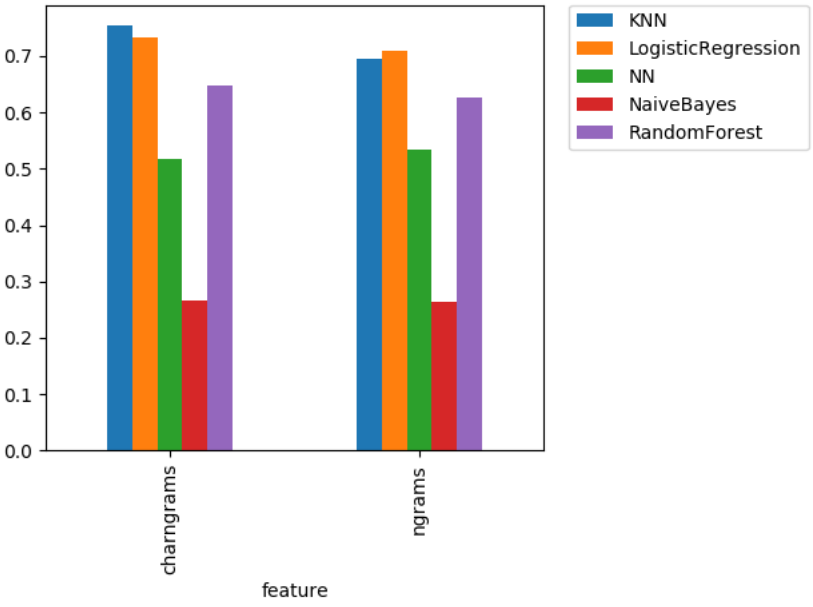

Figure 24: Micro F-Measure results for token and character n-grams

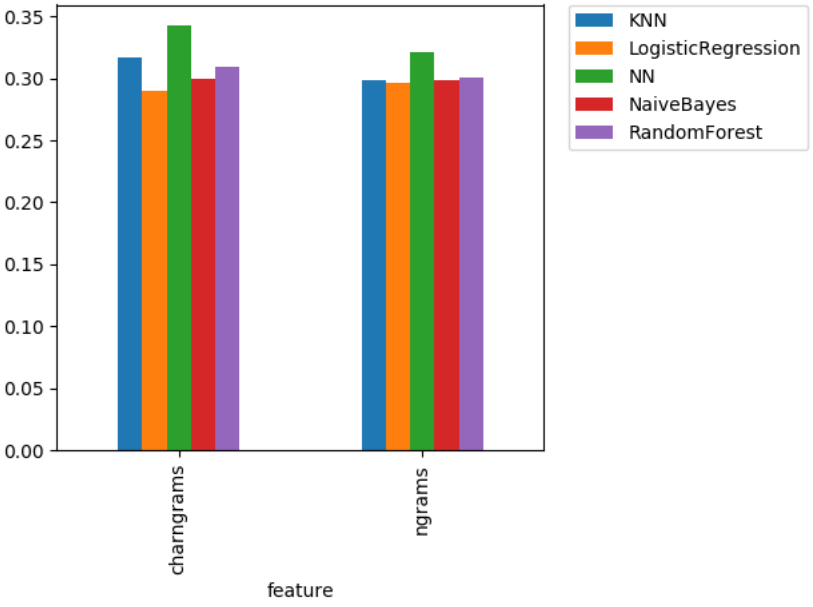

Figure 25: Macro F-Measure results for token and character n-grams

5.3 Significance testing

5.3.1 Binary Classification

In table 18 we present ANOVA results with respect to feature extractors and classifiers, under macro and micro F-measure scores. We can see that, for both metrics, the selection of both features and classifiers is statistically significant with a confidence level greater than 99.9%.

We continue by performing a set of Tukey's Honest Significance Difference test experiments in table 19. In the upper part we present results between feature combination groups, where the `best` combination is significantly different by the similar `all` and `vector` combinations by a large margin, as expected. The middle part compares individual features, where word2vec, BoW and NGGs are assigned to neighbouring groups and arise the most significant features, with the other approaches having a large significance margin from them. Spelling and syntax features are grouped together, as well as the n-gram approaches.

Finally, the lower part of the table examines classification algorithms. There, LR leads the ranking, followed by two significance groups; one with the ANN implementations, followed a group with NB and RF.

In table 20, we present the Tukey's test results in terms of macro F-Measure for our binary classification task. The results are similar to the micro F-Measure, with the `best` feature combination, BoW and word2vec have the most significant contribution and NGGs being slightly behind. ANNs in the scikit-learn implementation have the best result, followed by RF, KNN and LR.

The results and statistical tests on our work showcase the BoW, word2vec embeddings and the NGG model as the top performing feature-related configurations. BoW and word2vec score best in terms of micro and macro F-measure respectively, with NGG close behind despite the extreme dimensionality reduction incurred by the model vector representation of graph similarities. We remind the reader that the BoW encoding has built upon the HateBase word list, which embeds domain knowledge. The NGGs do not use such background knowledge, which is an added value. The combination of the top performing features improves the results over individual ones, with 0.831 micro F-Measure when employed on an LR classifier or NN with the scikit-learn implementation.

Regarding classification methods, the LR and ANN classifiers perform best when used with our top performing features (separately or combined). Statistical tests show that in both micro and macro F-Measure terms, both representation and classification approaches have a significant role in the performance results.

Finally, we understand from our study that the contribution of NGGs as a text representation is significant. They have a slightly worse performance from BoW — backed with background knowledge — and word embeddings — encoding significant information from a variety of corpora. The vector dimension of the NGG-based approach is equal to the

Table 18: ANOVA results with repect to feature and classifier selection, in terms of macro F-measure (top) and micro-Fmeasure (bottom) for the unified dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
features	10	11.082	1.108	193.188	< **2e-16**
classifiers	5	0.167	0.033	5.837	**2.77e-05**
Residuals	644	3.694	0.005		
features	10	5.319	0.531	148.564	< **2e-16**
classifiers	5	0.152	0.030	8.483	**8.65e-08**
Residuals	644	2.306	0.003		

Table 19: Tukey's HSD group test on micro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the unified dataset

config	microf	groups
best	0.787	a
all	0.695	cd
vector	0.693	d
word2vec	0.767	a
BoW	0.755	ab
NGG	0.730	bc
spelling	0.617	e
syntax	0.613	e
c-ngrams	0.574	f
w-ngrams	0.572	f
sentiment	0.500	g
LR	0.689	a
NN_keras	0.670	ab
NN_sklearn	0.668	ab
NB	0.661	bc
RF	0.655	bc
KNN	0.639	c

Table 20: Tukey's HSD group test on macro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the unified dataset

config	microf	groups
best	0.770	a
all	0.658	d
vector	0.669	cd
word2vec	0.749	ab
BoW	0.774	a
NGG	0.705	bc
spelling	0.547	e
syntax	0.483	f
c-ngrams	0.465	fg
w-ngrams	0.459	fg
sentiment	0.425	g
LR	0.617	abc
NN_keras	0.595	c
NN_sklearn	0.630	a
NB	0.589	c
RF	0.627	ab
KNN	0.599	bc

number of classes. On the other hand, the BoW and embeddings models produce vectors of 1000 and 50 dimensions respectively. Empirical evaluation shows that NGGs have a significant contribution, despite the low dimensionality of the generated feature vectors. Therefore NGGs can be seen as deep features, encapsulating rich information in a low dimensional representation, which helps achieve significant performance even when used by itself.

5.3.2 Multi-class Classification

5.3.2.1 RS Dataset

Table 21 shows the ANOVA test results in terms of micro and macro F-Measure for the RS dataset. In this classification task, both features and algorithms have significant role to the result, in both micro and macro F-Measure terms.

Tables 22 and 23 show the Tukey's test results for micro and macro F-Measure respectively. Regarding to the classification algorithms, there are not significant differences. In micro F-Measure results, only NB performs worse while the remaining algorithms seem to have similar contribution, while in the macro F-Measure results, NB and RF have the most significant contribution to the produced result.

Table 21: ANOVA results with repect to feature and classifier selection, in terms of macro F-measure (top) and micro-Fmeasure (bottom) for RS dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
features	10	4.980	0.4980	52.159	< **2e-16**
classifiers	5	0.224	0.0447	4.685	**0.000329**
Residuals	644	6.149	0.0095		
features	10	5.789	0.5789	21.67	< **2e-16**
classifiers	5	1.491	0.2983	11.16	**2.48e-10**
Residuals	644	17.208	0.0267		

Table 22: Tukey's HSD group test on micro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the RS dataset

config	microf	groups
best	0.571	bcd
all	0.628	abc
vector	0.618	abc
word2vec	0.354	f
BoW	0.585	abcd
NGG	0.560	cd
spelling	0.663	ab
syntax	0.653	abc
c-ngrams	0.450	e
w-ngrams	0.520	de
sentiment	0.677	a
LR	0.595	a
NN_keras	0.616	a
NN_sklearn	0.571	a
NB	0.469	b
RF	0.580	a
KNN	0.594	a

Table 23: Tukey's HSD group test on macro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the RS dataset

config	microf	groups
best	0.524	a
all	0.438	b
vector	0.423	b
word2vec	0.406	b
BoW	0.447	b
NGG	0.509	a
spelling	0.300	c
syntax	0.325	c
c-ngrams	0.293	c
w-ngrams	0.296	c
sentiment	0.273	c
LR	0.386	ab
NN_keras	0.351	b
NN_sklearn	0.382	ab
NB	0.391	a
RF	0.414	a
KNN	0.386	ab

One interesting result related to the features is the difference in micro and macro F-Measure for the syntax, spelling and sentiment analysis. In 22 they appear to be some of the features with the most significant contribution, while in 23 they have low score along with the token and character bag of n-grams. This is something that we expected, since these features always predicted the majority class. Therefore, in micro F-Measure they seem to have good prediction score (since the number of instances per class is included in the calculation), while in macro F-Measure results they have the less contribution (since macro F-Measure calculates the correct predictions per class by average).

From the remaining features, BoW, NGGs and their combination with word2vec have important contribution in the multi-class classification task. This can be assumed both from the micro and the macro F-Measure results, which means that these features do not predict only the majority class for all the tested instances.

5.3.2.2 HSOL Dataset

In table 24, the ANOVA test results are presented. The top part of the table shows the results in terms of macro F-Measure while the bottom part of the table presents the micro F-Measure results. In micro F-Measure terms, both features and classifiers have significant contribution to the produced result, while in macro F-Measure results, it seems that

Table 24: ANOVA results with repect to feature and classifier selection, in terms of macro F-measure (top) and micro-Fmeasure (bottom) for the HSOL dataset

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
features	10	7.631	0.7631	126.839	< **2e-16**
classifiers	5	0.078	0.0156	2.586	**0.025**
Residuals	644	3.875	0.0060		
features	10	2.615	0.2615	12.88	< **2e-16**
classifiers	5	4.006	0.8012	39.45	< **2e-16**
Residuals	644	13.079	0.0203		

features have more important contribution than the classification algorithm used.

Table 25 shows the results of Tukey's test on micro F-Measure for the HSOL dataset. The most significant features are the NGGs and word2vec followed by the tested feature combinations. Here we can notice that sentiment, syntax and spelling features seems to be more significant than BoW in terms of micro F-Measure. However, in the following table related to macro F-Measure, it is obvious that these are the less significant features. This happens since these features always predict the majority class, no matter the real label. Therefore, in micro F-Measure, where the number of instances per class are taken into account, they have similar score to the best performing features, but in macro F-Measure, where the average performance is calculated, they have low score.

Related to the classification algorithms, the top performing are ANNs in Keras implementation, LR, RF and KNN followed by the scikit-learn implementation of ANNs. Classifiers seems to have differences in terms of contribution when tested in micro F-Measure. On the other hand, in macro F-Measure score, they seem to have similar contribution to the classification result.

In table 26, we present the macro F-Measure results on Tukey's test for the HSOL dataset. Regarding to the classification algorithms, we can assume that all techniques had similar results in the classification task and therefore none had significant contribution. On the other hand, significance varies when it comes to the features used. Although all results are poor in terms of performance, the most significant contribution is achieved through the combination of our best performing features, followed by their separate use. This shows that the use of NGGs in the multi-class classification task has significant contribution, even when compared with features such as BoW which is represented by a 1000 dimension vector and also has knowledge on the task provided by the HateBase dictionary.

Table 25: Tukey's HSD group test on micro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the HSOL dataset

config	microf	groups
best	0.721	ab
all	0.746	ab
vector	0.750	ab
word2vec	0.798	a
BoW	0.694	bc
NGG	0.801	a
spelling	0.763	ab
syntax	0.746	ab
c-ngrams	0.615	cd
w-ngrams	0.599	d
sentiment	0.699	b
LR	0.765	ab
NN_keras	0.782	a
NN_sklearn	0.720	b
NB	0.553	c
RF	0.740	ab
KNN	0.767	ab

Table 26: Tukey's HSD group test on macro F-Measure between feature combination groups (top), individual features (middle) and classifiers (bottom) for the HSOL dataset

config	microf	groups
best	0.571	a
all	0.460	cd
vector	0.430	d
word2vec	0.549	ab
BoW	0.479	c
NGG	0.504	bc
spelling	0.303	e
syntax	0.309	e
c-ngrams	0.308	e
w-ngrams	0.300	e
sentiment	0.264	e
LR	0.410	a
NN_keras	0.396	a
NN_sklearn	0.420	a
NB	0.394	c
RF	0.421	a
KNN	0.400	a

6. CONCLUSION AND FUTURE WORK

In this study, we reviewed the reasons to deal with Hate Speech phenomenon. The increase of hatred against minorities in modern societies, accompanied with the massive use of Internet and social media, led to uncontrollable spread of Hate Speech through online platforms. The large amount of data shared through the Internet, made the human factor in detecting Hate Speech extremely ineffective. Therefore, automatic Hate Speech detection tools are necessary to Social Media platforms, which wish to eliminate hate content.

In order to implement this automatic tool, we tested a variety of text representation techniques and classification algorithms, performing a large number of experimental evaluations on the Hate Speech detection problem both in binary and multi-class classification tasks. Through these experiments, we investigated which are the best performing features and algorithms and whether feature combinations can achieve better results.

Results showed that BoW, word2vec and NGGs had an overall better performance than the remaining features. Moreover, their combination (i.e. `best` features in the results) has similar or sometimes better performance than testing them separately. Related to the classifiers, an overall better performance is achieved by using ANNs and Logistic Regression. Especially, when these classifiers are combined with our best performing features, had the best performance.

Additionally, we showed that n-gram graph-based features constitute deep/rich features, with significant contribution to the Hate Speech classification results. Compared with shallow features (e.g. Bow) in terms of performance and significance, through ANOVA and Tukey's tests, showed that NGGs have significant contribution to the classification result, even when tested in multi-class classification tasks (e.g. in HSOL dataset).

In the future, we aim to better evaluate the contribution of word roles (e.g. POS tags) and combine them with improved preprocessing, to avoid possible noise in the related features. Concerning NGGs in Hate Speech detection, we want to apply the findings from the work of [29] on NGG variations, to represent short texts with only the important n-grams of the text (e.g. through a TF-IDF filtering process and/or a named entity recognizer). The aim is to reduce the complexity and size of the NGGs, while retaining all the useful information. Additionally, we aim to use word NGGs and compare their performance with the character NGGs. Another avenue of research, is the enrichment of deep features with statistical pre-trained models (such as Latent Dirichlet Allocation) or semantic information (e.g. from thesauri) to further improve performance. Finally, another improvement would be the use of transfer learning techniques in order to fine-tune the word embedding features.

REFERENCES

[1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee, 2017. 31, 32, 34, 35, 37, 38, 39, 40, 47

[2] Anat Ben-David and Ariadna Matamoros-Fernandez. Hate speech and covert discrimination on social media: monitoring the facebook pages of extreme-right political parties in spain. *International Journal of Communication*, 10:1167–1193, 2016. 38

[3] Peter Bourgonje, Julian Moreno-Schneider, Ankit Srivastava, and Georg Rehm. Automatic classification of abusive language and personal attacks in various forms of online communication. In *International Conference of the German Society for Computational Linguistics and Language Technology*, pages 180–191. Springer, 2017. 31, 32, 34, 35, 36, 38

[4] Alexander Brown. What is hate speech? part 1: The myth of hate. *Law and Philosophy*, 36(4):419–468, 2017. 28

[5] Teresa Correa, Amber Willard Hinsley, and Homero Gil De Zuniga. Who interacts on the web?: The intersection of users' personality and social media use. *Computers in Human Behavior*, 26(2):247–253, 2010. 23

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 34

[7] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*, 2017. 31, 32, 34, 35, 36, 38, 39, 44, 47, 53, 68

[8] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. 2017. 32, 34, 35, 37

[9] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web*, pages 29–30. ACM, 2015. 32, 34, 35, 38

[10] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: Small sample performance. Technical report, CALIFORNIA UNIV BERKELEY, 1952. 50

[11] Natalia S Gavrilova, Victoria G Semyonova, Galina N Evdokushkina, and Leonid A Gavrilov. The response of violent mortality to economic crisis in russia. *Population Research and Policy Review*, 19(5):397–419, 2000. 24

[12] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, 2015. 32, 35, 37, 38

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 37

[14] Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In *AAAI*, 2013. 28, 31, 32, 35, 38

[15] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002. 49, 50

[16] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989. 31

[17] Scott W Menard. *Applied logistic regression analysis*. Number 04; e-book. 1995. 49

[18] Hamdy Mubarak, Kareem Darwish, and Walid Magdy. Abusive language detection on arabic social media. In *Proceedings of the First Workshop on Abusive Language Online*, pages 52–56, 2017. 31, 32, 38

[19] 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. 31, 32, 38

[20] George Papadakis, George Giannakopoulos, and Georgios Paliouras. Graph vs. bag representation models for the topic classification of web documents. *World Wide Web*, 19(5):887–920, 2016. 41

[21] Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer, 2010. 32, 35, 36, 38

[22] Claire M Renzetti. Economic stress and domestic violence. 2009. 24

[23] Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*, 2017. 38

[24] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80, 1995. 48

[25] Haji Mohammad Saleem, Kelly P Dillon, Susan Benesch, and Derek Ruths. A web of hate: Tackling hateful speech in online social spaces. *arXiv preprint arXiv:1709.10159*, 2017. 32, 34, 35, 37, 38, 44

[26] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, 2017. 25, 38

[27] Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabricio Benevenuto, and Ingmar Weber. Analyzing the targets of hate in online social media. In *ICWSM*, pages 687–690, 2016. 32, 38

[28] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009. 42

[29] Leonidas Tsekouras, Iraklis Varlamis, and George Giannakopoulos. A graph-based text similarity measure that employs named entity information. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 765–771, 2017. 83

[30] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics, 2012. 31, 32, 34, 35, 38

[31] Zeerak Waseem. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on NLP and computational social science*, pages 138–142, 2016. 28, 31, 32, 38, 39, 47

[32] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93, 2016. 28, 32, 35, 37, 38, 40, 53, 61

[33] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the*

21st ACM international conference on Information and knowledge management, pages 1980–1984. ACM, 2012. 32, 34, 35, 36, 37, 38

[34] Zhi Xu and Sencun Zhu. Filtering offensive language in online communities using grammatical relations. In *Proceedings of the Seventh Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pages 1–10, 2010. 31, 32, 38