Universität Zürich[UZH]

Masterarbeit
zur Erlangung des akademischen Grades
Master of Arts
der Philosophischen Fakultät der Universität Zürich

Data Augmentation in Deep Learning for Hate Speech Detection in Lower Resource Settings

Verfasserin: Michaela Benk
Matrikel-Nr: 15-710-973

Referent: Prof. Dr. Martin Volk

Betreuer: Dr. Simon Clematide

Institut für Computerlinguistik

Abgabedatum: 15.06.2019

Abstract

With the increase in online communication within the last decade, the need to automatize the detection and removal of hateful content has become ever more present. However, because of data scarcity especially in languages other than English, current state of the art neural approaches are limited in their ability to predict hate speech efficiently. To address this problem, this project focuses on applying several data augmentation techniques on German and Italian data, for both multi-class and binary classification. We experiment with a hybrid CNN/RNN architecture, as well as two classical, statistical machine learning models. Our findings show that augmentation can improve hate speech detection to varying degrees. For German multi-class classification, our models could achieve an absolute increase in overall macro F1-score of up to 8.5% and ∼78% in recall of the least represented class. For Italian binary classification, we observe an absolute improvement of up to ∼14% in precision for detecting the hate class and 6.8% in macro F1-score.

Zusammenfassung

Mit der Zunahme der Online-Kommunikation in den letzten Jahren ist die Notwendigkeit, das Erkennen und Entfernen von hasserfüllten Inhalten zu automatisieren, immer größer geworden. Aufgrund der Datenknappheit, insbesondere in anderen Sprachen als Englisch, sind die derzeitigen neuronalen Ansätze in ihrer Fähigkeit, Hasssprache effizient vorherzusagen, beschränkt. Um dieses Problem zu lösen, fokussiert dieses Projekt auf die Anwendung mehrerer Datenaugmentationmethoden auf deutschen und italienischen Daten, sowohl für die Klassifizierung mehrerer Klassen, als auch für die Binärklassifizierung. Wir experimentieren zu dem Zweck mit einer hybriden CNN / RNN-Architektur und mit zwei klassischen, statistischen Modellen des maschinellen Lernens. Unsere Ergebnisse zeigen, dass eine Verbesserung der Hasserkennung dank Datenaugmentation möglich ist. In der deutschen Mehrklassenklassifizierung erzielen unsere Modelle eine Steigerung des Makro-F1-Scores von bis zu 8,5%, sowie eine verbesserte absolute Trefferquote bei der Erkennung der am wenigsten representierten Klasse von bis zu ∼78%. Bei der italienischen Klassifizierung erreichen wir eine absolute Verbesserung der Trefferquote der Erkennung der Hassklasse von bis zu ∼14% und eine Steigerung des Makro-F1-Scores von bis zu 6,8%.

Acknowledgement

I would like extend my deepest gratitude to all those who have been there to support me on a personal, professional, and academic level throughout the last years.

Above all, I would like to sincerely thank my supervisor, Simon Clematide, for introducing me to the field of Machine Learning, for his guidance, dedication, invaluable feedback and the occasional nudge, which helped me stay on track and focused.

I would also like to thank my examiner, Martin Volk, whose enthusiasm for the field and dedication to his students motivated and inspired me throughout my studies.

Furthermore, I would like to thank the entire Vladimir's family for their invaluable friendship and support over the last years, without which I would not be where I am today; Roxana Porada, whom I could always count on for advice and moral support; and my parents and brother, for their support, their love and their belief in my success.

Lastly I would like to extend my deepest appreciation to Diego Antognini, for the many invaluable discussions, brainstorming sessions, his patience and continuous flow of moral support, encouragement, as well as his enthusiasm and dedication, which has been the biggest source of motivation and inspiration for me.

Ethical Disclaimer

The topic of Hate Speech requires us to work with and display language, which is mildly to severely offensive. We would like to emphasize that we choose examples at random and solely to illustrate and explain our findings. We do not wish to offend the reader in any way. With this work, we hope to contribute to the task of hate speech detection and consequently to a decrease in the use of hateful language online.

Contents

List of Figures

List of Tables

List of Acronyms

Adam	Adaptive Moment Estimation
CNN	Convolutional Neural Network
GRU	Gated Recurrent Unit
LSTM	Long-Short Term Memory
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
POS	Part-Of-Speech
RMSprop	Root Mean Square Propagation
RNN	Recurrent neural network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
XGBoost	eXtreme Gradient Boosting

1 Introduction

1.1 Motivation

Within the past decade, the internet has become a central means of communication, through channels such as social media, blogs and online forums, allowing for world-wide communication and knowledge-exchange within seconds. While online communication is a tremendous achievement of modern technology, it bears with it problematic consequences: "real-life" problems are transferred to the web and amplified through anonymity and lack of proper mechanisms for legal responsibility and regulation. One such aspect includes the use of verbal aggression in form of insulting and offensive comments, commonly referred to as "hate speech".
As mentioned in a paper by the Wikipedia Foundation (Wulczyn et al., 2017), a 2014 Pew Report found that 73% of adult internet users "have seen someone harassed online, and 40% have personally experienced it." Since manual moderation is not able to handle the flood of offensive language, there is an increasing need for automated methods to reliably identify such "toxic" content.
This task is not at all trivial for several reasons. First and foremost, hate-speech constitutes only a fraction of all available data online and therefore is in itself a sparse data problem. Additionally, many existing annotated datasets suffer from large class imbalance, with hate-speech being undersampled (Malmasi and Zampieri, 2017; Zhang and Luo, 2018), which can lead to overfitting on the majority class. Secondly, the term "hate-speech" is poorly defined, as seen by the amount of different terms used to describe the concept (van Aken et al., 2018). The lack of a clear definition of what constitutes hate-speech and how it can be distinguished from harmless usage of profanities leads to difficulties in the collection and annotation of data for the task and to major inconsistencies from one research project to another, making it difficult to compare and share data in the field (Davidson et al., 2017).
Data sparsity, class imbalance, unclear class labels, and inconsistent annotations are hence key problems researchers need to address, when intending to train an effective machine learning model for hate speech detection.

1.2 Research Questions

The aim of this project is to address two of these problems, namely that of data sparsity and class imbalance. As shown by Hemker (2018), data augmentation is an effective way of improving hate speech detection, achieving an average of 4-6% improved accuracy. Following similar methods, this project aims to answer the following research questions:

1. Do distributional-semantics-based augmentation techniques yield similar improvements in lower-resource settings, as in its higher-resource variant? More specifically, can we achieve similar results in German and Italian hate speech detection, as previous work has shown for English?

2. Can we transfer knowledge from a comparable dataset in a high-resource language, by means of automatic translation?

3. Is data augmentation equally effective for both binary and multi-class classification of hate speech?

The project's main objective is hence to improve hate speech classification, by addressing the problems of data sparsity and class imbalance, using data augmentation techniques. To explore the research questions of this project, we will initially discuss how previous work in the field has addressed these problems and where augmentation techniques have shown promising results. We will also address the challenges posed by the lack of a clear hate-speech definition and the resulting labeling inconsistencies. We then use three augmentation methods to test the hypothesis that data augmentation techniques can improve classification results in German and Italian: the first method implements a Recurrent Neural Network (RNN) to generate more training samples automatically; the second method uses distributed word representations (embeddings) to substitute words with close neighbors, using a cosine distance threshold of their respective word vectors. The third method adds automatically translated samples from a different dataset in English to the datasets in German and Italian, to see if the use of additional data, even from a different corpus and language, can help improve classification results. To perform the hate-speech classification, we adapt a previously used model architecture, which achieved promising results in Hemker (2018) and Zhang and Luo (2018).

The main contributions of this thesis are as follows:

1. We show that data augmentation techniques improve hate speech detection in both Italian and German data, with an overall increase of up to 8.5 macro F1

percentage points.

2. We show the effects of data augmentation on both neural and classical machine learning models.

3. We report the efficacy of different augmentation techniques using distributional-semantics-based substitution.

4. We introduce a novel, hybrid method using distributional-semantics-based substitution and translation-based augmentation.

5. We show that translation-based augmentation methods can increase the absolute precision of hate detection by up to ∼14% in Italian binary classification and the absolute recall for under-represented classes by up to ∼78% in German multi-class classification.

1.3 Thesis Structure

This work is organized as follows. In this first chapter we introduced the topic of hate speech detection and state our research motivations. In Chapter 2 we take a closer look at the previous work in this field, more specifically with regards to finding a definition for hate speech, the data that has been collected for the purpose of hate speech classification, different automated approaches, augmentation methods, and different types of classification. In Chapter 3 we provide details on the German, Italian and English datasets we use for this project. Chapter 4 lays the theoretical foundation for the NLP and Machine Learning methods we use for the augmentation of our data and subsequent classification, the details and implementation of which we discuss in Chapter 5. In Chapter 6 we report and discuss our findings, as well as the limitations of this project and our suggestions for future research in the field. We conclude this work by summarizing our findings in Chapter 7.

2 Related Work

Abusive language detection has become a hot topic in the Natural Language Processing and Machine Learning research communities within the last years. There are numerous reasons: from a regulatory perspective, the flood of online information necessitates filtering systems, which can differentiate between different types of offensive language in a sensitive manner as to not obstruct freedom of speech. From a psychological perspective, certain forms of offensive speech can be offensive but generally deemed harmless, while other forms can have much graver consequences. Such forms include online harassment, racist and sexist comments, and cyber-bullying, which can trigger decreased online participation, negative behavior in others, hate crimes, and even suicide (Dinakar et al., 2011; Sood et al., 2012; Waseem and Hovy, 2016; Nobata et al., 2016).
Many efforts have been made with regard to three key aspects: to define the task itself, given the vague and culturally specific nature of the term "hate speech"; to create adequate datasets in order to train classification models; and finally to find ways to create and optimize robust and suitable models. This chapter is hence going to look at previous work within these three areas.

2.1 Defining Hate Speech

One of the most challenging aspects of hate speech detection, which is addressed by the majority of related work, is the lack of a clear definition of what constitutes hateful or abusive language (Schmidt and Wiegand, 2017; Kshirsagar et al., 2018; van Aken et al., 2018). The difficulty humans face in identifying hateful language becomes evident, when looking at the amount of synonymous terms used interchangeably in the literature, including toxicity (Georgakopoulos et al., 2018; Gunasekara and Nejadgholi, 2018; Wulczyn et al., 2017), hate speech (Badjatiya et al., 2017; Burnap and Williams, 2016; Davidson et al., 2017; Gambäck and Sikdar, 2017; Gao et al., 2017; Schmidt and Wiegand, 2017; Petrocchi and Tesconi, 2017; Warner and Hirschberg, 2012), cyberbullying (Caines et al., 2018; Dadvar and Eckert, 2018; Dinakar et al., 2011; Chen et al., 2012; Zhong et al., 2016), sexism and

racism (Waseem and Hovy, 2016), abusive language (Mehdad and Tetreault, 2016; Nobata et al., 2016; Park and Fung, 2017), or flaming (Razavi et al., 2010). Some key problems arise due to confusion around the terminology and definition: first, most classification models require large amounts of human annotated training data, yet annotations become unreliable if annotators lack expertise knowledge or clear guidelines (Nobata et al., 2016; Ross et al., 2017; Waseem and Hovy, 2016); clear guidelines in turn rely on a clear task definition.

Consider the following two statements:

1. When you realize how curiosity is a b*tch #CuriosityKilledMe (Davidson et al., 2017)

2. Gas the skypes (Magu and Luo, 2018)

The first sentence includes an English expression, which does not reflect hateful intent, but rather expresses a sense of frustration and unfairness, using a profanity. If the task is a binary classification, should this sentence be considered offensive? The second example is clear case of hateful speech directed at the Jewish community, yet the term "skypes" may not be known to annotators without the proper knowledge of far-right code terms and might be mislabeled if used in a more implicit context.

Another difficulty caused by the lack of a clear definition and inconsistencies in labeling arises when trying to compare research results. For instance, hate may be expressed differently in racist comments than in general hateful speech and the same model may not work equally well across all sub-tasks (Waseem and Hovy, 2016).

Several attempts have been made to deal with the problem of class label inconsistency and better define offensive speech and its subtasks. Some work focuses solely on the distinction between hate and non-hate (Djuric et al., 2015; Caines et al., 2018), while others find fine-grained labeling more effective (Badjatiya et al., 2017; Hemker, 2018). Razavi et al. (2010) refers to the general hate speech task as "flame detection" and further categorizes the task into several subtasks. Singh et al. (2018) propose three classes: covertly aggressive, consisting of indirect attacks or insincere polite expressions); overtly-aggressive, consisting of the use of lexical items or specific syntactic structures; and non-aggression. Zhang and Luo (2018) distinguishes between hate speech, abusive language, offensive/profane language, and bullying. Waseem et al. (2017) created a typology in order to break down the broader task of hate-speech into several subtasks, depending on whether or not the hateful language is directed toward an individual or a group, and whether or not it is implicit or explicit, which a number of researchers in the field deemed useful (Fortuna et al., 2018; Hemker, 2018; ElSherief et al., 2018).

2.2 Data and Annotation

As of now, most work has focused on English (Schmidt and Wiegand, 2017; Caines et al., 2018), resulting in several publicly available datasets. Only few hate-speech datasets exist in other languages, such as German (Wiegand et al., 2018), Arabic (Mubarak et al., 2017), Italian (Sanguinetti et al., 2018), or Slovene (Fišer et al., 2017) and not all are publicly available, due to privacy constraints. Some of the most common hate-speech data sources are Twitter (Xiang et al., 2012; Chen et al., 2012; Burnap and Williams, 2016; Burnap et al., 2015; Burnap and Williams, 2015; Silva et al., 2016; Wiegand et al., 2018), Yahoo! (Nobata et al., 2016; Djuric et al., 2015; Warner and Hirschberg, 2012), or YouTube (Dinakar et al., 2011). As no benchmark corpus exists, many authors collect and label their own data, which results in the labeling inconsistencies mentioned above (Hemker, 2018; Schmidt and Wiegand, 2017). As most NLP tasks, hate speech detection requires large amounts of data to be able to train accurate models. However, hate speech generally suffers from large class imbalance, since most online comments are not hateful; each corpus will therefore by default contain a much larger number of non-hateful samples than hateful ones. Creating a balanced dataset is hence one of the major challenges in hate-speech detection (Schmidt and Wiegand, 2017). Manual data labeling is usually done by expert annotators (Wiegand et al., 2018) or by using a crowdsourcing platform, such as CrowdFlower (Davidson et al., 2017), or Amazon Mechanical Turk (Nobata et al., 2016). Each method has their trade-off; while crowdsourcing can be a cost-effective source of annotations, it may not generate the highest quality. Employing expert annotators on the other hand is much more costly, but does not guarantee high quality results, as shown by Ross et al. (2017). Zhang and Luo (2018) offered one potential solution, by employing a mixture of amateurs and experts, and weighing expert annotations more heavily. Wulczyn et al. (2017) decided to employ an automatic labeling approach, by using the empirical distribution of human-ratings as training data, rather than majority vote.

Annotations vary significantly across tasks and datasets. Waseem and Hovy (2016) used the classes *Racism*, *Sexism* and *Neither*; Davidson et al. (2017) used the classes *Hate*, *Offensive* or *Neither*; Nobata et al. (2016) labeled their data with *Hate Speech*, *Derogatory*, *Profanity* and *Neither*. The data published by the Kaggle Toxic Comment Classification challenge, which we describe in Section 3.3 uses an even more fine grained approach, with the labels *Toxic*, *Severe Toxic*, *Obscene*, *Threat*, *Insult*, and *Identity Hate*, while the GermEval dataset, described in Section 3.1 uses the labels *Other*, *Profanity*, *Insult*, and *Abuse*.

2.3 Automated Approaches

2.3.1 Rule based

The earliest methods to detect hate speech used lexical approaches, such as manually developed regular expressions (Yin et al., 2009), or blacklists, which can consist of general hate terms, verbs, or adjectives (Sood et al., 2012; Xiang et al., 2012; Burnap and Williams, 2015; Nobata et al., 2016) in order to detect hateful key words. Chen et al. (2012) was able to reduce the false negative rate by using a rule-based approach, with profanities and obscenities as features. Gitari et al. (2015) built a semantic lexicon of hate related words, using subjectivity and semantic feature extraction, with promising applications in online discourse, such as forums and blogs. These methods can be effective in the detection of profanities, but their ability to filter all hate speech accurately is limited; as seen in the earlier paragraphs, not all hate is expressed explicitly or by use of profanities and not all uses of profanities are necessarily hateful (Davidson et al., 2017). Furthermore, intentional obfuscation of profanities (e.g. b*tch, f$ck), insults directed at minorities which may otherwise be meaningless words (e.g. skypes), or implicit insults all suggest that language cannot be viewed as static, rendering rule- and keyword-based approaches less effective on their own (Nobata et al., 2016; Schmidt and Wiegand, 2017; Wu et al., 2018).

2.3.2 Classical/ Statistical Machine Learning

Classical, supervised Machine Learning methods have been employed successfully in the past, but require a large amount of manual feature engineering. Some of the models employed include Naïve Bayes (Razavi et al., 2010), SVMs (Warner and Hirschberg, 2012; Chen et al., 2012; Dadvar and Eckert, 2018; Schofield and Davidson, 2017), logistic regression (Burnap and Williams, 2015; Waseem and Hovy, 2016; Burnap and Williams, 2015; Wang et al., 2016; Davidson et al., 2017; Wulczyn et al., 2017), or decision trees (Caines et al., 2018). Useful features include "surface" features such as ngram frequencies, URL mentions, and hashtags, or other features, such as vector representations, sentiment, POS tags, or meta information of the tweet, such as gender, age or location (Zhang and Luo, 2018).

2.3.3 Neural Machine Learning

In recent years, employing Neural Networks for NLP tasks has become a promising trend and hate speech detection is no exception.

Currently, models employing deep learning algorithms, particularly CNNs, RNNs or a combination thereof, achieve the best results (Karan and Šnajder, 2018). Some of the early works include the use of simple RNNs (Badjatiya et al., 2017), RNNs with self attention (Pavlopoulos et al., 2017), CNNs (Gambäck and Sikdar, 2017), or a combination of CNN and GRU (Zhang and Luo, 2018; Wang, 2018). All methods use simple word / character-based ngrams as input features and hybrid structures, such as CNN and LSTM or CNN and GRU appear most popular and most beneficial. (Badjatiya et al., 2017) achieve best results with an ensemble method, using LSTM in combination with randomly initialized word embeddings and Gradient Boosted Decision Trees.

2.4 Augmentation

Data augmentation has been employed extensively in Computer Vision, but less so in NLP (Hemker, 2018). Zhang et al. (2015) experiment with the use of an English thesaurus, which replaces words by semantic closeness and most frequently seen meaning. (Jafarpour et al., 2018) use use two methods to augment data: the first one finds similar words, by considering concepts from ConceptNet[1], a freely-available semantic network, and using the definitions of Wikidata[2], a collaborative knowledgebase. The second method uses text generation on the augmented data. (Hemker, 2018) uses synonym replacement techniques, based on cosine similarity and POS tags, as well as text generation approach create additional samples. Most recently, (Yu et al., 2018) introduce a novel augmentation technique, which generates new data by backtranslation from a neural machine translation model.

[1] `http://conceptnet.io/`

[2] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

3 Data

After careful screening of publicly available hate-speech datasets, we decided to use two datasets to test the effect of data augmentation on new, non-English data: one Italian dataset, made available through the HaSpeeDe (Hate Speech Detection) shared task organized by Evalita 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (Bosco et al., 2018), and one German dataset collected for the GermEval 2018 Shared Task on the Identification of Offensive Language (Wiegand et al., 2018). We chose these datasets primarily for three reasons. First, it appeared sensible to use data from completed shared tasks, as it allowed us to use the resulting scores as a baseline and compare our results with the reported winning results and methods. Secondly, both datasets were published for general hate-speech classification, which is our main research interest, rather than for task-specific classification, such as racism or sexism. Thirdly, our corresponding language competences allowed for a qualitative analysis of the results, which we deemed preferable over using data without any understanding of the language and simply relying on evaluation scores that our model produces.

To add training samples to the Italian and German datasets, it appeared sensible to look for an English dataset, used for a similar task. As mentioned in Section 2.2, most work in this field has been conducted on English data, hence we consider it a high-resource language, which is well suited for translation into lower-resource languages. We considered two potential datasets: on the one hand the dataset published by Davidson et al. (2017), containing 24,783 Tweets, which have been annotated by CrowdFlower workers with the labels "hate speech", "offensive but not hate speech" and "neither offensive nor hate speech"; on the other hand the Toxic Comment Classification Challenge dataset, published by Google Jigsaw in December 2017, containing 223,549 user comments, collected from Wikipedia[1] and annotated by human raters with six fine-grained class labels. We decided to use the latter, because of the larger sample size and because of the finer granularity of hate speech classes, making it possible to create a direct mapping with the German classes.

[1]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data

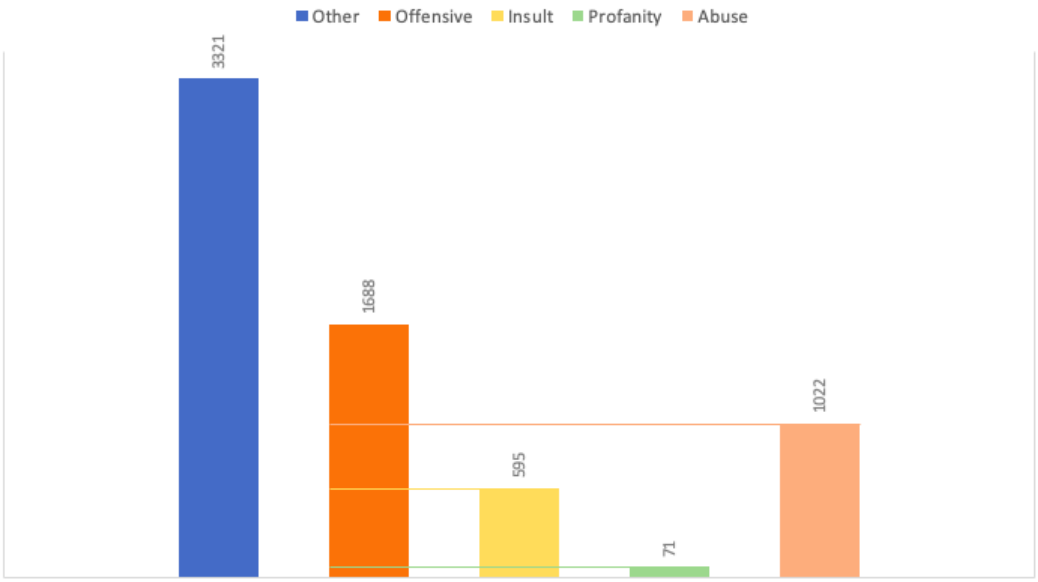


Figure 3.1: Class distribution the GermEval Hate Speech training dataset

The following sections provide further details on all datasets.

3.1 German Data

The data for the GermEval Task 2018 consist of a training set with 5009 annotated tweets and a testing set of 3000 tweets in standard German language collected from Twitter in an unspecified period of time. The collection of tweet was labeled by three annotators (however, no inter-annotator agreement scores have been reported) and is provided as a simple tab-separated file, each line containing a tweet and the corresponding labels. The tweets are in random order and have been annotated for a course-grained and fine-grained classification task. Figure 3.1 visualizes the number of tweets for each label in the training dataset.

The non-hate speech majority class labeled as Other makes up 66% of the dataset, while the hate-speech class labeled as Offensive constitutes 34% of the dataset and is further broken down into the sub-classes Insult, Profanity and Abuse.
The following sections will look at each of the tasks and corresponding labels in further detail.

3.1.1 Coarse-grained Classification

The goal of Task 1 is to decide, whether or not a tweet contains offensive language (labeled as Offensive or Other). The following sentences are example tweets from the binary task.

OFFENSIVE:

(3.1) **Original:** Juhu, das morgige Wetter passt zum Tag SCHEIßWETTER.
Translation: Yay, tomorrow's weather fits to the SHITTY WEATHER day.

(3.2) **Original:** @KarlLagerfeld ist in meinen Augen strunzdumm wie ein Knäckebrot.
Translation: @KarlLagerfeld is in my view stupid as a crispbread

OTHER:

(3.3) **Original:** Endlich hat Kurz einen Verbündeten aus Frankreich, der auch die ungesetzliche Einwanderung von jungen Afrikanern unterbinden will.
Translation: Finally Kurz has an ally from France, who also wants to prohibit unlawful immigration of young africans

(3.4) **Original:** Die Türkei führt einen Angriffskrieg und die @spd inkl. @sigmargabriel rüstet noch ihr Panzer.
Translation: Turkey is leading a targeted war and the @spd incl. @sigmargabriel is still arming their tanks.

3.1.2 Fine-grained Classification

Task 2 differentiates hate-speech further, by breaking down the tweets labeled as Offensive into three subcategories, as seen in Figure 3.1: Profanity (usage of profane words, however without intention to insult anyone), Insult (the tweet intends to offend one or several individuals), and Abuse (not just an insult against a person, but a targeted form of abusive language). As visualized in the figure, the classes are not evenly distributed: the class Other constitutes 66% of the dataset, the class Insult, Profanity, and Abuse 12%, 0.01% and 20% respectively. The class Profanity consisting of only 71 tweets is extremely sparse and may not be sufficiently represented for training purposes. The following sentences are examples of each category.

PROFANITY:

(3.5) **Original:** Als SPD wäre ich jetzt maximal angepisst.
Translation: If I were the SPD [Social Democratic Party of Germany], I would now be majorly pissed off.

(3.6) **Original:** @annaIlna Kann man diesen ganzen Scheiß noch glauben..?
Translation: @annaIlna Can one still believe this shit..?

INSULT:

(3.7) **Original:** Wo ist #Kubicki heute? Ist er schon besoffen im Puff?
Translation: Who is #Kubicki today? Is he already drunk in the brothel?

(3.8) **Original:** @HeikoMaas Dummes Gelaber. Was Sie für Steuerbetrug halten gilt nur in Deutschland und in Ihrem Hirn.
Translation: @HeikoMaas Dumb talk. What you consider tax fraud only exists in Germany and in your brain.

ABUSE:

(3.9) **Original:** Ich würde auch nicht mit einer schwarzen #Schwuchtel zusammenarbeiten #Tatort
Translation: I would also not work together with a black #faggot #crimescene

(3.10) **Original:** @KURIERat Besser spät als nie, ist ja echt nicht mehr tragbar der Juden hass der Moslems!
Translation: @KURIERat Better late than never, it's really not bearable anymore, the Jews hate of Muslims!

One of the aspects of the data which we examined further, concerns the length of each tweet, which can vary significantly. The shortest tweet in the dataset consists of 23 characters, while the longest consists of 573. While the number of characters in Twitter is limited to 280, we include mentions in our calculation, which are not included in Twitter's character limit. Figure 3.2 shows a rather even distribution of average characters per tweet, but differences in outliers. To analyze this difference, we further examined if certain classes contain more lengthy or short tweets than others and found that the non-hate class contains two-thirds more tweets with a character count above 380 characters than the offensive class, and slightly above two-thirds more tweets with a character count below 50. The Abuse class contains significantly more short tweets than the other hate-speech classes, while the Insult class contains most lengthy tweets.

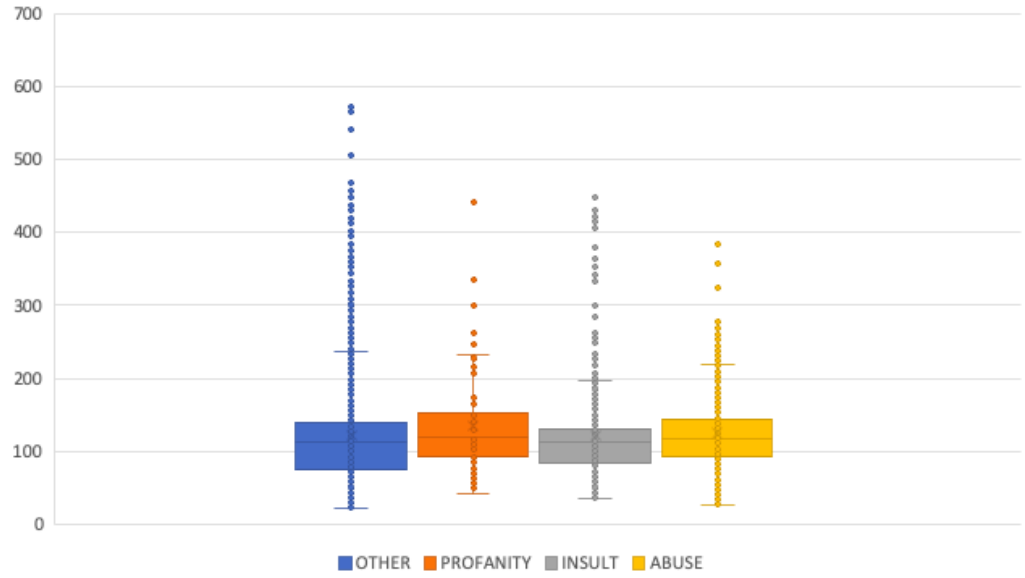


Figure 3.2: Box and whisker plot of characters per tweet for each class in the GermEval dataset

3.2 Italian Data

The dataset from the Evalita HaSpeeDe Shared Task consists of two previously collected and labeled datasets: the first being a collection of Facebook posts (Petrocchi and Tesconi, 2017) and the second a collection of Tweets about ethnic origins and religion (Poletto et al., 2017; Sanguinetti et al., 2018). We decided to only use the Twitter corpus for this project, in order to use a similar format for both the German and Italian hate speech classification.
The training dataset comprising of 3000 tweets and testing dataset of 1000 tweets are made available as a tab separated file, where the annotation format includes one tweet per line with a binary label (1 is hate speech, and 0 is not). The following subsection provides more details about the classes.

3.2.1 Binary Classification

The dataset originally consisted of a fine-grained annotation of different types of hate-speech, including aggressive (such as Example 3.11 below) and offensive language (Example 3.12), as well as general hate speech (Example 3.13). The data

made available through the shared task, however, contains simplified binary labels, for unknown reasons. Poletto et al. (2017) mention significantly low inter-annotator agreement for the fine-grained approach, which may be a reason for the simplified choice.

The majority class in this dataset consists of non-hate samples, comprising 68% of the data, while 32% consists of hate-speech samples. The following sentences serve as examples for each category.

Hate speech:

(3.11) **Original:** I migranti sanno solo ostentare l'ozio.
Translation: Migrants can only show off their laziness.

(3.12) **Original:** Cacciamo i rom dall'Italia
Translation: Let's kick Roma out of Italy

(3.13) **Original:** Ci vuole la guerra per salvare l'Italia dai criminali filo islamici
Translation: We need a war to save Italy from pro-Islamic criminals

Non-hate speech:

(3.14) **Original:** Ancora una volta, anche a capodanno! Fate l'opposto di ciò che vuole lo Stato islamico! #isisfuckyou
Translation: Once again, even at New Year's Eve! Do the opposite of what the Islamic State wants! #isisfuckyou

(3.15) **Original:** Roma, campi nomadi dappertutto. Persino sottoterra a Trastevere
Translation: Rome, nomad camps everywhere. Even underground in Trastevere

The shortest tweet has the length of 15 characters, while the longest consists of 139 characters, with an even distribution of short tweets (under 50 characters) and long tweets (over 120 characters). Figure 3.3 shows the average number of characters per tweet, where mentions, hashtags, and punctuation count as one character respectively. As seen in the figure, the average length of the tweets is similar across classes, with the non hate speech class having a slightly larger range of tweet lengths.

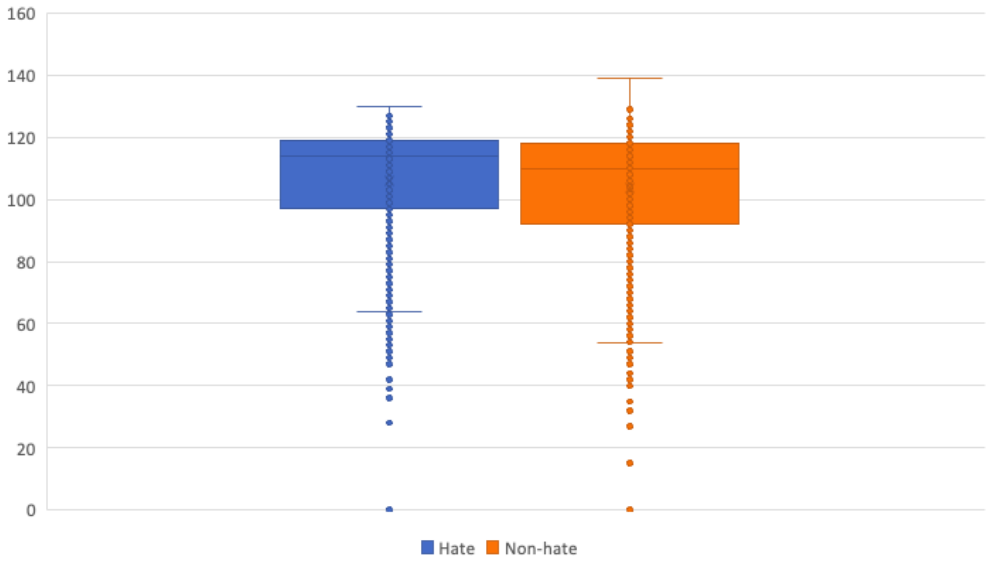


Figure 3.3: Box and whisker plot, showing average number of characters in hateful (1) and non-hateful (0) Italian tweets

3.3 English Data

We use the English training dataset from the Jigsaw Toxic Comment Classification Challenge[2], to translate samples into German and Italian. The dataset contains Wikipedia comments, labeled by human raters for toxic behavior and is intended for multi-class, multi-label classification with six classes, namely *Toxic, Severe Toxic, Obscene, Threat, Insult, Identity Hate*. The data is made available as a comma separated file, where the annotation format includes one sample per line with a binary label 1 or 0 for each class, signifying the presence or absence of a class label. A sample labeled with all 0s is hence a non-hateful sample. No official definitions for the classes have been published, however, the website states that comments may be "rude, disrespectful or otherwise likely to make someone leave a discussion".

The dataset contains 201'081 non-hateful samples and 49'596 hateful samples in total. The dataset is highly imbalanced, with the non-hateful sample size being 4 times larger than the hateful sample size and the *Toxic* class being significantly larger than the other classes. Figure 3.4 visualizes the distribution of samples per hate-class.

[2]https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/

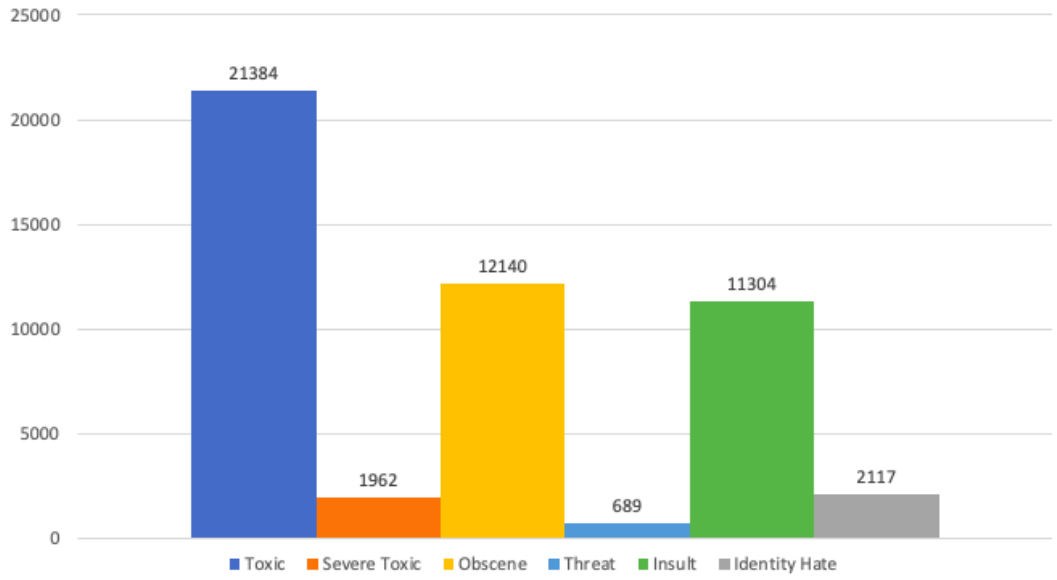


Figure 3.4: Number of samples per class for the English Jigsaw Toxic Comment Classification dataset

Furthermore, the data is quite noisy, containing samples with only one token, such as *hyper_individualist@yahoo.com* or *User:Wipeouting*. On average, samples contain 67 tokens and 394 characters. The largest found sample contains 5000 characters and 1411 tokens. This may be due to the fact, that certain samples are extremely repetitive, where the same sentence or token n-gram may be repeated dozens of times. Below are examples of samples and their labels.

(3.16) HOPE YOUR HEAD GETS CUT OFF AND SOMEONE WIPS THERE ASS WITH IT AND THEN STABS YOU IN YOUR HEART
Label(s): Toxic, Severe Toxic, Obscene, Threat, Insult

(3.17) I'm Sorry I'm sorry I screwed around with someones talk page. It was very bad to do. I know how having the templates on their talk page helps you assert your dominance over them. I know I should bow down to the almighty administrators. But then again, I'm going to go play outside....with your mom.
Label(s): Toxic

(3.18) A pair of jew-hating weiner nazi schmucks.
Label(s): Toxic, Obscene, Insult, Identity Hate

(3.19) Your vandalism to the Matt Shirvington article has been reverted. Please don't do it again, or you will be banned.
Label(s): Non-hate

(3.20) How could I post before the block expires? The funny thing is, you think I'm being uncivil!
Label(s): Non-hate

(3.21) Fuck off You are NOT an administrator. You DON'T have the authority to tell me what to do.
Label(s): Toxic, Obscene

To summarize, the English dataset differs from the ones in German and Italian, in that it is 50 times larger than the former and 80 times larger than the latter, with samples being generally more noisy and, from our observations, more extreme in their language.

4 Preliminaries

In this chapter, we explain some key concepts and algorithms from the fields of Machine Learning and NLP, which we apply in this project. In section 4.1 we lay the foundation for the distributional approach we use to encode the data. Sections 4.2 and 4.3 provide some technical background about the models we used. For all explanations, we closely follow the terminology and definitions used by Géron (2017) and Goldberg (2017).

4.1 Word Embeddings

Machine learning models cannot directly process discrete, categorical variables, such as words, characters, or POS tags, so they need to be mapped to real numbers before they can be used as training input. One simple solution is to map variables to a sparse vector of 0s and a single 1, signaling one dimension for each possible feature; this approach is also known as one-hot-encoding. For instance, in a bag-of-word approach with a vocabulary size of 10'000 items, each vocabulary item is one dimension, resulting in a vector with 10'000 dimensions. Taking one of the examples from our English dataset, the tweet containing the five words *can one believe this shit* would result in a vector, where at most five items are non-zero values. In such a vector, the word *can* may appear in position 3'210, *believe* in position 1'230 and *shit* in position 7'870. Table 4.1 shows how a one-hot encoding could look like for each of these words.

There are a number of problems with such a representation. First, as each dimension is encoded separately, a vector's size will increase with the number of features. For instance, a feature representing the co-occurrence of *this* and *shit* must be encoded explicitly as a separate feature dimension. This results in extremely high-dimensional, sparse vectors, which are difficult to process efficiently on a computational level. Second and more importantly, as each feature is encoded separately, they are independent and not comparable by a similarity measure. As seen in Table 4.1, the word vector for *shit* is as different from *crap* as it is from *believe*, yet *shit*

Word	Sparse representation	Dense representation
can	(1, 0, 0, 0, ...)	(-0.071549 0.093459 0.023738 ..., -0.096652)
believe	(0, 1, 0, 0, ...)	(1.1005, -0.22599, -0.41616, ..., -0.22211)
crap	(0, 0, 1, 0, ...)	(0.68398 -0.35945 -0.38269, ..., -0.1211)
shit	(0, 0, 0, 1, ...)	(0.61815 -0.61925 -0.53806 ..., -0.51783)

Table 4.1: Examples of English words and their possible representation as one-hot (sparse) encoding or word embedding (dense).

and *crap* are semantically closely related.

Distributional word representations (word embeddings) are a solution to both problems and a powerful NLP tool originating in the field of distributional semantics. Conceptually, an embedding is a mapping of discrete variables (such as a word, a character, or a POS tag) to a vector of real numbers, where each feature is embedded into a d dimensional space. Table 4.1 shows what an embedding vector might look like. An individual item within the vector is not meaningful, but the entire vector allows for a representation of the word within an distributed embedding space, where similar vectors are found in close distance to each other. This way, the representation learned for the above words *shit* and *crap* should be similar, assuming a large training dataset, where both words occurred often enough in similar context.

There are several methods to compute word embeddings using neural networks, such as Word2Vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), and fastText (Bojanowski et al., 2017). These methods are based on the assumption that similar words appear in similar contexts; following this logic, two words with similar context words should be related in some way. In order to compute word embeddings, most methods use some type of neural network architecture, which loops over the training data (w,c), calculates the probability of a word (e.g. Word2Vec) or character n-gram (e.g. fastText) w given its context c and uses an optimization algorithm to maximize the likelihood. Word2Vec is a widely used method, which implements two types of modeling architecture: Continuous Bag of Words (CBOW), which calculates the probability of a target word t, given its context words, or Skip-Gram, which calculates the probability of the context words, given a target word. Figure 4.1 visualizes the two architectures with a window of four context words, where *W(t)* is the target word and *W(t-2),...,W(t+1)* are the context words.

In this project, we use embeddings, which have been trained using the skip-gram variants of Word2Vec and fastText (Cieliebak et al., 2017). The fastText and Word2Vec model architectures differ mainly in the fact, that the former treats

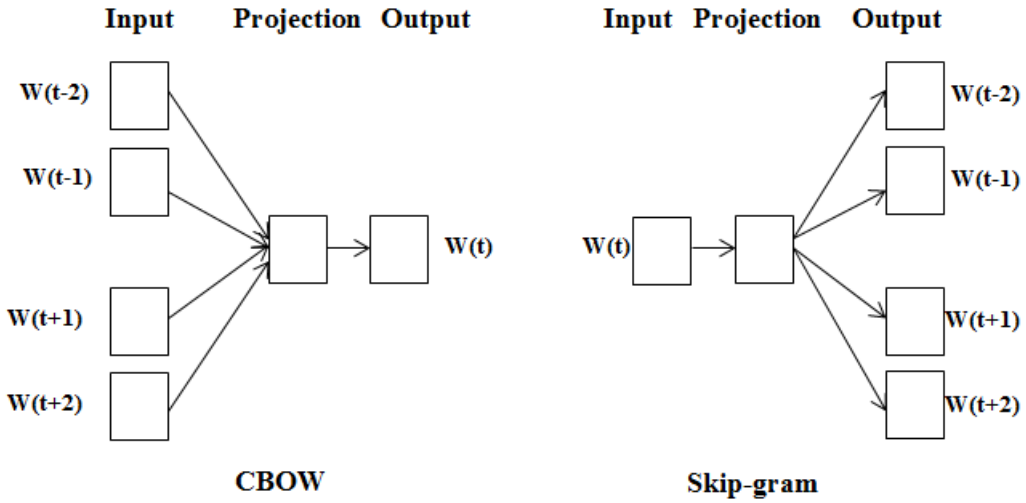


Figure 4.1: Figure from Mikolov et al. (2013): Word2Vec's Skip-Gram vs. CBOW model with context window of 4

words as atomic units, while the latter decomposes each word into character n-grams. fastText's use of character n-grams instead of words, allows it to capture sub-word information, which can be especially advantageous for words that have not been seen during training and to detect compound words. Formally, the skip-gram model is defined as

$$p(c|w;\theta) = \frac{exp(v_c * v_w)}{\sum_{c' \in C} exp(v'_c * v_w)} \tag{4.1}$$

where w is either a target word (in the case of Word2Vec) or a bag of character n-grams (in the case of fastText), c is one of the context words or bag of n-grams, and the objective is to compute $p(c|w)$, using a softmax probability distribution over all word-context pairs in the corpus. Taking the example of Figure 4.1, the goal would hence be to compute the probability of the target word W(t) appearing with the context words W(t-2),...,W(t+2). For a more detailed explanation, please refer to Mikolov et al. (2013).

For tasks with smaller amounts of data, such as hate-speech detection, it makes sense to use embeddings, which have been pre-trained on large amounts of data, such as Wikipedia or Common Crawl [1], instead of training them ad hoc on the task's data. Since the language used on Twitter may differ from language used on Wikipedia and general web articles, we decided to use embeddings, which have been pre-trained on

[1] http://commoncrawl.org/the-data/

German and Italian Twitter data (Cieliebak et al., 2017). The German embeddings were trained on 50 million tweets using 300 dimensions. The Italian embeddings were trained on 200 million tweets using 200 dimensions. Dimensions for embeddings typically range between 10 to 1000, where higher dimensions usually increase quality but decrease computing efficiency.

4.2 Classical Machine Learning in NLP

In order to create a point of comparison for our neural classification models, we initially train two baseline models, using traditional, feature-based machine learning algorithms. The following sections provide the necessary background information.

4.2.1 Support Vector Machine (SVM)

SVMs are very good baseline models for smaller datasets, as they are fast to train, interpretable, and robust to overfitting (Yin et al., 2009). They are the most frequent method used in hate-speech classification (Schmidt and Wiegand, 2017) and were the most popularly used method in the GermEval competition (Wiegand et al., 2018). SVMs perform large margin classification; in other words, they try to draw a decision boundary by maximizing the margin between all data points. However, it is not always possible to perform hard margin classification, where all instances need to be on one side or the other, since not all data is perfectly linearly separable. Additionally, outliers may skew the decision boundary and hinder the classifier from generalizing well. Instead, it makes sense to perform soft margin classification, where margin violations are balanced with margin size. Figure 4.2 shows the contrast between them: on the left, fewer margin violations exist, but the margin is low, which can lead to overfitting. On the right, more margin violations exist, but the margin is larger, which means the model might perform better on unseen data. The goal is hence to find a good balance between the two.

Often, the training data is not linearly separable at all. In this case, the "kernel trick" can be used, which adds additional features and hence projects the feature vector to another dimension, where data points might be linearly separable. The easiest way to visualize the kernel trick is to draw different points on a piece of paper and then fold the paper, resulting in two separate sheets, where points are on top or on bottom and a line can be drawn between them. The more features are added, the more times the paper is folded, resulting eventually in as many data points as possible being separable.

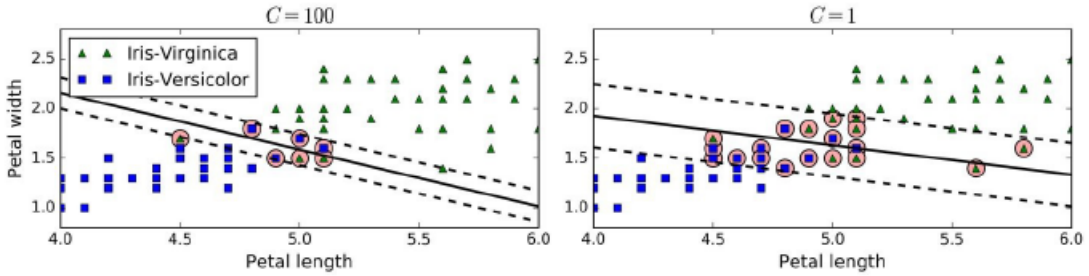


Figure 4.2: Figure from Géron (2017): Fewer margin violations versus large margin

For this project, we use the Scikit-learn implementation of the SVM classifier. As input we use TF-IDF vectors, which were calculated on the tweet texts, using unigrams and bigrams. To regulate the margin, Scikit-Learn's SVM class includes the **C** hyperparameter, which can be optimized, where (counterintuitively) a small **C** means a higher margin and larger **C** means a lower margin. Furthermore, Scikit-Learn allows us to choose between different Kernel types, such as Polynomial, Linear, or Gaussian RBF. In case of the Gaussian RBF, the hyperparameter gamma (γ) is used, where a small gamma means instances have a larger range of influence, but may be prone to underfitting, and a larger gamma decreases the range of influence, but may be prone to overfitting. **C**, Kernel type, and Gamma are hence hyperparameters to be optimized.

4.2.2 Gradient Boosted Decision Trees

Boosted Decision Tree algorithms refer to ensemble methods, where single decision trees are combined to sequentially learn from each other. The most popular methods include AdaptiveBoosting (AdaBoost) and Gradient Boosting. For this project, we use XGBoost [2], a library that implements the gradient boosted decision tree algorithm. It has been gaining popularity especially in machine learning competitions, due to it's computational speed and model performance (Chen and Guestrin, 2016). While AdaBoost adapts the weights for each instance after each iteration, Gradient Boosting updates the weights by computing the residual error, using the gradient descent algorithm to minimize the loss when adding new trees. The following hyperparameters are to be optimized:

[2]https://xgboost.readthedocs.io/

1. the **learning rate**, which scales the influence of each tree.
2. the **maximum depth**, or the length of the longest path from the tree root to a leaf, where the root node has a depth of 0.
3. the **number of estimators**, or number of trees
4. the **minimum child weight**, which is the minimum sum of instance weight needed in a child.
5. **gamma**, a threshold which decides how conservatively the algorithm decides to make a further partition on a leaf node of the tree.
6. **subsampling**, where less of the training data can be sampled, which reduces the risk of overfitting.

4.3 Deep Learning in NLP

In the following sections, we provide some preliminary explanations of the deep learning algorithms, which build the basis for our model architecture.

4.3.1 Convolutional Neural Network (CNN)

CNNs are a class of feedforward neural networks, which are used to identify informative n-grams in sequences of text, taking their respective local ordering, but not global position into account. They are often referred to as feature extractors, since they ideally capture useful textual aspects for the prediction task at hand. Originating in the field of Computer Vision, they capture information by means of a *convolution*, a sliding window of size k, which runs over a sequence of length n and applies a filter function to each window. Formally, given a sequence of words $w_1, ..., w_n$ represented by their corresponding d-dimensional word embedding $\mathbf{w_1}, ..., \mathbf{w_n} \in \mathbb{R}^d$, we concatenate all vectors inside the window, resulting in the following sentence representation

$$\mathbf{x}_{1:n} = \mathbf{w}_1 \oplus \mathbf{w}_2 \oplus ... \oplus \mathbf{w}_n$$

where $\oplus$ is the concatenation operator (Kim, 2014). We feed the resulting n x k matrix into a convolutional layer, where a convolution operation using a *filter* $\mathbf{f} \in \mathbb{R}$ is applied to a window of k words, resulting in a new feature. The resulting scalar

value of the convolution for one window is computed as follows

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b),$$

where $\mathbf{w} \in \mathbb{R}^{hk}$ is the weight matrix of the convolution unit, $\mathbf{x}_{i:i+h-1}$ the window of words, $b \in \mathbb{R}$ the bias and f the non-linear activation function. After applying the filter to each window, resulting in a feature map $\mathbf{c} = [c_1, c_2, ..., c_n]$, we apply a max-over-time pooling operation $max_c = max(c)$ over it, in order to determine the most useful feature in each dimension of the convolution layer. This pooled vector $\mathbf{c}$ is then fed into a fully-connected softmax layer, outputting the probability distribution over the labels.

4.3.2 Recurrent Neural Networks (RNNS)

While CNNs are very good at capturing local features, they do not perform as well in learning tasks involving long-distance relation patterns, as they assume items are independent and identically distributed (commonly known and abbreviated as i.i.d.). RNNs are a class of neural networks, which allow for modeling of sequences of arbitrary length, which are not i.i.d., by looking at long-distance dependencies around a target word. Essentially, an RNN differs from a feedforward network by adding a memory variable, where at each time step t it considers not only the input vector $\mathbf{x}$, but also the output from the previous state. This process allows RNNs to process sequences of varied lengths and thereby (ideally) capture each word's dependencies. More formally, as defined by Goldberg (2017), an RNN is a function, which takes a weighted state vector s_{i-1} and a sequence of n d-dimentional vectors $\mathbf{x_{1:n}} = \mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n}, \mathbf{x_i} \in \mathbb{R}^{d_{in}}$ as input and returns a new state vector s_i, which is mapped to a single d-dimensional vector $\mathbf{y}_n \in \mathbb{R}^{d_{out}}$ as output. An RNN therefore considers the state at time t given the entire history and updates the state vector accordingly.

It is generally known that Vanilla RNNs are difficult to train, due to the vanishing and exploding gradient problems, which Pascanu et al. (2013) describes in detail. To solve this problem, two techniques using gated architectures have been introduced: Long-Short Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs). As this project implements both RNN variants, we will describe the respective architectures in more detail.

4.3.3 Long-Short Term Memory Network (LSTM)

Gated mechanisms were designed to solve the vanishing gradient problem, by introducing update and reset gates, two vectors which decide which information will be passed through to the output. LSTMs, like Vanilla RNNs, include a state vector, which captures the previous output in addition to the current one. However, unlike in RNNs, the state is split in two vectors: h_t and c_t, where one can think of h_t as the short-term state and c_t as the long-term state (Géron, 2017).

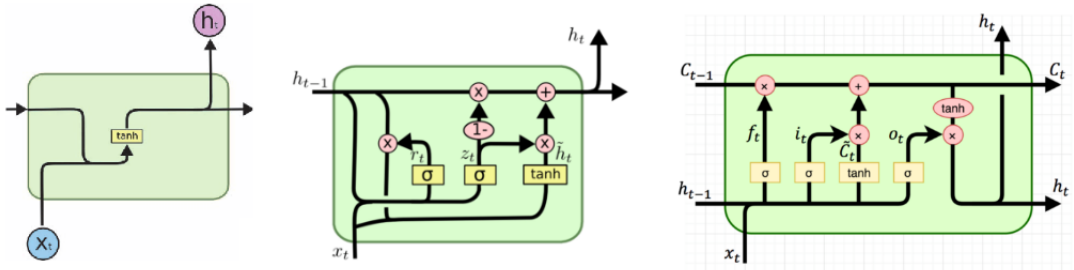


Figure 4.3: Figure from dProgrammer (2019): Basic architectures of RNN (left), GRU (middle) and LSTM (right) cells

As seen in Figure 4.3, first, the current input vector x_t and the previous short-term state $h_{(t-1)}$ are fed to four fully connected layers, where the main output layer g_t analyzes them and partially stores this information in the long-term state c_t. Three more layers serve as "gates", deciding which information gets passed on. The forget gate (f_t) controls which parts of the long-term state should be erased, while the input gate (i_t) controls which parts of (g_t) should be added to the long-term state. As opposed to the basic RNN architecture, LSTMs hence keep important input, by storing it in the long-term state and erase any unneeded information, which allows them to be more efficient and less prone to vanishing or exploding gradients.

4.3.4 Gated Recurrent Unit (GRU)

Cho et al. (2014) introduced the GRU as an alternate, computationally simpler version of LSTMs, where both state vectors are merged into a single vector h_t and only one gate handles both the forget and input gate. If it outputs a 1, the forget gate is open and the input gate is closed; if it outputs a 0, the opposite happens. Instead of an output gate, a full state vector is output at every time step, but a new gate controls which of the past information is passed through to the main layer. Figure 4.3 shows how the architecture is simpler than that of the LSTM.

5 Methods

In Chapters 2 and 3, we discussed how data sparsity is one of the main challenges in hate speech detection and that data augmentation can provide a solution to this problem. This chapter will outline the technical methods used for this project, building on the theoretical foundation, which we laid in Chapter 4.
In Section 5.1, we describe the methods used to pre-process the data for this task. In Section 5.2 we discuss the methods used to augment the data. Section 5.3 provides an overview over our classifier's model architecture.

5.1 Pre-processing

One of the main goals of data pre-processing is to filter out noise, which may hinder subsequent tasks. We employ a number of commonly used NLP practices to prepare the data for augmentation and classification. After initial tokenization using SpaCy, we normalize each tweet containing "mentions", which are references made by one user to other users in the form of an "@" sign, followed by a username. We further normalize URL's, since they do not add any useful information and since we do not want the classifier to learn features from these tokens. We replace such tokens with the placeholders "MENTIONHERE" and "URLHERE". We do not normalize "hashtags", which are keywords used to categorize and easily identify tweets in public Twitter searches. They appear in form of a hashtag symbol (#) before a relevant keyword or phrase, which often contains hateful terminology and therefore may add useful information, that the classifier can learn from.

We then split the training files for both German and Italian into a training and validation set. The training set consists of 80% of the original training set and the remaining 20% are used for (hyper-)parameter tuning. The separately provided test files are used as held-out evaluation sets.

As one of the augmentation tasks employs POS tagging, an NLP technique described in more detail in Section 5.2.1, we do not lower-case the data immediately, in order to allow the tagger to identify parts-of-speech accurately. We do lower-case the data

in a later step, in order to find the appropriate word embedding, which we describe in Section 5.2.1.

Since the English data pre-processing and translation are closely linked, we describe both in detail in Section 5.2.2.

5.2 Augmentation

To test our hypotheses, we implement three data augmentation techniques: substitution, based on distributed word representations, described in Section 5.2.1; language generation, described in Section 5.2.3; and finally translation of an English dataset into the respective language, which we describe in Section5.2.2.

5.2.1 Word Substitution

Adding training samples, where certain words have been substituted by new words, is an intuitive way of augmenting existing data, as it ideally produces more data with similar meaning and therefore adds more variation in the training data without having to manually collect more samples. Take for instance Example 3.12 from Chapter 3: *We need a war to save Italy from pro-Islamic criminals.* If we replace *pro-Islamic* by a similar word, such as *pro-Muslim*, or *pro-Arabic*, the sentence would still be considered hateful and adding the augmented sample would increase the model's ability to generalize.

We test the following two variations of word substitutions for this task:

1. **Threshold-based substitution**

   This method replaces words in a sentence with similar words, by retrieving each word's vector from a pre-trained word embedding corpus and by calculating the distance between the respective vectors, using a similarity or distance function. We use cosine similarity, which is a frequently used similarity measure. It calculates the angle between two n-dimensional vectors in an n-dimensional space, by taking the dot product of the two vectors, divided by the product of the two vectors' lengths. Values range between -1 and 1, where -1 is perfectly dissimilar and 1 is perfectly similar. Formally it is defined as follows

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} * \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{5.1}$$

Word	Translation	Cosine Similarity	Word	Translation	Cosine Similarity
glaube	believe		israelischen	israeli	
denke	think	0.92	palästinensischen	Palestinian	0.97
dachte	thought	0.90	iranischen	Iranian	0.97
wahrscheinlich	probably	0.87	syrischen	Syrian	0.97
glaubste	you believe	0.86	irakischen	Iraqi	0.95
bezweifle	doubt	0.86	ägyptische	Egyptian	0.95

Table 5.1: German words with close cosine similarity and their English translation

where A and B are n-dimensional word vectors to be compared. More specifically, for each word vector t, we collect the five most similar vectors and check, whether or not they pass a cosine similarity threshold (similarity >= positive threshold), which we pre-define as a hyperparameter. Table 5.1 shows examples of words retrieved from the German dataset and their potential replacements, using word embeddings pre-trained on German tweets (Cieliebak et al., 2017).

After checking each word of a sentence for replacement candidates and selecting up to n closest embeddings, we may end up with n new samples for all words w in the sentence. However, creating new sentence combinations of each candidate with each of the other candidates, resulting in w^n words would be too expensive on a computational level. One alternative method employed by Hemker (2018) generates a list of n new samples for each word w in a sentence (provided they pass the threshold) and produces s new sentences, where s_1 contains all first candidates in each n, s_2 all candidates n2, etc. However, we hypothesize that this method may be biased towards picking candidates with the highest threshold and not create enough semantic variation. We decided instead to generate new samples by randomly sampling from a geometric distribution, having as probability mass function

$$f(k) = (1-p)^{k-1}p$$

where p is a parameter, indicating the success probability of a trial. For instance, if p is set to 0.5 and a fair coin is thrown repeatedly, k indicates the number of trials it took to reach one success (for instance to throw head). *F(k)* computes the probability of achieving success at the k^{th} trial. Table 5.2 shows sample tweets with their augmented replacements.

Finally, the class label for each augmented sample remains identical to the one of the original sentence.

Original German	**MENTIONHERE Das zeig mir mal mit einem Bild.**
Augmented TH 75	mentionhere dieses zeig mir mal mit einem bild. mentionhere das zeigs mir dochmal mit einem profielbild. mentionhere das zeig dir mal mit einem bild.
Augmented TH 25	mentionhere das zeig mir nochmal mit einem bild. : (mentionhere dieses zeig dir mal mit einem titelbild. mentionhere das zeigs mir nochmal inklusiven einem bild.
Augmented POS	mentionhere das zeig mir mal mit einem titelbild. mentionhere das zeig mir mal mit einem profielbild. mentionhere das zeig mir mal mit einem profielbild.

Table 5.2: Augmented examples German

Original Italian	**Torturatore di migranti in Libia arrestato a Milano Lombardia URLHERE**
Augmented TH 75	torturatore di profughi in libia denunciato a milano lombardia urlhere torturatore di migranti in libia arrestato a milano piemonte urlhere torturatore di migranti in siria arrestato a milano lombardia urlhere urlhere
Augmented TH 25	giustiziato di migranti in libia denunciato a milano lombardia urlhere urlhere torturatore di migranti in libia denunciato . milano piemonte urlhere torturatore di immigrati in libia arrestato a milano veneto urlhere
Augmented POS	giustiziato di profughi in Libia arrestato a Milano piemonte urlhere Torturatore di migranti in siria arrestato a Milano Lombardia urlhere Torturatore di migranti in Libia arrestato a firenze Lombardia urlhere

Table 5.3: Augmented examples Italian

POS-tag-based substitution

The second method replaces words of a pre-specified part of speech (such as nouns) with their most similar vector of the same tag, regardless of their similarity score. To tag the tweets with their parts of speech, we use spaCy[1], an open-source library for Python, which automatically tokenizes each tweet and assigns POS tags following the Universal Dependencies (UD)[2] scheme. UD is a framework for universal, cross-linguistic grammatical annotation, currently available for approximately 70 languages. The German spaCy model has been trained on the TIGER [3] and WikiNER [4] corpora, and the Italian model on the Universal Dependencies and WikiNER corpora[5], both by using a multi-task CNN.

To limit the number of generated samples, we use the same random sampling method as in the threshold-based approach.

[1]https://spacy.io/

[2]http://universaldependencies.org/u/pos/

[3]https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html

[4]http://schwa.org/projects/resources/wiki/Wikiner

[5]https://spacy.io/models/it

5.2.2 Translation

The second augmentation technique used in this project adds new data by translating a subset of the English training dataset from the Jigsaw Toxic Comment Classification Challenge (see Chapter 3.3) into German and Italian. Due to it's significantly larger size (223,549 training samples vs. 5009 German and 3000 Italian samples), as well as larger sample size (up to 1403 characters vs. the Twitter allowance of 280) we employed several methods to reduce the data to a more feasible size. To do this, we initially remove the classes *Toxic*, *Severe Toxic* and *Threat'* and only keep the classes *Obscene*, *Insult*, *Identity hate*, which we can map directly to the classes *Profanity*, *Insult* and *Abuse* in the German dataset, and which can also be collectively mapped to the Italian hate class.

As the English dataset contains multiple labels for each sample, but our German fine-grained training set contains only single labels, we further have to convert the multi-label classification into a single-label one. To do this, we check if a sample contains multiple labels; if it does not, we directly map it to the appropriate class in the German training set. If it does, we randomly choose and assign a label from the selection. For example, the translation of the sentence *"Hi! I am back again! Last warning! Stop undoing my edits or die!"* with the original labels [obscene: 0, insult: 1, identity hate: 0], is mapped to the label [2] for insult in the German dataset, and [1] for hate in the binary German and Italian datasets. For the translation of the sentence *"What a motherfucking piece of crap those fuckheads for blocking us!"* with the original labels [obscene: 1, insult: 1, identity hate: 0], we use a random choice between the labels obscene and insult, resulting in the mapping [1] for obscene in the German dataset, and [1] for hate in the binary German and Italian datasets.

Furthermore, we keep only those samples, where the unique token / length of text ratio is over 0.5, and the length of the text is under 400 characters, in order to avoid too repetitive language and to assimilate them to the length of tweets.

To translate the samples, we use Googletrans [6], a free python library that implements the Google Translate API, which can dynamically translate text and supports over 100 languages. We initially considered a stratified sampling approach, where we add different amounts of translated samples to see the effects on the original dataset (for instance, adding 500, 1000, 10'000 translated samples in total). However, due to the API's limited allowance of requests, we could only translate 2000 samples at most. We therefore decided to add up to 500 translated samples to each class in the German dataset. The final distribution of new English samples can be seen in

[6] https://pypi.org/project/googletrans/

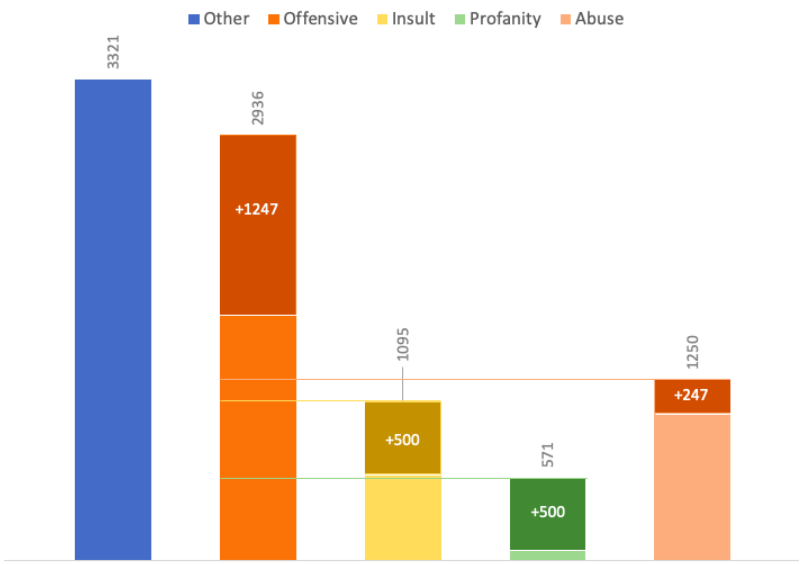


Figure 5.1: Class distributions of German data with addition of translated samples

Figure 5.1. The figure shows that we keep the original class distribution in tact, but that for each class, especially for Profanity, the number of samples are raised by a significant amount.

Examples 5.2, 5.3 and 5.4 show English samples and their translations into Italian and German. While from our initial observation, we find the translations to be of high quality, we notice some differences between English and German. As can be seen in Example 5.5, the original sample has been (mostly) correctly translated into German, while most has been left untranslated in Italian. We observed this trend in several samples, which we examined, and assume that the translation method might not work as well for the Italian dataset, as for the English one.

(5.2) **Original:** Stupid peace of shit stop deleting my stuff asshole go die and fall in a hole go to hell!
Labels: Obscene, Insult
Italian: Stupida pace di merda smettila di cancellare le mie cose stronzo vai a morire e cadi in un buco vai all'inferno!
Label: Hate
German: Dumme Scheiße, hör auf, mein Zeug zu löschen, Arschloch, stirb und falle in ein Loch, fahr zur Hölle!
Label: Profanity

(5.3) **Original:** Would you both shut up, you don't run wikipedia, especially a stupid kid.
Label: Insult
Italian: Vorresti entrambi zitto, non eseguire wikipedia, specialmente un bambino stupido.
Label: Hate
German: Würden Sie beide den Mund halten, laufen Sie keine Wikipedia, vor allem ein dummes Kind.
Label: Insult

(5.4) **Original:** I think your just a cuntt.
Labels: Obscene, Insult
Italian: Penso che tu sia solo una fica.
Label: Hate
German: Ich denke, du bist nur eine Fotze.
Label: Insult

(5.5) **Original:** Hi IM SO GAY AND RETARTED WE SUCK SO MANY BALLS OOOOOOOOOOO I LOVE MEN !
Label: Identity Hate, Insult
Italian: Ciao IM SO GAY AND RETARTED SUCCHIAMO MOLTE PALLINE OOOOOOOOOOO I LOVE MEN!
Label: Hate
German: Hallo ich bin so schwul und neugestartet wir saugen so viele Bälle OOOOOOOOOOO Ich liebe Männer!
Label: Abuse

5.2.3 Language Generation

The third augmentation technique uses recurrent neural networks (RNNs) to generate new samples of text for a given class automatically. In this project, we use a two-layer Long Short Term Memory (LSTM) network to learn word representations of each class and generate new samples. We use textgenrnn[7], which is a Python 3 module on top of Keras/TensorFlow. Figure 5.2 shows the default architecture of the model.

The model is initialized by a random token and converts N input words or characters (input: InputLayer; in this case 40 words) into a 100-D embedding vector (embed-

[7]https://pypi.org/project/textgenrnn/

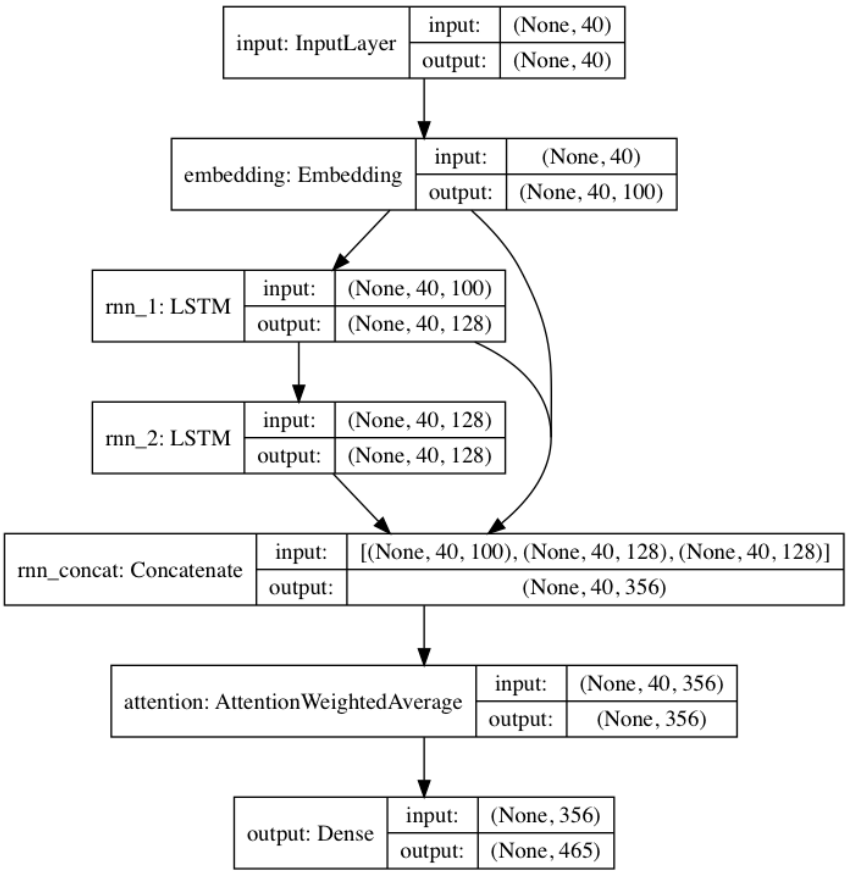


Figure 5.2: Default textgenrnn model

ding: Embedding). It uses the 40X100 embedding matrix as input to two 128-D LSTM layers (rnn: LSTM; rnn2: LSTM). The output of the embedding and the two LSTM layers is concatenated to a 40x356 feature matrix (rnn: Concatenate) and then fed into an attention weighted layer (attention: Attention WeightedAverage), which weighs the most important features and averages them together. It is finally fed into fully connected layer (output: Dense).

We initially assumed that generating text with a small data size may not necessarily result in coherent text, but may still be useful to create similar, abstracted semantic structures, which our classification model could learn from. However, after examining the samples generated by this model, we concluded that they were unusable, presumably because of the small data size. We therefore did not include this method in our experiments, but believe the method could be reevaluated in future works, if more training data is available.

5.3 Model Architecture

The previous chapter provided the necessary technical background information for the deep learning techniques used in this project. Having laid the foundation, this section will outline the model architecture we use to classify hate speech, which builds upon the topology seen in Figure 5.3, which was proposed by Hemker (2018) and Zhang and Luo (2018).

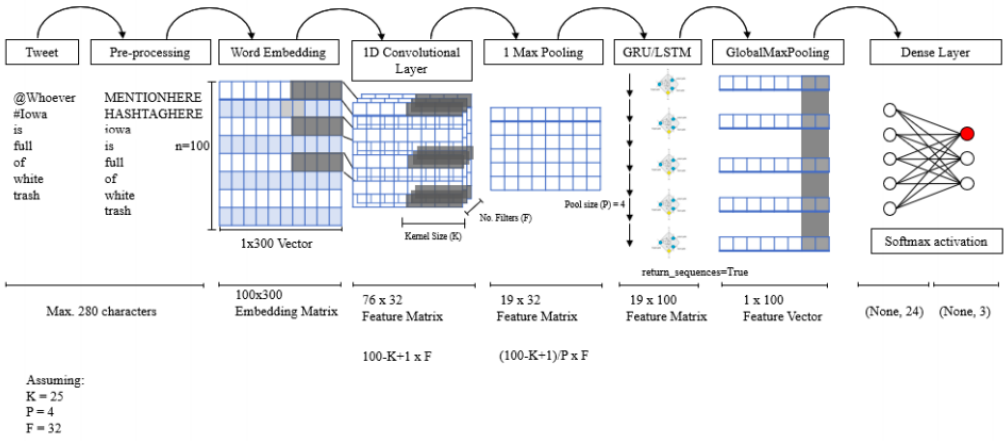


Figure 5.3: Model architecture taken from Hemker (2018), with kernel size K = 25, padding P = 4, Filter size F = 32, and embedding size 300

We initially pre-processed the tweets, as described in 5.1, and map them to the respective word embedding, resulting in a 300 x d embedding matrix, where d corresponds to the embedding dimension (in Figure 5.3 $d = 300$). The embedding matrix is used as input for the convolutional layer, whose window and kernel size are hyperparameters to be optimized. The resulting convolved feature representation is passed into a max-pooling layer, which extracts the best features for each convolution and reduces the feature dimension (in Figure 5.3 to a 19x32 feature matrix). A dropout layer is subsequently used to avoid overfitting, which is further detailed in 5.4.3. The feature map is then passed into a GRU or LSTM, which outputs 100 hidden units per timestep, allowing the LSTM or GRU to learn the features extracted by the CNN. This way, each feature dimension at recurrence t of an input text may be useful to predict the next feature at point *(t + 1)*. We then use a dense layer with softmax activation function, which turns an input vector of K real numbers into a normalized probability distribution consisting of K probabilities.

5.4 Training and Optimization

We use random search hyper-parameter tuning, rather than the traditional grid search, as it has been shown to perform well and efficiently in high-dimensional search spaces with more than two hyper-parameters (Bengio, 2012), especially for neural networks (Bergstra and Bengio, 2012). The latter also show in their paper that in different datasets, different and limited numbers of subspaces are important; due to the curse of dimensionality, the number of wasted grid search trials is exponential, making it highly inefficient. In contrast, random search has the practical advantages of grid search, such as conceptual simplicity, ease of implementation and potential to run searches in parallel, while at the same time being more efficient. Figure 5.4, taken from Bergstra and Bengio (2012), shows how random and grid search differ in dealing with low dimensionality. With grid search, nine trials only test a function g(x) (which is to be optimized), in three distinct places, whereas with random search, all nine trials explore distinct values of g. One can also consider the intuition behind the efficiency of random search from a probabilistic point of view. Suppose that the subspace of optimal and good solutions are contained in around 5% of the overall space of candidates. If the number of trials is represented with the variable T, the probability to find one such solution is $1 - (1 - 0.05)^T$. With $T = 59$, we already reach a probability of 0.95. If the subspace represents 1% of the overall space, to have the same probability, T has to be at least greater than 297.

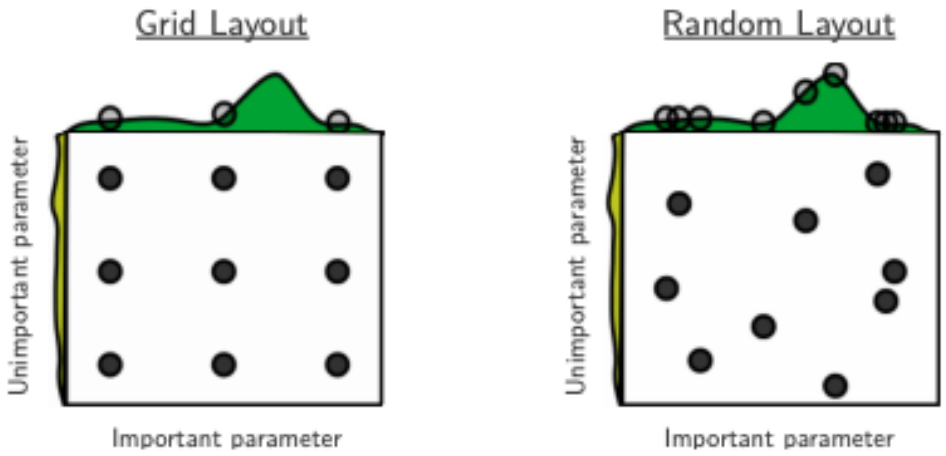


Figure 5.4: Figure from Bergstra and Bengio (2012) showing grid search and random search of nine trials for optimizing a function with low effective dimensionality.

Parameter	Value
Learning Rate	0.01, 0.001, 0.003, 0.005, 0.007
Dropout	0, 0.1, 0.2, 0.3, 0.4, 0.5
Filters	32, 64, 128
Batch Size	32, 64, 80
Kernel Size	4, 5, 6
Strides	1, 2, 3, 4, 5
Momentum	0, 0.7, 0.8, 0.9
λ_{L2}	0.2, 0.1, 0.01, 1e-4
Optimizer	Adam, RMS, SGD

Table 5.4: Hyperparameters and corresponding search values

5.4.1 Hyperparameters

We first consider uniform hyperparameter sampling in the logarithmic space, in order to better observe the impact of the different values on the performance, and later sample in a combination of logarithmic and linear space around the best candidates, for a more fine tuned search. Table 5.4 shows the hyperparameters and all possible values we sample from, using random search.

The following sections provide more details on the individual hyperparameters.

5.4.2 Optimization

The goal of any neural training procedure is to find the optimal model parameters (or weights) that minimize the cost function over the training set. Gradient-based optimization algorithms, which are commonly used, iteratively update the parameters, by calculating the gradient (or the instantaneous rate of change of y with respect to x) of the error function, given a parameter vector θ, and then move in the opposite direction of the gradient, until a local (or ideally global) minimum is found. For this project, we test the following three optimization algorithms:

Stochastic Gradient Descent is a popular optimization algorithm, which repeatedly samples a random training instance and computes the gradient on that instance (as opposed to calculating it on the entire training set, like in traditional Gradient Descent). Since only one training sample at a time is held in memory, computation is much faster, making it easier to train on large datasets. Due to the random (or stochastic) sampling, parameter updates have high variance, which can on the one hand help avoid local minima in case of non-convexity (which is most often the case),

but means on the other hand, that the algorithm will fluctuate to different degrees and will not reach convergence (Goldberg, 2017; Géron, 2017).

RMSProp is a type of adaptive learning algorithm and an extension of the AdaGrad algorithm by Duchi et al. (2011), which, as it's name suggests, computes adaptive learning rates for each parameter θ_i, as opposed to the entire set of parameters θ, like SGD. Like AdaGrad, it performs larger updates for infrequent parameters and smaller updates for frequent parameters. While AdaGrad is designed to converge rapidly and takes into account the entire history of the squared gradient, which may result in a too small learning rates, RMSProp uses an exponentially decaying average to find how quickly the most recent gradients evolved in one direction, before updating the weights.

Adam is an optimizer introduced by Kingma and Ba (2014), which combines attributes of both AdaGrad and RMSProp. It uses an exponentially decaying average, but not only takes into account the average first moments of the gradients (the mean), but also the second moments (the uncentered variance). RMSprop and Adam are hence similar algorithms, but Adam has been shown to slightly outperform RMSprop, making it the most promising algorithm for our task (Ruder 2017).

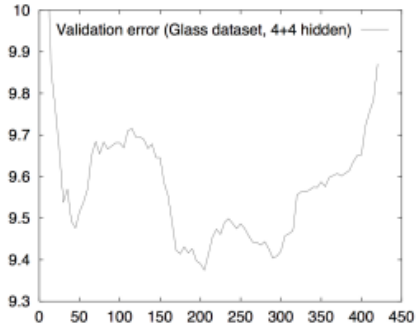
We further test different degrees of **Momentum** for SGD (for Adam and RMSProp, the momentum value is by default the estimate of the first order moment). Momentum is an algorithm, which introduces an extra variable to the SGD algorithm, which controls the direction and speed at which the parameters move through parameter space (Goodfellow et al., 2016). In other words, it helps accelerate the optimizer in the right direction and dampens fluctuations.

5.4.3 Overfitting

The goal when training a neural network is to build a model, which generalizes well over unseen data. Most standard neural networks are prone to overfitting (Prechelt, 2012), therefore it is an important issue to address in our task. Overfitting can be detected, when comparing the loss function's output of the training data with that of the evaluation data. Prechelt (2012) shows why it is a non trivial problem for neural networks with Figure 5.5(a) and Figure 5.5(b). The first figure shows an ideal validation error curve, while the second one shows a more realistic validation error curve for a neural network, which includes many valleys and local minima.

The following are popular techniques, which are designed to prevent overfitting and which we employ for this task.

(a) Ideal training and validation error curves.



(b) Real validation error curve for a neural network.

Figure 5.5: Figure from Prechelt (2012) showing the difference between ideal and real training and validation errors curves. Horizontal axis: time in training epochs, vertical axis: validation set error.

Regularization is used for neural and classical machine learning models, by adding a regularization term R to the optimization algorithm, in order to control the loss and complexity of the parameter value. The most commonly used choices for R are the L1 or L2 norm; we use the latter for this project, which is also referred to as the gaussian prior or weight decay. The L2 regularizer punishes high parameter weights, as opposed to punishing low and high parameter weights uniformly, like L1. λ_{L2} is the weight decay value.

Dropout (Srivastava et al., 2014) can be seen as a special type of ensemble method to prevent overfitting. At every training step, a new neural network is generated, where every neuron has a probability p of being masked or "dropped out." This way, it might be ignored at the current training step, but active during the next step, resulting in 2^N possible networks, where N is the total number of neurons, which can be either present or absent (Géron, 2017).

Early Stopping is another simple and effective method to prevent overfitting. It determines when to stop training, which is not always trivial. We use Keras' built in EarlyStopping method, which stops training when there has been no improvement after k epochs, returning the best model seen so far.

5.5 Implementation

All augmentation techniques and neural models were implemented using Python and commonly used Python NLP libraries, including NLTK[8], pandas[9], scikit-learn[10] and spaCy. The neural networks were implemented using Keras[11], a deep learning API based on Tensorflow with a straight-forward interface and fast implementation in Python. All word embedding techniques were implemented using the Gensim[12] library, which provides a number of operations to analyze and compare vectors. For the embeddings themselves, we use SpinningBytes pre-trained embeddings in German and Italian. We further use the Python textgenrnn[13] module for the RNN text generation.

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

[9]https://pandas.pydata.org/

[10]https://scikit-learn.org/stable/

[11]https://keras.io/

[12]https://pypi.org/project/gensim/

[13]https://pypi.org/project/textgenrnn/

6 Results

This chapter reports the results of our classification models, described in Chapter 5, for each task (German multi-class, German binary, Italian binary).

Before training the models, we pre-process and split the datasets, as described in 5.1. In order to fairly compare our models, we perform a random search, as described in 5.4, with 200 iterations and 40 epochs per model to find their best parameters. Additionally, we measure the stability of all models, by running each of them with their best parameters 30 times, to compute the mean and the variance. For neural models, this means changing the seed of the random generators, in order to initialize the weights differently each time, while for ensemble tree-based models, it means each tree uses a different subset of the data. In contrast, SVMs are not affected by randomness, as the algorithm does not rely on weight initialization or sampling.

For each model and augmentation method, we report the mean macro F1 score with its variance across 30 trials, as well as the accuracy, in accordance with reported scores in related work and officially reported competition scores. The macro-average computes precision and recall per class and then averages them to compute the F1 score, where it's best results are reached at 1 and worst at 0, making it a good metric for imbalanced datasets. While we report the mean accuracy as well to stay in line with previous work, we do not consider it to be a good measure of performance for imbalanced classes. For example, a baseline, which outputs 100 predictions correctly and 20 wrongly, would get an accuracy of 100/120 or 83%. However, if all 20 wrong predictions belong to one class and the model only predicts the majority class, the high accuracy does not reflect the performance of the model. The macro F1 score provides a more balanced measure of performance, by taking into account the F1 score of each class.

Note that in all tables, we report augmentation methods as follows: *none* meaning no augmentation; *POS* meaning augmentation using POS-tag-based substitution; *Th + value* meaning augmentation using threshold-based substitution with the indicated threshold value; *Trans* meaning augmentation with English translated samples; *Tr+Th75* meaning augmentation using both translated samples and

Aug. M.	Opt.	Batch	Drop	Stride	Kernel	LR	Filters	Mom.	λ_{L2}.
None	adam	64	0.5	3	5	0.001	64	-	0.010
POS	adam	80	0.3	3	5	0.001	32	-	0.1
Th75	rms	64	0.3	5	4	0.007	64	0.7	0.0001
Th50	rms	80	0.4	1	5	0.001	64	0.9	0.01
Th25	rms	64	0.4	1	4	0.005	128	0.9	0.0001
Trans	adam	32	0.1	5	5	0.001	64	-	0.1
Tr+Th75	rms	32	0.3	3	5	0.007	128	0.8	0.0001

Table 6.1: Best hyperparameters per augmentation method for German multi-class classification using CNN/GRU

threshold-based substitution with a threshold of 75.

For the quantitative analyses, we report the confusion matrices for each task, which help visualize the performance of a classifier for each class and interpret its results. In the matrix, columns represent the predicted relations while the rows the gold truth classes. A perfect classifier should have values only in the diagonal (showing correct predictions) while having zeros elsewhere (showing no errors). Precision corresponds to the ratio of the relevant instances among the retrieved ones, while recall is the number of relevant instances retrieved over the whole set of relevant elements. These can be easily computed with the confusion matrix.

Two popular diagnostic tools exist to analyze the trade-off between precision and recall, namely Precision-Recall (PR) curves and Receiver Operating Characteristic (ROC). ROC curves display the trade-off between the true positive rate and false positive rate, while PR curves display the trade-off between precision and recall for different threshold, making them more useful for imbalanced datasets (Davis and Goadrich, 2006). Since we are more interested in the latter trade-off, we use PR curves in our analysis. A perfect classifier should have a curve at 1.0 (the top-right part), whereas a classifier with a high precision and low recall should have high values at the top left side of the plot.

Finally, we perform a qualitative error analysis, since interpreting our models' output is an important part of predictive analysis, especially given the fact that neural networks are usually considered black boxes.

	CNN + GRU		SVM		XGBoost	
Aug. M.	Acc	Macro F1	Acc	Macro F1	Acc	Macro F1
None	66.51	30.89 ± 2.15	59.71	34.71 ± 0.00	68.10	27.71 ± 0.09
POS	66.60	29.90 ± 3.28	68.83	38.85 ± 0.00	68.10	31.85 ± 0.50
Th25	67.45	30.43 ± 4.10	**68.16**	$\mathbf{40.10 \pm 0.00}$	69.78	35.14 ± 0.00
Th50	66.75	31.30 ± 2.79	69.01	39.62 ± 0.00	68.39	30.90 ± 0.31
Th75	67.27	31.00 ± 3.16	64.16	39.45 ± 0.00	69.22	32.41 ± 0.65
Trans	67.31	$\mathbf{35.49 \pm 2.06}$	57.59	36.97 ± 0.00	67.57	$\mathbf{36.29 \pm 0.49}$
Tr+Th75	62.65	32.15 ± 3.22	63.95	37.94 ± 0.00	68.51	35.14 ± 0.17

Table 6.2: Results German multi-class classification

6.1 Results German Multi-Class Classification

6.1.1 Overall Results

Our resulting mean macro F1 scores for the German multi-class classification, shown in Table 6.2, are comparable with the reported overall mean macro F1 score of 39.71 from the GermEval 2018 competition, although we do not come close to the top performing team, which achieved an macro F1 score of 52.71. However, as our goal was not to outperform the best model, we use these scores and corresponding findings as points of comparison only. Wiegand et al. (2018) report that the best systems do not use neural approaches and SVMs were the most commonly used non-neural type of classifier. We conclude that SVMs perform well on this task, which can be confirmed by the results of our SVM baseline model, which outperforms our neural approach and XGBoost significantly.

We found that the translation-based augmentation method yields significant improvements over not using any augmentation. As can be seen in Table 6.2, the macro F1 score increases by 4.6 in the neural model, and by 8.5 in the XGBoost model, compared to the scores using no augmentation. For our SVM model, the translation-based method still achieves an improvement over 1.58, but is outperformed by all threshold-based methods, of which a threshold of 25% achieves the best results with an increase of 5.39. Surprisingly, the POS-based method performs slightly worse than using no augmentation at all in our neural model. We suspect that this may be due to the fact, that the performance of the POS tagger is negatively affected by the specific language of tweets (e.g. writing variations, misspellings). In contrast, the threshold-based method does not use any filtering based on parts of speech. Looking at the overall accuracy, we can observe that in

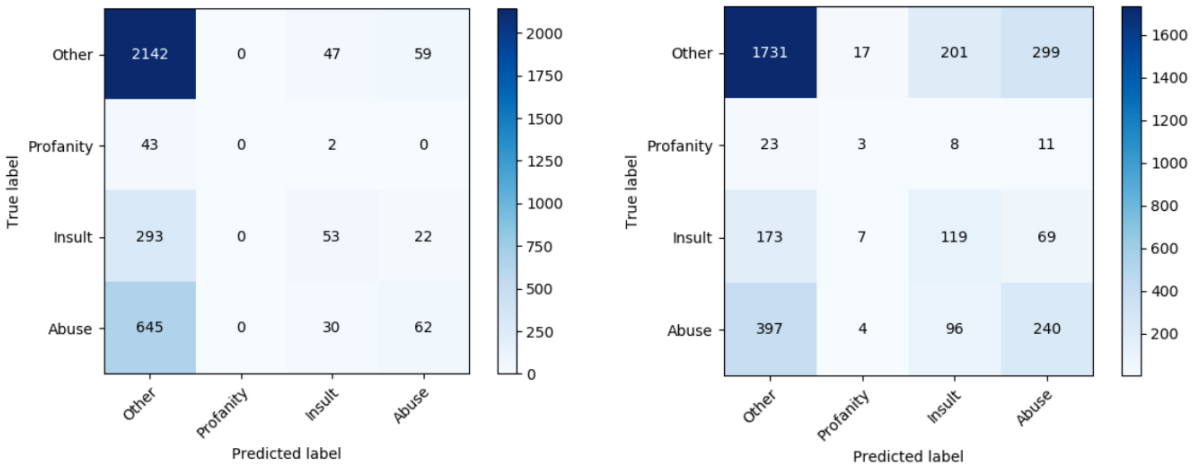


Figure 6.1: Confusion matrices of model without augmentation (left) and model achieving highest macro-average F1 score (right) in German multi-class classification

	Other			**Profanity**			**Insult**			**Abuse**		
Aug.M.	P	R	F1	P	R	F1	P	R	F1	P	R	F1
None	94.88	29.38	44.87	0.00	0.00	0.00	8.74	33.82	13.89	20.47	40.08	27.10
POS	92.88	28.51	43.63	0.00	0.00	0.00	9.46	50.03	15.91	19.23	39.39	25.84
Th25	94.51	28.00	43.20	0.00	0.00	0.00	10.73	52.97	17.85	15.74	38.09	22.28
Th50	92.41	28.97	44.11	0.00	0.00	0.00	15.17	56.80	23.94	18.29	37.98	24.69
Th75	92.32	28.92	44.04	0.00	0.00	0.00	14.23	57.24	22.79	18.80	39.56	25.49
Trans	89.09	**31.88**	46.96	03.56	**77.53**	6.81	**19.84**	42.81	**27.11**	**25.02**	37.44	**30.00**
Tr+Th75	89.48	30.44	45.43	01.19	40.68	2.00	15.81	43.76	23.23	22.42	35.81	27.58

Table 6.3: Mean precision (P), recall (R) and macro F1 scores per class and augmentation method for German multi-class classification

all models, accuracy decreases when using the translation-based or translation and threshold combination.

With our hyperparameter search for each augmentation method using our neural model, we found that all threshold-based method perform best by using RMS as optimizer and the translation-based method, which achieves the best results, uses Adam, with a low batch size (32), low dropout (0.1) and low learning rate (0.001). The values can be seen in Table 6.1.

6.1.2 Quantitative Analysis

Looking more closely at the correctly and wrongly classified samples by our neural model, we notice that in all non-translation-based augmentation methods, the Profanity class is rarely considered at all; only in a handful of all runs do we observe a few tweets (mis-)classified as containing profanity. Table 6.3 further shows that the precision and recall for the Profanity class are 0 for all non-translation-based augmentation methods and for not using any augmentation. This comes to no surprise, given that the class contains only 75 samples. As pointed out in Wiegand et al. (2018), the overall performance scores for this task were significantly lower, since all systems struggled to detect the Profanity class. Looking closely at the confusion matrices of our best model and model without augmentation, seen in Figure 6.1, we find that adding the translated samples helps raise the number of predictions for the Profanity class. Overall we can observe that more samples within the hate classes are correctly predicted by using augmentation (e.g. 119 instead of 53 for Insult), but also less non-hate samples are correctly predicted (i.e. 1731 instead of 2142). The PR curves seen in Figure 6.2 confirm this, by showing that both in terms of precision, but especially in terms of overall recall, our model with augmentation clearly performs better. Using the translation-based method, our neural models improve the Profanity recall on average by 78 percentage points, as can be seen in Table 6.3.

6.1.3 Qualitative Analysis

To better understand the performance of our model, we analyzed the correctly and wrongly predicted samples more closely. We initially noticed that a large number of samples was misclassified as Insult (201 non-hateful samples and 96 with the actual label Abuse). Looking at some relevant examples, we found that many of them labeled as Insult or Abuse contain political terms, such as Merkel (referring to German chancellor Angela Merkel), SPD, FDP (both referring to German political parties), or religious terms (Islam, muslim, jew). For instance, Examples 6.1, 6.2, and 6.3 below all include the use of several political terms.

(6.1) **Original:** Boh ich bin RICHTIG sauer, was denkt sich diese @spd @jusos ?????? Das alle Wähler so blöd sind, wie die mit Merkel verhandelt haben? Sogar Herr Seehofer hat seine Obergrenze. Leute jetzt Reicht es wirklich mit dieser SPD sorry
Translation: Ugh I'm REALLY angry, what does this @spdde @jusos ?????? That all voters are as stupid as they have negotiated with Merkel? Even Mr. Seehofer has his upper limit. People, it's really enough with this

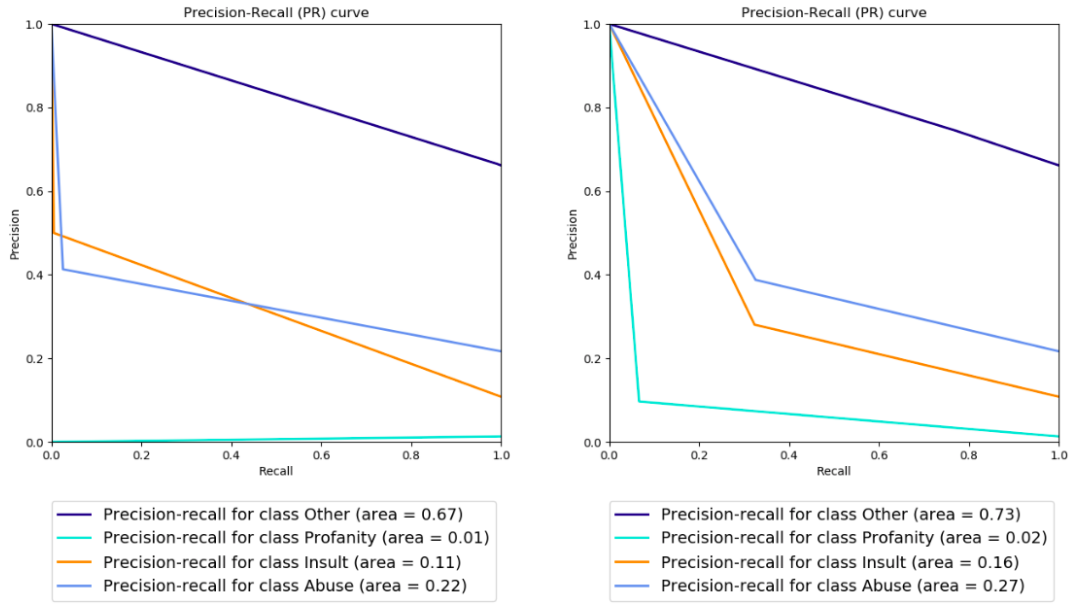


Figure 6.2: PR curves for results without augmentation (left) and model achieving highest macro-average F1 score (right) in German multi-class classification

SPD sorry
Predicted: Abuse, **Actual:** Other

(6.2) **Original:** Frau Merkel und Konsorten! Ihr seid die Sklaven der Ostküste. Jetzt lacht ihr,doch der Wind wird sich drehen,und dann gnade euch gott!
Translation: Mrs. Merkel and friends! You are the slaves of the East Coast. Now you are laughing, but the wind will turn, and then God have mercy on you!
Predicted: Insult **Actual:** Abuse

(6.3) **Original:** @POTTRlOT Hoffentlich finden die ihn und stecken ihm seine Nagelbombe in den Arsch !!!! der kleine drecks Islamist!!!!
Translation: @POTTRlOT Hopefully they will find him and stick his nail bomb in his ass !!!! the little dirt Islamist !!!!
Predicted: Insult **Actual:** Abuse

Since it is possible that certain words are statistically biased, as they appear often in hateful context, we made a list of various political, religious, and profane words and counted their occurrences in samples of each class, in order to analyze potential trends. Note that words were stemmed before, using the NLTK stemmer (Weissweiler and Fraser, 2018). Our findings can be seen in Table 6.4. We observe that

		Profanity			Insult			Abuse		
Search Term	Occ.	Pred.	Act.	Cor.	Pred.	Act.	Cor.	Pred.	Act.	Cor.
schwul	35	0	0	0	2	0	0	18	15	9
Islam	89	1	0	0	6	3	0	47	54	26
Muslim	36	0	0	0	2	1	0	22	20	13
Merkel	110	0	3	0	35	20	8	23	23	6
Flüchtling	72	0	0	0	4	2	0	20	18	6
Nazi	41	0	1	0	2	6	1	29	20	15
Deutschland	145	0	1	0	14	17	4	38	39	15
Pädophil	26	0	0	0	6	0	0	5	26	5
FDP	33	0	0	0	12	13	7	7	2	0
SPD	159	0	1	0	35	37	11	19	44	9
Türke	40	0	0	0	2	2	0	14	13	7
grün	116	1	0	0	21	16	6	24	47	12
fuck	30	1	0	0	4	12	2	19	18	13
Arsch	45	3	4	1	17	20	8	12	14	4
Bastard	9	1	0	0	2	4	2	5	5	3
Religion	38	0	0	0	0	3	0	16	16	9
Schwein	19	1	0	0	4	5	1	7	9	6

Table 6.4: Overall Appearance (Occ.) of search terms in test samples and their predicted (Pred.) vs. actual (Act.) class and correctly labeled (Cor.) samples

there indeed appears to be a correlation between words, which appear in heated or hateful contexts and their occurrence in samples classified as hateful. For instance, 24 of 36 samples containing the word *Muslim*, 24 of 30 samples containing the word *fuck*, 32 of 45 samples containing the word *Arsch*, and 19 of 33 samples containing the word *FDP*, were predicted to be hateful. The distribution of predictions further reflects the class imbalance; we can see that the samples rarely were labeled as Profanity, and even less often labeled as such by our classifier.

6.2 Results German binary Classification

6.2.1 Overall Results

For the German binary classification, the macro F1 scores are significantly higher than the fine-grained equivalent, as seen in Table 6.6, which was to be expected. Most of our methods with augmentation improve the scores compared to not using any augmentation, by up to 4.48 F-score points and the translation-based in combi-

Aug. M.	Opt.	Batch	Drop	Stride	Kernel	LR	Filters	Mom.	λ_{L2}.
None	adam	64	0.3	1	5	0.001	128	-	0.1
POS	adam	80	0.4	1	5	0.003	64	-	0.0001
Th75	adam	64	0.5	1	4	0.001	32	-	0.1
Th50	adam	80	0.1	1	5	0.007	32	-	0.01
Th25	rms	64	0.5	5	5	0.001	128	0.7	0.1
Trans	adam	64	0.3	3	5	0.003	64	-	0.2
Tr+Th75	rms	64	0.3	5	5	0.003	32	0.8	0.1

Table 6.5: Best hyperparameters per augmentation method for German binary classification using CNN/ GRU

	CNN + GRU		**SVM**		**XGBoost**	
Aug. M.	Acc	Macro F1	Acc	Macro F1	Acc	Macro F1
None	69.95	60.88 ± 2.18	67.33	60.20 ± 0.00	70.92	57.65 ± 0.19
POS	69.00	61.73 ± 2.40	72.13	60.14 ± 0.00	70.60	57.59 ± 0.12
Th25	69.93	61.24 ± 5.19	71.87	60.89 ± 0.00	71.28	59.05 ± 0.10
Th50	68.97	61.59 ± 3.17	69.63	62.56 ± 0.00	70.78	58.90 ± 0.16
Th75	70.26	61.51 ± 2.69	67.25	61.82 ± 0.00	71.78	59.91 ± 0.29
Trans	67.73	61.83 ± 2.24	66.39	61.70 ± 0.00	66.45	65.92 ± 0.17
Tr+Th75	68.95	$\mathbf{63.23 \pm 1.85}$	72.48	$\mathbf{64.68 \pm 0.00}$	70.75	$\mathbf{61.81 \pm 0.08}$

Table 6.6: Results German binary classification

nation with threshold-based method, using a threshold of 75%, outperforms all other methods across all models. The POS method achieves the least improvements. Compared to the officially reported scores, our models perform in a similar range, as in the fine-grained task; the best performing GermEval 2018 system achieved a score of 76.77, compared to the mean of 66.35 across models and lowest reported score of 49.03. With regards to the best retrieved hyperparameters for each augmentation method using our neural model, we found that using Adam and a smaller stride achieves the best results. The values can be seen in Table 6.5.

6.2.2 Quantitative Analysis

Looking at the confusion matrices of our best performing model compared to the model using no augmentation, seen in Figure 6.3, we notice that a significantly higher number of samples has been correctly classified as offensive (545 as opposed to 313), but also incorrectly as such (447 as opposed to 204). The PR curve shown in Figure 6.4 displays the trade-off; we see that the precision and recall have increased for the hate class, but consequently, the precision for the non-hate class has decreased.

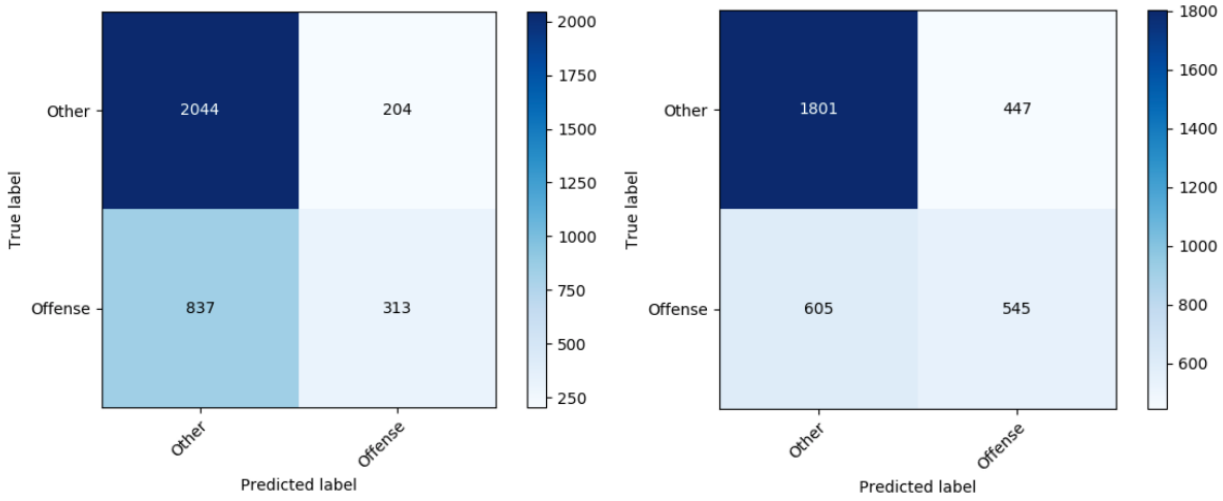


Figure 6.3: Confusion matrices of model without augmentation (left) and model achieving the highest macro-average F1 score (right) in German binary classification

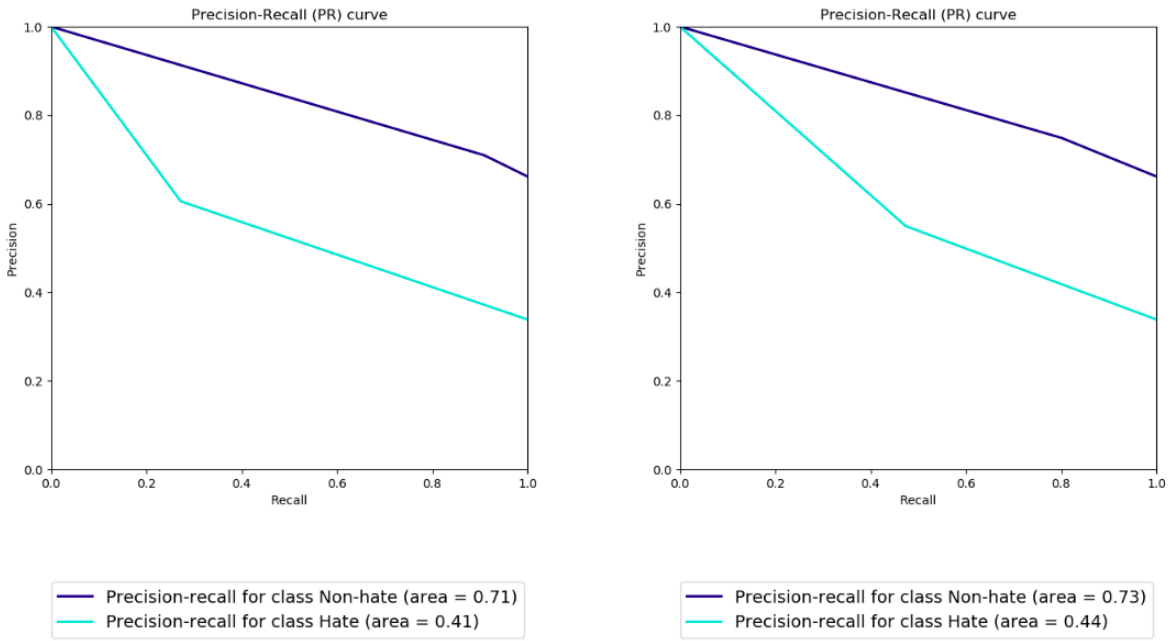


Figure 6.4: PR curves of model without augmentation (left) and best model (right) in German binary classification

6.2.3 Qualitative Analysis

Upon examining the predictions of the best model, we notice that Examples 6.1 and 6.3 in Section 6.1.3 were correctly classified as offensive, while 6.2 was still considered hate-speech.

(6.4) **Original:** @MDRINFO @KontraKulturell ab nach Hause mit dem Pack.
Translation: @MDRINFO @KontraKulturell home with the pack.
Predicted: Other, **Actual:** Offense

(6.5) **Original:** @StapelChipsYT @ichbinmuckmuck @tonino85 Weil ich ein ungewöhnlicher Schwuler bin. Glaub ich zumindest.
Translation: @StapelChipsYT @ichbinmuckmuck @tonino85 Because I am an unusual gay man. At least I think so.
Predicted: Offense , **Actual:** Other

When analyzing further predictions, we noticed that our classifier appears to be biased towards classifying those tweets as hateful, which contain lexical items that are often used in a hateful context. For instance, Example 6.4 was misclassified as non-hateful; we see that neither profanities, nor obviously hateful words are used and when considered out of context, it is possible that we would have not necessarily labeled this tweet as hateful ourselves. On the other hand, Example 6.6 contains the word *Schwuler* (gay man), which may appear more often in hateful contexts. Upon further analysis, we found that 35 tweets in the test set contain the word *schwul* in German, of which 27 were classified as offensive by our classifier, but only 12 were actually offensive. To see if there is a correlation between tweets being classified as hateful, and words, which generally appear in hateful or heated contexts, and their occurrence in hateful tweets, we performed a relative word count (number of words appearing in hateful tweets, compared to overall occurrence), using similar words as in Table 6.4, which may appear in more or less hateful contexts. Figure 6.5 visualizes our findings.

We can see that our classifier is biased toward classifying tweets as hateful, which contain words that are directed at certain groups or people, such as *Nazi, Koran, schwul* and *Merkel*. More neutral words on the other hand, such as *Deutschland, Amerika, Kirche* and *radikal* are less frequently classified as hateful. The words found in those tweets, that have been annotated as hateful, as well as our correctly predicted hateful tweets appear more evenly balanced.

Figure 6.5: Wordclouds showing German words and their relative frequencies in hateful tweets, based on our model's predictions (left), annotations (center), and agreement between both (right)

Aug. M.	Opt.	Batch	Drop	Stride	Kernel	LR	Filters	Mom.	λ_{L2}.
None	adam	32	0.3	5	5	0.007	32	-	0.1
POS	rms	32	0.5	3	5	0.005	64	0.9	0.1
Th75	rms	64	0.3	5	5	0.007	128	0.9	0.1
Th50	rms	64	0.1	5	5	0.009	64	0.8	0.01
Th25	rms	32	0.2	5	5	0.001	64	0.9	0.1
Trans	adam	32	0.5	5	5	0.007	32	-	0.01
Tr+Th75	rms	64	0.1	5	5	0.007	64	0.9	0.1

Table 6.7: Best hyperparameters per augmentation method for Italian binary classification using CNN/ GRU

6.3 Results Italian binary Classification

6.3.1 Overall Results

For the Italian classification task, our findings show that, while the best results are achieved with the threshold-based method with a threshold of 25%, our retrieved mean macro F1 scores differ only up to 1.39 percent points (see Table 6.8). Given a variance of up to 2.74, we cannot make claims of improvement, based on these scores. However, our baseline models show a more significant improvement; for the SVM, we achieve an increase in mean macro F1 score over 2.57, compared to not using any augmentation, and for XGBoost, an increase of 3.8 respectively. For the SVM, we achieve highest score using the threshold-based method with a threshold of 50%, while for XGBoost, the best results are achieved with a threshold of 25%, supporting the neural model's result. Surprisingly, our models perform worst with the translation-based method, resulting in lower scores than using no augmentation.

	CNN + GRU		SVM		XGBoost	
Aug. M.	Acc	Macro F1	Acc	Macro F1	Acc	Macro F1
None	75.10	69.90 ± 2.74	70.20	68.18 ± 0.00	63.30	65.35 ± 0.46
POS	74.25	69.01 ± 2.42	74.10	70.51 ± 0.00	72.20	68.17 ± 0.10
Th25	77.70	$\mathbf{70.40 \pm 1.71}$	70.75	70.32 ± 0.00	69.24	$\mathbf{69.24 \pm 00.24}$
Th50	76.20	69.11 ± 1.99	72.90	$\mathbf{70.75 \pm 0.00}$	70.40	66.12 ± 0.33
Th75	72.70	69.80 ± 1.46	72.50	68.12 ± 0.00	71.40	67.18 ± 0.53
Trans	75.20	69.18 ± 1.76	68.10	64.49 ± 0.00	60.50	60.05 ± 0.13
Tr+Th75	74.90	69.60 ± 1.27	68.52	65.20 ± 0.00	67.30	65.32 ± 0.28

Table 6.8: Results Italian binary classification

With regards to overall accuracy, we can see that the threshold-based methods generally perform better, while the translation-based method shows no significant improvement for our neural model, and worsens the accuracy for our baseline models.

When comparing our results with the official results from the Evalita competition, we find that our systems are comparable to the average results; the winning system achieved a macro F1 score of 79.93, while the worst system got a score of 40.33.

Regarding the best retrieved hyperparameters for each augmentation method using our neural model, we notice that using RMS and a smaller batch size, as well as a kernel size of 5 achieves the best results. The values can be seen in Table 6.7.

6.3.2 Quantitative Analysis

Looking at the confusion matrices seen in Figure 6.6, we notice a slight increase in correctly classified offensive samples and decrease in falsely classified samples. The PR curves shown in Figure 6.7 that the increase in precision is slightly higher for the non-hate class, than the hate class. We further conducted an analysis of the mean precision, recall and macro F1 score for both classes and could see a clear steady decrease in precision for the non-hate class and simultaneous increase for the hate class. Table 6.9 shows the trade-off in more detail. We can see that for the Threshold 25 augmentation method, which has the highest overall F1-score for this task, the precision of the non-hate class and recall of the hate class are higher than for the translation-based methods, while the non-hate recall and hate precision are lower, but not as low as the other threshold-based methods. We conclude that this model best balances the trade-off between precision and recall for each class, where the others perform worse in one or the other. We further notice that the

	Non Hate			Hate		
Aug.M.	P	R	F1	P	R	F1
None	87.94	63.95	74.06	50.93	79.90	62.20
POS	85.72	63.95	73.26	51.39	77.99	61.95
Th75	85.13	64.51	73.40	52.78	78.20	63.02
Th50	86.32	63.57	73.22	50.62	78.19	61.45
Th25	84.99	65.50	73.98	55.09	78.91	64.88
Trans	79.73	66.10	72.28	58.95	74.37	65.77
Tr+Th75	77.81	68.30	72.75	**64.81**	73.87	**69.05**

Table 6.9: Mean precision, recall and macro-average F1 scores per class in Italian binary classification

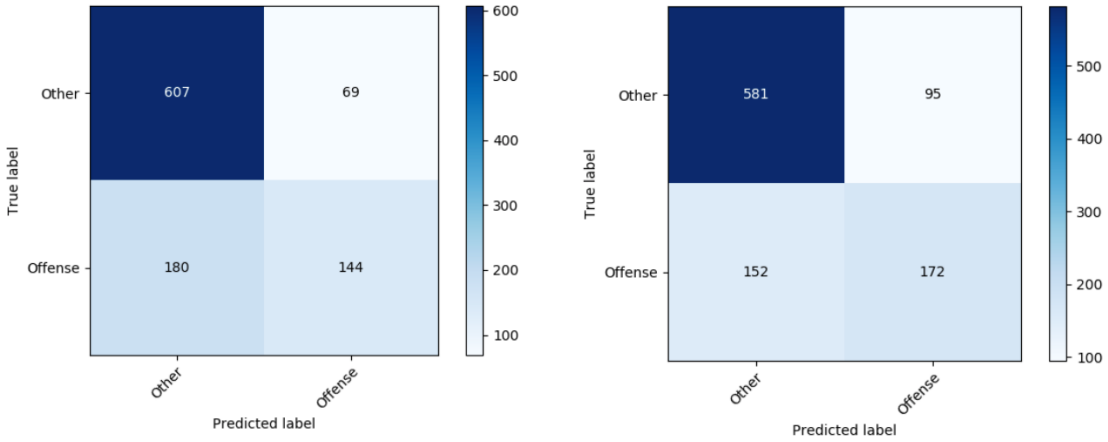


Figure 6.6: Confusion matrices of model not using augmentation (left) and model achieving highest macro-average F1 score (right) in Italian binary classification

translation-based methods manage to capture hateful tweets best; as can be seen in Table 6.9, our method combining Translation and Threshold 75 achieves an absolute increase in hate precision of approximately 14% and 6.85% in macro F1 score. Translation alone achieves an absolute increase of approximately 8% and 3.57% in macro F1 score. However, these methods perform worst with regards to identifying non-hateful samples. Not using any augmentation results in high precision for non-hateful samples but low precision for hateful samples.

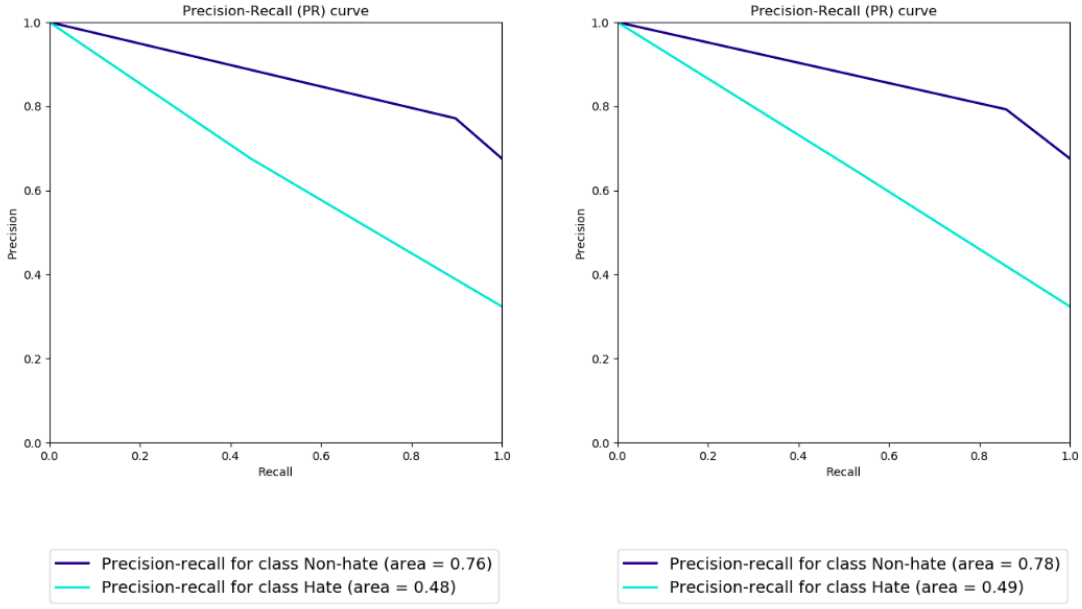


Figure 6.7: PR curves of model without augmentation (left) and model achieving highest macro F1 score (right) in Italian binary classification

6.3.3 Qualitative Analysis

Looking more closely at some of the generated predictions, we notice similar trends in terms of statistical biases, as already seen in the German examples. The occurrence of terms, often used in hateful contexts appears to heighten the chances of the sample being classified as hateful. Tweets that do not contain any hateful or biased terms, such as Examples 6.9 and 6.10 are misclassified as non-hateful, while ones that do contain biased terms, such as Muslims, Islamic, or Religion are misclassified as hateful (see Examples 6.6, 6.7, and 6.8).

(6.6) **Original:** Tutti i MMS parlano dei morti islamici in Canada. Ma non parlano dei cristiani che giornalmente vengono uccisi dagli islamici.
Translation: All MMS talk about Islamic deaths in Canada. But they do not talk about Christians who are killed by Muslims every day.
Predicted: Hate, **Actual:** Non-hate

(6.7) **Original:** Devono rovinare il Natale all'occidente perché invidiosi della religione altrui. #animalifanatici #berlino #terrorismo
Translation: They must spoil Christmas in the West because they are envious of the religion of others. #animalifanatici #berlino #terrorismo
Predicted: Hate, **Actual:** Non-hate

(6.8) **Original:** Non tanto sincero anche chi difende gli immigrati e condanna i gay ...tutti uguali si dovrebbe essere...credo
Translation: Not so sincere even those who defend immigrants and condemn gays ... everyone should be the same ... I think
Predicted: Hate, **Actual:** Non-hate

(6.9) **Original:** #quintacolonna in diretta da Roma: questi sarebbero i poveri immigrati che scappano dall'isis e dalla guerra??
Translation: #quintacolonna live from Rome: would these be the poor immigrants who escape from the isis and the war ??
Predicted: Non-hate, **Actual:** Hate

(6.10) **Original:** @PiazzapulitaLA7 @salvaperiodico gli stranieri tutto dovuto. Gli italiani. Fame e zero diritti ...
Translation: @PiazzapulitaLA7 @salvaperiodico foreigners are allowed all. Italians. Hunger and zero rights ...
Predicted: Non-hate, **Actual:** Hate

6.4 Discussion

Having looked at and analyzed our models' performances using data augmentation methods, this section will summarize and discuss our findings. Our research hypothesis stated that data augmentation can improve classification results in German and Italian. We have shown that this is the case, albeit to different degrees. Having tested several models, we found that the neural approach performs worse than classical machine learning approaches in the multi-class setting, whereas all models perform similarly in the binary tasks. Since neural models rely on large amounts of data, we conclude that the small number of samples, even with additional augmented samples, keeps the model from learning adequate representations. We further see a higher degree of variance in the results, showing that neural models are sensitive to weight initialization.

Regarding our first research question, we conclude that data augmentation techniques based on distributed word representation substitutability improve hate speech classification, where the POS-based methods performs slightly worse and threshold-based methods best. As mentioned previously, we believe this to be due to the fact that our tagger might not correctly tag all samples, given the language used in tweets is highly colloquial. For the threshold-based approach, we experimented with different thresholds and did not find any clear indication that one performs overall

significantly better than the others.

These methods were slightly outperformed by our translation-based methods, which shows with regards to our second research question, that we can indeed transfer knowledge from a comparable dataset in a high-resource language by means of automatic translation. However, the term *outperformed* should be taken with a grain of salt. While we found that overall, the translation-based methods increase precision on the hate-class, particularly in the multi-class setting, where it improves the problem of data imbalance, they also decrease the precision on the non-hate class. This leads to an interesting question: are we more interested to detect hateful content precisely, taking into account that some samples may be missed, or do we prefer to catch as many hateful samples as possible, with a risk of many false positives? There is no clear answer, as both may be useful. On the one hand, we could imagine a system, which automatically detects hateful tweets and sends them to humans for further validation; in this case, it would be better to have a high recall, covering all the real hateful speech, but also including false positives, which would be processed by humans in a later step. On the other hand, if a system is supposed to automatically hide offensive content, it is more desirable to have a system with high precision and low recall, since removing comments might obstruct our freedom of speech and expression. We can conclude that data augmentation is more useful for the former case, as we are generally able to catch more hateful content.

With regards to our third research question, we have found data augmentation to be more beneficial for fine-grained than for binary classification. While we can see a general trend in the better detection of hateful tweets, augmentation works especially well to increase the precision of each class, as seen in the German multi-class task. However, the performance of our model is still dependent on the original dataset. The Profanity class containing only 75 samples could not be detected easily, even with additional translated samples.

6.5 Limitations and Future Work

While we have been able to show the efficacy of data augmentation for hate speech classification, the task is far from solved and this project can be extended in many different ways. We would like to address a few of these potential directions for future research.
In this project, our focus was to show that data augmentation helps improve the performance of our models. However, we are aware that our models could have generally performed better, for instance by using ensembling methods, as is often

done in competitions, such as GermEval or Kaggle. For instance, the winning team in the GermEval 2018 competition used an ensemble of random forests with a large amount of feature engineering (Montani, 2018). It would be interesting to apply our augmentation techniques to such a model, to see if the trends we have seen hold true in combination with different types of features and higher performance.
Furthermore, we only used one type of word embedding for both the augmentation based on distributed word substitutability, as well as for the classification task itself. However, many more types of embeddings exist, such as emoji embeddings (Eisner et al., 2016) or most recently contextualized embeddings (Devlin et al., 2019), where part of the words are masked and their position is used to infer information in an unsupervised manner. Another recent contextualized method is ELMO (Peters et al., 2018), which has shown promising results. Furthermore, for one of our augmentation experiments, involving natural language generation, we did not achieve good enough results to consider this method further. Since this method heavily depends on a large corpus, we do not find it to be a useful method for a low-resource setting. It could be interesting to see, if generating new samples from a mixture of the original dataset and added translated samples would change the performance significantly. Lastly, we would like to emphasize that all models and augmentation experiments still rely on the original data and as we have shown by considering related work, competition results, and our own experiments, two of the great limitations of this field are data sparsity and lack of a clear definition of hate speech, leading to different annotations and hindering comparability and transfer of knowledge.

7 Conclusion

In this work, we analyzed the hypothesis that data augmentation techniques can improve the detection of hateful content in lower resource settings. We first introduced the topic of hate speech detection and showed that the task is non-trivial, largely because of the lack of annotated data and clear definition of hate speech. We further showed how previous work has addressed this task. We then provided an overview over the German, Italian and English datasets used in this project, followed by preliminary background information for the Machine Learning and NLP techniques, which were needed for our project. We further described the three augmentation methods, which we use to test our hypothesis: the first method implements a Recurrent Neural Network (RNN) to generate more training samples automatically; the second method uses distributed word representations (embeddings) to substitute words with close neighbors, using a cosine distance threshold of their respective word vectors. The third method adds automatically translated samples from a different dataset in English to the datasets in German and Italian. As generating new samples did not produce any usable samples, presumably due to the small number of samples, we did not pursue this method further, but see it as a method that could be used in the future, if larger training sets become available.

We further described the neural and classical machine learning approaches, which we implemented for the classification tasks; for the neural approach, we implemented a hybrid CNN-GRU architecture, which achieved promising results in Hemker (2018) and Zhang and Luo (2018). For the classical approach, we used XGBoost, an implementation of gradient boosted decision trees, as well as SVMs.

Our findings show that using data augmentation techniques can improve hate speech detection in both Italian and German data. We experimented with different augmentation methods and introduce a novel, hybrid method using distributional-semantics-based substitution and translation-based augmentation, which achieves promising results across all classification tasks. Translation-based augmentation methods have been particularly successful for German data, and could increase the absolute recall for the most under-represented class on average by up to 78%. For Italian, we could increase the absolute precision of hate detection by up to approximately 14%. We

discuss how the trade-off between precision and recall is an important consideration, depending on the ultimate goal when creating a hate-speech detection model.

We concluded this work by suggesting possible extensions to this project, including the use of different types of machine learning architectures and embeddings. We pointed out that data sparsity and labeling inconsistencies continue to be problematic and should be further addressed in the future.

Glossary

Accuracy Accuracy is a measure of how many observations have been correctly classified by a machine learning model, compared to all observations.

Ensemble Learning Ensemble learning is a method to learn a strong classifier by combining multiple weak classifiers.

F1-Score The F1-Score is the harmonic mean between Precision and Recall, used to measure a model's performance.

Inter-Annotator Agreement Natural Language Processing tasks usually require large amounts of human-annotated data. However, different people might not always label the same information the same way. Inter-Annotator Agreement is a measure, which shows to what extend several annotators agree or disagree in an annotation task.

POS Tagging Part-of-Speech (or POS) Tagging is the task of automatically assigning a word classes (e.g. noun, adjective, verb) to each word within a text, helping to disambiguate its meaning.

Precision Precision is the fraction of relevant instances among the retrieved instances.

Recall Recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

Tokenization In NLP, tokenization means segmenting for instance a sentence or a text into so called "tokens", or smaller pieces, often single words.

References

P. Badjatiya, S. Gupta, M. Gupta, and V. Varma. Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion (WWW 2017)*, pages 759–760, 2017.

Y. Bengio. Practical Recommendations for Gradient-based Training of Deep Architectures. Neural networks: Tricks of the Trade, pages 437–478. Springer, 2012.

J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146, 2017.

C. Bosco, F. Dell'Orletta, F. Poletto, M. Sanguinetti, and M. Tesconi. Overview of the EVALITA 2018 Hate Speech Detection Task. *EVALITA Evaluation of NLP and Speech Tools for Italian*, pages 67–74, 2018.

P. Burnap and M. L. Williams. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making: Machine Classification of Cyber Hate Speech. *Policy & Internet*, 7(2):223–242, June 2015.

P. Burnap and M. L. Williams. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(1), Dec. 2016.

P. Burnap, O. F. Rana, N. Avis, M. Williams, W. Housley, A. Edwards, J. Morgan, and L. Sloan. Detecting Tension in Online Communities with Computational Twitter Analysis. *Technological Forecasting and Social Change*, 95:96–108, June 2015.

A. Caines, S. Pastrana, A. Hutchings, and P. Buttery. Aggressive Language in an Online Hacking Forum. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 66–74, 2018.

T. Chen and C. Guestrin. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 785–794, 2016.

Y. Chen, Y. Zhou, S. Zhu, and H. Xu. Detecting Offensive Language in Social Media to Protect Adolescent Online Safety. *Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust*, pages 71–80, 2012.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

M. Cieliebak, J. M. Deriu, D. Egger, and F. Uzdilli. A Twitter Corpus and Benchmark Resources for German Sentiment Analysis. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media (SocialNLP 2017)*, pages 45–51, 2017.

M. Dadvar and K. Eckert. Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study. *arXiv preprint arXiv:1812.08046*, 2018.

T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the eleventh International AAAI Conference on Web and Social Media (ICWSM 2017)*, pages 512–515, Mar. 2017.

J. Davis and M. Goadrich. The Relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 233–240, 2006.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019.

K. Dinakar, R. Reichart, and H. Lieberman. Modeling the Detection of Textual Cyberbullying. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM-11)*, pages 11–17, 2011.

N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati. Hate Speech Detection with Comment Embeddings. *Proceedings of the 24th International Conference on World Wide Web (WWW 2015)*, pages 29–30, 2015.

dProgrammer. Simple RNN vs GRU vs LSTM : Difference lies in More Flexible Control. Apr 2019. URL `http://dprogrammer.org/rnn-lstm-gru`.

J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

B. Eisner, T. Rocktäschel, I. Augenstein, M. Bošnjak, and S. Riedel. emoji2vec: Learning Emoji Representations from their Description. *Proceedings of the 4th International Workshop on Natural Language Processing for Social Media at EMNLP 2016*, pages 48–54, 2016.

M. ElSherief, S. Nilizadeh, D. Nguyen, G. Vigna, and E. Belding. Peer to Peer Hate: Hate Speech Instigators and Their Targets. *Proceedings of the Twelfth International AAAI Conference on Web and Social Media (ICWSM 2018)*, pages 52–61, Apr. 2018.

D. Fišer, T. Erjavec, and N. Ljubešić. Legal Framework, Dataset and Annotation Schema for Socially Unacceptable Online Discourse Practices in Slovene. *Proceedings of the First Workshop on Abusive Language Online (ALW1)*, pages 46–51, 2017.

P. Fortuna, J. Ferreira, L. Pires, G. Routar, and S. Nunes. Merging Datasets for Aggressive Text Identification. *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC 2018)*, pages 128–139, 2018.

B. Gambäck and U. K. Sikdar. Using Convolutional Neural Networks to Classify Hate-Speech. *Proceedings of the First Workshop on Abusive Language Online (ALW1)*, pages 85–90, 2017.

L. Gao, A. Kuppersmith, and R. Huang. Recognizing Explicit and Implicit Hate Speech Using a Weakly Supervised Two-path Bootstrapping Approach. *arXiv preprint arXiv:1710.07394*, Oct. 2017.

S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos. Convolutional Neural Networks for Toxic Comment Classification. *Proceedings of the 10th Hellenic Conference on Artificial Intelligence (SETN 2018)*, 2018.

N. D. Gitari, Z. Zhang, H. Damien, and J. Long. A Lexicon-based Approach for Hate Speech Detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230, Apr. 2015.

Y. Goldberg. *Neural Network Methods for Natural Language Processing*, volume 10. Morgan & Claypool Publishers, 2017.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.

I. Gunasekara and I. Nejadgholi. A Review of Standard Text Classification Practices for Multi-label Toxicity Identification of Online Content. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 21–25, 2018.

A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2017.

K. Hemker. Data Augmentation and Deep Learning for Hate Speech Detection. Master's thesis, Imperial College London, 2018.

B. Jafarpour, S. Matwin, et al. Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 107–114, 2018.

M. Karan and J. Šnajder. Cross-domain Detection of Abusive Language Online. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 132–137, 2018.

Y. Kim. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Oct. 2014.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, Dec. 2014.

R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor. Predictive Embeddings for Hate Speech Detection on Twitter. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 26–32, 2018.

R. Magu and J. Luo. Determining Code Words in Euphemistic Hate Speech Using Word Embedding Networks. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 93–100, 2018.

S. Malmasi and M. Zampieri. Detecting Hate Speech in Social Media. *arXiv: 1712.06427*, Dec. 2017.

Y. Mehdad and J. Tetreault. Do Characters Abuse More Than Words? *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2016)*, pages 299–303, 2016.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

J. P. Montani. TUWienKBS at GermEval 2018: German Abusive Tweet Detection. *14th Conference on Natural Language Processing (KONVENS 2018)*, pages 45–50, 2018.

H. Mubarak, K. Darwish, and W. Magdy. Abusive Language Detection on Arabic Social Media. *Proceedings of the First Workshop on Abusive Language Online (ALW)*, pages 52–56, Aug. 2017.

C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang. Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*, pages 145–153, 2016.

J. H. Park and P. Fung. One-step and Two-step Classification for Abusive Language Detection on Twitter. *arXiv:1706.01206 [cs]*, June 2017.

R. Pascanu, T. Mikolov, and Y. Bengio. On the Difficulty of Training Recurrent Neural Networks. *Proceedings of the International Conference on Machine Learning (ICLM)*, pages 1310–1318, 2013.

J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. Deeper Attention to Abusive User Content Moderation. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1125–1135, Sept. 2017.

J. Pennington, R. Socher, and C. Manning. Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365*, 2018.

M. Petrocchi and M. Tesconi. Hate me, hate me not: Hate speech detection on Facebook. *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95, 2017.

F. Poletto, M. Stranisci, M. Sanguinetti, V. Patti, and C. Bosco. Hate Speech Annotation: Analysis of an Italian Twitter Corpus. *Proceedings of the Fourth Italian Conference on Computational Linguistics (CLiC-it 2017)*, pages 263–268, 2017.

L. Prechelt. Early Stopping — But When? Neural Networks: Tricks of the Trade: Second Edition, pages 53–67. Springer, 2012.

A. H. Razavi, D. Inkpen, S. Uritsky, and S. Matwin. Offensive Language Detection Using Multi-level Classification. *Advances in Artificial Intelligence*, 6085:16–27, 2010.

B. Ross, M. Rist, G. Carbonell, B. Cabrera, N. Kurowsky, and M. Wojatzki. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. *arXiv:1701.08118 [cs]*, Jan. 2017.

M. Sanguinetti, F. Poletto, C. Bosco, V. Patti, and M. Stranisci. An Italian Twitter Corpus of Hate Speech against Immigrants. *Proceedings of the 11th Language Resources and Evaluation Conference (LREC 2018)*, pages 2798–2805, May 2018.

A. Schmidt and M. Wiegand. A Survey on Hate Speech Detection using Natural Language Processing. *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media (SocialNLP 2017)*, pages 1–10, 2017.

A. Schofield and T. Davidson. Identifying Hate Speech in Social Media. *XRDS: Crossroads, The ACM Magazine for Students*, 24(2):56–59, Dec. 2017.

L. Silva, M. Mondal, D. Correa, F. Benevenuto, and I. Weber. Analyzing the Targets of Hate in Online Social Media. *Proceedings of the Tenth International AAAI Conference on Web and Social Media (ICWSM 2016)*, pages 687–690, 2016.

V. Singh, A. Varshney, S. S. Akhtar, D. Vijay, and M. Shrivastava. Aggression Detection on Social Media Text Using Deep Neural Networks. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 43–50, 2018.

S. O. Sood, E. F. Churchill, and J. Antin. Automatic Identification of Personal Insults on Social News Sites. *Journal of the American Society for Information Science and Technology*, 63(2):270–285, Feb. 2012.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a Simple Way to prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

B. van Aken, J. Risch, R. Krestel, and A. Loser. Challenges for Toxic Comment Classi[U+FB01]cation: An In-Depth Error Analysis. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42, 2018.

C. Wang. Interpreting Neural Network Hate Speech Classi[U+FB01]ers. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 86–92, 2018.

N. Wang, B. Varghese, and P. D. Donnelly. A Machine Learning Analysis of Twitter Sentiment to the Sandy Hook Shootings. *2016 IEEE 12th International Conference on e-Science (e-Science)*, pages 303–312, Sept. 2016.

W. Warner and J. Hirschberg. Detecting Hate Speech on the World Wide Web. *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, June 2012.

Z. Waseem and D. Hovy. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. *Proceedings of the NAACL Student Research Workshop*, pages 88–93, 2016.

Z. Waseem, T. Davidson, D. Warmsley, and I. Weber. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. *Proceedings of the First Workshop on Abusive Language Online (ALW1)*, pages 78–84, May 2017.

L. Weissweiler and A. Fraser. Developing a Stemmer for German Based on a Comparative Analysis of Publicly Available Stemmers. *Language Technologies for the Challenges of the Digital Age*, 10713:81–94, 2018.

M. Wiegand, M. Siegel, and J. Ruppenhofer. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. *14th Conference on Natural Language Processing (KONVENS 2018)*, pages 1–10, 2018.

Z. Wu, N. Kambhatla, and A. Sarkar. Decipherment for Adversarial Offensive Language Detection. *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 149–159, 2018.

E. Wulczyn, N. Thain, and L. Dixon. Ex machina: Personal attacks seen at scale. *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*, pages 1391–1399, 2017.

G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose. Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus. *Proceedings of the 21st ACM international conference on Information and knowledge management (CIKM 2012)*, pages 1980–1984, 2012.

D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards. Detection of Harassment on Web 2.0. *Proceedings of the Content Analysis in the WEB*, 2:1–7, 2009.

A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *arXiv:1804.09541 [cs]*, Apr. 2018.

X. Zhang, J. Zhao, and Y. LeCun. Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems (NIPS 2015)*, pages 649–657, Sept. 2015.

Z. Zhang and L. Luo. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter. *Semantic Web*, (Preprint), Feb. 2018.

H. Zhong, H. Li, A. C. Squicciarini, S. M. Rajtmajer, C. Griffin, D. J. Miller, and C. Caragea. Content-Driven Detection of Cyberbullying on the Instagram Social Network. *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI 2016)*, pages 3952–3958, 2016.