MASTER'S THESIS 2020

# Exploring Methods for Hate Speech Detection in Text

Lucas Molsby


ISSN 1650-2884
LU-CS-EX: 2020-31

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-31

# Exploring Methods for Hate Speech Detection in Text

**Lucas Molsby**

# Exploring Methods for Hate Speech Detection in Text

Lucas Molsby
tpi15lmo@student.lu.se

June 25, 2020

Master's thesis work carried out at The Swedish Security Service.

Supervisor: Pierre Nugues, pierre.nugues@cs.lth.se
Examiner: Jacek Malec, jacek.malec@cs.lth.se

**Abstract**

Detection of hate speech can be used for many applications. Most commonly, it is used for creating a safe and just setting for online communication, but it can also be an asset when working with prevention of violent extremism. In this report, we train classifiers to detect hate in the Gab Hate Corpus, a corpus collected from the social media platform Gab. Our results show that, by fine-tuning pre-trained models and excluding a selection of data, we can outperform the current state of the art on this task.

**Keywords**: Hate speech, Violent Extremism, Natural Language Processing, Machine Learning, Neural Networks, Deep Learning, Transformers

# Acknowledgements

Firstly, I would like to thank The Swedish Security Service for giving me the opportunity to carry out this work and for providing GPUs for training. I would like to give a special thanks to my supervisor at The Swedish Security Service who has contributed a lot to this work by always being at ones disposal for brainstorming, discussions and for acquiring the dataset that has been used in this work. Lastly, I would like to thank my supervisor Pierre Nugues at LTH for giving me constant feedback and guidance throughout the work and for always keeping me on my toes.

# Contents

# Chapter 1
# Introduction

On the morning of October 27, 2018, Robert Gregory Bowers entered a synagogue in Pittsburgh, USA and took the life of eleven people and injured six. Bowers had an account on a social media platform, Gab, where he had earlier posted anti-Semitic comments against the Hebrew Immigrant Aid Society (HIAS). On the day of the mass shooting, he posted the following:

> "I can't sit by and watch my people get slaughtered. Screw your optics, I'm going in." – Robert Gregory Bowers

## 1.1 Background

Today, there are a countless number of social medias available on the internet. The social media platforms are available to anyone who has access to the internet and are used by all age groups from children, students, and elders to even presidents and many others. As a result, we have an immense collection of platforms used for expressing opinions, thoughts and feelings. While this creates extraordinary opportunities for discussions and simplifies communication, it also comes with the risk of setting the scene for bullying, abuse, offensive speech, and hate speech.

It is not uncommon that actors of terrorism share opinions, feelings and, sometimes even, action plans on social media. In some cases, we can see that terrorists have posted implications of an action on the actual day of, or day before, an action. As mentioned in the beginning of this chapter, in the Pittsburgh synagogue shooting, Robert Bowers posted an offensive message on Gab on the same day that he performed the mass shooting. Similar traits can be found in other acts of terror. On April 23, 2018 Alek Minassian killed 10 people and injured 16 while ramming a van through a street in Toronto, Canada. Shortly before the attack Alek posted on Facebook:

> Private (Recruit) Minassian Infantry 00010, wishing to speak to Sgt 4chan please.

> C23249161. The Incel Rebellion has already begun! We will overthrow all the Chads and Stacys! All hail the Supreme Gentleman Elliot Rodger!

To clarify, Incel, refers to *Involuntary celibate*, members of an online subculture who defines themselves as unable to find romantic or sexual company despite desiring it. Chads and Stacys are, by Incels definition, popular, attractive, sexually active men and women respectively. Elliot Rodger was an Incel who performed an attack in 2014, killing 6 and injuring 14 others. Because we can trace terrorists who share their opinions and thoughts online, an opportunity arises to detect and prevent attacks at a planning stage by analyzing data from online forums and social media.

The Swedish Security Service is interested in detecting hate speech in text in order to use it as an asset in prevention of violent extremism. The goal is to create a model that can identify hate speech in text and thus make screening of text much less time consuming. The Cambridge dictionary defines hate speech as public speech that expresses hate or encourages violence toward a person or group based on something such as race, religion, sex, or sexual orientation. Detection of hate speech is a part of an ongoing effort to trace and limit verbal abuse against targeted communities. The negative effects on these communities from experiencing hate speech are often neglected but nevertheless very important (Gelber and McNamara, 2016).

Hate speech is commonly found on the internet, particularly in social media. Many, but not all, social media platforms attempt to prevent hate speech from appearing in order to create a more just and safe environment. Twitter writes in their Hateful conduct policy that they are:

> committed to combating abuse motivated by hatred, prejudice or intolerance, particularly abuse that seeks to silence the voices of those who have been historically marginalized.

In this work, we will investigate and explore methods of detecting hate speech in text in order to create a software that can classify and select parts of text that most likely contain hate speech. To do this, we will work with a corpus gathered from a social media platform known for its far right user-base that has been annotated by multiple instructed annotators. As methods, we will consider LSTM (Section 3.2.1) models at first and later turn to Transformer-based models which have turned out to be very promising because of their superior increase in performance on multiple natural language processing (NLP) tasks. Transformer-based models report (as of today) state-of-the-art performance on multiple text classification tasks (e.g. The AG News corpus and The DBpedia ontology dataset, both from Zhang et al. (2015)) and Sentiment analysis tasks (e.g. IMDB dataset (Maas et al., 2011) and The Stanford Sentiment Treebank (Socher et al., 2013)).

## 1.2 Related Work

### 1.2.1 The Gab Hate Corpus

This work is very much based on the work by Kennedy et al. (2020a), who introduced the Gab Hate Coprus (GHC). The article is a collaboration between the Departments of Computer

Science, Psychology, and Political Science at the University of Southern California. We will dive into the Corpus and analyze it in detail in Section 2.2. For now it is sufficient to know that GHC is a dataset consisting of posts from a social media site called Gab. Posts are annotated as hate or non-hate according to the *hate-based rhetoric* described in the article. The authors also introduce a baseline model for detecting hate speech. The GHC is the the largest theoretically-justified, annotated corpus of hate speech.

Work done on this corpus include Kennedy et al. (2020b). In this paper, the authors describe the difficulties of hate speech classifiers dealing with unbalanced data. Their results show that classifiers have problems with group identifiers such as "gay" or "black". Namely, the issue is to distinguish if the words are used in a offensive/prejudice way or not. We will call this *group identifier bias*.

To deal with this problem, they introduce a novel regularization technique, which improves the performance of classifiers by limiting false negatives. The reported classifiers were trained on GHC. Each classifier was run 10 times and the mean value and standard deviation were reported in the paper. On the average, the best performing classifier achieves a mean F1 score of 69.52 with a standard deviation of 1.3.

### 1.2.2 Offensive Language

One of the most noted offensive language task is OffensEval, first introduced in 2019 in (Zampieri et al., 2019). The task included three subtasks:

1. Subtask A was to identify offensive speech,

2. subtask B to classify the type of offense and

3. C to identify the target of the offense.

The data consisted of a set of 14,100 annotated English posts from the social media site Twitter, of which 4,640 were annotated as offensive.

In OffensEval 2020, a similar task was proposed, this time in multiple languages including English, Danish, Arabic, Greek, and Turkish. Nearly 800 teams signed up in 2019 and 115 of them submitted results. The most popular, subtask A had 104 participants and among the top-10 teams, seven used a pre-trained model called BERT, which we will cover in more detail in Section 3.2.5. The top performing team used a BERT model with max sentence length of 64 and 2 epochs. The F1 score was 82.9%.

Other works on offensive language include Wester et al. (2016) who introduced and worked on the Youtube Threat Corpus. The corpus consists of 9,845 comments which correspond to 28,643 sentences. Out of all comments, 1,285 of them included threat. This corresponds to 1,384 sentences. The authors perform a more linguistic approach to classification, working with features such as word form, part of speech tag, semantic cluster labels and more. Their results show that a combination of lexical features outperform the use with more complex syntactic and semantic features. The best performance was obtained by a support vector machine model trained on a feature set with lexical n-grams, F1 score of 68.9%.

### 1.2.3 State of the Art

In our work, we will study many classifiers, most of which are based on the structure of BERT introduced in Devlin et al. (2018). The authors introduced a novel classifier utilizing the Transformer architecture. We will go into this in more detail in Section 3.2.5 and 3.2.4.

BERT is trained on a large dataset with a pre-defined task. The goal is to create a general model base that can be fine tuned for many NLP tasks. Their results show that a rich, unsupervised pretraining is an integral part of many language understanding systems.

Models evolved from BERT include XLNet (Yang et al., 2019). XLNet has a similar architecture with the major deviation being that it uses a different task during pre-training. XLNet is the best performing model on all of the NLP tasks mentioned in Section 1.1. We will cover this model and more along with the theory behind them in Section 3.2.

# Chapter 2

# Datasets

## 2.1 What is Gab?

Gab (`https://gab.com/`) is a social media platform created by Andrew Torba and publicly launched on May 8, 2017. It calls itself:

> A social network that champions free speech, individual liberty and the free flow of information online. All are welcome.

Gab is similar to many other social media platforms, where each user has a personal account with a username, profile picture, and personal description also known as *bio*. The latter two are optional while a username is mandatory. Users have the possibility to upload posts, called *gabs*, containing text, media (e.g. image or video), polls or links that consist of at most 3000 characters. To clarify, we will distinguish the web page (Gab) from posts (gabs) with a capital letter. As a user on Gab, you have access to two *feeds*, *All* and *Home*. A feed is a collection of gabs. The All-feed consists of all gabs posted on the platform, while on the Home-feed, you find gabs posted by users that you have chosen to follow. Both users that you have chosen to follow and users that have chosen to follow you are visible by all.

All new users on Gab will automatically follow three accounts, *Gab*, *Gab support*, and *Andrew Torba*. Gab uses a hashtag system, where typing a hashtag in front of a word in a gab will make it visible for searches on that specific hashtag. Gab first differentiates from other platforms when it comes to content restriction. In Gabs first point in *Content Standards* (`https://gab.com/about/tos`), one can read that, User Contributions must NOT:

> Be unlawful or be made in furtherance of any unlawful purpose. User Contributions must not aid, abet, assist, counsel, procure or solicit the commission of, nor constitute an attempt or part of a conspiracy to commit, any unlawful act. For avoidance of doubt, speech which is merely offensive or the expression of an offensive or controversial idea or opinion, as a general rule, will be in poor taste but will not be illegal in the United States.

Many other social media platforms such as Twitter, Instagram, and Facebook prohibit the use of e.g. hate speech by removing content and, in severe cases, suspending users. Gab allows this type of content as long as it is not unlawful.

Possibly, as a result of its liberal policies on speech censoring, Gab has attracted a large amount of far-right users. This hypothesis is based on claims by *The New York Times* (Hess, 2016) and *National Public Radio* (Selyukh, 2017) among others. Recent studies have shown that there exists a high frequency of hate speech on the social media platform. Zannettou et al. (2018) used the hatebase vocabulary (`https://hatebase.org/`) in their work to analyze the amount of posts containing hate words in Gab. Results showed that 5.4% of all posts on Gab included hate words, 2.4 times the rate when compared to Twitter.

## 2.2 The Gab Hate Corpus

The dataset that we have worked on is the *Gab Hate Corpus* (GHC) (Kennedy et al., 2020a). GHC consists of 27,655 gabs collected from Gab in 2018. The posts were sampled at random, the only restriction was a textual threshold. Each gab is then labeled with 14 binary labels by trained annotators. However, in this report, we will only consider two, calls for violence, (*CV*) and assault on human dignity (*HD*).

All annotators were instructed to label a gab as either CV, HD, both or neither. Additionally, we have a class called Hate (hate speech). This class is not to be labeled by the annotator but is generated based on CV and HD. If a post has been labeled with CV and/or HD it will be classified with the label *Hate*. On the case where it is neither HD or CV it will be classified as Non-Hate (not hateful). The authors of GHC define CV and HD as follows:

**CV:** Calls for violence include any verbalization or promotion of messages, which advocate or endorse aggression towards a given person or group on account of their status as member of a given sub-population. This aggression can take the form of violence, genocide, exclusion, and segregation. Threats, which do not name the target's group membership as cause for the threat are not hate speech under our definition.

**HD:** A document should be labeled as HD if it assaults the dignity of group by: asserting or implying the inferiority of a given group by virtue of intelligence, genetics, or other human capacity or quality; degrading a group, by comparison to subhuman entity or the use of hateful slurs in a manner intended to cause harm; the incitement of hatred through the use of a harmful group stereotype, historical or political reference, or by some other contextual means, where the intent of the speaker can be confidently assessed.

A good take-away from these definitions is that both imply that a post must be targeted at a group or person from a specific sub-population. This implies that general offensive language is not hate speech *per se*. As an example of how the annotations are made, Table 2.1 shows how annotator number 10 (A10) labeled gab number 4,900. In total, there are 18 annotators and 27,655 gabs. Every gab is labeled by, at least, two annotators with no upper limit. Figure 2.1 shows both the number of annotators per post and the number of labeled posts per annotator.

To determine if the annotators are in agreement, we turn to measure the inter annotator agreement (IAA). Both Fleiss Kappa and PABAK have been considered in the GHC paper.

| Text | Hate | CV | HD |
|---|---|---|---|
| Irish people are scum | 1 | 1 | 0 |

**Table 2.1:** A10's labels of gab 4900.

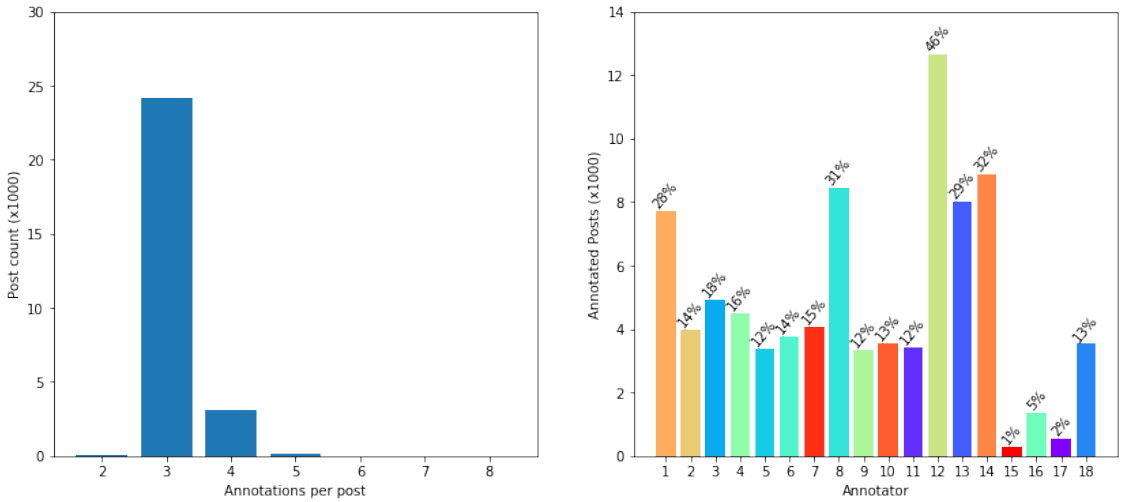


**Figure 2.1:** Left: A histogram over the number of annotators per post. Right: Number of posts labeled by each annotator. On top of each bar the annotators coverage in percentage of the dataset is shown.

Fleiss Kappa is a way to measure the agreement between annotators, where if we have perfect agreement the kappa value is equal to 1 and where we have total disagreement kappa would be 0. The major issue with a kappa value is that it is affected by both the bias between observers and the distribution of data over the different categories, called prevalence. Thus, if we were to have very unbalanced data, the kappa value would not be very representative of the IAA.

To estimate the true distribution of hate speech in our corpus, we looked at the ratio of total number of Hate (and Non-Hate) annotations against the total number of annotations. In Figure 2.2, we can see that 13% of the annotations are labeled as Hate and the remaining 87% Non-Hate. Thus, we estimate that about 13% of the posts to include hate speech, this would be considered fairly unbalanced data. Therefore, we cannot rely on Fleiss Kappa and instead the authors have considered the *Prevalence-Adjusted and Bias-Adjusted Kappa* (PABAK) (Byrt et al., 1993). As seen in Table 2.2, the Fleiss kappa score is very low for both HD and CV but the PABAK is reasonably high. Therefore we can expect that the annotators are in enough agreement for the data to have two distinct classes.

**Table 2.2:** Fleiss kappa and PABAK of original dataset.

| Kappa | Hate | HD | CV |
|---|---|---|---|
| Fleiss | 0.248 | 0.231 | 0.241 |
| PABAK | 0.662 | 0.670 | 0.969 |

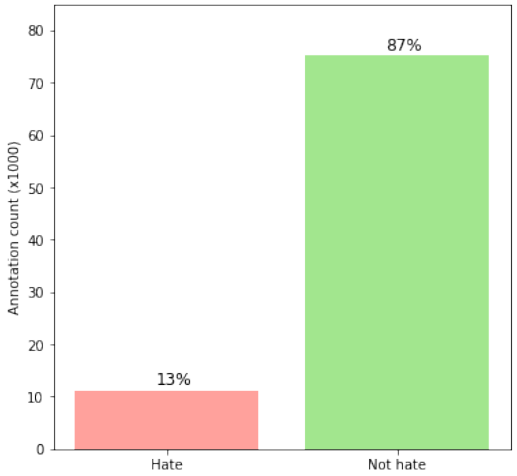


**Figure 2.2:** Distribution of hate annotations and non hate annotations. In total, there are 86,529 annotations on 27,655 posts.

## 2.3 The Majority Dataset

To overcome the issue of multiple labels per post, the authors created a dataset which we will refer to as the Majority dataset. This dataset aggregates each annotator's labels into one label set per post by a majority vote (ties broken towards positive). In this work, we will only consider the hate speech label and disregard the subcategories CV and HD.

Take gab number 4900 as an example. In Table 2.3, we can see that A14 and A10 labeled the post *Irish people are scum* as CV, not HD (i.e. Hate) while A18 labeled it as neither (Non-Hate). The resulting annotation in the majority dataset is therefor Hate.

| Annotator | Text | Hate | CV | HD |
|---|---|---|---|---|
| 14 | Irish people are scum | 1 | 1 | 0 |
| 10 | Irish people are scum | 1 | 1 | 0 |
| 18 | Irish people are scum | 0 | 0 | 0 |
| **Majority:** | Irish people are scum | 1 | – | – |

**Table 2.3:** All annotations on gab number 4900 along with the resulting datum in the majority dataset.

### 2.3.1 Exploratory Data Analysis

We will now explore the Majority dataset by examining different lexical properties. In Figure 2.3, we can see the class distribution of the majority dataset. As suspected, our data is slightly unbalanced.

#### Character and Word Properties

In Figures 2.4 and 2.5, we can see the number of words and characters in each post. In both classes, we see a similar result. It is most common to include about 100 characters or 20-

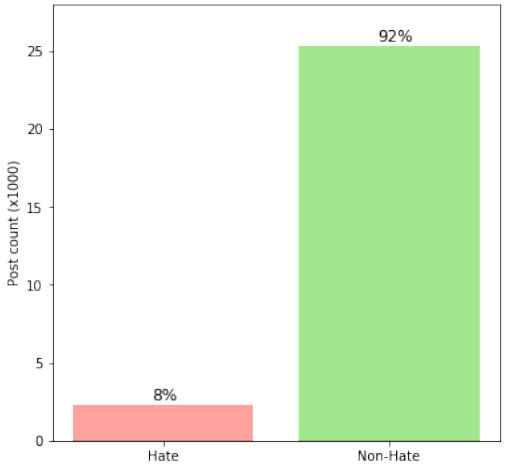


**Figure 2.3:** Distribution of hate and non-hate posts in the majority dataset.

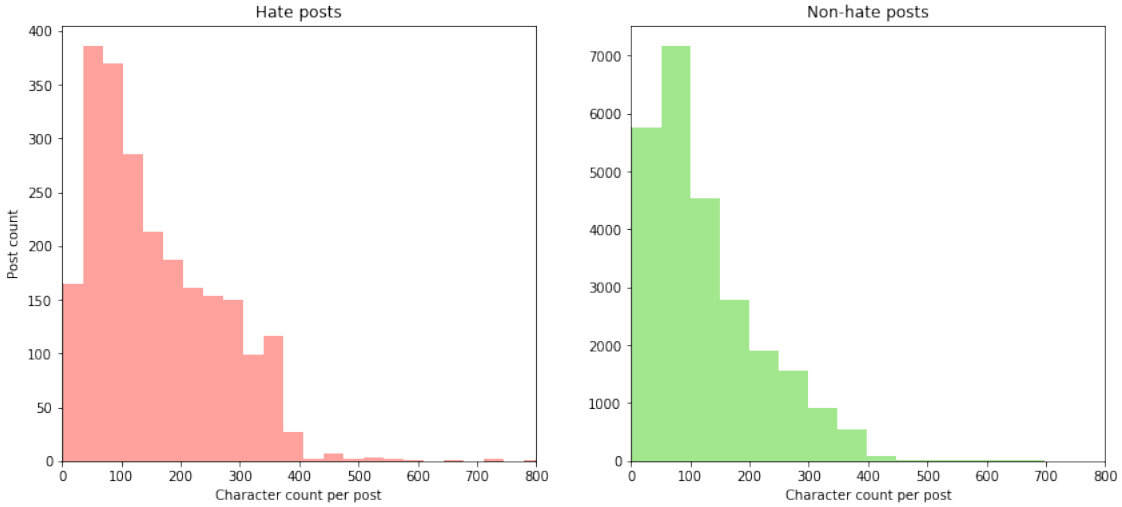


**Figure 2.4:** Number of character in each post per class.

25 words. The upper limit of 3,000 characters does not imply a restriction on any of the classes since posts with more then 500 characters are very rare. We can observe that Hate posts have a lower frequency of short posts (less then 8 words or less then 35 characters). One explanation of this could be that it is difficult to formulate what is considered as hate speech in such a short text sequence. Observing now the distributions, it appears that the character distribution is the same for the two classes while there is a possibility of the two word distributions being different. In Non-Hate, there is a more distinct spike at 25 words and drastically dropping whilst in Hate, there is a much smoother transition.

## Words and Bigrams of Importance

As seen in Figures 2.6 and 2.7, the most common words and bigrams are not so surprising. This gives us little to no information about the different classes. As an attempt to gain more awarding information, we will define a new type of top-list called the unique top-list. Let

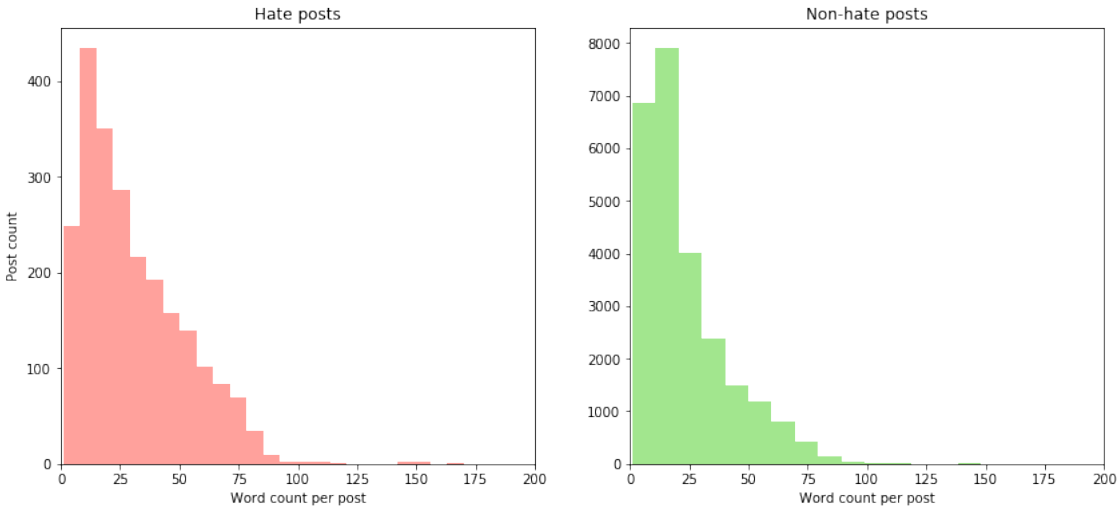


**Figure 2.5:** Number of words in each post per class.

*top_words* be the function returning the $n$ most frequent words of a given class. The unique top list for two classes C1 and C2 is given by:

**while true do**
  $common \leftarrow top\_words(C1) \cap top\_words(C2)$
  **if** *common* is empty **then**
    **return:** top words for $C1$ and $C2$
  **else**
    **for all** $w \in common$ **do**
      remove $w$ from $C1$ and $C2$
    **end for**
  **end if**
**end while**

In Figure 2.8, we can see the most common unique words for each class. There is a clear distinction of the two classes. Hate posts include words of sub-population groups often exposed to hate speech e.g. Jews and Muslims. It also more common to address gender and (most likely) ethnicity e.g. white, black, women, men and so on.

Since creating a unique list gave insights for single words, we did the same for bigrams. In Figure 2.9, we can see the most common unique bigrams for each class. Looking at hate posts, we see less words tied to gender, ethnicity or religions. What we can see instead is a clear distinction of the object of sentences. In Hate posts, we can see a frequent use of plural pronouns e.g. *They are*, *of them* and *that they*. For Non-Hate posts we see a much higher frequency of first person singular pronouns e.g. *if you*, *I would* and *I can*.

We recall the definition in Sect. 2.2, which implies that offensive speech against a person does not suffice as hate speech unless it is directed at a sub-population. This would explain the scarcity of *I* and *you* in Hate posts. As a last exploration of important words we looked at words ranked with TF×IDF. In Figure 2.10, we can see the top ranked word w.r.t. TF×IDF. For us, these results did not contribute to understanding the data.

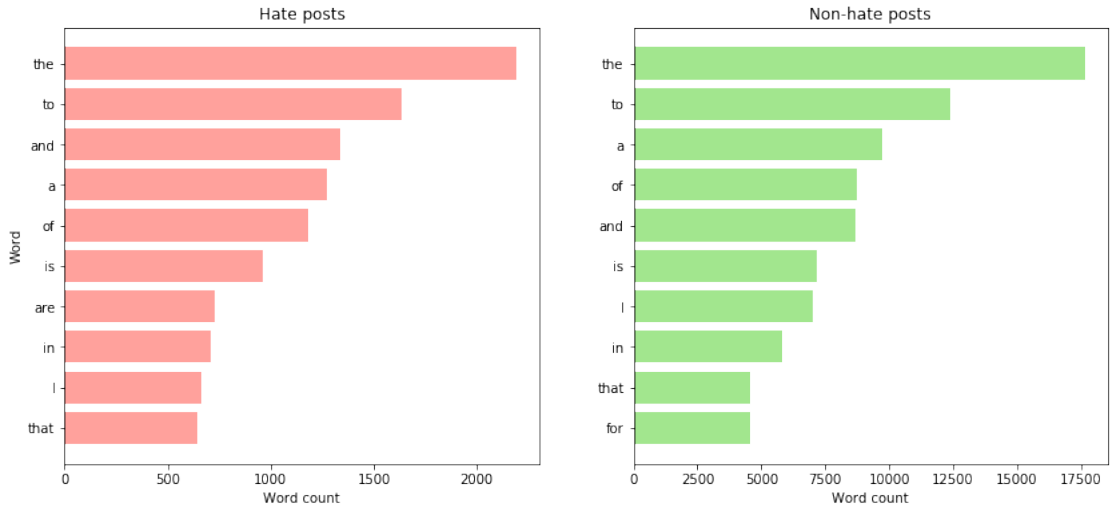


**Figure 2.6:** Most common words in each class.

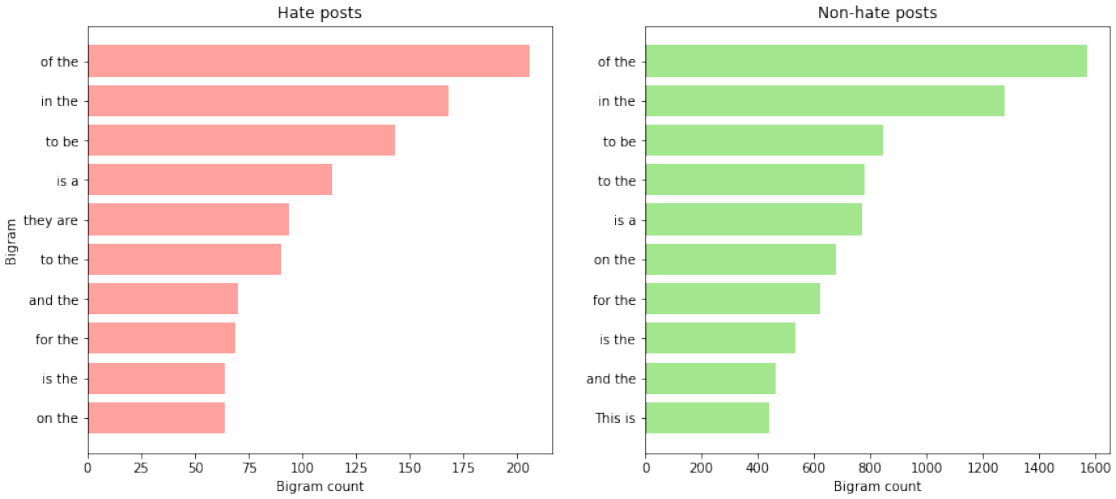


**Figure 2.7:** Most common bigrams in each class.

## Lexical Diversity

Lexical diversity (LD) is to measure of how diverse the language is in a text. There are multiple measures available and all capture different aspects of diversity. McCarthy (2010) suggests to use three different measures to get an appropriate expectation of LD, they are: MTLD, HD-D (or vocd-D), and Maas. An explination of the different measures will follow below. A recurring instrument is the Type-Token Ratio (TTR), which is the ratio between unique words ($V$) and the total number of words ($N$), i.e. $100 \cdot V/N$.

**MTLD:** MTLD is the mean length of sequential word strings in a text that maintain a pre set TTR limit. Take the word string, *of the people by the people for the people*, and a TTR limit of 0.72 as an example. The rolling TTR is, of (1.00) the (1.00) people (1.00) by (1.00) the (.800) people (.667) for (.714) the (.625) people (.556). After each violation of the TTR limit the rolling TTR is reset and one increment is added to our sequential counter. Thus, given the previous example, MTLD would execute, of (1.00) the (1.00)

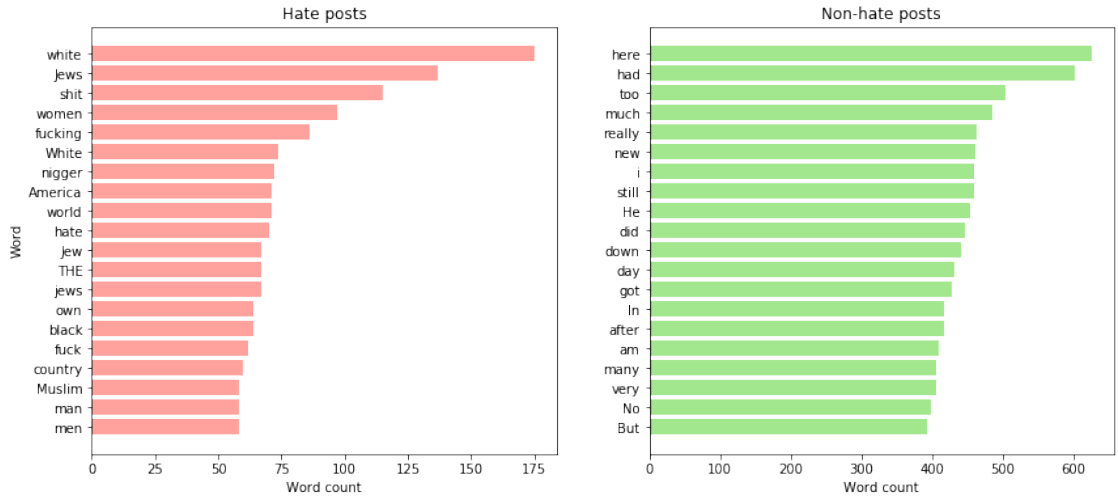


**Figure 2.8:** Most common unique words in each class.

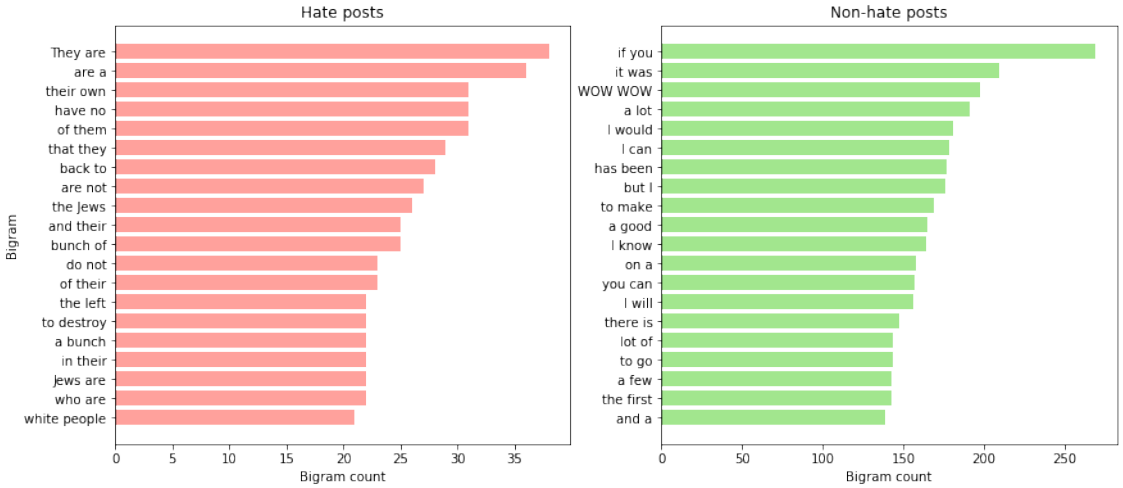


**Figure 2.9:** Most common unique bigrams in each class.

people (1.00) by (1.00) the (.800) people (.667) ‖COUNTER = COUNTER + 1‖ for (1.00) the (1.00) people (1.00), and so forth. MLTD is then given by COUNTER / $N$.

**HD-D:** For each token $v_i$, calculate the probability of encountering the token in a random sample ($A$) of 42 tokens drawn from the text. HD-D is then given by taking the sum of these probability for all token types, i.e. $\sum_{i=1}^{V} Pr(v_i \textbf{ in } A)$.

**Maas:** This measure is also denoted $a^2$ and is given by:

$$a^2 = \frac{\log N - \log V}{\log N^2}$$

.

In Figure 2.11, we show the lexical diversity of 430 individual posts containing more than 45 words from each class. We also calculated a common lexical diversity for each of the classes. To do this, we created two corpora: the hate corpus consisting of all hate posts and a non

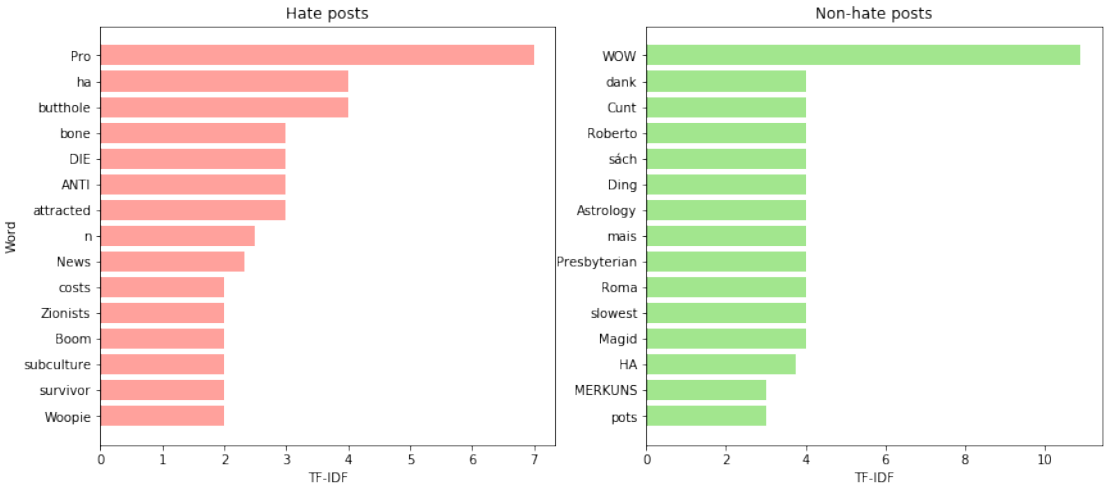


**Figure 2.10:** Top rated words in each class with respect to TF×IDF.

hate corpus consisting of posts randomly sampled from the non hate class. Sampling was done such that the corpora have similar lengths. The lexical diversity of the two corpora are shown in Figure 2.12. All three of the above mentioned measures (HD-D instead of vocd-D) are used in both cases to achieve a good representation. Neither one of the methods showed results that implied that the lexical diversity of one class is different from the other. In Figure 2.12, Non hate scored best on MTLD but the worst on Maas, the HD-D score was a tie. Further, in Figure 2.11, the distributions are not separated at all.

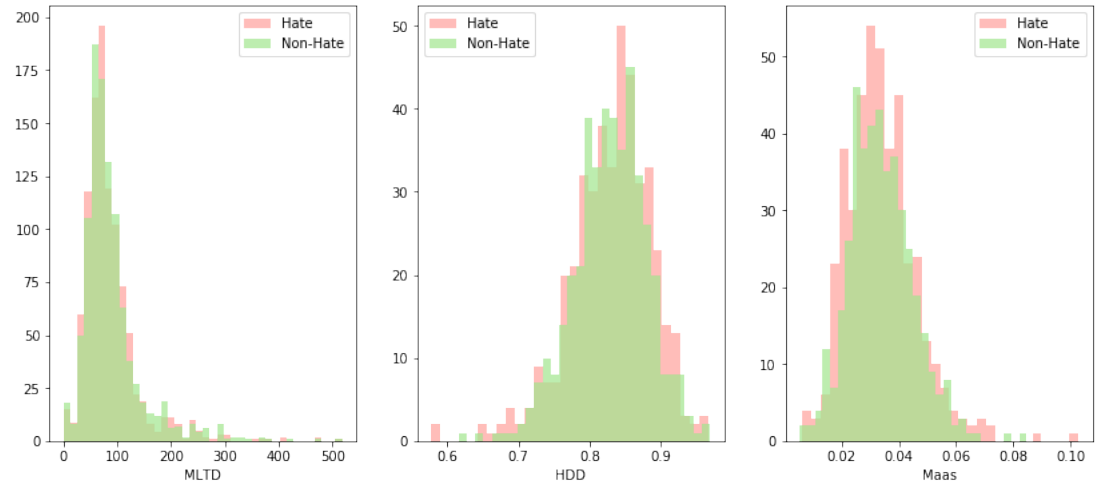


**Figure 2.11:** Histogram of lexical diversity per post for 430 posts with 45 from each class.

## Word Cloud

Lastly, we give a word cloud representation of the two classes in Figure 2.13 to give a intuition of the two classes. In the images, we see similar words as shown in Figure 2.8 and this implies that a reasonable annotation has been made. Consequently, we consider the dataset sufficient

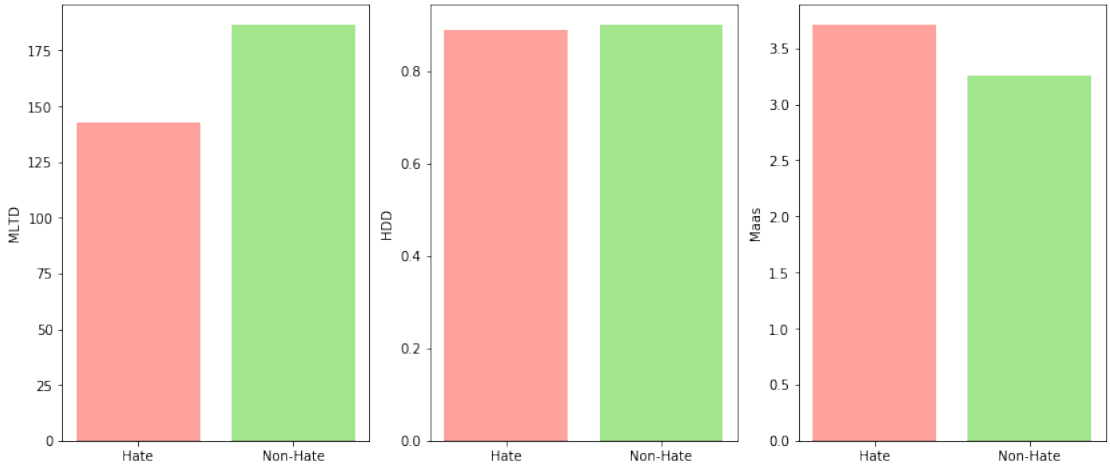


**Figure 2.12:** Lexical diversity of Hate corpus and Non hate corpus. The corpuses contain 375,980 and 376,060 characters respectively.

for the usage in this work.


**Figure 2.13:** Word cloud representation of both classes.

### Uniqueness of the Dataset

After analysis, it has become clear that there are 99 posts which appear with more than one ID in the Gab Hate Corpus, and thus appears more than once in the majority dataset. This creates some ambiguity, especially since 3 posts get different labels in the majority dataset. However, in order to have comparable results with the authors, we ignored this issue.

## 2.4 More Alternatives of GHC

Throughout this work, we will work with a few other variants of GHC as well. Each one is listed below:

**Consensus** The consensus dataset is a subset of the Majority dataset. It includes only the posts, where all annotators are in complete agreement, i.e. all Hate or all Non-Hate. There is a total of 20,436 posts in this dataset and 2.85% are Hate posts.

**Extended Consensus** The extended consensus dataset is also a subset of the Majority dataset. It contains all Hate posts from Majority but only the Non-Hate where all annotators are in complete agreement. This dataset could also be interpreted as the Majority dataset without uncertain Non-Hate posts. There are a total of 22,415 posts in this dataset and 11.43% are Hate posts.

Table 2.4 shows the distribution of classes and Table 2.5 shows the IAA for posts qualifying for Consensus and Consensus Extended. The Majority datasets contains all posts and is therefore ignored.

**Table 2.4:** Distribution over classes in sub-corpora.

| Dataset | Size | Hate |
|---|---|---|
| majority | 27,655 | 8.45% |
| consensus | 20,436 | 2.86% |
| consensus extended | 22,415 | 11.43% |

**Table 2.5:** Fleiss kappa and PABAK of Consensus and Consensus Extended.

| Dataset | Kappa | Hate | HD | CV |
|---|---|---|---|---|
| | Fleiss | 1.000 | 0.949 | 0.578 |
| Consensus | PABAK | 1.000 | 0.995 | 0.993 |
| | Fleiss | 0.251 | 0.232 | 0.250 |
| Consensus Ext. | PABAK | 0.663 | 0.670 | 0.968 |

## 2.5 Alt-Right Forum

In order to have an completely unbiased dataset for evaluation of our model, we have annotated 1594 posts extracted from an Alt-Right forum. We will call this data set the Alt-Right dataset. Just like with GHC, posts from this forum have been labeled as either Hate or Non-Hate. Annotations on this data have not been done following the exact procedure used in GHC, but with the same goal: to label posts which were overly aggressive towards a group or an individual. To get an understanding of the data, we have looked at the distribution of classes (Figure 2.14), the number of words per post (Figure 2.15) and wordcloud for each of the two classes (Figure 2.16).

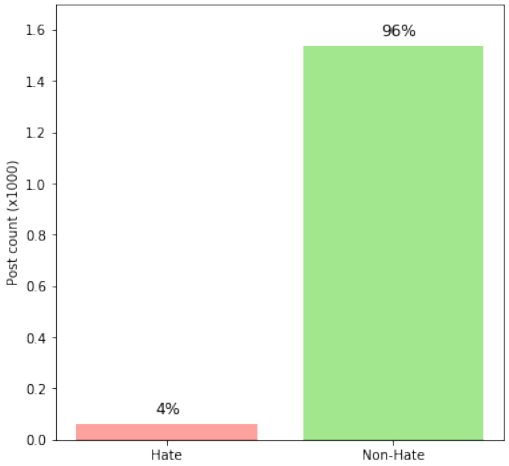


**Figure 2.14:** Distribution of classes of alt-right dataset.

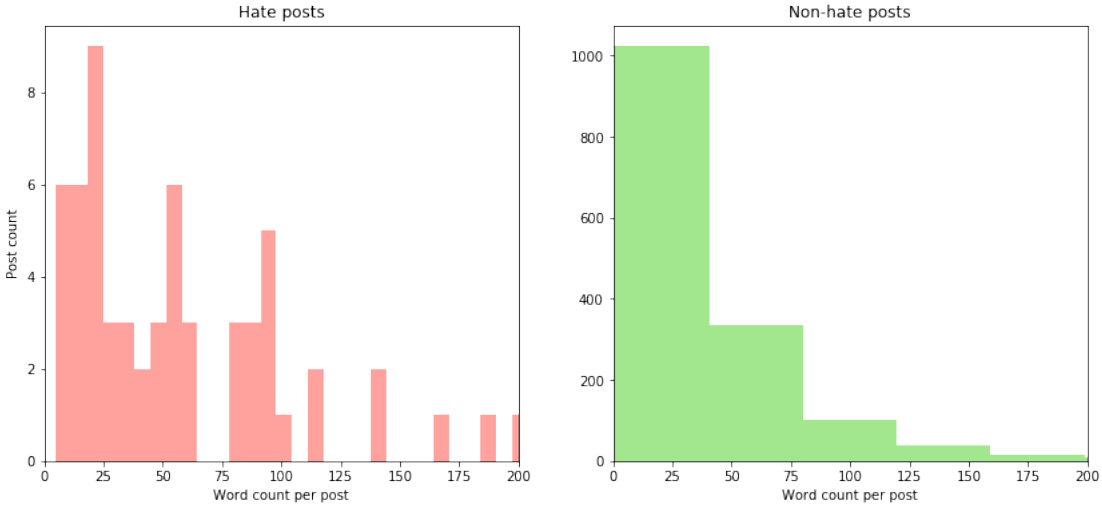


**Figure 2.15:** Word count of posts from alt-right dataset.


**Figure 2.16:** Word cloud representation of both classes from alt-right.

# Chapter 3
# Approach

## 3.1 Method

In this work, we have followed the *Cross-industry standard process for data mining*, also known as CRISP-DM, introduced by IBM. CRISP-DM consists of six major phases of which there are multiple ways of navigating through. Figure 3.1 shows the possible directions one can navigate. How the different phases were implemented in this work is described below.

**Business Understanding:** In this phase, we explored what The Swedish Security Service had to gain from the work. At first, we aimed at detecting threats in a more general sense, which got narrowed down to hate speech detection after conclusions drawn in the Data Understanding phase.

**Data Understanding:** During this phase, we explored the data to evaluate if the data was separable into two classes or not. This was done by looking at the distribution of classes and IAA.

**Data Preparation:** In this phase, we prepared all the sub datasets explained in Sections 2.3 and 2.4.

**Modeling:** Here we explored and tuned a hand full of classifiers, a LSTM as a baseline and thereafter Transformer based classifiers.

**Evaluation:** Evaluating models created in the previous phase on the Majority dataset. Also, final evaluation of best performing model on the Alt-Right dataset.

**Deployment:** Selecting model and creating a application.

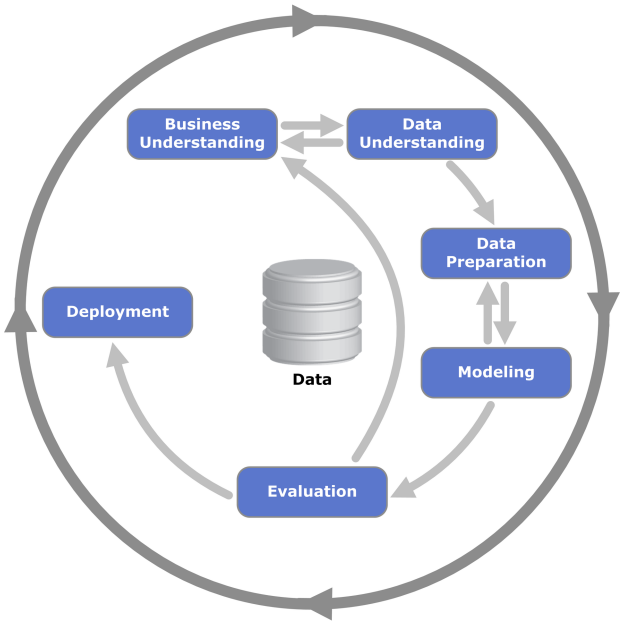


**Figure 3.1:** Visualization of CRISP-DM. Source: `https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining`

## 3.2 Theory

### 3.2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of gated recurrent neural network commonly used in NLP. A recurrent neural network (RNN) is a type of network structure that uses recent output from the model as input.

Let $\mathbf{X}$ be a set of inputs to the network, $\mathbf{h}$ be a set of outputs from hidden nodes and $\mathbf{y}$ be the set of outputs. Then the most simple RNN has as output $y_i = \theta(h_i(\mathbf{X}) + w \cdot \mathbf{h}_{i-i}(\mathbf{X}))$, where $\mathbf{w}$ is the the weight on the recurrent node and $\theta$ is the activation function.

A big problem with RNN's is keeping track of long term dependencies. When training a model with back-propagation, gradients tend to vanish and sometimes explode. This creates an issue for optimization as demonstrated in Goodfellow et al. (2016, sect. 10.7).

The clever idea of LSTM is to also pass gradients through the network, sometimes without weighting them at all, i.e. unchanged. LSTMs consist of *LSTM cells* which have internal recurrent units. These cells contain three *gates* regulating the flow of information. An *input gate*, *output gate*, and a *forget gate*. The input gate regulates the input, the output gate regulates the output and the forget gate regulates the memory – stored information from previous states.

### 3.2.2 WordPiece

The WordPiece embeddings were first introduced by Wu et al. (2016). The tokenizer starts with splitting each word into parts until all parts are part of a vocabulary. The vocabulary

consist of the $N$ of most common word pieces, $N$ is usually around 8 and 32 thousands. Since the most common word pieces always include all Latin characters, a word can, in the worst case, be broken down into a sequence of characters.

Take as an example gab number 4,900 from Sect. 2.2. When passing it through a word piece tokenization from a BERT model with a vocabulary size of 30,000, the result would be *[Irish, people, are, s, ##cum]*. Implying that all words except *scum* are a part of the dictionary. *Scum* will therefore be broken down until all word pieces are a part of the vocabulary, in this case with *s* and *##cum*. The *##* implies that it is the ending of a word. The actual split is chosen by a word piece model that maximizes the language-model likelihood. Next, each word piece is mapped to the corresponding index, in our example this would be `[2600, 1234, 1132, 188, 19172]`.

### 3.2.3 Embedding Matrix

An embedding matrix of size $K \times E$, where $K$ is the size of the vocabulary and $E$ a chosen dimension is a set of vectors (usually word or token vectors) collected in a matrix. Each token has a given index and each index $i$ corresponds to row $i$ in the embedding matrix.

Thus, for the example in Sect. 3.2.2, the token *Irish* would have its word vector on row 2600 in the embedding matrix. A word vector is meant to give a representation of that word or token. In a well orchestrated embedding matrix, words with similar meaning are similar in the embedding space. Usually similarity is measured with the cosine similarity,

$$\text{similarity}(\mathbf{u}, \mathbf{v}) = \cos(\theta) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||||\mathbf{v}||} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}},$$

where $\theta$ is the angle between $\mathbf{u}$ and $\mathbf{v}$. In all transformer-based models the embedding matrix is trained along with the model. However, there exists pre-trained embedding matrices which can be used as well. We will give an example of embeddings from Mikolov et al. (2013) to get an understanding for the concept. Not only did similar words have a similar representation, the embeddings have also captured the relations between word-vectors. For instance, one offset direction in the vector space corresponded to plural/singular and one corresponded to gender, demonstrated in Figure 3.2. Unexpectedly allowing for arithmetic operations on word vectors, for instance King − Man + Woman resulted in a vector very similar to Queen.

Although an embedding matrix usually consists of token (or word) vectors, it could also consist of other things. For example, in many transformer models, there are two other types of embeddings, positional, and segmental embeddings. In positional embedding, the first row in the embedding matrix consists of the embedding vector for the first position in a sentence. While in the segmental case, the first row corresponds to the first input sequence (some models can take multiple sequences as input).

### 3.2.4 Transformers

Vaswani et al. first introduced the transformers in 2017 in their paper *Attention is all you need*. I will give a brief explanation to what transformers are but for a more profound description see the article. Figure 3.3 shows the overall architecture of a transformer model. We will first go through some relevant theory in order to later understand the architecture.

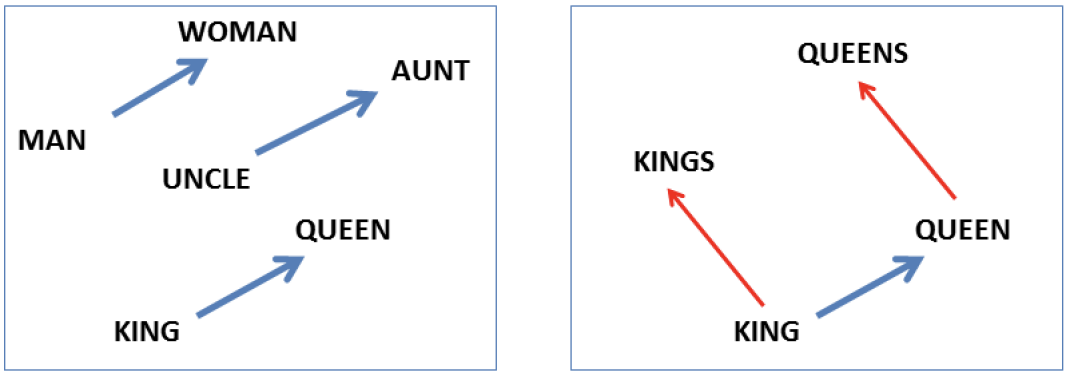


**Figure 3.2:** Left: Vector offsets for three word pairs representing the gender relation. Right: Demonstrating the singular/plural relation for two words. After Mikolov et al. (2013).

## Encoder – Decoder

The encoder–decoder architecture consists of an encoder, which object is to map a sequence of tokens $\mathbf{x}$ to a continuous representation $\mathbf{z}$, and a decoder, which objective is to map the representation $\mathbf{z}$ to an output sequence of tokens $\mathbf{y}$, one element at a time. This is usually done in a auto-regressive manner, taking the previously predicted tokens $(y_0, \ldots, y_{i-1})$ as input when predicting $y_i$.

## Attention

Attention is a mapping from three input vectors (a key-value pair and a query) to an output. The output can be interpreted as weighted values, where the weights depend on the key and query. In this particular case, the authors use the scaled dot product attention which takes, as weights for the value, the dot product of the key and query and scales it with $\sqrt{d_k}$, where $d_k$ is the dimension of the key (and query). In the transformer architecture, they have used attention in three ways:

**Self-attention:** Keys, values, and queries come from the same source. This let each token of the sentence attend to every other token in the sentence. Thus a way of weighting the importance of tokens in a sentence.

**Masked self-attention:** This is used in the *decoder*. This is identical to self-attention apart from the fact that tokens that have yet to be decoded are masked. This to ensure that illegal connections are not used.

**Attention:** Used in the *encoder-decoder*. Here the keys and queries come from the output of the encoder and the values come from the output of the masked self-attention in the decoder. Hence, a way of weighting previously predicted tokens against the output from the encoder. This is then used for predicting the next token.

It is a lot more efficient to calculate attention in batches so the key, values and queries become matrices, $\mathbf{K}$, $\mathbf{V}$, $\mathbf{Q}$. The attention is then calculated with the following function.

$$Attention(Q, V, K) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{3.1}$$

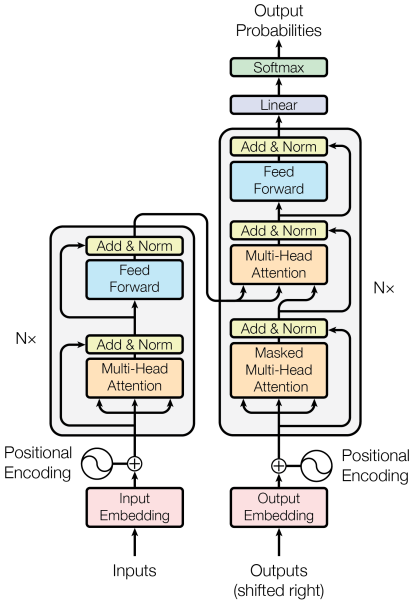


**Figure 3.3:** Transformer – the model architecture. Left half: Encoder, right half: Decoder. After Vaswani et al. (2017).

In order to capture more properties about the input, we can perform attention multiple times (i.e. with multiple *Heads*), with a different linear transformation of $\mathbf{K}$, $\mathbf{V}$, $\mathbf{Q}$ for each attention mapping. In Figure 3.3, this is called *Multi-Head Attention*.

## The Architecture

Observing Figure 3.3, the transformer consists of two parts, one encoder stack with $N = 6$ identical layers and one decoder stack also consisting of $N$ identical layers.

The input of the encoders is always the input sequence, while the input of the decoder are the previously predicted outputs. Let us, as an example, consider the task of machine translation. We want to translate the Swedish sentence *Jag bor i Sverige.* to English, *I live in Sweden.*. The input to the encoder at all steps is *Jag bor i Sverige.* while the first input of the decoder is a start of sentence tag. The first output is *I*. The input of the decoder will then be *I* in the second step, *I live* in the third step and so on.

The encoder consists of a self attention, adding a residual connection followed by normalizing. Next the output goes into a feed-forward neural network and is then added with another residual connection to lastly be normalized again.

In the decoder, we see that the input is first passed through a self attention called Masked Multi-Head Attention. This is the action of masking tokens to prevent attention on subsequent tokens. For instance in the last step we have as input in the decoder layer *I live in Sweden.* When calculating the attention of *in*, it cannot depend on *Sweden.* It must only depend on *I* and *live*. The remaining part of the decoder, after adding a residual connection and normalizing, has the same structure as the encoder. The only difference is in the Multi-Head Attention, which this time is not self-attention but actual attention.

This is the architecture of what is called one *Transformer block*.

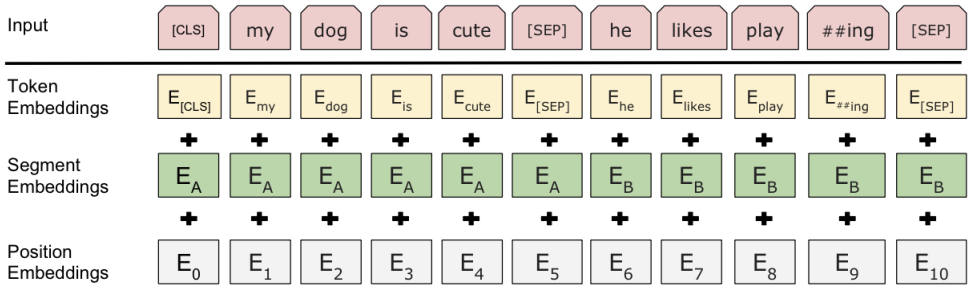


**Figure 3.4:** Visualization of input to the BERT model. After Devlin et al. (2018).

### 3.2.5 Pre-Trained Models

Before we go into pre-trained models, we will briefly introduce transfer learning. Transfer learning is a research problem within machine learning that aims to store knowledge from one task for later use on a different, possibly related, task. For instance, if a model trained on cat images with the task of identify cats, the model is probably well suited to identify dogs as well. Even more so if the cat model is trained on additional dog data, called fine-tuning.

Transfer learning could also be used in NLP. Any model that can capture the sentiment of a text sequence would serve as a good candidate as a pre-trained model for many NLP tasks. In this work, we have studied four different pre-trained models, BERT, RoBERTa, ALBERT and XLNet. We will first go through BERT in detail and later give a brief description of the remaining models to get an understanding of how they differentiates from each other.

#### BERT

BERT stands for: Bidirectional Encoder Representations from Transformers. Bidirectional implies that the model is a combination of a forward and backward language model. Given a sequence of $\boldsymbol{N}$ tokens, a forward language model predicts token $t_k$ based on the previous tokens $(t_0, \ldots, t_{k-1})$ i.e. the history. A backward language model, on the other hand, predicts token $t_k$ based on $(t_{k+1}, \ldots, t_N)$. Encoder representation implies that the model uses encoders as explained in 3.2.4 and transformers implies that the model uses transformers blocks in its architecture. BERT takes up to two sequences of text as input, known as segments, and embeds the input with three type of embeddings, token, position and segment. In Figure 3.4, we can see an input example.

There are a few hyper-parameters that are good to have knowledge about, the number of layers (i.e. Transformer blocks) is denoted as $\boldsymbol{L}$, the size of *all* hidden states $\boldsymbol{H}$, the embedding dimensions $\boldsymbol{E}$ and the number of self-attention heads $\boldsymbol{A}$. There is also a cap on the number of tokens set to 512.

In this work, we have worked with two BERT models sizes, $\boldsymbol{BERT}_{BASE}$ ($L$=12, $H$=$E$=768, $A$=12, Total Parameters=110M) and $\boldsymbol{BERT}_{LARGE}$ ($L$=24, $H$=$E$=1024, $A$=16, Total Parameters=340M). Both models were pre-trained on a 3.3 billion word corpus with a batch size of 128,000 on approximately 40 epochs, taking roughly four days per model to train on 16 cloud TPU's.

Training was done with two unsupervised tasks. The first task is called Masked Language Modelling which simply means masking a word in a sequence and letting the model predict

| Model | | Parameters | *L* | *H* | *E* | *A* | Parameter sharing |
|---|---|---|---|---|---|---|---|
| BERT | base | **110M** | 12 | 768 | 768 | 12 | False |
| | large | **340M** | 24 | 1024 | 1024 | 16 | False |
| ALBERT | base-v2 | **11M** | 12 | 768 | 128 | 12 | True |
| | large-v2 | **17M** | 24 | 1024 | 128 | 16 | True |
| | xlarge-v2 | **58M** | 24 | 2048 | 128 | 16 | True |
| | xxlarge-v2 | **223M** | 12 | 4096 | 128 | 64 | True |
| RoBERTa | base | **125M** | 12 | 768 | 768 | 12 | False |
| | large | **355M** | 24 | 1024 | 1024 | 16 | False |
| XLNet | base | **110M** | 12 | 768 | 768 | 12 | False |
| | large | **340M** | 24 | 1024 | 1024 | 16 | False |

**Table 3.1:** The configurations of all models considered in this work

the masked word. The second task is Next Sentence Prediction: given two sentences the model will predict if the second sentence follows the first.

Let us see two examples for clarification. We will consider three segments of text, "the man went to the store", "he bought a gallon of milk" and "penguin's are flightless birds". Note that [CLS] is a classification token used in BERT and is always the first token in the input. [SEP] is the separator between segments and [MASK] is the masking token and indicates the word that is to be predicted.

**Input:** [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
**Label** = *IsNext*

**Input:** [CLS] the man [MASK] to the store [SEP] penguin [MASK] are flight ##less birds [SEP]
**Label** = *NotNext*

As we can see in example 3.2.5 the word *flightless* has been split into two parts. This is because BERT uses a WordPiece tokenization with a 30,000 token vocabulary.

## Other Models

The additional models used in this work are listed below. We will give a brief description of them and explain how they differ from the rest. For more information about the models, please see the corresponding papers.

**ALBERT:** Lan et al. (2019) introduced ALBERT, which uses the same architecture as BERT with slightly different hyper-parameters. The distinguishing attribute of ALBERT is parameter-sharing. All ALBERT models will share parameters (e.g. feed-forward weights and attention weights) across all layers (i.e. transformer blocks). In this work, we used a second version of ALBERT (v2) which has been trained for a longer period of time on additional data.

**RoBerta:** Liu et al. (2019) analyzed the pre-training of BERT and concluded that BERT was significantly undertrained. They carefully measured the impact of many key pre-training hyperparameters and training data size in order to Robustly Optimize the BERT Pretraining Approach.

**XLNet:** A model which is architecturally similar to BERT with alternations in pre-training task. Yang et al. (2019) discovered that during BERT's task of predicting masked words BERT neglects dependency between the masked positions, which XLNet does not. For instance, take the sentence *I live in Stockholm.* and mask *in* and *Stockholm*. BERT would predict $P(\text{in} \mid \text{I live}) + P(\text{Stockholm} \mid \text{I live})$ while XLNet would predict $P(\text{in} \mid \text{I live}) + P(\text{Stockholm} \mid \text{I live } \mathbf{in})$.

In Table 3.1, we can see the differences in hyperparameters and total number of parameters for all models.

### 3.2.6 Techniques to Enlarge Dataset

In this work, we have considered two techniques of enlarging data: sampling and augmenting.

#### Oversampling

Oversampling is the act of sampling the minority class until the dataset is balanced. Sampling is done with replacement, thus there will be multiple duplicates in the minority class. The over-represented class will remain unchanged. Oversampling data eliminates the bias towards the over-represented class but raises the risk of over-training. This is one out of many sampling techniques used on unbalanced data.

#### Text Augmentation

Augmenting data is the art of slightly altering data in order to enlarge a dataset. In image analysis, it is common to rotate, zoom, flip and skew images in order to augment the data but in NLP we use some different techniques. In this work, we have used two techniques TF×IDF and Back-translation.

TF×IDF stands for Term Frequency × Inverse Document Frequency. Term Frequency counts the total number of occurrences of the word in all of the documents while Document Frequency counts the number of documents the word appears in. By taking term frequency times the inverse document frequency we give each word a measure of importance. E.g. if a word appears a lot in a single document this will raise the score of the word, the word seems to be important. However, if the word is very common and appears in most of the documents this will lower the score, the word might not be so important after all.

The most common augmentation techniques used on text sequences are the following:

**Synonym Replacement:** Selecting a random word and replacing it with a synonym.

**Random Insertion:** Inserting a random word at a random position.

**Random Swap:** Selecting to words at random and swapping the place of the words.

**Random Deletion:** Selects a word at random and deletes this word from the sequence.

Data augmentation with TF×IDF uses a TF×IDF model (trained on GHC) to get a score for each word. Next it will perform the common augmentation techniques. However, instead of randomly selecting words it will select words w.r.t. the TF×IDF score. Thus, augmenting less important words with higher probability than more important words.

The other type of augmentation, Backtranslation, is simply the act of translating a sequence to a foreign language and then translating it back to the original language. In this work, we have used the program from Xie et al. (2019) in order to backtranslate the data. The foreign language that have been used in this case is French.

### 3.2.7 Statistics

In this section, we will mainly focus on comparison of two sets of samples that are assumed to have been drawn from two normal distributions. Namely, we want to determine if one population is significantly larger than the other. To do this, we turn to Welch's t-test, a Student's t-test with different variances.

We will assume to have two identically large sets of samples ($n$ samples) with potentially varying sample variances. The sample mean of population 1 is denoted $\bar{X}_1$ and the sample mean of population 2 is $\bar{X}_2$, are both assumed to be normally distributed with potentially different estimated values and variances. The sample variances are denoted $s_1^2$, $s_2^2$, respectively. Both follow $\chi^2$ distribution with $n-1$ degrees of freedom. In our test statistics, our null hypothesis, $H_0$ is $\bar{X}_1 \leq \bar{X}_2$, thus we have a one tailed test. To summarize:

$$\bar{X}_1 \sim \mathcal{N}(\mu_1, \sigma_1), \quad \bar{X}_2 \sim \mathcal{N}(\mu_2, \sigma_2), \quad s_1^2 \sim \chi^2_{n-1}, \quad s_2^2 \sim \chi^2_{n-1}.$$

The test statistics are given by,

$$H_0 : \bar{X}_1 \leq \bar{X}_2, \quad H_1 : \bar{X}_1 > \bar{X}_2,$$

$$t = \frac{\bar{X}_1 - \bar{X}_2}{s_{\bar{\Delta}}}, \quad s_{\bar{\Delta}} = \sqrt{\frac{s_1^2 + s_2^2}{n}}, \quad \nu = \frac{(n-1)(s_1^2 + s_2^2)}{s_1^4 + s_2^4}, \quad p = 1 - F_\nu(t),$$

where $t$ is the test statistics $\nu$ is the degree of freedom, $p$ is the $p$-value and $F_\nu(t)$ is the cumulative distribution function for Student's t-distribution with $\nu$ degrees of freedom. Thus, we reject $H_0$ on significant level $\alpha$ if $p < \alpha$. In other words, if $p < \alpha$ we conclude that we have statistical evidence that the mean value of the first population is larger then the second.

# Chapter 4
# Evaluation

## 4.1 Experimental setup

### 4.1.1 Pre-Processing

In GHC, each post had multiple annotators. This is an ambiguity issue for any classifier for all cases where annotators are not in complete agreement. To solve this, we created three datasets with one unique label per post, Majority, Consensus and Consensus Extended. The different datasets are described in detail in Sections 2.3 and 2.4. Apart from the creation of datasets no pre-processing was done. This was mainly to create results comparable with the authors of GHC.

### 4.1.2 Classifiers

We considered five different classifiers, LSTM, BERT, RoBERTa, ALBERT, and XLNet. All transformer-based models use a base model and for ALBERT, we use the second version, see base models in Table 3.1 for details. LSTM was implemented with Keras while the transformer-based classifiers were implemented with pytorch.

#### Hyperparameters of Transformer Based Classifiers

All classifiers with the transformer architecture used hyperparameters suggested by the authors of BERT for all except max sequence length which was set to 64. Limiting the sequence length to 64 makes training easier on the GPU while only effecting 17% of the data. The hyperparameters used are number of epochs (2), the learning rate of the optimizer ($510^{-5}$) and batch size (32).

### 4.1.3 Text Augmentation

We experimented with two types of augmentation, Backtranslation and and TF×IDF. backtranslation creates one extra post for each post, thus doubling the size of the dataset. However, for TF×IDF, we have the possibility of creating an optional number of posts from each post. In an attempt to kill two birds with one stone, we created an additional 9 posts for all Hate posts and one additional post for all Non-Hate posts, thus making the data less imbalanced. In Figure 4.1, we can see how the size of the dataset and distribution of hate speech is changed after augmentation.

**Table 4.1:** The size and distribution of Hate for each of the datasets after augmentation.

| Dataset | Augmentation | Size | Hate |
|---|---|---|---|
| | Backtranslation | 55 310 | 8.45% |
| | TF×IFD | 74 006 | 31.6% |
| Majority | Both | 129 316 | 21.7% |
| | Backtranslation | 40 872 | 2.86% |
| | TF×IFD | 45 544 | 12.8% |
| Consensus | Both | 86 416 | 8.11% |
| | Backtranslation | 44 830 | 11.4% |
| | TF×IFD | 65 326 | 39.2% |
| Consensus Ext. | Both | 110 156 | 27.9% |

### 4.1.4 Imbalanced Data

In order to deal with the issue of imbalanced data, we will apply two strategies. The first strategy is oversampling with replacement on the minority class. This means that we will end up with multiple duplicates of Hate posts in the training file. The reason for doing this is to eliminate the bias towards Non-Hate class. The second strategy is to add a weight to each data point. In order to do this with transformer-based models, one must alter the source code slightly.

### 4.1.5 Evaluation

Before training, we have held out 10% of the majority dataset for evaluation. This test set will be used for evaluation on *all* experiments. The reason for this is because the authors of GHC reported results of a classifier trained and evaluated on the Majority dataset. Because our data is imbalanced, we will evaluate our data with F1 score with macro average. This simply means that all classes will be weighted the same. Also, this is the score used by the authors of GHC.

Due to random initialization in the latent space, each time one trains a classifier, the results will be slightly different. In order to evaluate, we will therefore train each classifier 10 times and report the mean value F1 score along with the standard deviation. As a final evaluation, we will test the performance of the model on the self-annotated Alt-Right dataset.

### 4.1.6 Greedy Assumption

To summarize, we have 3 datasets, 5 classifiers, 4 options of augmentation (including none) and 4 combinations of balancing data strategies (also including none), resulting in 240 combinations. Training each combination 10 times gives us a total of 2400 training sessions. Each session takes about 15 minutes, resulting in a total training time of 25 days. Clearly, we need to change our strategy in order to have a reasonable training time. Instead of exploring all possible combinations, we will consider classifiers with or without weighting of data on all datasets. From this test, we will continue with the best model/models and optimize with augmentation of data and balancing of data with sampling. Thus, applying the greedy assumption to save time. To clarify we will:

1. Determine the optimal dataset and classifier with or without weights.

2. Optimize model with augmentation and balancing of data.

This results in 24 + 8 combinations resulting in a training time of about 80h, much more reasonable time frame.

## 4.2 Results

In our results, we will compare results with the results achieved by the authors of GHC. This model will be referred to as the *baseline*. Further we will also compare our results to the *lazy classifier*, classifying all as the majority class (i.e. Non-Hate). In order to compare results, we will investigate the sample mean value and the sample standard deviation. Comparison will be done by t-test on a significance level of $\alpha = 0.1$.

### 4.2.1 Classifier and Data

In Table 4.2, we see the performance of all classifiers on all datasets. Pairwise comparisons shows that all transformer classifiers perform best on the Consensus Extended dataset. Furthermore, roberta-base trained on Consensus Extended performs significantly better then all other classifiers. Weighting data has been implemented as an attribute of the classifier, thus we will next determine if weighting data has a significant effect on the model. In Table 4.3, we see the performance of the classifiers with weighting on the datasets. bert-base-cased on Majority seems to be the best model at first glance. However, it is not significantly better than roberta-base (also on Majority), this is due too the high variance of roberta-base. Despite this, bert-base-cased achieves the largest F1 score. It is also clear that most models trained on Consensus Extended have a smaller standard deviation and are therefor more stable. T-tests shows that all classifiers perform significantly better without weighting. Comparing models in Table 4.2 and 4.3, we can conclude that the non-weighted roberta-base trained on Consensus Extended is significantly better than *all* other models. We will therefore continue in the next section with optimization of this model.

| Dataset | Classifier | Mean F1 | Std F1 | Max F1 |
|---|---|---|---|---|
| | lazy classifier | 47.5 | 0 | 0 |
| – | baseline | 58.0 | 2.00 | - |
| | LSTM | 48.0 | - | - |
| | roberta-base | 66.6 | 8.60 | 73.1 |
| | bert-base-cased | 70.7 | 0.97 | 72.1 |
| | xlnet-base-cased | 70.7 | 1.45 | 72.0 |
| Majority | albert-base-v2 | 68.1 | 1.64 | 70.8 |
| | roberta-base | 64.3 | 6.00 | 69.9 |
| | bert-base-cased | 64.7 | 6.00 | 68.8 |
| | xlnet-base-cased | 67.9 | 2.30 | 70.3 |
| Consensus | albert-base-v2 | 57.4 | 7.70 | 67.4 |
| | roberta-base | **73.1** | 0.50 | **74.1** |
| | bert-base-cased | 72.4 | 0.60 | 73.5 |
| | xlnet-base-cased | 72.0 | 0.47 | 72.7 |
| Consensus Ext. | albert-base-v2 | 71.6 | 1.80 | 73.3 |

**Table 4.2:** Mean value, standard deviation and max value of macro weighted F1 score (%) for multiple classifiers.

| Dataset | Classifier | Mean F1 | Std F1 | Max F1 |
|---|---|---|---|---|
| | lazy classifier | 47.5 | 0 | 0 |
| – | baseline | 58.0 | 2.0 | - |
| | roberta-base | 64.0 | 18.47 | **72.1** |
| | bert-base-cased | **70.7** | 1.03 | **72.1** |
| | xlnet-base-cased | 69.0 | 1.21 | 70.1 |
| Majority | albert-base-v2 | 68.0 | 1.43 | 70.3 |
| | roberta-base | 55.4 | 10.01 | 69.9 |
| | bert-base-cased | 63.5 | 9.07 | 71.0 |
| | xlnet-base-cased | 58.1 | 10.67 | 70.1 |
| Consensus | albert-base-v2 | 57.6 | 8.07 | 67.4 |
| | roberta-base | 69.4 | 0.57 | 70.0 |
| | bert-base-cased | 69.7 | 0.64 | 70.9 |
| | xlnet-base-cased | 68.4 | 0.96 | 70.4 |
| Conensus Ext. | albert-base-v2 | 68.1 | 1.80 | 70.1 |

**Table 4.3:** Mean value, standard deviation and max value of macro weighted F1 score (%) for multiple *weighted* classifiers. Note that lazy classifier and baseline does not weight data.

### 4.2.2 Optimization

Results in Table 4.4 show that no augmentation technique had an effect on this model. This was also confirmed by t-tests. One might argue that the model with only backtranslation achieves a similar result with lower standard deviation and thus a more stable model. However, we would have to execute more experiments to prove a significant difference. For now,

we conclude that we cannot say that backtranslation has a positive effect on the models performance.

| $TF \times IDF$ | Backtranslation | Oversample | Mean F1 | Std F1 | Max F1 |
|---|---|---|---|---|---|
| True | True | True | 68.9 | 0.82 | 69.9 |
| True | True | False | 71.9 | 0.64 | 72.7 |
| True | False | True | 71.6 | 0.68 | 72.8 |
| True | False | False | 71.6 | 0.63 | 72.5 |
| False | True | True | 71.0 | 0.74 | 72.1 |
| False | True | False | **73.2** | 0.30 | 73.7 |
| False | False | True | 72.3 | 0.27 | 72.7 |
| False | False | False | 73.1 | 0.50 | **74.1** |

**Table 4.4:** Mean value, standard deviation and max value of macro weighted F1 score (%) of non weighted roberta-base, trained on Consensus Extended with different optimization techniques.

## 4.3 Evaluation on Alt-Right Data

To get an unbiased evaluation of our model, we turned to the Alt-Right data. We selected a roberta-base classifier trained on the Consensus Extended dataset with no additives. As we can see in Table 4.5, we achieve a F1 score of 0.67, almost matching the performance on the GHC dataset.

| | Precision | Recall | F1 |
|---|---|---|---|
| **Non-Hate** | 0.98 | 0.93 | 0.96 |
| **Hate** | 0.27 | 0.62 | 0.38 |
| macro average | 0.63 | 0.78 | 0.67 |

**Table 4.5:** Caption

Inspection of the result was done on post level for each of the false positives, i.e. those posts wrongly classified as hate. This evaluation was done by the same person that had annotated the sentences. For roughly half of these sentences, classifying them as hate seems reasonable. They contain spiteful language and derogatory terms pointing out groups of people in a generalising and negative way. However, in the opinion of the annotator, this had not been on such a level as to be categorised as hate.

It seems very plausible that another annotator could just as well have marked these posts as hateful. Inspecting the false negatives (i.e. posts wrongly classified as Non-Hate) was also done. It is hard to say something definitive about why these particular Hate posts were missed, but a majority of them are to be considered racist. While racism is abundant in the forum posts, racist thoughts or references to antisemitism alone have not been enough to qualify a sentence as hate when annotating the data (though some people would argue that it should).

To better understand the performance, we analyzed results with Lime(Ribeiro et al., 2016). Lime provides a heatmap and a table over words which contribute to the classification. Thus, we can see on what basis the classifier made its choice. In Figure 4.1, we can see a post, correctly classified as Hate (true positive). In Figure 4.2, we see a post incorrectly classified as Hate (false positive).

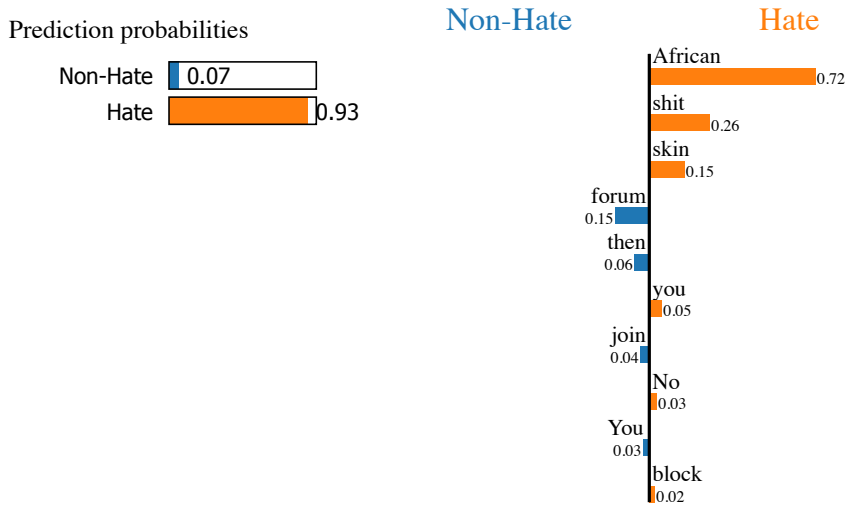


**Figure 4.1:** Lime representation of a *correctly* classified Hate post. Top Left: Class probabilities. Top Right: Table over most influential words. Bottom: Heatmap over analyzed text.

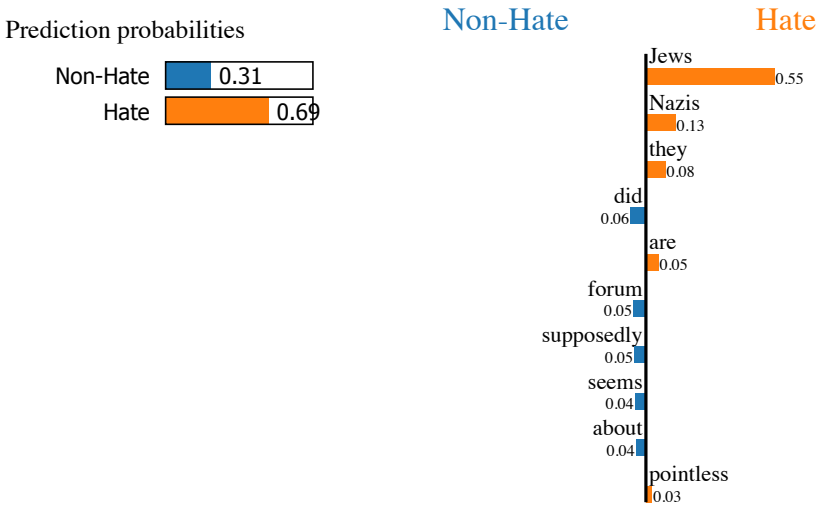


**Figure 4.2:** Lime representation of a *incorrectly* classified Hate post. Top Left: Class probabilities. Top Right: Table over most influential words. Bottom: Heatmap over analyzed text.

# Chapter 5

# Discussion

In this section, we will first discuss results and then continue with addressing positive and negative experience throughout the work. Last we will conclude the overall work.

## 5.1 Interpretations of Results

The variable with the greatest impact on performance was choice of training data. Models trained on Consensus Extended which excludes about 5000 posts, perform significantly better than those trained on the Majority datasets which includes all posts. This result is most likely due to the fact that Consensus Extended has a *cleaner* dataset than the Majority dataset. All excluded posts are posts where: majority vote would result in Non-Hate but annotators are not in complete agreement. Thus, the excluded posts contain uncertainty.

One could take this further and only include posts with complete agreement, i.e. Consensus. While this seems like a good idea in theory, our results show that models trained on Consensus perform worse than those trained on Consensus Extended, most probably due to the small size of the Hate-class in this dataset.

Optimizing the best performing model had no impact at all. This was surprising. Authors of the backtranslation method used in this work showed that models trained on data with backtranslated data outperformed the state-of-the-art models on multiple NLP tasks, one of the models being a fine-tuned BERT. Despite this, the models did not improve their performance. Implying that pre-trained classifier with transformer architecture that has been trained on Consensus Extended is well adapted for the task and does not improve performance with common optimization techniques.

Lastly, looking at the results in Section 4.3, we can see from the false positive in Figure 4.2 that the classifier had issues with targeted group labels, in this case “Jews”. As mentioned in the introduction, we can see in Kennedy et al. (2020b) that these results are not unexpected.

## 5.2 Experience

Our initial objective was to detect text strings which are related to and/or contain threats. Due to security reasons, the Swedish Security Service could not provide a dataset for the task. Therefore, we had to find an open sourced dataset well suited for the task. We ended up working with GHC.

While this dataset seemed to have all we needed, it became clear that the class of CV – the one most representative of the initial task, was to small for a classifier to learn the task. Hence, we turned to Hate speech recognition, a much larger class and a reasonable compromise. We have therefore come to focus more on violent extremism, instead of general threats.

After settling on task and dataset, we soon ran into our next problem, the baseline model. Our LSTM classifier was barley able to separate the data into two classes, performing only slightly better than the lazy classifier. We suspect this is due to the complexity of the task. We tried many different architectures, one of them used by a finalist of OffensEval 2019. After poor performance from LSTM, we turned to transformer-based classifiers which succeeded in separating the data after only a bit of fine tuning.

Exploring the data and creating different sub dataset was a large part of this work. It turned out, using a well constructed dataset with multiple annotators, was not as trivial as first thought. At first, we were satisfied with the majority dataset.

Only after examining the inter annotator agreement, we realized that we might be better of by excluding some data. Hence, there was a trade off between exclusion of *bad* data versus the size of the dataset. With limitless data one could exclude all data with disagreement and this would likely result in a great classifier. On the other hand, with very limited data, we might not want to exclude any data at all. Results show that a compromise between exclusion of bad data and class size was the best alternative, i.e. Consensus Extended.

We attempted to optimize the model with no success. The used augmenting techniques serve the purpose of teaching the model to perform better in a general setting. However, this had little to no effect on classifiers trained on the Consensus Extended dataset.

Lastly, large models were also implemented for all transformer-based models with some modifications to hyperparameters (max sequence length = 64, batch size = 8, epochs = 2). Due too the time consumption, we only trained these models for five sessions. All models had identical results on every single session, macro F1 score of 0.475. We have yet to understand this unexpected behaviour.

## 5.3 Future Work

Our first recommendation of future work is to explore all proposed combinations of hyper-parameters such as model-architecture, type of augmentation and so on explained in Section 4.1. As well as, exploring the model-level-hyperparameters such as learning rate, batch size and number of epochs, not only the ones recommended in the paper introducing BERT. Doing this is very time consuming, however, the greedy assumption is a naive approach. A much more suitable apporach is to assume that everything is possible, therefore one must explore all possible options.

Moreover, we suggest an investigate why all large transformer-based classifiers perform

as they do. It is most unlikely that all models are performing in this manner. We suspect that this might be consequence of the smaller batch sizes. However, it is most likely due to an error in the code. Solving this would presumably result in a model with a slightly better performance. Likewise, an improvement of performance is plausible if one where to implement the techniques proposed by Kennedy et al. (2020b), reducing the group identifier bias.

## 5.4 Conclusion

This work consists of two main parts. Part one is an Exploratory Data Analysis of the relatively new dataset GHC. In this part we get an insight on this new dataset. Noting the clear difference in language between the classes, we can conclude that an annotation has been done in a reasonable way. Additionally we have deduced that the data is indeed separable into two distinct classes. Lastly, we have observed that there exist a lot off inter annotator *disagreement*. An issue, but also an opportunity if one where to utilize this observation.

In part two we have seen that when using a corpus with multiple annotators, models can benefit from excluding data with inter annotator disagreement from the over-represented class. We have also seen that a transformer-based model with the RoBERTa architecture, trained on a cleaned dataset (i.e. Consensus Extended), surpassed the state-of-the-art for the task of identifying Hate speech on the Gab Hate Corpus. Furthermore, our results showed that a pre-trained transformer-based model that has been fine tuned on a the Consensus Extended dataset is not effected by either adding augmented data, weighting data nor sampling data.

The task of identifying hate speech is a global issue. This work serves as a complement to the many other works done on the subject. Further, we believe that there is a lot more knowledge to gain from the Gab Hate Corpus. E.g. by additional exploration of the dataset, implementation of suggested future work, and by the creation of brand new models. We have especially high hopes of exploring methods on selection of data with respect to inter annotator agreement since this was the key component of our results.

# References


Byrt, T., Bishop, J., and Carlin, J. B. (1993). Bias, prevalence and kappa. *Journal of clinical epidemiology*, 46(5):423–429.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Gelber, K. and McNamara, L. (2016). Evidencing the harms of hate speech. *Social Identities*, 22(3):324–341.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. `http://www.deeplearningbook.org`.

Hess, A. (2016). The far right has a new digital safe space. *New York Times*.

Kennedy, B., Atari, M., Davani, A. M., Yeh, L., Omrani, A., Kim, Y., Koombs, K., Havaldar, S., Portillo-Wightman, G., Gonzalez, E., et al. (2020a). The gab hate corpus: A collection of 27k posts annotated for hate speech.

Kennedy, B. J., Jin, X., Davani, A. M., Dehghani, M., and Ren, X. (2020b). Contextualizing hate speech classifiers with post-hoc explanation. *ArXiv*, abs/2005.02439.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

McCarthy, P.M., J. S. (2010). Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42:381—-392.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144.

Selyukh, A. (2017). Feeling sidelined by mainstream social media, far-right users jump to gab. *NPR*.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wester, A., Øvrelid, L., Velldal, E., and Hammer, H. L. (2016). Threat detection in online discussions. In *Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 66–71, San Diego, California. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M., and Le, Q. V. (2019). Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., and Kumar, R. (2019). Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Zannettou, S., Bradlyn, B., Cristofaro, E. D., Sirivianos, M., Stringhini, G., Kwak, H., and Blackburn, J. (2018). What is gab? A bastion of free speech or an alt-right echo chamber? *CoRR*, abs/1802.05287.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-12

**EXAMENSARBETE** Exploring Methods for Hate Speech Detection in Text
**STUDENT** Lucas Molsby
**HANDLEDARE** Pierre Nugues (LTH)
**EXAMINATOR** Jacek Malec (LTH)

# Utforskande av metoder för igenkänning av hets mot folkgrupp i text

POPULÄRVETENSKAPLIG SAMMANFATTNING **Lucas Molsby**

I detta arbete har vi tränat modeller att identifiera hets mot folkgrupp i en mängd text hämtat från Gab, en social mediaplattform känd för sin stora andel högerextrema användare. Genom noga utvald träningsdata och för-tränade modeller lyckades vi åstadkomma den bäst presterande modellen som finns att tillgå idag.

Identifiering av hets mot folkgrupp har många användningsområden. Den mest uppenbara användningen är hur sociala medier utnyttjar den för att sortera bort olämpliga inlägg. I Sverige är hets mot folkgrupp olagligt, därmed bryter det inte mot yttrandefriheten att exkludera sådana inlägg. Däremot är hets mot folkgrupp inte är ett lagbrott i alla länder. Trots det har de flesta sociala medier (inte alla) valt att förbjuda sådant innehåll. Ett annat användningsområde för identifieringen är som ett verktyg i arbetet mot våldsbejakande extremism. Syftet är då att identifiera en delmängd text som högst sannolikt innehåller hets mot folkgrupp. På så sett kan en granskare spara mängder med timmar av läsande. Det är av denna anledning som vi har gjort detta arbete på uppdrag av Säkerhetspolisen.

I detta arbete har vi jobbat med data från Gab, en social mediaplattform som *inte* sorterar bort hets mot folkgrupp. Datan vid namnet "Gab Hate Corpus" (GHC), innehåller ca 30,000 inlägg. Varje inlägg har blivit granskat av minst 3 av totalt 18 granskare på uppdrag av forskare vid University of Southern California. Granskarens uppgift är att markera ett inlägg som hets mot folkgrupp (HF) eller ej (EJ). Detta arbete är uppbyggt av två delar. Första delen är en analys av GHC, ett relativt nytt dataset. Den senare delen går ut på att träna modeller på uppgiften att identifiera hets mot folkgrupp med GHC som träningsdata. I del ett konstaterar vi att det är möjlig för en modell att lära sig att identifiera hets mot folkgrupp utifrån den givna datan. Vi visar sedan i den andra delen att en för-tränad modell som dessutom är finjusterad presterar bäst på uppgiften. Att en modell är för-tränad innebär att den under en längre tid har tränat på en stor mängd text. På så sett har modellerna en viss förståelse för betydelsen av en text, redan innan vi tränar den ytterligare på vår data. Finjusterad innebär att modellen har fått träna ytterligare på data från GHC. Analysen av GHC avslöjade att granskare sällan var helt överens. Vi valde därför att exkludera viss data där de fanns en osäkerhet.

Den slutgiltiga produkten består av ett program som har förmågan att träna och utvärdera modeller. Vidare är programmet skapat så att det finns möjlighet att integrera det i en applikation ämnad för granskning av text. Dessutom är arbetet en del i en gemensam globalt ambition att minska hets mot folkgrupp. Detta i syftet att dämpa verbala förtryck och attacker på utsatta folkgrupper på internetbaserade plattformar så som sociala medier och forum.