Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Maria Hilmo Jensen

Detecting hateful utterances using an anomaly detection approach

Master's thesis in Computer Science
Supervisor: Heri Ramampiaro
June 2020

NTNU
Norwegian University of Science and Technology

Maria Hilmo Jensen

Detecting hateful utterances using an anomaly detection approach

Master's thesis in Computer Science
Supervisor: Heri Ramampiaro
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Research on safety in social media has grown substantially in the last decade. With the widespread use of online services and social media, it has become easy to disseminate hateful messages. Freedom of speech is considered a human right in the Norwegian society; however, several statutory restrictions are prohibiting discriminatory and hateful statement. These posts are intended to be derogatory, humiliating or insulting, and are defined as hate speech. Many online communities dedicate massive amounts of resources towards the removal of such hateful contents, but the methods often rely on manual effort. A manual approach scales poorly, and for hate speech detection to be practically feasible, there is a need for systems that can automatically detect hateful expressions. Such automatic detection is a challenging task, and the majority of the research in the field is targeting the task using text classification approaches. However, despite the emerging scientific studies using these approaches, state-of-the-art solutions still suffer many drawbacks. This thesis explores the effects of re-phrasing the problem of hate speech detection by re-conceptualising hate speech detection as anomaly detection. Hence, this research aims at discovering if the problem can rightfully be considered an anomaly detection problem. Moreover, most of the existing methods use English datasets, so an enduring challenge in the research field is the lack of methods performing well on non-English datasets. Therefore, this thesis also investigates the possibility of creating a system that is language-independent.

A thorough literature review related to hate speech detection and anomaly detection was conducted to attain valuable insights. Based on the obtained knowledge, a deep semi-supervised anomaly detection approach to hate speech detection was proposed, which is based on the principle of entropy minimisation and consists of pre-trained Word Embeddings and a Convolutional Neural Network. Additionally, a Norwegian dataset, including a representative selection of topics, was collected and annotated. This dataset is a major contribution to the field of hate speech detection in Norwegian since an annotated baseline dataset did not previously exist. The developed system was used for experimentation with both an English and Norwegian dataset, and it achieved relatively good performance using both datasets. Utilising anomaly detection systems have several advantages over regular classification algorithms, such as not assuming similarities between the hateful content and being more suited for handling a real-scenario distribution between neutral and hateful content online. This indicates that using an anomaly detection approach might solve several persistent issues in the research field. The findings from this thesis suggest a huge potential for detecting hateful utterances using anomaly detection techniques, but it is still necessary to conduct more research for the system to be practically usable.

Sammendrag

Forskning på sikkerhet i sosiale medier har vokst betydelig det siste tiåret. Med den utbredte bruken av nettbaserte tjenester og sosiale medier, har det blitt enkelt å spre hatefulle meldinger. Selv om ytringsfrihet anses som en menneskerettighet i det norske samfunn, er det flere lovbestemte begrensninger som forbyr diskriminerende og hatefulle uttalelser. Slikt innhold er med hensikt nedsettende, ydmykende eller fornærmende, og er definert som hatefulle ytringer. Mange nettmiljøer dedikerer enorme mengder ressurser til fjerning av slikt hatefullt innhold, men metodene er ofte avhengige av manuelt arbeid. En slik manuell tilnærming skalerer dårlig, og for at deteksjon av hatefulle ytringer skal være mulig i praksis, er det behov for systemer som automatisk kan oppdage hatytringer. Slik automatisk deteksjon er en utfordrende oppgave, og det meste av relevant forskning prøver å løse problemet ved å ta i bruk metoder for tekstklassifisering. Til tross for de fleste vitenskapelige studiene bruker disse tilnærmingene, har de fremdeles mange problemer og ulemper. Derfor undersøker denne oppgaven virkningene av å konseptualisere deteksjon av hatefulle ytringer som anomalideteksjon. På denne måten utgjør hatefulle ytringer en avvikende variant av vanlig tale. Denne forskningen har altså som mål å avgjøre om problemet rettmessig kan betraktes som et anomalideteksjonsproblem. Videre bruker de fleste eksisterende metoder engelske datasett, så en varig utfordring innenfor forskningsfeltet er mangelen på metoder som gir gode resultater på ikke-engelske datasett. Derfor undersøker denne oppgaven også muligheten for å lage et språkuavhengig system.

For å oppnå verdifull innsikt ble en grundig litteraturgjennomgang relatert til både deteksjon av hatefulle ytringer og anomalideteksjon utført. Basert på den innhentede kunnskapen ble det foreslått en dyp anomalideteksjonsmetode basert på delvis veiledet læring til deteksjon av hatefulle ytringer. Metoden er basert på prinsippet om entropiminimering og består av forhåndstrente ord-vektorer og et konvolusjonelt nevralt nett (CNN). I tillegg ble det samlet og annotert et norsk datasett, bestående av et representativt utvalg av emner. Dette datasettet er et stort bidrag til forskningsfeltet som omhandler deteksjon av hatefulle ytringer på norsk, ettersom et slikt annotert datasett ikke eksisterte. Det utviklede systemet ble brukt til eksperimentering med både et engelsk og norsk datasett, og det oppnådde relativt god ytelse ved bruk av begge datasettene. Å benytte anomalideteksjonssystemer har flere fordeler sammenlignet med tradisjonelle klassifiseringsalgoritmer. For eksempel antar de ikke likheter mellom det ulike hatefulle innholdet og de er mer egnet for å håndtere en reell fordeling mellom nøytralt og hatefullt innhold på nett. Dette indikerer at bruk av en anomalideteksjontilnærming kan løse flere vedvarende problemer i forskningsfeltet. Funnene fra denne oppgaven antyder et enormt potensial for å oppdage hatefulle ytringer ved bruk av anomalideteksjonsteknikker, men det er fremdeles nødvendig å utføre videre forskning for at systemet skal være praktisk anvendelig.

Preface

This Master's Thesis was written by Maria Hilmo Jensen in the Spring of 2020, as a part of the Master of Science degree in Computer Science at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. I want to thank my supervisor Herindrasana Ramampiaro for all guidance, discussions and help throughout the semester. Moreover, I would like to express my appreciation to Tora Seim Gunstad and Marie Andreassen Svanes for good collaboration on the specialisation project and the collection of the dataset, as well as for interesting discussions on the topic.

I also want to express my deepest gratitude to fellow students, friends and family that agreed to help with the annotation of the collected Norwegian dataset. Furthermore, I would like to thank Lukas Ruff at TU Berlin for help and inspiration.

Maria Hilmo Jensen

Trondheim, 13th June 2020

Contents

1. Introduction 1

1.1. Background and motivation . . . 1

1.2. Goals and research questions . . . 3

1.3. Research method and environment . . . 4

1.4. Contributions . . . 5

1.5. Thesis structure . . . 5

2. Background Theory 7

2.1. Definition of hate speech . . . 7

2.2. Machine learning . . . 9

2.2.1. Learning algorithms . . . 9

2.2.2. Classical methods . . . 10

2.3. Anomaly detection . . . 11

2.3.1. Definition . . . 11

2.3.2. Approaches and algorithms . . . 13

2.3.3. Detecting anomalies in text . . . 14

2.3.4. Challenges . . . 15

2.4. Deep learning . . . 17

2.4.1. Artificial neural networks . . . 17

2.4.2. Recurrent neural network . . . 19

2.4.3. Convolutional neural network . . . 20

2.4.4. Autoencoders . . . 21

2.4.5. Attention . . . 21

2.5. Natural language processing . . . 22

2.5.1. Textual preprocessing . . . 22

2.5.2. Text representation . . . 24

2.6. Evaluation methodologies . . . 27

2.6.1. Techniques . . . 27

2.6.2. Metrics . . . 28

2.6.3. Inter-annotator agreement metrics . . . 30

2.7. Tools and libraries . . . 32

3. Related Work 33

3.1. Hate speech detection . . . 33

3.2. Existing data collections . . . 34

3.3. Anomaly detection . . . 36

3.4. Features in hate speech detection 39
3.5. Classification methods 42
3.6. Hate speech detection for non-English languages 45
3.7. Summary 46

4. Preparation of Data Collection 51
4.1. Collecting data 51
4.1.1. Preprocessing 53
4.2. Annotation procedure 54
4.2.1. User-based annotation 54
4.2.2. Guidelines 55
4.2.3. Inter-annotator agreement 57
4.3. Challenges 61
4.3.1. Annotation 62
4.3.2. Language issues 62

5. The ADAHS Approach 65
5.1. Text preprocessing 65
5.1.1. Semi-supervised setting 67
5.2. Model architecture 67
5.3. System functionality 70
5.3.1. Optimisation and regularisation 71

6. Experiments and Results 73
6.1. Experimental plan 73
6.2. Experimental setup 75
6.2.1. Datasets 75
6.2.2. Semi-supervised setup 78
6.2.3. Configurations and hyperparameters 79
6.2.4. Evaluation methodology 80
6.3. Experimental results 81
6.3.1. Results using the English dataset 82
6.3.2. Results using the Norwegian dataset 87
6.3.3. Results from the baseline methods 91

7. Evaluation and Discussion 93
7.1. Evaluation 93
7.1.1. General trends and observations 93
7.1.2. Results using the English dataset 95
7.1.3. Results using the Norwegian dataset 101
7.2. Discussion 107
7.2.1. Overall performance 107
7.2.2. Advantages 110
7.2.3. Disadvantages and challenges 112

7.2.4. Improvements 114
7.2.5. Language independence 115
7.2.6. Dataset annotation 116
7.2.7. Revisiting the research questions 117

8. Conclusion and Future Work **121**
8.1. Conclusion 121
8.2. Contributions 122
8.3. Future Work 123

Bibliography **127**

Appendices **135**

A. Additional Experimental Results **135**
A.1. Results using the English dataset 135
A.2. Results using the Norwegian dataset 142

B. Annotation Guidelines **145**

C. Collecting Social Media Content **151**
C.1. Twitter search words 151
C.2. Facebook 151
C.2.1. Sites and posts 151
C.2.2. Names 152

List of Figures

2.1. A feed-forward network architecture 18
2.2. A simplified architecture of a convolutional neural network 20
2.3. A usual text preprocessing pipeline 22
2.4. Word n-gram representation 26

5.1. An overview of the system architecture 67
5.2. All the layers in the neural network model 69
5.3. A detailed illustration of the CNN architecture 70

6.1. The distribution of comments in each category in the Norwegian dataset . 77
6.2. Loss and validation AUC for the English dataset 84
6.3. The ROC curve and Precision-Recall curve for the English dataset 84
6.4. Histograms of the anomaly scores for the English dataset 85
6.5. Loss and validation AUC for the Norwegian dataset 88
6.6. The ROC curve and Precision-Recall curve for the Norwegian dataset . . 88
6.7. Histograms of the anomaly scores for the Norwegian dataset 89

7.1. Confusion matrices for the English dataset 99
7.2. Confusion matrices for the Norwegian dataset 104

A.1. Validation AUCs for the English dataset in the unsupervised setting . . . 139
A.2. Validation AUCs for the English dataset in the normal setting 140
A.3. Additional confusion matrices for the English dataset 141
A.4. Additional confusion matrices for the Norwegian dataset 144

List of Tables

2.1. The interpretation of the kappa coefficient 31

3.1. Features in hate speech detection 41
3.2. Overview of related approaches 47

4.1. Pairwise utterance coherence between the annotators 58
4.2. The number of comments in each category labelled by the annotators . . 58
4.3. Hateful comments categorised by at least one annotator 59
4.4. Percentage agreement between each pair of annotators 60
4.5. Calculated inter-annotator metrics 60
4.6. Additional calculated inter-annotator metrics 61

6.1. Two random samples from the English dataset 75
6.2. The number of comments in each category in the English dataset 76
6.3. The number of neutral and hateful comments in the English dataset . . . 76
6.4. The number of comments in the annotated Norwegian dataset 77
6.5. Preprocessed combined annotated Norwegian dataset 78
6.6. AUC scores from testing hyperparameter values 82
6.8. AUC scores of the experiments on test 1 with the English dataset 83
6.9. Classification metrics using the English dataset 86
6.10. AUC scores of the experiments on test 2 with the English dataset 86
6.11. AUC scores for different values of η using the Norwegian dataset 87
6.12. AUC scores of the experiments on test 1 with the Norwegian dataset . . . 87
6.13. Classification metrics using the Norwegian dataset 90
6.14. AUC scores of the experiments on test 2 with the Norwegian dataset . . . 90
6.15. AUC scores for the experiments with OC-SVM on both datasets 91
6.16. AUC scores for the experiments with CVDD model on both datasets . . . 92

A.1. Results without decreasing normal samples in the English dataset 136
A.2. Comparison between the two approaches on the English dataset 137
A.3. Results with and without network bias terms 138
A.4. Alternative results from test 1 with the Norwegian dataset 142
A.5. Alternative results from test 2 with the Norwegian dataset 142
A.6. Comparison between the two approaches on the Norwegian dataset 143

B.1. Annotation guidelines 146

1. Introduction

The rise of social media and digital platforms has contributed to more people using their freedom of speech to participate in public debates, which is a positive factor for democratic participation. However, it is apparent that the debate is becoming increasingly polarised where thoughts and ideas based on hatred and fear are spread fast to many people via social media. The debate about hate speech has been central in recent years, and there has been an increasing legal pressure to remove this content from digital platforms. Even though hate speech is commonly defined as abusive language that targets specific group characteristics, it still does not have a formal definition. This makes it harder to moderate and detect hateful comments, and thus challenging to annotate new data. Despite that there is no general agreement on what expressions should be considered hateful, or what the limits of freedom of speech should apply to, there is still a broad consensus that hateful expressions are a significant societal problem. Hate speech keeps prejudice alive and deprives people of their dignity. Furthermore, repetitive hate speech may lead to a normalisation of negative attitudes in the population towards particular groups, which can lead to increased discrimination, harassment and violence towards people of these groups (Veledar, 2018). Hence, the work against hate speech is an essential contribution to the work for equality. This thesis focuses on the detection of hate speech in social media using anomaly detection approaches. It includes both an extensive literature review, a collection and annotation of a Norwegian dataset and a series of experiments.

1.1. Background and motivation

Freedom of speech is considered a human right and is a mainstay in the Norwegian society. However, there are several statutory restrictions prohibiting threats, defamation, harassment and various discriminatory and hateful statements (Elden et al., 2018). Despite this, it appears as if the threshold for writing hateful utterances online is lower due to anonymity and lack of supervision. Technology giants such as Facebook, Twitter and Google spend enormous amounts of resources on moderation, and they are still not able to reach a satisfactory level of moderation. The process of manual moderation is becoming increasingly time-consuming for human annotators due to information overflow, and automation of hate speech detection would thus allow earlier detection of harmful situations.

Previous studies focus mainly on dataset construction, text classification and automatic identification of hateful language. In later studies, many state-of-the-art approaches use word embeddings, often in combination with deep learning methods (Badjatiya et al., 2017; Zhong et al., 2016). A commonality for most of these studies is that they use text classification approaches and collect and label their own data, due to the lack of a standard corpus. Even though there is an abundance of research going on within the field, there are still some challenges. While the methods perform relatively well when presented, Gröndahl et al. (2018) found that several methods achieved poor performance when they were trained on one dataset and tested against another. The classifiers typically assume that similar data are likely to be a part of the same class, which does not necessarily hold for the hateful contents. Additionally, text classification systems require a balanced dataset to perform sufficiently. However, in a real-life scenario, the hateful statements only account for a small fraction of the total amount of data. This stresses the need for a system that can handle this drastic imbalance, and hence function efficiently on a dataset that represents real-life data. Creating a system that does not rely on a fully labelled dataset may also make it easier to utilise for several platforms, which may have access to a considerable amount of unlabelled data. Hence, this thesis explores the idea suggested by Gröndahl et al. (2018), to re-phrase the problem of hate speech detection by re-conceptualising hate speech detection as anomaly detection (AD), where hate speech constitutes an anomalous variant of ordinary speech. This concept is similar to how the common reader may perceive hate speech. To the best of our knowledge, there exist no methods that explore this idea.

Nobata et al. (2016) stated that abusive language evolves with time since people create new slurs and inventive ways to avoid being detected. For a regular classification model, this involves that the hateful content no longer fits into its assigned class. When using anomaly detection approaches, one does not assume similarities between the abnormal/hateful data. This is a major advantage compared to the classification algorithms, and AD methods might, therefore, be better suited to handle these language changes. This is a motivating factor for re-phrasing the problem.

Even though hate speech detection has previously only been considered a supervised learning problem, there exists no benchmark labelled dataset for this purpose, and the labelling of a large enough dataset is a very time-consuming process. One might not have a large labelled dataset, but one might have access to a small amount of labelled data verified as either neutral or hateful. An advantage would be to utilise these labelled samples in addition to the large portion of unlabelled samples. Anomaly detection is usually treated as an unsupervised learning problem, but semi-supervised approaches to AD aim to utilise such labelled samples. Therefore, this thesis intends to use semi-supervised learning in order to facilitate the utilisation of large amounts of unlabelled data.

Another challenge in the field is that most studies are conducted in English, which results in inadequate hate speech detection methods in other languages such as Norwegian. This is highly motivating for research in the field of Norwegian hate speech and collecting a standard Norwegian corpus.

1.2. Goals and research questions

This section presents the overall goal and proposed research questions for this thesis. The goal of this thesis is to conduct further research in the field of hate speech detection and apply anomaly detection techniques to separate hateful and neutral utterances. Based on this, the overall goal was formulated as follows:

Goal Investigate how to accurately detect hate speech in text using anomaly detection techniques.

Using anomaly detection techniques to detect hateful utterances are not yet explored. Hence, the work of this thesis aims at discovering if anomaly detection techniques can be applied to solve the problem of hate speech detection and thus if the problem can rightfully be considered an anomaly detection problem.

The following research question is formulated in order to achieve the main goal:

Research question How can effective hate speech detection be achieved by applying anomaly detection?

It is desirable to discover if there exists a potential for using anomaly detection approaches to detect hate speech. The objectives of this work are set to be achieved through practical experiments with a developed anomaly detection method, as well as a theoretical literature study, including relevant research in the field of hate speech detection and anomaly detection. The main research question proposed is a question formulated based on the overall goal, and can be decomposed into the following sub-questions:

Research question 1 Which principles and models are effective when using anomaly detection on textual data?

In order to develop an anomaly detection model that can detect hateful utterances, it is necessary to determine which principles, techniques and models to utilise. In order to gain insights into how to best make these choices, an extensive literature review is conducted. This study contains reviews of previous research within the field of anomaly detection, as well as in the field of hate speech detection. The findings are evaluated comprehensively to obtain valuable knowledge which is then used to determine which principles and what model to implement.

Research question 2 Would a semi-supervised deep learning model for anomaly detection be effective at correctly determining hateful social media comments?

Since it would be advantageous to be able to utilise unlabelled data, as well as a smaller part of labelled data, it is desirable to implement an anomaly detection model that uses semi-supervised learning. This implemented model is then applied to the task of detecting hate speech through a series of experiments. The experiments investigate the anomaly detection method's ability to distinguish the hateful language from offensive and neutral language. They are designed to explore different scenarios, which includes investigating the effect of labelled samples on system performance, and the system's ability to handle novelties by simulating the unpredictable nature of anomalies. Furthermore, the method's robustness to increasing pollution of unlabelled anomalous samples is investigated. This is in order to imitate the real-life scenario where it is difficult to determine if all of the unlabelled samples are normal samples.

Research question 3 How to develop a method for hate speech detection based on anomaly detection that is language independent?

This research question is closely related to Research Question 2 and investigates the method's ability to handle a non-English dataset. To be able to test the method with a dataset of another language, it is necessary to collect and annotate a dataset consisting of Norwegian social media comments. The method's performance is tested using this created dataset, as well as an English dataset.

1.3. Research method and environment

To answer the research questions and accomplish the overall goal, several methodologies have been used. The first step to accomplish the overall thesis goal was to adopt an exploratory approach that was used to conduct a qualitative analysis of relevant research in the field of hate speech detection and anomaly detection, i.e. conduct a detailed literature review. This was necessary to achieve theoretical insights on both topics, and understand how existing solutions attempt to solve the problem at hand. Furthermore, this was important in order to gain relevant knowledge that could be utilised to propose a new possible solution. This step is also related to research question 1, which requires exploration of existing principles, techniques and models related to anomaly detection on textual data.

Experiments were conducted primarily to answer the second and third research question, which involves the implementation of a semi-supervised anomaly detection method that can separate between hateful and neutral speech, as well as an extensive set of experiments used to test this method. Here, the primary strategy is the design and creation of a practical solution. The experiments focus on discovering if there is a potential for using anomaly detection techniques to detect hateful utterances, as well as how the amount of labelled training data and pollution in the training data, affect performance. The focus is to do a quantitative research experiment and to use the achieved results to determine if hate speech detection can, in fact, be addressed as an anomaly detection problem. Hence, I plan on conducting a quantitative data analysis on the results of this thesis. Another vital part of the experimental research was to test if the implemented model is language independent, as proposed in the third research question. To be able to test the implemented model using a non-English dataset, a Norwegian dataset consisting of social media comments was created. The collection of this dataset started in the specialisation project in the Autumn of 2019 (Jensen et al., 2019) and continued in this thesis. Both the specialisation project and the further work related to the creation of the dataset was conducted together with Tora Seim Gunstand and Marie Andreassen Svanes.

1.4. Contributions

The exploratory work conducted in this thesis contributes to further research in the field of hate speech detection, by being the first to investigate if anomaly detection techniques can be used to separate hateful and neutral utterances. It provides a deeper insight into an evolving research field, and hopefully, it helps to bring the research one step closer to a reliable method for detecting hate speech in social media. The main contributions of this work are listed below:

1. A thorough literature review of existing research related to hate speech detection.
2. The creation of a large labelled dataset containing Norwegian tweets and comments from Facebook and Resett.
3. The development of a deep learning method for detecting hate speech based on anomaly detection techniques, using semi-supervised learning, pre-trained word embeddings and a convolutional neural network.
4. Experimentation with the implemented method on two datasets; one in English and another one in Norwegian.

1.5. Thesis structure

The remainder of this thesis is organised as follows:

Chapter 2 introduces and describes relevant theories, technologies and methods used in the work of this thesis and related work.

Chapter 3 provides a detailed overview of related work in the field of hate speech detection, including literature related to the definition of hate speech, existing data collections, features and methods. Further, the chapter includes relevant work conducted within the field of anomaly detection.

Chapter 4 outlines the creation and preparation of a labelled dataset consisting of Norwegian comments. The chapter includes the collection of data, preprocessing, annotation procedure and guidelines, as well as the inter-annotator agreement calculations. Moreover, it finishes with a discussion of important challenges related to annotation and language.

Chapter 5 explores the developed method that utilises an anomaly detection approach to distinguish hateful and neural utterances. The chapter includes a detailed description of the system, including text preprocessing steps, the implemented architecture and the system's functionality.

Chapter 6 presets the conducted experiments, including the experimental plan, setup and results.

Chapter 7 evaluates the obtained experimental results and discusses the findings in relation to the proposed research questions.

Chapter 8 gives a conclusion to the thesis and presents suggestions for future work in order to improve anomaly detection approaches to hate speech detection.

The thesis also contains three appendices, which presents additional results, complete annotation guidelines and information related to the collection of the Norwegian dataset.

2. Background Theory

When addressing the problem of detecting hateful expressions, it is crucial to provide a precise definition of hate speech. Hence, this chapter starts with defining hate speech and how it can be separated from offensive language. Furthermore, it provides the relevant background theory within the fields of machine learning, deep learning, anomaly detection and Natural Language Processing (NLP), including necessary concepts and techniques. It presents both technologies and architectures, as well as tools and libraries used in this thesis and in relevant work presented in Chapter 3. A significant part of the relevant background material was written as a part of the specialisation project preceding this thesis (Jensen et al., 2019). New background information about relevant topics such as anomaly detection, pre-trained word embeddings, transfer learners, autoencoders and the attention mechanism has been added to this chapter. Most of the sections initially written in the specialisation project have been modified, but still contain parts that have not been altered.

2.1. Definition of hate speech

The term "hate speech" is ambiguous, and there is no unified national or international definition of the phenomenon. The term may also appear misleading because it gives the impression that the sender must have a subjective sense of hatred for an utterance to be hateful. However, according to Veledar (2018, p. 37), whether there is a sense of hatred that drives the sender, or whether it is a political conviction, ideology, prejudice, xenophobia or otherwise, is not decisive for an utterance to be hateful. What is decisive is how the ordinary audience perceives the utterance, given the context in which it is presented.

Although there is no universal definition of hate speech, the most accepted definition in the research field is provided by Nockleby (2000): “any communication that disparages a target group of people based on some characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics”. In all, there seems to be a pattern shared by most of the literature reviewed (Davidson et al., 2017; Dennis Gitari et al., 2015; Djuric et al., 2015; Nobata et al., 2016; Nockleby, 2000; Schmidt and Wiegand, 2017; Silva et al., 2016) where hate speech is defined as a deliberate attack directed towards a specific group of people motivated by actual or perceived aspects that form the group’s identity.

According to Schmidt and Wiegand (2017), it is difficult to define what is hateful, because what is considered a hateful expression might be influenced by aspects such as the domain of an utterance, its discourse context, the exact time of posting and world events at this moment, identity of the author and target recipient, as well as context consisting of co-occurring media objects (images, videos, audio). This thesis will not focus on the identity of authors and co-occurring media objects; however, the importance of these areas are emphasised.

Another factor to take into consideration is that hate speech may have strong cultural implications. As pointed out by Davidson et al. (2017), an utterance may be perceived as offensive or not depending on one's cultural background and perceived relation to this culture. For example, the words *hoe* and *bitch* are rather normal when quoting rap lyrics, but in another context, they should be perceived differently. Besides, whether or not an utterance is hateful is often subjective. As Xiang et al. (2012) states "The notion of vulgarity is rather subjective and the degree of offensiveness varies considerably among people." This statement can be further substantiated by Ross et al. (2017) and Schmidt and Wiegand (2017). They both emphasise the issues regarding annotation of hate speech datasets, and even though the annotators have common annotation guidelines, the agreement score amongst the annotators are often deficient.

The understanding of hate speech depends on whether one uses a legal (and thus narrow) understanding of the term, as stated in the Norwegian Penal Code §185 and which includes only the most severe statements, or whether one uses a broader social science understanding of the phenomenon. Examples of hate speech which have been penalised by the Norwegian Supreme Court is:

(1) *Fandens svarte avkom reis tilbake til Somalia og bli der din korrupte kakkelakk.*

(2) *Det er vel bedre at vi fjerner disse avskyelige rottene fra jordens overflate selv tenker jeg!!*

(3) *Ja de forsvinner den dagen disse steppe bavianene reiser dit de hører hjemme!*

Comment (1)[1] can by the ordinary reader be perceived as a severe offence, with reference to skin colour and thereby ethnic origin. Regarding comment (2) and (3),[2] the Supreme Court found that the first comment was related to Muslims, while the second comment was aimed towards dark-skinned people. Hence, all of these statements were considered a violation of §185 in the Norwegian Penal Code (often referred to as the clause of racism).

Applying a too narrow understanding of the term "hate speech" will offer some methodological challenges, because it will be complicated to draw the line between the "illegal" and "legal" hate speech (Veledar, 2018). Therefore, in this thesis, the definition of hate

[1] https://www.domstol.no/Enkelt-domstol/hoyesterett/avgjorelser/2020/hoyesterett---straff/hr-2020-184-a/

[2] https://www.domstol.no/Enkelt-domstol/hoyesterett/avgjorelser/2020/hoyesterett---straff/hr-2020-185-a/

speech is the one provided by Veledar (2018, p. 11), which is based on the definitions by the European Commission against Racism and Intolerance's (ECRI) and The Danish Institute for Human Rights:

> Stigmatising, derogatory, abusive, harassing or threatening statements affecting the dignity, reputation and status of an individual or group through linguistic and visual means that promote negative emotions, attitudes and perceptions based on characteristics such as ethnicity, religion, gender, disability, sexual orientation, age, political outlook and social status.

As opposed to hate speech, offensive language is defined as terms that are applied to hurtful, derogatory or obscene comments made by one person to another person or towards a group. The difference from hate speech is the severity of the statement. It is vital to separate hate speech from other instances of offensive language; just because a message contains a particular term does not make it hateful, and neither makes it automatically offensive. It is often challenging to separate offensive from hateful speech, and a reason is the lack of a universal definition, which leads to subjective opinions. Furthermore, hate speech and offensive speech often contains many of the same terms and are particularly difficult to distinguish because of the many nuances of natural language. In general, hate speech is more than profane words; it can be precise and sophisticated. Thus, this is another challenge when separating hateful language from offensive language.

2.2. Machine learning

Machine learning is a field in computer science concerned with the study of algorithms and statistical models aiming to create techniques for solving complex problems without using explicit instructions. Such problems are hard to solve using conventional programming methods, but machine learning algorithms can solve many of these severe problems in a generic way by relying on patterns and inference (Rebala et al., 2019). Essentially, the algorithms learn from datasets of variable size by examining the data to find common patterns and explore differences. Machine learning is an application of artificial intelligence that provides computer systems with the ability to learn from experience. By comparison, artificial intelligence is a much broader field of study, where the focus is to understand and build intelligent entities (Russell and Norvig, 2010). The following sections present the different types of learning algorithms used in machine learning and some classical methods commonly used in the field of hate speech detection.

2.2.1. Learning algorithms

In order to understand the terminology used throughout the rest of this thesis, general types of learning algorithms used in machine learning are presented. Machine learning algorithms differ in how they learn and what data they input and output, as well as the type of problem they are trying to solve. Therefore, they are usually divided into different categories/learning models. The most prominent learning models are *Supervised Learning, Unsupervised Learning* and *Semi-supervised Learning.*

Supervised machine learning algorithms utilise a labelled training dataset, i.e. the training set contains both the inputs and the known desired output. After sufficient training, new input data can be provided to the algorithm. Based on the key characteristics, the model predicts the most likely output (Rebala et al., 2019; Russell and Norvig, 2010). Typical problems supervised algorithms are designed to solve are classification and regression problems, and it is frequently used to classify text. On the other hand, unsupervised algorithms are used when the dataset used to train is neither classified nor labelled, i.e. an unlabelled dataset. In other words, the algorithm learns patterns and trends of similarity based on the input even though no explicit feedback is supplied (Russell and Norvig, 2010). Unlike supervised algorithms, these algorithms cannot find a correct output, but instead, they can draw an inference to describe hidden structures. The algorithms can be used in, for example, pattern detection and text clustering, as well as anomaly detection. Semi-supervised learning algorithms fall somewhere between supervised and unsupervised learning algorithms. These algorithms are provided with both labelled and unlabelled data, typically a small amount of labelled and a larger amount of unlabelled data. One of the biggest advantages of this approach is that it is not necessary to spend much time labelling the entire dataset (Rebala et al., 2019).

2.2.2. Classical methods

Schmidt and Wiegand (2017) stated that, previously, classical machine learning methods were mainly used in the field of hate speech detection, but more recently, neural networks and deep learning methods tend to outperform these methods. However, much research utilises these methods, either as their primary model or as a baseline. Since hate speech detection has previously only been regarded as a classification problem, these methods are used in a large proportion of relevant research. These commonly used classic methods involve supervised learning and include naïve Bayes, logistic regression, support vector machines, gradient boosted decision trees and random forest.

Naïve Bayes classifiers are a set of simple, yet powerful probabilistic models used for solving classification tasks. The decisive part of all the classifiers are based on Bayes' theorem (Bayes, 1763), and the classifiers differ mainly in their chosen decision rule. Even though the classifiers assume conditional independence, which is rarely accurate in real-world situations, they have proven useful in text classification tasks (Russell and Norvig, 2010). Cox (1958) first proposed logistic regression (LR), which is a supervised machine learning model used for classification that originated from the field of statistics. Initially, the LR model provided a binary prediction indicating if a specific outcome would be achieved or not, but it was later expanded to also work on multi-nomial values. Support vector machines (SVMs) were first introduced by Cortes and Vapnik (1995) and are particularly prominent when you do not have any prior knowledge about a domain (Russell and Norvig, 2010). SVMs often produce significant accuracy with less computation power, and it can be used for both regression and classification tasks. A decision tree is a decision support tool that represents a function for making a decision based on input data. The decision tree has a tree-like structure, and the full paths from the root node to the leaves serve as the classification rules. Russell and Norvig (2010) states that the decision tree learning algorithm uses a greedy divide-and-conquer approach, and as a result, the main problem can be divided into smaller sub-problems that can be solved recursively. Gradient boosting is a technique that produces a prediction model in the form of an ensemble of weak prediction models and hence, converts weak learners to strong learners.[3] Decision trees are used as the weak learner in gradient boosting; thus gradient boosting decision trees are decision trees that use the gradient boosting technique. Random forest is an ensemble learning method and a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset. The concept behind random forest is that a large number of relatively uncorrelated models, in this case, decision trees, operating together will outperform any of the individual models.[4]

2.3. Anomaly detection

Anomaly detection is the process of finding data objects with behaviours that differ from the norm or the expectation. If using anomaly detection techniques to solve the problem of detecting hateful utterances, it implies that hate speech must be considered anomalous variants of normal speech. This consideration involves creating an anomaly detection model that can efficiently separate between normal and abnormal textual utterances. In order to understand how to address this problem, this section presents the definition of anomalies and anomaly detection and explains approaches and common algorithms. One of the sections will focus specifically on anomaly detection on text data, as this is of particular interest in this thesis. The last part of this section addresses important challenges related to the utilisation of these techniques to detect hate speech.

2.3.1. Definition

Generally, an anomaly is an outcome or value that can be considered a variation of the norm, which means that anomalies deviate from the expected. According to Alla and Adari (2019), anomalies typically fall into three categories:

Data point-based anomalies: These anomalies can seem comparable to outliers in a set of data points. Outliers are data points that differ from the norm but are still expected to be present in the dataset. These instances may occur in the dataset due to unavoidable random errors or systematic errors.

Context-based anomalies: These consist of data points that at first glance appear to be

[3]https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14

[4]https://towardsdatascience.com/understanding-random-forest-58381e0602d2

normal, but are considered anomalies in specific contexts. For example, if a person owns an electric car and suddenly buys gasoline, this purchase would seem out of place. On the other hand, for a person that owns a gasoline car, this would be completely normal. In hate speech detection, some comments might be normal in some contexts but hateful/anomalous in others.

Pattern-based anomalies: These anomalies are patterns and trends that deviate from their historical counterparts. It can often be time-series data, where the goal is to identify periods of abnormal behaviour.

As mentioned, anomaly detection is the process of discovering or detecting anomalies using advanced algorithms. Hence, the purpose is to identify unexpected items or behaviour in datasets that differ from the norm. Related to this is both outlier detection and novelty detection, which one can call options of anomaly detection. Outlier detection is according to Alla and Adari (2019, p. 15-19), a technique that aims at discovering outliers within a given dataset. These models are given a dataset and then decides which of the instances are outliers and which are normal.

On the other hand, novelty detection aims at discovering novelties. Novelties are data points that are not previously seen by the model, e.g. they are not a part of the training dataset. Novelty detection and outlier detection are very similar. However, the key difference is that the novelty detection models learn what is considered regular data points and then try to determine if new data instances are anomalies or not (Alla and Adari, 2019, p. 18-19). The detection of hateful utterances can hence be categorised as both outlier detection and novelty detection since we expect some hateful comments, but the hateful comments might not be similar to what we have seen before. Nevertheless, this thesis uses novelty detection techniques.

Anomaly detection approaches are based on the assumption that normal data are stationary, which means that the underlying processes do not change significantly over time. Hence, the approaches are based on past data. The models assume that statistics characterising a system in the past will continue to characterise the system in the future. If one is dealing with data that changes over time, the data may, in some cases, be characterised by long-term trends or cyclic behaviour (Mehrotra et al., 2017, p. 4).

In many anomaly detection problems, one can separate the data into two classes; normal and abnormal/anomalous. In this case, it might be tempting to address this problem using classical machine learning classification algorithms, such as support vector machines or decision trees. However, in the case of anomaly detection, this approach will rarely be successful due to the drastic imbalance between the two classes. The anomalous instances are sporadic compared to the normal instances, which will often result in too many false negatives (Mehrotra et al., 2017, p. 6). Another issue using normal classification is that the anomalies might not resemble each other. Classifiers typically assume that similar data are likely to be a part of the same class. This assumption often holds for the normal class but is crucially invalid for the anomalies (Ruff et al., 2020), because all of the anomalous data might not fit into a single class. Also, if an anomalous data point is a novelty, the algorithm will not be capable of labelling the point as a part of the anomaly class. When detecting hate speech in a real-life scenario, the majority of the utterances will be neutral and hence, normal instances. According to Veledar (2018), approximately 10% of the comments on the Facebook pages of *NRK* and *TV 2* were hateful. From a social perspective, this can be regarded as a large proportion of the comments, but for a classification model, a dataset with only 10% hateful comments will be considered too imbalanced.

2.3.2. Approaches and algorithms

According to Han et al. (2012), one can differentiate between several approaches to anomaly detection:

Proximity-based: These methods assume that data objects that are far away from other objects in feature space are considered anomalous. The effectiveness of these methods relies heavily on the proximity measure used. There are two major types of proximity-based anomaly detection; distance-based and density-based.

Clustering-based: These methods assume that the normal data objects belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or do not belong to any clusters.

Classification-based: The idea of classification-based anomaly detection methods is to train a classification model that can distinguish normal data from outliers.

Distance (similarity) measures are the basis of distance-based anomaly detection. Popular applied similarity measures include direct measures, such as Euclidean and Minkowski distances, but also measures such as cosine similarity and Jaccard index (Mehrotra et al., 2017, p. 34). Distance-based anomaly detection algorithms are assured to work well only when the different neighbourhoods, consisting of data points, are characterised by approximately equal densities. This assumption is often violated, giving unfortunate results. On the other hand, in a density-based approach, anomalies are considered to be those data objects that are in regions with relatively low density. Hence, these methods review the density of a specific point and compare it with the density associated with its neighbours. Well-known density-based algorithms include local outlier factor (LOF) and connectivity-based outlier factor (COF)(Mehrotra et al., 2017, p. 107-113).

Clustering-based approaches detect anomalies by examining the relationship between objects and clusters. The methods declare anomalies to be those data points that are outside of clusters, in relatively dense clusters or near the boundaries of clusters (Mehrotra et al., 2017, p. 33-55). Methods that fall into this category include clustering methods based on distance, such as k-nearest neighbours and k-means clustering, as well as methods based on density such as DBSCAN.

One can use classification-based methods when labelled data is available. As previously mentioned, in an anomaly detection problem, there is often a significant imbalance between the number of ordinary data objects and the number of anomalies. Consequently, we are left with an insufficient representation of the anomalous data. Therefore, to overcome this challenge, these approaches often use a *one-class model*, which is a model build only to describe the normal data (Han et al., 2012, p. 571). The one-class support vector machine model is an example of a one-class model and is described further in the following section.

One-Class SVM

One of the main issues with using regular SVMs for anomaly detection is that they are designed to handle *two* classes which needs to be separated using a decision boundary. As previously mentioned, assuming two classes in anomaly detection will be invalid for the anomaly class, and hence, an alternative algorithm is needed. A One-Class Support Vector Machine (OC-SVM) is a modified support vector machine used to identify unusual or anomalous data from a given dataset. Hence, it is well-suited for both anomaly detection and novelty detection. The models build upon the idea that they are trained solely on normal data and can consequently be used to detect anomalies/novelties in new data when they are presented. According to Zhou et al. (2008, p. 645), OC-SVMs have been widely applied in many areas, such as outlier ranking and density estimation. Moreover, the models have played an essential role in the field of intrusion detection. Generally, OC-SVMs is considered an unsupervised learning algorithm. Nevertheless, since training on only one class is considered training on "partially labelled" data, the algorithm can also be used for semi-supervised learning (Alla and Adari, 2019, p. 51).

Furthermore, according to Alla and Adari (2019, p. 51-52), an OC-SVM model is good at handling high-dimensional data and has a great ability to capture the shape of the data. However, according to Aggarwal (2017, p. 92), an issue with OC-SVMs is that they can be sensitive to the choice of kernels and many hidden parameters associated with the method. Hence, it is essential to set the regularisation hyperparameters and kernel hyperparameters in order to obtain satisfactory results (Zhou et al., 2008, p. 645).

2.3.3. Detecting anomalies in text

Mahapatra et al. (2012) stated that in the textual domain, anomaly detection techniques aim at uncovering novel and interesting topics and words in a document corpus. Furthermore, they affirm that anomaly detection in text data finds broad applicability in several domains. Because of the ubiquity of text in social media, emails and blogs, there are several applications of anomaly detection on textual data in web applications. For instance, it can be used to detect important events or unusual topics from Twitter streams. Furthermore, it can be used to detect a subset that corresponds to spam in a stream of emails. There are many conceptual similarities between high-dimensional and sparse data and text data. The reason is that the text data is often represented using a vector format, which is usually high-dimensional and sparse. Therefore, Aggarwal

(2017, p. 262) claims that most of the probabilistic, linear, and proximity-based methods for multidimensional data can be generalised to text. However, there are differences in how these models are implemented caused by the sparsity and non-negative nature of text. Besides, a principal challenge related to models using a textual data corpus is the well-known *curse of dimensionally.* With this increasing dimensionality, many of the conventional anomaly detection methods do not produce satisfactory results. It also becomes difficult to effectively detect anomalies when analysing the data in full dimensionality. The reason is that the anomalies may be masked by the noise effects of having multiple irrelevant dimensions (Aggarwal, 2017, p. 149).

Probabilistic models are models that assign probabilities of memberships to classes or clusters, instead of assigning a distinct label to data objects. These models are often used for probabilistic clustering (also called *soft* clustering) of text documents, which is essentially an application of the Expectation-Maximization (EM) algorithm to text data (Aggarwal, 2017, p. 262).

2.3.4. Challenges

This section presents and discusses important challenges that should be addressed when using anomaly detection methods to detect hate speech.

Modelling data normality

When solving a problem using anomaly detection techniques, the quality of the results highly depend on the modelling of normal and abnormal data objects. In order to achieve valuable results, the model has to represent the two classes effectively. According to Han et al. (2012), one challenge that may arise is that building a comprehensive model for data normality is hard, or even impossible. The reason is that it is often difficult to find all possible normal behaviours in a system. Furthermore, when AD techniques are applied in the textual domain, is it challenging to handle the significant variations in documents belonging to each class or topic (Chalapathy and Chawla, 2019). Additionally, in many systems, there is not a clear cut between what is normal and what is not. Hence, many data instances can be labelled wrongfully. A possible solution to this exact problem can be to measure the degree of "outlier-ness" instead of giving a distinct evaluation as either normal or abnormal.

In many domains, normal behaviour evolves frequently, and the current notion of normality might not be sufficient to represent normal instances in the future (Chalapathy and Chawla, 2019). This is a major challenge with regards to using AD techniques on hate speech detection. First of all, defining the boundary between neutral and hateful language is challenging because what is considered hateful might be influenced by other aspects such as the domain, its context and world events at this moment. Also, what is considered hateful varies over time, and hence, so does what is considered normal (neutral or offensive). As a result, defining a normal region which encompasses every possible normal behaviour overtime is practically impossible.

Choosing the correct similarity measure

For an anomaly detection algorithm to perform sufficiently, it is crucial to determine the correct similarity measure. Unfortunately, these choices tend to be application-specific because different application domains often have very different requirements. Hence, anomaly detection is highly dependent on the application type, which makes it exceptionally difficult to develop a universal method (Han et al., 2012). Furthermore, anomaly detection methods are prone to noise in the dataset as this can distort the data and make it challenging to create a distinction between the ordinary data objects and the anomalies.

Choosing the optimal threshold

According to Aggarwal (2017), most anomaly detection algorithms assign an anomaly score to each data sample in the test set. This score indicates the extent to which the model believe the sample is an anomaly, which results in a ranking of the data samples. Hence, to be able to induce binary labels and determine if a data sample is an anomaly or not, a threshold is chosen. Samples with a score above this threshold are considered anomalous. Choosing the optimal threshold can be difficult, and is often domain-specific. If the threshold is selected too restrictively, then the algorithm misses real anomalies, and hence the number of false negatives increase. On the other hand, if the algorithm declares too many data samples as anomalies, then it leads to too many false positives (Aggarwal, 2017). Related to hate speech detection, this involves choosing to flag too many comments as potentially hateful versus letting some of these comments remain unnoticed.

Sarcasm and subjectivity

Related to all hate speech detection problems, is the problem of handling aggressive comments disguised as sarcastic irony. *Irony* is when an individual state the opposite of what is implied, and *sarcasm* is the mockery, derision or ridicule of a person using irony. Sarcasm is intentionally ambiguous, and even humans may struggle to understand and interpret the content. Sarcasm disguises the actual intention of the statement, which is challenging to recognise for a machine. A system typically identifies a statement as neutral, when it is, in fact, sarcastic. Of course, not all sarcastic comments can be considered hate speech, but according to Frenda (2018), sarcasm is a commonly used figure of speech to express negative opinions. Furthermore, they discovered that hate speech detection systems might experience difficulties detecting sarcastic, abusive tweets. Hence, this is a significant issue and is also considered its own research field. Handling sarcastic irony is not considered in this thesis, but we emphasise its importance.

Moreover, as was mentioned in Section 2.1 and will be further discussed in Section 3.1, the subjective interpretation of hate speech is an important challenge. As Davidson et al. (2017) stated; an utterance may be perceived as offensive depending on one's cultural background, which can lead to issues when determining what is hateful and what is not.

2.4. Deep learning

Deep learning is a sub-field of machine learning which in recent years has experienced noticeable growth in popularity. A significant difficulty in many artificial intelligence applications is the impact variation has on the observable data, and how challenging the extraction of features can be on such data. As opposed to classical supervised and unsupervised methods, deep learning automatically extracts relevant features during training and in this way solves this significant problem.

The machine learning methods described in Section 2.2.2 require choosing features carefully to function well and extracting these features can be challenging. Deep learning extract features and thus solves this issue by building representational hierarchies containing multiple abstraction levels. Goodfellow et al. (2016) describe deep learning as a type of machine learning that achieves great power and flexibility by being able to learn complex concepts out of simpler ones. The lowest level of the hierarchy contains simple concepts, and it is typically working on much simpler representations of data than what is used in other machine learning approaches. On the other hand, the higher hierarchical levels use increasingly complex concepts, based on the lower simpler levels.

The main challenge with deep learning models is that they generally require a large amount of data to perform well, along with a great deal of computational power. Today, when data availability and computational power is not an issue, deep learning is used increasingly to solve many machine learning problems, including anomaly detection problems. In recent years, there has been an increasing interest in deep anomaly detection algorithms (Chalapathy and Chawla, 2019). These approaches are motivated by the limited scalability of shallow AD techniques, and the need for methods that can handle large and complex datasets.

There are several variations to deep learning models, and this section will briefly describe some of the models used in natural language processing and anomaly detection.

2.4.1. Artificial neural networks

Artificial neural networks (ANNs), also called multi-layer perceptrons, are networks inspired by the human brain and is one of the models used in deep learning. It is a set of networks that consists of highly interconnected processors, called nodes or neurons, that imitate biological neurons. These biological neurons are connected through synapses, which in neural networks corresponds to weighted links that send signals between nodes. The network has a fixed number of external inputs to specific nodes, as well as a fixed number of outputs from other specific nodes. Each node takes several input signals, sums them and produces an output based on an activation function (Rebala et al., 2019). This function performs a non-linear transformation and is the reason that neural networks are capable of learning both linear and non-linear functions. A node can then be mathematically described as:

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right) \tag{2.1}$$

where a_i is the output from node i, g is the activation function and $w_{i,j}$ is the weight of the connection between node i and j. The learning happens by adjusting the weights between each node using gradient descent, which is a method for optimizing a function (Rebala et al., 2019). Neural networks can be used to create both supervised, semi-supervised and unsupervised machine learning models and are very useful for solving complex problems where other conventional methods do not produce accurate results.

The simplest way to connect a neural network is as a feed-forward network. A feed-forward network is a network that only has connections one way, from the input layer, through hidden layers (if some) to the output layer and in this way forms a directed acyclic graph. There are no internal states in the network and in this way it represents a function of its current inputs (Russell and Norvig, 2010). Feed-forward networks are generally arranged in layers where each node only receives inputs from its immediately preceding layer and the computations are done layer by layer (Rebala et al., 2019). One often distinguishes between single-layer networks where the information precedes immediately from the input nodes to the output nodes and multi-layer perceptrons/networks (MLPs) that contains one or more hidden layers. A simple feed-forward network containing one hidden layer is shown in Figure 2.1.

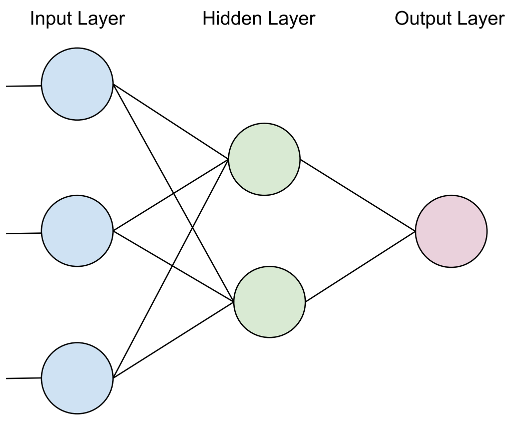


Figure 2.1.: A feed-forward network architecture with one input layer, one hidden layer and one output layer

Feed-forward networks can solve many problems, but they are not the only kind of networks used in modern deep learning.

Deep neural networks

Deep neural network (DNN) is a variant of neural networks composed of several layers. These networks are distinguished from the single-hidden-layer neural networks by their depth, which is the number of layers the data must pass through. According to Rebala et al. (2019), deep neural networks usually refer to neural networks with many layers and a large number of neurons where each extra layer increases the complexity of the network. This allows them to represent more complex functions than shallow neural networks.

Both recurrent neural networks and convolutional neural networks are examples of neural networks that can be categorised as deep, which are explained in the following sections.

2.4.2. Recurrent neural network

Recurrent neural networks (RNNs) presented by Rumelhart et al. (1986) have recurrent values, meaning that they have units that are linked in cycles. In other words, the network feeds its output back to its inputs and hence uses feedback. The presence of these cycles has a profound impact on the network's learning capability. Unlike feed-forward networks, RNNs enables short-term memory and can use this internal state to process a series of inputs (Russell and Norvig, 2010). In this way, the output from the system will depend on the internal state which in turn may depend on previous inputs. These dynamic networks are best suited for processing sequential data, e.g., text or time-series data (Rebala et al., 2019). Furthermore, they can handle sequences of much greater length than regular MLPs.

Long Short-Term Memory

A Long Short-Term Memory (LSTM) network is a variation of a recurrent network and was proposed by the German researchers Hochreiter and Schmidhuber (1997). These gradient-based networks included so-called Long Short-Term Memory cells and were introduced as a solution to the RNNs vanishing gradient problem; The gradient expresses the change in all weights concerning the change in error. When the gradient vanishes, the weights cannot be adjusted and learning will stop. The LSTM networks are used to address the problem of modelling long-term dependency in recurrent neural networks and they can solve complex long-time-lag tasks that are not possible to solve with a basic recurrent network.

Rebala et al. (2019) states that LSTM networks have been very successful in modelling problems related to natural language processing with strong long-range dependency modelling. LSTM can be used to learn the long-distance contextual dependency (order information) among words. Wang et al. (2018) conducted experimental results which showed that given enough training data the methods can learn the word usage in the context of social media. These findings can be useful for further experiments with textual data.

2.4.3. Convolutional neural network

A convolutional neural network (CNN) is a variation of a feed-forward network. Goodfellow et al. (2016) describe convolutional networks as neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Convolution is a technique that automates the extraction and combination of important features which is necessary for identifying a target class. Simply put, thy can be though of as sliding window functions applied to a matrix. This sliding window is often called a kernel or a filter, and it can have variable sizes. A CNN usually consists of several layers that combine convolution and pooling, followed by a neural network. The pooling layer(s) reduce the dimensions of the inputs. A simplified architecture of CNN can be seen in Figure 2.2.

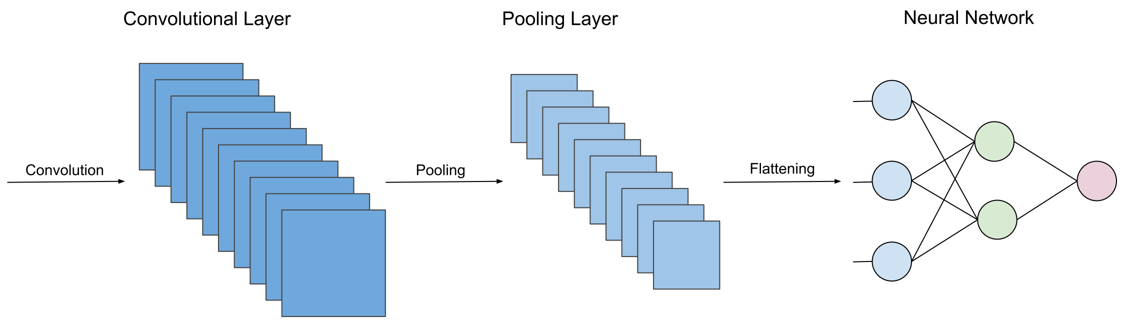


Figure 2.2.: A simplified architecture of a Convolutional Neural Network. The network contains one convolution layer, one pooling layer and a fully connected neural network

As opposed to regular multi-layer networks, the first layers involved in convolution in a convolutional network are not fully connected. This means that all nodes in one layer are not connected to all nodes in the preceding layer. Goodfellow et al. (2016) states that CNNs are mainly used for processing data that has a grid-like topology such as images, but they can also successfully be applied to problems within the field of natural language processing. For instance, CNN can be used on text by splitting sentences into words and represent the words as numerical vectors. These features are then fed into a convolutional layer. The filters can be of different height and correspond to the number of adjacent rows considered jointly, i.e. the n-grams (a 1xn filter) within the text. A representative number is given as output from pooling the results of the convolution and sent to a fully connected neural network. If one is considering a classification problem, then the network may output a probability for each class, whereas if one is considering an anomaly detection problem, then the output might be a vector. Either way, the decision is based on weights assigned to each feature. Thus, CNN is effective as "feature extractors" as they are good at extracting combinations of words or characters.

2.4.4. Autoencoders

Autoencoders are a type of neural networks that can learn efficient data representations and uses this to reconstruct its inputs to its outputs. Hence, they are useful for detecting anomalies. The model consists of two parts: an encoder and a decoder. The encoder's job is to reduce high-dimensional data into a lower-dimensional and dense representation, which is also known as a latent representation. The decoders job is to convert and hence, expand, this low-dimensional data into the original input (Alla and Adari, 2019, p. 123-126). As a result, the autoencoder neural network has the same amount of input nodes as output nodes. The model uses the neural network's property of backpropagation to learn normal behaviour, and hence being able to detect when anomalies occur. When the network is trained on solely normal data instances, the aggregated error of reconstruction will be higher for data that does not fit the description of "normal". Hence, this reconstruction error can be used to quantify the outlier score (Aggarwal, 2017, p. 102). The hidden layer, and hence the encoding, have fewer units than the input layer, the model is forced to prioritise which aspects of the input to keep and what to discard (Goodfellow et al., 2016, p. 505). In this way, it learns to preserve the useful parts of the input and discard the irrelevant parts.

2.4.5. Attention

The key idea behind the attention mechanism is to pay "attention" to the relevant source content, and thus create short-cut connections between the source and target. The attention mechanism was first introduced by Bahdanau et al. (2015) followed by Luong et al. (2015), where they both used it to improve the quality of Neural Machine Translation (NMT) systems. An NMT system is a sequence to sequence (Seq2Seq) model, which is a particular case of the recurrent neural network typically used to solve complex text problems, such as language translation, speech recognition and question answering. The most common architecture used to build Seq2Seq models is encoder-decoder architecture. The encoder builds a numerical context vector based on the input sequence, and the decoder processes this vector in order to return the output sequence, which is a translated sentence in the case of NMT. This setup works fine for short and medium sentences, but this fixed-size vector becomes a bottleneck when handling longer sentences.[5] In this case, the network might forget the earlier parts once the whole sequence is processed. The attention mechanism aims at solving this problem by capturing global information over all the items in an input sequence.

[5]https://github.com/tensorflow/nmt

2.5. Natural language processing

Natural language processing (NLP) is a subfield of computer science and linguistics, concerned with the interaction between humans and computers using natural language. It is challenging for a computer system to be able to interpret ambiguous and unstructured language correctly. Therefore, the field of NLP offers methods and techniques to make human languages possible to understand and process for a computer system. NLP is an important part of hate speech detection. It is crucial to preprocess and make the text mathematically computable so that a machine learning model can make sense of it. In this section, text preprocessing and text representation used in Natural Language Processing are presented.

2.5.1. Textual preprocessing

For a machine to be able to understand natural language, the text needs to be preprocessed. This is an essential step in NLP, especially when handling user-generated content such as comments or short messages from social media. A typical textual preprocessing pipeline is illustrated in Figure 2.3. All the steps are not necessary for every situation, but this pipeline includes the most common preprocessing steps.

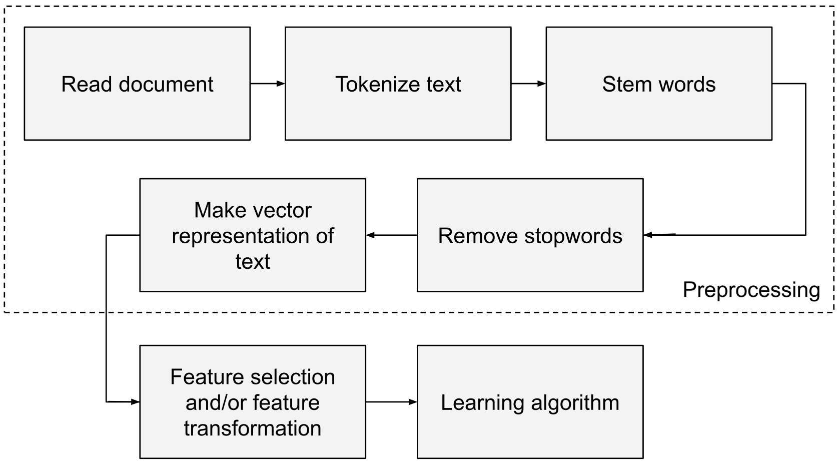


Figure 2.3.: A usual text preprocessing pipeline followed by feature selection and a learning algorithm (Ikonomakis et al., 2005).

The process starts when a document is read. Next, the text is typically tokenised, meaning the document is converted into a sequence of tokens. A token provides the link between documents and queries and is usually itself a sequence of alphanumeric characters from A to Z and from 0 to 9. According to Büttcher et al. (2016), the tokenisation is critical in the process of text classification because it allows for more effective processing of search queries by limiting the number of queries that the system can process. Once the text is tokenised, word stemming or lemmatisation may be applied. Word stemming considers the morphology, more commonly known as the internal structure, of terms and reduces the term to a word stem. The word stem is just a smaller form of the word and is not necessarily the same root as a dictionary-based morphological root. Stemming is done to easily compare words such as "runs" and "running", which both are reduced to "run", and also because stemming increases retrieval effectiveness for specific queries (Büttcher et al., 2016). *Porter's Algorithm*, or *Porter Stemmer*, is known as the most common algorithm used for stemming English texts according to D.Manning et al. (2009). Lemmatisation is a similar but more calculated process than stemming. It focuses on removing inflectional endings only and return a word that is either the base or a lemma. A lemma is a word in its dictionary form. To be able to find the lemma of each word, a part of speech (POS) tagger is needed. A POS Tagger is a software that reads the text and assigns each word with a part of speech, such as noun, verb or adjective. An example to display the difference between stemming and lemmatisation is the word *saw*. While stemming might return only the character *s*, lemmatisation might return either *see* or *saw* depending on the context and to what part of speech the POS Tagger assigns the word.

Another factor for more effective queries is the removal of stopwords. Many common words may potentially have little value for retrieval purposes, so at query time these are removed from the query. These stopwords are usually frequent terms, thus removing them leads to a size reduction of the indexing structure, which implies reduced execution times. The retrieval is then based solely on the remaining terms (Baeza-Yates and Ribeiro-Neto, 2011). Examples of stopwords in the Norwegian language are "alle", "det" and "og". Other preprocessing methods include the removal of special characters, such as emoticons, or frequently used characters such as hashtags (#) and mentions () from Twitter messages.

The next step is to represent the text as a numerical vector. This is useful when the ordering of terms is relevant or when terms are repeated (Baeza-Yates and Ribeiro-Neto, 2011). A common approach is that each term is represented by a vector filled with zeros except for a one that represents the element at the index corresponding to the word in the text. These vectors can be used further as features. The topic of text representation is crucial in NLP tasks and will be further elaborated in Section 2.5.2.

After the text is represented as numerical vectors, it might be necessary to select the relevant features from the data. The default for text classification is to use terms as features. However, only a few classifiers operate directly on the textual representation (Büttcher et al., 2016). Furthermore, it is possible to increase the performance of the classifiers by adding additional features which are suited to a specific problem, so that each document is represented as a collection of features. The process of defining and extracting features that might be relevant is generally referred to as *feature engineering*. Various features may be applied, and this can include features derived directly from the text or extrinsic information related to it. An example of features that are purely based on the given text is simple surface features, such as the document length, number of characters or number of symbols, while the time an e-mail arrived can be an example of extrinsic information that may be a relevant feature. The goal of *feature selection* is to improve effectiveness and computational efficiency by filtering irrelevant or redundant features. To be able to reduce dimensionality, it is necessary first to identify relevant features and eliminate the ones which do not add significant value. Another method to achieve this is by feature extraction. Here, the original feature space is converted to a new and reduced space, where all the original features are replaced by a smaller representative set. This is useful when the number of features in input data is too large (Indira Gandhi et al., 2015).

Once the data is preprocessed and represented numerically, it can be used as input to a learning algorithm. The learning algorithm may, for instance, be a classification or clustering model, or in the case of this thesis, an anomaly detection model. Since deep learning models extract features automatically, these steps are typically used when handling a classic/shallow machine learning model.

2.5.2. Text representation

After the text is preprocessed, it needs to be represented numerically. Each term of the document representation is considered a separate variable, or *feature*. This is a technique generally referred to as *text representation*, and is concerned with the achievement of a numerical representation of the unstructured text, thus making it mathematically computable. The rest of this section will focus on methods used for text representation, and hence present various popular ways to represent text numerically.

Bag of words

Bag of words, or BoW, is a simple representation of queries and documents. Here, the text is represented as a bag that contains its words, with no regards to word order or grammar. Thus the text is represented very simply by term conjunctive components which reflect the terms they contain. The number of occurrences of a particular term in the text is counted because the important factor is the presence of a word, and not where it occurs. This makes it possible to use the frequency of each term to find the keywords of the document and make decisions based on the presence or absence of a particular word. The bag-of-words-model is used for feature extraction and modelling, based on the assumption that documents are similar if they have similar content. Another usage is to calculate term frequency, which is explained in the following section (Baeza-Yates and Ribeiro-Neto, 2011).

TF-IDF

TF-IDF is, according to Baeza-Yates and Ribeiro-Neto (2011), the most popular term weighting scheme used in information retrieval. TF-IDF is based on term frequency (TF) and inverse document frequency (IDF) and determines the importance of a term in a document.

Baeza-Yates and Ribeiro-Neto (2011) defines TF and IDF as follows:

Term Frequency The value, or weight, of a term k_i that occurs in a document d_j is simply proportional to the term frequency $f_{i,j}$.

Inverse Document Frequency Let k_i be the term with the r-th largest document frequency, i.e., $n(r) = n_i$. Associated with the term k_i the inverse document frequency, IDF_i, is given by:

$$IDF_i = \log \frac{N}{n_i} \tag{2.2}$$

where N is the number of documents in the collection.

This leads to the definition of the TF-IDF weighting, as proposed by Salton and Yang (1973):

Let $w_{i,j}$ be the term weight associated with the pair (k_i, d_j). Then, we define

$$w_{i,j} = \begin{cases} (1 + \log f_{i,j}) \times \log \frac{N}{n_i} & \text{if } f_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

which is referred to as TF-IDF weighting scheme (Baeza-Yates and Ribeiro-Neto, 2011).

N-grams

N-gram is a simple language model that assigns probabilities to word and character sequences. An n-gram is a sequence of n words or n characters, such as 1-gram (unigram), 2-gram (bigram), 3-gram (trigram) and so on. The optimal value for n will vary from language to language (Büttcher et al., 2016). Character n-grams treat overlapping sequences of n characters as tokens. A particular use case for character n-grams is to reduce the problem of spelling variation in user-generated data. An example of this can be the word "f@aen" which is a variation of "faen", but the words still convey the same message. With $n = 4$ and the word "nordmenn" as an example, the result is the following character 4-gram:

nor nord ordm rdme dmen menn enn

Word n-grams can be used to estimate the probability $P(w|h)$ of word w given a history h. The n-gram model looks n words into the past to estimate the probability of word w. An example of a trigram model would be $P(en|han\ er)$. Assigning probabilities to n-grams is useful to help decide which n-grams that can form single entities together. Use cases include spelling error corrections, likely suggestions for misspelt words or prediction of the next word or characters in a sequence. For instance, the sentence "drikk kafe" could be corrected to "drikk kaffe" if the word "kaffe" had a higher probability of occurring after the word "drikk". An example of a word n-gram representation of the sentence "Katten liker ikke å bade" is shown in Figure 2.4.

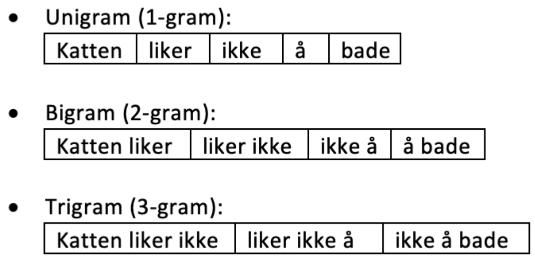


Figure 2.4.: Word n-gram representation

Word embeddings and transfer learners

A word embedding is a numerical and distributed word representation based on a word's context. For each word, a vector representation is induced from a text corpus. A word embedding can capture the context of a word in a document, relation with other words and semantic similarities. The idea is that words with the same meaning, such as "great" and "good" in these examples "*Have a great day!*" and "*Have a good day!*", occupy close spatial positions and are not categorised as having nothing to do with each other. Thus, similar words may end up having similar vectors. This is an advantage compared to regular one-hot encodings, where all words are represented independently of each other.

Word embeddings were first introduced by Bengio et al. (2003). They propose to fight the curse of dimensionality by learning a distributed representation for words, where each of the training sentences provides the model with several semantically similar sentences. The vector representations are typically created by using neural networks, and hence, they are well suited as input features for other neural networks. Word embeddings have become highly popular, and it exists several pre-trained word embeddings that have already been trained on a large document corpus. Commonly known pre-trained models are Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and FastText (Mikolov et al., 2017).

Word2vec is a predictive algorithm which is built upon two models; the Continuous Bag-of-Words (CBOW) model and the Skip-gram model. The purpose of the CBOW model is to predict a word from surrounding words, whereas the skip-gram model aims at predicting a word's surrounding words. The GloVe (short for Global Vectors) algorithm is an extension of this model. It is a statistical approach that leverages global matrix factorisation and local context window methods to create a global log-bilinear regression model. Both of these models capture the local word-to-word co-occurrences from a corpus, but in addition, GloVe also captures global co-occurrence statistics. FastText tries to solve Word2Vec's problem of generalisation to unknown words. It is doing so by accounting for word parts and characters. This makes it possible to train on less data since more information is extracted from the text.

Recently, large language models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) have gained popularity within the field of NLP. These models learn dynamic sentence embeddings in an unsupervised manner and have proven to be very useful for transfer learning. However, these large models are very computationally intensive.

Word2vec, GloVe and FastText word embeddings are context-independent, which means that they output one vector for each word. This is a consequence of not considering the order of words. On the other hand, ELMo and BERT can generate different word embeddings for a word that captures the context of a word, i.e. its position in a sentence.

2.6. Evaluation methodologies

When dealing with a machine learning problem, it is vital to evaluate and test the performance of the implemented model. While training the model is an important step, how the model performs on unseen data is a crucial aspect that should be considered. It is necessary to know if it actually works and if its predictions are trustworthy. The model might merely memorise the data it was trained on, and consequently, be unable to predict correctly on new data. Therefore, it is essential to measure the performance of the model. This section describes techniques for evaluation of a machine learning model and also presents various metrics that are often used when evaluating hate speech detection systems. All of the presented metrics are common when dealing with both classification problems and anomaly detection problems.

2.6.1. Techniques

The first step in the evaluation of a machine learning model is to split the data into three categories: a *training* set, a *validation* set and a *test* set. In order to improve accuracy, the training set is used to build predictive models by adjusting the weights of the model. The validation set is not a part of the training data and is used to assess the performance of the model and avoid overfitting to the data in the training set. With a validation set, it is possible to select the best performing model after fine-tuning the data, and validate that the model is improving. The test set consist of unseen data and is used to assess the performance of the finalised model.

The accuracy of this evaluation can vary based on how the data was split into categories, and unwanted bias can become part of the model. Cross-validation is a regression method used to avoid this. *K-Fold Cross Validation* is where a dataset is partitioned into K folds, and each fold is then used as a testing set. In the first iteration, the first fold is used to test the model and the rest to train the model. This process is repeated until each of the K folds have been used to test the model. Regression is then performed on the combined results, which are then regarded as one validation set (Büttcher et al., 2016).

2.6.2. Metrics

Evaluating the quality of the results using performance metrics provides a quantitative measure which makes it easier to effectively determine a model's accomplishments. Evaluation metrics are used to make informed decisions based on quantitative measures to increase performance and discover better approaches. Comparing the effectiveness of one method against another in the same situation is useful when deciding which method best achieves their intended purpose and meets the user's need (Büttcher et al., 2016). In this section, various evaluation metrics are presented. All the presented metrics require having ground-truth labels available. In this case, where the datasets are adapted from imbalanced classification problems, the ground-truth labels can be used to measure the performance of the anomaly detection system.

Precision and recall

Precision and *recall* are two of the most widely used evaluation metrics. Usually, the metrics are used for measuring the effectiveness of set-based retrieval, and they can be used to evaluate anomaly detection models that possess labelled data samples (Aggarwal, 2017, p. 27-28). Assume that an anomaly detection model outputs an anomaly score. By setting a threshold, the score can be converted into a binary label for each data sample. For any given threshold t on the anomaly score, the declared set of anomalies is denoted by $S(t)$. Furthermore, let G denote the true set consisting of all anomalies (based on the ground-truth labels). Then, for any given threshold t, the precision is defined as the fraction of retrieved documents that are in fact outliers:

$$Precision(t) = \frac{|S(t) \cap G|}{|S(t)|} \tag{2.4}$$

On the other hand, recall is the fraction of ground-truth outliers that have been retrieved:

$$Recall(t) = \frac{|S(t) \cap G|}{|G|} \tag{2.5}$$

Recall is used to evaluate the effectiveness of tasks where the user wants to find all relevant documents. This evaluation metric measures how meticulously the search results meet the user's information need. By varying the parameter t, it is possible to plot a curve based on the precision and recall values. This is called a precision-recall curve (PRC), and it shows the relationship between precision and recall for fixed recall levels from 0% to 100%. For each recall point, the curve plots the maximum precision achieved at that recall level or higher. The curves are normally used to compare retrieval quality of distinct retrieval algorithms because it allows evaluation of both the fraction of relevant documents found and the quality of the results itself. A precision-recall curve can also be used to clarify where a retrieval algorithm is most suitable. For instance, if an algorithm has higher precision at lower recall levels, it will be more suitable for the Web. On the other hand, if the algorithm has higher precision at higher recall levels, it may be more suitable for use cases such as legal applications (Baeza-Yates and Ribeiro-Neto, 2011). The baseline of PRC is determined by the ratio of positives (P) and negatives (N) as y = P / (P + N). Hence, for a balanced dataset, the baseline is 0.5.

F-measure

Several of the previous works on hate speech detection use another measure to evaluate their models. F-measure, also known as the F_1-score, is an accuracy measure that provides another way of combining precision and recall. It is defined as follows:

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}} = \frac{2 * R * P}{R + P}. \tag{2.6}$$

Here, R represents recall and P represents precision. F-measure is the harmonic mean between these two metrics, and it brings a balance between recall and precision. Its use cases include serving as a measure for search, query classification and document classification performance, as well as evaluating named entity recognition or word segmentation. An F_1-score varies between 0 and 1, where 1 equals perfect precision and recall and is the optimal value. If needed, the formula can be weighted to focus more on either recall or precision.

Area under the ROC curve

A *receiver operating characteristic* (ROC) curve is a probability curve that is closely related to the precision-recall curve (Aggarwal, 2017, p. 28). It is used as a performance measure, and it tells how much a model is capable of distinguishing between classes. The ROC curve provides a geometric characterization of filter effectiveness, plotted with the true positive rate (TPR), or recall (Equation 2.5), on the y-axis against the false positive rate (FPR) on the x-axis. FPR gives the proportion of falsely predicted positives out of the ground-truth negatives. For a dataset D with ground-truth labels G and threshold t, the false positive rate is given by Equation 2.7.

$$FPR(t) = \frac{|S(t) - G|}{|D - G|} \tag{2.7}$$

The degree of separability between classes is measured in the *area under the curve (AUC)*. The higher the AUC, the better the model is at predicting the correct label (Büttcher et al., 2016). A model with a good measure of separability will have AUC near to 1. Likewise, a poor model will have AUC near 0. AUC near 0.5 means the model cannot separate between classes which equal random guesses.

2.6.3. Inter-annotator agreement metrics

When creating a linguistic data collection, it is common to have multiple people annotate the same data and then compare the annotations. There might be several reasons why this is desired, for example, to validate the annotation guidelines or identify difficulties within the annotation procedure. The evaluation, which often is a comparison, can, for instance, be a qualitative examination of the annotations or a quantitative examination based on the calculations of agreement metrics. Either way, the annotation variation between annotators must be examined to assure the quality and reliability of the dataset. According to Artstein (2017), an annotation process is reliable if the annotations yield consistent results. This section presents the most common inter-annotator agreement metrics, which are used in Chapter 4. The metrics are intended to provide a quantitative measure of the magnitude of agreement between observers.

Observed agreement

The easiest way to measure the level of agreement between annotators is to use the *observed agreement*, or *raw agreement*. This measure equals the percentage agreement between the annotators. Hence, it is calculated by counting the number of items for which the annotators provide identical labels and divide this by the total number of annotated items. The metric is easy to understand and calculate, and according to Bayerl and Paul (2011), it is the most common way of reporting agreement. The drawback to this approach is that it does not account for the chance that the agreement might be by accident (Artstein, 2017, p. 299).

Kappa and alpha

Coefficients in the kappa and alpha family are intended to calculate the amount of agreement that was attained above the level expected by chance (Artstein, 2017, p. 300). Hence, they attain the expected level of agreements given a scenario.

Cohen's kappa (Cohen, 1960) measures agreement between two annotators while considering the possibility of an agreement by chance. **Fleiss kappa** (Fleiss, 1971) have many similarities to Cohen's kappa, but it allows for more than two annotators. When there are more than two annotators, the agreement is calculated pairwise. However, according to Artstein (2017), the coefficients are not compatible because they differ in their conceptions of an agreement by chance.

Let $\mathrm{A_o}$ be the actual/observed agreement and $\mathrm{A_e}$ be the expected agreement. Then, both Cohen's and Fleiss' kappa (κ) can be calculated using the simplified formula presented in Equation 2.8:

$$\kappa = \frac{\mathrm{A_o} \text{ - } \mathrm{A_e}}{1 - \mathrm{A_e}} \tag{2.8}$$

Viera and Garrett (2005) provided an overview of how to interpret the kappa score. This is presented in Table 2.1.

Table 2.1.: The interpretation of the kappa coefficient

Kappa	**Agreement**
< 0	Less than chance agreement
0.01−0.20	Slight agreement
0.21−0.40	Fair agreement
0.41−0.60	Moderate agreement
0.61−0.80	Substantial agreement
0.81−0.99	Almost perfect agreement

Unlike Cohen's and Fleiss' kappa, **Krippendorff's Alpha** can assess agreement among a variable number of annotators and also accepts non-annotated examples (Bobicev and Sokolova, 2017). Krippendorff's α is similar to Fleiss' κ, but is expressed in terms of disagreement, rather than agreement. α is calculated from the simplified formula in Equation 2.9, where $\mathrm{D_o} = 1 - \mathrm{A_o}$ and $\mathrm{D_e} = 1 - \mathrm{A_e}$.

$$\alpha = 1 - \frac{\mathrm{D_o}}{\mathrm{D_e}} \tag{2.9}$$

α does not treat all disagreements equally and uses a distance function in order to set a specific level of disagreements between each pair of labels. The observed disagreement is then calculated by counting the number of disagreeing pairs, rather than the agreeing pairs. Furthermore, each disagreement is scaled by the appropriate distance given by the distance function (Artstein, 2017).

2.7. Tools and libraries

Many tools and open-source libraries are developed to provide easily accessible and reusable components. There exist many such tools and libraries for working with textual data, and they make the job more convenient. `Python` was chosen to be the main programming language, primarily because of its extensive support for machine learning libraries. `PyTorch`[6] with `PyTorch-NLP`[7] was used to implement the deep learning method. This was chosen because of its ability to utilise a machine's graphics processing unit (GPU) that supports CUDA. Furthermore, `pandas`[8], `skikit-learn`[9] and `NLTK`[10] were used for a variety of tasks, such as reading and handling data, creating the OC-SVM baseline method and calculate evaluation metrics. Lastly, Twitter's REST API was used to collect tweets.

[6]https://pytorch.org/
[7]https://pytorchnlp.readthedocs.io/en/latest/
[8]https://github.com/pandas-dev/pandas
[9]https://scikit-learn.org/stable/
[10]https://www.nltk.org/

3. Related Work

The state of the art and related work on hate speech detection were reviewed and carried out in the specialisation project preceding this thesis (Jensen et al., 2019). This is amended with a discussion of papers based on anomaly detection that have become relevant for this thesis, as well as a few additional papers on hate speech detection that have become available after the project.

In this chapter, the current state of the art within the field of hate speech detection are discussed. This includes the existing research on hate speech detection, hate speech datasets and an overview of research on anomaly detection, both in general and on textual data. Furthermore, relevant features and text representation methods commonly used for hate speech detection are discussed. Even though this thesis aims at using anomaly detection techniques to detect hate speech, a significant part of the relevant research on hate speech detection involves classification models. Hence, an overview of the current classification methods used in the research field is also included. In addition, research on hate speech detection for non-English languages is provided. The chapter finishes with a table containing a complete summary of all the research directly related to hate speech detection, which is presented previously in this chapter.

3.1. Hate speech detection

Hate speech is a field which has continuously received increased attention in the research community, and thus the amount of research has also grown accordingly. Despite this, there are still many challenges within the field. Nobata et al. (2016) summed up some of the major challenges including; the subjective interpretation of hate speech and offensive language, the difficulty of annotating data, the evolving language, and the lack of a benchmark dataset.

The subjective interpretation of hate speech and offensive language is already mentioned in the definition of hate speech in Section 2.1 and will be discussed further in the annotation procedure described in Section 4.2. Dinakar et al. (2012) stressed that “the presence of profane content does not in itself signify hate speech. General profanity is not necessarily targeted towards an individual and may be used for stylistic purposes or emphasis. On the other hand, hate speech may denigrate or threaten an individual or a group of people without the use of any profanities.” Thus, hate speech detection is more than just detecting profane words. Hate speech can be both grammatically correct and fluently written. In addition, as already mentioned in Section 2.1, Schmidt and Wiegand (2017) underlined what is considered a hate speech message might be influenced by a lot of different aspects and Davidson et al. (2017) point out that different offensive terms often have cultural implications; a word can have a different meaning to specific groups. Due to these difficulties, Ross et al. (2017) and Waseem and Hovy (2016) both investigate the difficulty of annotating data by analysing the reliability of annotators, which includes the difference between the annotation quality when annotators have a definition or not, and whether they are experts. They both concluded that the reliability did not improve when they were given a definition. However, annotators should be given a more detailed definition of hate speech before annotating. This highlight the problem of subjective interpretation of hate speech detection.

Another problem for hate speech detection is an evolving language. As users get moderated, they will use new variations of spellings, words and other methods to learn how to avoid getting moderated. Lastly, the lack of a benchmark dataset and how the data is collected is also a highly relevant and debatable topic. This will be further discussed in Section 3.2.

As the amount of research increase, the number of different approaches also increase. Fortuna (2018) made an overview of the different existing datasets and annotation procedures, and Schmidt and Wiegand (2017) also made an overview and introduction to the hate speech field. The first discussion point is whether hate speech classification should differentiate only between hate and non-hate (binary) or multiple classes. As binary classification is considered more straightforward, method-wise, this is previously the most commonly used approach (Malmasi and Zampieri, 2017). As noted by Dinakar et al. (2012), models trained on such data often rely on the frequency of offensive or profane words to distinguish between the classes. Therefore, it is crucial to discriminate between hate speech and profane words. As a result, classifying multiple labels has recently become more widespread through either multi-label classification (Malmasi and Zampieri, 2017; Sharma et al., 2018) or multi-step classification (Park and Fung, 2017). H. Liu et al. (2019) looked into multi-step learning because classifying into a single class would not be optimal for the real world. They proposed a three-level framework which first did a polarity classification, determining whether something is hate or not hate, followed by a multi-task classification for the identification of the different types of hate speech and then in the last stage it detected topics and context of hate speech.

3.2. Existing data collections

Despite the proliferation of hate speech as a research field, one commonly accepted corpus does not exist yet. Because of this, authors usually have to collect and label their own data. The datasets are often constructed specially for the domain. Since the datasets have been constructed for different purposes, they may display different sub-types of hate speech and have unique characteristics. As an example, the data collected at a white supremacy forum will differ from the data collected at more general sites such as Twitter, due to amongst other the difference in demographic. Because of the lack of a benchmark dataset, a lot of the studies conducted use a variety of different annotations and data, making it harder to compare methods and results. In addition, creating datasets is very time-consuming because the number of hateful statements is much lower than for neutral statements. Thus, to get a sufficient number of hate speech instances, a larger number of comments has to been annotated. Also, a lot of the datasets has not been made publicly available. One reason can be that due to the offensive and profanity language of the data, the authors do not want the content to be publicly available.

Even though there is no benchmark dataset there exist some that are widely used in recent papers. One of these is from Waseem and Hovy (2016). It was made publicly available on GitHub with approximately 16k messages from Twitter which were labelled as racism, sexism or neither. The tweets were collected through a manual search of common hateful terms and hashtags related to religion, sex, gender and ethnic minorities. The dataset is quite small and also contain a significant proportion of neutral tweets, which makes it unbalanced. The authors state in their paper that this is intentional for a better real-world representation. At the time being, many tweets are no longer available due to users being deleted or blocked from Twitter.

Davidson et al. (2017) proposed a dataset with data collected from Twitter using the hate speech lexicon from *Hatebase.org*. They did labelling through employing CrowdFlower workers who did manual labelling on each tweet into one of three categories: hate speech, offensive but not hate speech or neither. The workers were given definitions of hate speech and told to consider the context of the tweet. The authors concluded that specific lexical methods are effective to identify the offensive language, but not as accurate when identifying hate speech; just a small percentage of the flagged Hatebase lexicon was considered hate speech by humans. Their analysis showed that a hateful term could both help and hinder accurate classification, and their study pointed out that some terms are especially useful for distinguishing between offensive language and hate speech. Nonetheless, if a text does not contain any offensive terms or curse words, we are most likely to misclassify hate speech.

Another publicly available dataset[1] from Kaggle contains around 150k Tweets where 16k are toxic. The dataset separates between six different types of hateful; toxic, severe toxic, obscene, threat, insult and identity hate. Sharma et al. (2018) created a new dataset of tweets based on ontological classes and degrees of harmful speech, with the granularity they claim other publicly available datasets to be missing. They also take into consideration the degree of harmful content, the intent of the speaker and how this affects people on social media when labelling the data. Z. Zhang et al. (2018) also created a dataset which extended the at time currently available datasets which consist of Waseem and Hovy (2016) and Davidson et al. (2017). More recently, Founta et al. (2018) published a dataset containing 100k English tweets with cross-validated labels. This is more useful

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

for deep learning models since they require larger amounts of data. The authors proposed a methodology for annotating large-scale datasets and used CrowdFlower workers for the labelling process. The dataset is more balanced compared to earlier datasets, with roughly half of the samples labelled as "Normal" and the rest as either "Offensive", "Spam" or "Hateful".

Gröndahl et al. (2018) tried to reproduce five state of the art hate speech models using datasets from Waseem and Hovy (2016), Wulczyn et al. (2017) and Z. Zhang et al. (2018). Their results show that the models only perform well when tested on the same type of data they were trained on. Thus, this underpins what Davidson et al. (2017) and Waseem (2016) emphasized; the lack of a definition of hate speech result in a difference in the annotation of the dataset, which leads to models predicting offensive speech as being hateful. However, if the models are retrained with the training set from another dataset and then tested using the test set from the same dataset, all models perform equally well. Thus, they are largely independent of the model architecture.

Lastly, several national and international hate speech workshops have recently proposed datasets in their respective languages for their workshops and competitions. Zampieri et al. (2019) presented OLID, the Offensive Language Identification Dataset, which contains over 14k English tweets. Thus, this indicates that there is still not one commonly accepted corpus.

3.3. Anomaly detection

As previously mentioned, Gröndahl et al. (2018) compared five state-of-the-art hate speech models with three well-known datasets and found that all of the models had poor performance when they were trained on one dataset and tested against another. Therefore, they suggest to re-phrase the problem. Hate speech detection has previously only been referred to as a classification problem, but should instead be addressed as detection. Hence, they suggest reconceptualising hate speech detection as anomaly detection, where hate speech constitutes an anomalous variant of ordinary speech. To the best of our knowledge, there are currently no existing methods that have experimented with the suggestion made by Gröndahl et al. (2018) to utilise anomaly detection to solve the problem of detecting hateful utterances.

Even though anomaly detection has not been used in the field of hate speech detection, it has been well-studied within diverse research areas and application domains. This includes areas such as fraud detection, industry damage detection, image processing, video network surveillance, and intrusion detection (Han et al., 2012). Most of the related work treats anomaly detection as an unsupervised learning problem. Typical anomaly detection methods assume that most of the data samples are normal and attempts to learn a "compressed" representation of this data. Moya et al. (1993) implemented a neural network one-class classifier for target recognition, while Schölkopf et al. (2001) and Tax and Duin (2004) implemented one-class SVMs for detecting novel data or outliers.

The methods aim at finding a subset of the data which contains the normal instances. Data samples that do not fall into this set are deemed anomalous. Furthermore, Kim and Scott (2012) and Vandermeulen and Scott (2013) use Kernel Density Estimation and F. T. Liu et al. (2008) use Isolation Forest to deal with the anomaly detection problem. A drawback to these shallow unsupervised anomaly detection methods is that they often require manual feature engineering to be effective on high-dimensional data. This process is time-consuming, which entails that they are limited in their scalability to large datasets.

In recent years, there has been an increasing interest in deep anomaly detection algorithms (Chalapathy and Chawla, 2019), a line of research which has already shown promising results. These approaches are motivated by the limited scalability of shallow AD techniques, and the need for methods that can handle large and complex datasets. Furthermore, the deep anomaly detection methods aim at overcoming the need for manual feature engineering by being able to learn the relevant features from the data automatically. The methods have been applied to a diverse set of tasks like video surveillance, image analysis, health care and cyber-intrusion detection. There have been proposed several novel deep approaches to unsupervised anomaly detection, including the work done by Abati et al. (2019) that designed an unsupervised deep autoencoder that learns the underlying probability distribution through an autoregressive procedure. Furthermore, Erfani et al. (2016) presented a hybrid model where an unsupervised Deep Belief Network is trained to extract features, and a one-class SVM is trained from these features. Other approaches include the works by Hendrycks et al. (2018), Ruff et al. (2019) and Pang et al. (2019).

All the methods previously mentioned are relying on unsupervised learning. On the other hand, semi-supervised learning utilises some labelled data samples in addition to unlabelled data. Many real-world applications have access to a small portion of data that might, for example, be labelled by a domain expert, and this knowledge is not exploited in the unsupervised setting. According to Ruff et al. (2020), the term *semi-supervised anomaly detection* has been used to describe two different settings for anomaly detection. These settings include only adding labelled normal data and adding both labelled normal data and anomalies. Most of the existing work adopts the first setting, i.e. only incorporates labelled normal data. Shallow approaches that adopt this setting include the work done by Blanchard et al. (2010) that created a semi-supervised method for novelty detection, assuming that only labelled examples of the normal class was available. They argue that the problem could be solved by reducing it to a Neyman-Pearson (NP) classification, which is the problem of binary classification subject to a constraint on the false positive rate. One deep approach is developed by Akcay et al. (2018) by using a conditional generative adversarial network by employing encoder-decoder-encoder sub-networks. There are a few authors that have investigated the second setting, where one utilises labelled anomalies in addition to the labelled normal data. This includes the works conducted by Görnitz et al. (2013) and Ergen et al. (2017) among others.

Ruff et al. (2020) introduce a deep end-to-end method for general semi-supervised anomaly detection using an information-theoretic perspective. It involves deriving a loss motivated by the idea that the entropy for the latent distribution of normal data should be lower than the entropy of the anomalous distribution. Generally, semi-supervised approaches to anomaly detection aim at utilising labelled samples, but most proposed methods are limited to merely including labelled normal samples. This method also takes advantage of labelled anomalies. They have conducted extensive experiments with three widely used datasets containing images, along with other anomaly detection benchmark datasets (where none contains text data). They argue that their method outperforms shallow, hybrid, and deep competitors, yielding increased performance even when provided with only a little labelled data.

There are a limited amount of works that address anomaly detection on text data. L. M. Manevitz et al. (2001) study one-class classification of documents using OC-SVM, where their model is based on identifying "outlier" data as representative of the second-class. L. Manevitz and Yousef (2007) later experimented with a simple autoencoder (feed-forward network) on text, where they developed a filter to examine a corpus of documents and choose those of interest. They did this by only using positive information, i.e. normal data, training on the Reuters-21578 data collection (Lewis et al., 2004).[2] Steyn and De Waal (2016) constructed a Multinomial Naïve Bayes classifier and enhanced it with an augmented Expectation-Maximization (EM) algorithm in an attempt to simplify the problem of textual anomaly detection. Kannan et al. (2017) use block coordinate descent optimisation to create a matrix factorisation method for anomaly detection on text, which they claim has significant advantages over traditional methods. Gorokhov et al. (2017) implemented a convolutional neural network for unsupervised learning with an RBF activation function and logarithmic loss that was tested on the Enron Email dataset.[3]

Recent work has found that proper text representation is crucial for designing well-performing machine learning algorithms. Several existing methods within the field of hate speech detection and text classification, in general, utilises word embeddings. This will be further discussed in Section 3.4. However, existing methods for anomaly detection often rely on bag-of-words (BoW) to represent text, such as the works by L. M. Manevitz et al. (2001), L. Manevitz and Yousef (2007), Kannan et al. (2017) and Mahapatra et al. (2012). Neither of these methods makes use of unsupervised pre-trained word models, like word embeddings. Ruff et al. (2019) is currently the only text-specific method for anomaly detection that utilises pre-trained models for distributed vector representations of words. They introduce a one-class classification method which uses unsupervised learning, builds upon word embedding models and learn multiple sentence representations via self-attention. These sentence representations capture multiple semantic contexts, which enables the performance of contextual anomaly detection concerning the multiple themes and concepts present in the unlabelled text corpus. The datasets they experimented

[2]Available at: http://www.daviddlewis.com/resources/testcollections/reuters21578/
[3]https://www.kaggle.com/wcukierski/enron-email-dataset

with was the *Reuters-21578*, *20 news-groups* and *IMDB Movie Reviews*. They tested both GloVe and FastText embedding and BERT language model and found that the improvements using BERT on these datasets were insufficient and did not justify the increased computational cost.

3.4. Features in hate speech detection

Feature extraction aims to transform input data into a new dataset by creating new features. Schmidt and Wiegand (2017) did a summary of important features used within hate speech detection, which includes a wide range of features. In this section follows a presentation of the state of the art within feature extraction.

Simple surface features are features that can be derived without advanced methods. Bag of words (BoW), word and character n-grams are popular methods used to find the presence and frequency of words in a document. URL mentions, hashtags, punctuation, word and document lengths and capitalisation are also used widely in hate speech classification by authors such as Burnap and Williams (2015), Waseem and Hovy (2016) and Nobata et al. (2016). Waseem and Hovy (2016) explored which features that are most prominent when detecting hate speech and found that character n-grams contribute the most to the result. Furthermore, Mehdad and Tetreault (2016) concluded that character n-grams outperform word n-grams and other methods. Character n-grams are superior within hate speech detection due to the ever-evolving language on social media. Users learn blacklisted words and can thus avoid them by using slurs and disguising the language in other manners. Using character n-grams is also more efficient to catch spelling mistakes.

Several *lexical resources* can be found on the web. Burnap and Williams (2015) created a word list containing specific negative words such as insults and slurs. Dennis Gitari et al. (2015) built a list of hate verbs and more recently *hatebase.org*[4] has been widely used. Davidson et al. (2017) showed that when detecting hate speech, one cannot rely completely on only these word lists. Most approaches today use it in addition to other features, such as simple surface features and word embeddings.

Linguistic features or syntactic features utilise syntactic information in the language such as dependency relationships and part-of-speech (POS) which are employed in the feature set of Dennis Gitari et al. (2015), Burnap and Williams (2015), Van Hee et al. (2015) and Z. Zhang et al. (2018). Nobata et al. (2016) looked into several different features, such as surface features and linguistic features. They found that character n-grams perform very well alone, but in combination with other features, their method became even more powerful. By adding these methods, one can capture long-range dependencies between words which n-grams may struggle to do. Burnap and Williams (2015) found that using typed dependencies, a representation of a syntactic grammatical relationship in a sentence, reduced the false negatives by 7% over the baseline BoW. This is useful

[4]https://hatebase.org/

for differentiating between utterances such as "send de hjem" and "la de være". Here, the POS pattern is the same, but the first utterance is more frequent in hateful comments. *Sentiment analysis* is the degree of polarity expressed in a message. A popular feature used is the presence of positive or negative words. Thus, hate speech and sentiment are often affiliated; often negative sentiment belongs to a hateful utterance, and several approaches such as Dennis Gitari et al. (2015) and Van Hee et al. (2015) look into this.

With hate speech detection, one might encounter the problem of data sparsity and high dimensionality. *Word generalisation*, which generally consists of word clustering and word embeddings, have been used to face these problems. With word clustering, induced cluster IDs representing a set of words are used as additional generalised features. Algorithms such as *Brown Clustering* (Brown et al., 1992), assigning each individual word to one particular cluster, and *Latent Dirichlet Allocation* (LDA) (Blei et al., 2003), topic distribution for each word were used as features in Warner and Hirschberg (2012), Malmasi and Zampieri (2017) and Zhong et al. (2016). However, more recently, word embeddings, distributed word representations based on neural networks, have been proposed for similar purposes. Word embeddings can be useful in hate speech detection since semantically similar words such as "dog" and "cat" may end up having a more similar vector than "dog" and "boat". There exist several popular word embedding models including Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017). Word embedding has also been quite popular in recent research within hate speech. Nobata et al. (2016), Park and Fung (2017) and Gambäck and Sikdar (2017) used Word2vec, Badjatiya et al. (2017) used GloVe and fastText and Pavlopoulos et al. (2017) used both GloVe and Word2vec. Djuric et al. (2015) showed that Word2vec outperformed the current state of the art model when compared to using BoW with a logistic regression classifier, addressing the aforementioned issues of data sparsity and high dimensionality. Furthermore, recently Devlin et al. (2018) presented a new word embedding model called BERT, Bidirectional Transformers for Language Understanding. BERT is a bidirectional unsupervised language representation, meaning that it can represent a word as having different meanings. For example, the word "bank" would have different representation, whether it is used in the word "bank deposit" or "riverbank". Their model outperformed the current state of the art, and this model can further improve many different types of NLP tasks. In addition to BERT, ElMo (Peters et al., 2018) and ULMFiT (Howard and Ruder, 2018) are some popular transfer learning models used in hate speech detection.

Meta information is information about the context such as user characteristics or whether the user has a high frequency of certain negative words in their user history. Waseem and Hovy (2016), Pitsilis et al. (2018) and Unsvåg (2018) all look into this. Today's social media consist of both images, videos and audio content. This content can also be hateful, and some studies try to use this *multi-modal information* as a predictive feature. Hosseinmardi et al. (2015) and Zhong et al. (2016) employ features based on images. World knowledge is used in *Knowledge-based features* to better get the context of a sentence. It requires a lot of manual coding, and therefore, to the best of our knowledge, only Dinakar et al. (2012) use a knowledge base in their work. In many papers, several different features are used and tested to gain the best result. This includes Nobata et al. (2016), Z. Zhang et al. (2018), Badjatiya et al. (2017), Davidson et al. (2017) and Waseem (2016). However, it is often difficult to select useful features since existing supervised models heavily rely on carefully engineered features. Robinson et al. (2018) conducted an extensive feature selection analysis, looking at surface features, linguistic features and sentiment features. They concluded that automatic feature selection could reduce the carefully engineered features by over 90%. The authors were able to select a small set of the most predictive features which achieves much better results than models using carefully engineered features. In addition, to the best of our knowledge, the general trend favours employing n-grams and different types of word embeddings, where transfer learning models have been the newest addition to the current state of the art.

The features commonly used in hate speech detection, as presented in this section, are summarised in Table 3.1.

Table 3.1.: Features in hate speech detection

Feature type	**Description**
Simple surface features	Features that can be derived without advanced methods
Bag of Words (BoW)	It is a representation of text that describes the occurrence of words within a document
N-grams	N-gram assigns probabilities to word and character sequences. It is a sequence of *n* words or characters
Linguistic features	Utilise syntactic information in the language such as dependency relationships and part-of-speech (POS) tagging
Typed dependencies	A representation of a syntactic grammatical relationship in a sentence
Sentiment analysis	The degree of polarity expressed in a message
Word generalisation	Generally consists of word clustering and word embeddings
Word clustering	Cluster ids is representing a set of words used as additional generalised features
Word embeddings	Distributed word representations based on neural networks, such as Word2Vec, GloVe and fastText
Transfer learners	Language models that represent a word as having different meanings, such as BERT and ElMo
Meta information	Information about the context, such as user characteristics
Knowledge-based features	World knowledge that is used to better understand the context of a sentence

3.5. Classification methods

Existing work within the field of hate speech detection can be divided into two categories; classic methods and deep learning methods. Previously, classical methods were mainly used, but more recently, neural networks and deep learning methods tend to outperform these methods, as stated by Schmidt and Wiegand (2017). This section presents the current state of the art within the field of classification methods, including classic methods and deep learning.

Classic methods include support vector machines (SVM), naïve Bayes (NB), logistic regression (LR), gradient boosted decision trees (GBT), random forest (RF) and general NLP. Despite deep learning being the most popular research area nowadays, some classical methods still compete with certain methods, and they are often used as a baseline for deep learning methods.

As stated by Schmidt and Wiegand (2017), LR and SVM have been the most popular choices among the classic methods within hate speech. Waseem and Hovy (2016) used LR in combination with extra-linguistic features and character n-grams to detect hate speech, while Gaydhani et al. (2018) proposed an approach which combined n-gram and TF-IDF with NB, LR and SVM. LR outperformed the previous state of the art methods at that time. However, these results have shown difficult to reproduce, and it turned out that 74% of the test data were either already in the training data or a duplicate (Isaksen, 2019). Davidson et al. (2017) first used LR to reduce the data dimensionality before testing a variety of models such as NB, decision trees, RF, and linear SVMs. They found that LR and linear SVMs performed significantly better than the other models. Even though they managed to get a high F_1-score on the best performing model, this model still misclassified almost 40% of the hate speech instances in the dataset.Lee et al. (2018) used the Founta et al. (2018) dataset to create the first baseline model using the most frequently studied classic machine learning and deep learning methods. They use BoW, TF-IDF and n-grams as features and they experimented with NB, LR, SVM, RF and GBT. For the neural network-based models, they used CNN and RNN. For their variant models, they use a pre-trained GloVe word embedding model. The models performed more or less equal, with an F_1-score ranging from 0.73 to 0.805, where RNN performed the best.

It is quite common to combine different machine learning classifiers. Burnap and Williams (2015) used Bayesian LR, GBT, SVM and an ensemble method to detect cyber hate speech. MacAvaney et al. (2019) proposed a multi-view SVM (combining multiple SVMs) approach that achieved near state-of-the-art performance while being simpler and producing easier interpretable decisions than neural methods. Malmasi and Zampieri (2017) and Robinson et al. (2018) looked into using SVM in combination with typical NLP features and achieved acceptable results, while Nobata et al. (2016) and Sharma et al. (2018) used a supervised model combining various surface and linguistic features.

As previously mentioned, supervised machine learning methods are often used as a baseline when testing deep learning methods. Fagni et al. (2019), Z. Zhang et al. (2018) and Badjatiya et al. (2017) used SVM, NB and LR as a baseline in their deep learning approach. Even though at present, classical methods are mostly outperformed by deep learning methods, some recent papers find SVM and LR models outperforming current deep learning architectures. Biesek (2019) tested SVM with TF-IDF vectors, bidirectional gated recurrent unit (GRU) and a contextual string embeddings model and found SVM outperforming the other methods. The reason probably being that it was a small and unbalanced dataset which can make more complicated models overfit. Classical methods are often chosen and preferred due to simplicity and when there is a smaller amount of data.

In most recent papers, deep learning with convolutional neural network (CNN), recurrent neural network (RNN), Long Short-Term Memory (LSTM) and a combination of methods have been the preferred approach. Both Zampieri et al. (2019), Badjatiya et al. (2017), Z. Zhang et al. (2018) and Fagni et al. (2019) showed that deep learning methods outperforms the more classical machine learning methods. Gambäck and Sikdar (2017) proposed to use CNN to classify hate speech by training four different CNN models using character n-grams, word vectors, and a combination of those. Park and Fung (2017) did one-step and two-step classification of abusive language. First detecting whether a tweet was hateful or not, and afterwards using a different classifier to detect whether it was racist or sexist. They, as well as Gambäck and Sikdar (2017), used a HybridCNN model, which is a variant of CNN that uses both words and characters to classify. For the one-step method, their proposed HybridCNN performs the best and for the two-step approach combining two LR classifiers performs as well as the HybridCNN in one step. This was surprising considering LR used fewer features than the HybridCNN. In addition, using HybridCNN for the first step and LR for the second step worked better than just using HybridCNN. Z. Zhang et al. (2018) introduce a new method based on deep neural network combining CNN and GRU, where word embeddings are first fed into CNN, which produces input vectors for the GRU. They evaluated it against several baselines and state of the art methods on a large collection of publicly available datasets. The results outperformed the baseline methods on six out of seven datasets.

Mehdad and Tetreault (2016) experimented with RNN and outperformed previous state-of-the-art methods. Pavlopoulos et al. (2017) used RNN in combination with GRU to further improve the previous state of the art approaches. It outperformed state of the art, which used LR or ML with features such as character n-grams or word n-grams. They also beat a standard CNN using word embeddings. Badjatiya et al. (2017) experimented with a supervised learning model based on deep learning architectures. They experimented with multiple classifiers such as LR, RF, SVMs, fastText, CNNs and LSTM. The best result was achieved using the LSTM model, assisted by gradient boosted decision trees and the features extracted by character n-grams. Their methods outperform previously considered state of the art methods such as character and word n-grams methods. However, other researchers have failed to reproduce their best-performing experiments, stating that their cross-evaluation method had a bug which increased the final F_1-score for each iteration

(Fortuna et al., 2019). However, Gröndahl et al. (2018) showed that they still achieved competing results without these implementations. De Gibert et al. (2018) tried different types of machine learning techniques such as SVM, CNN and LSTM. The LSTM based classifier obtained better results, but the SVM model still achieved decent results. Founta et al. (2019) experimented with different types of RNN architecture; GRUs, LSTMs and bidirectional RNNs. They found that the simple GRUs performed as good as more complex units.

In the aforementioned methods, they have separated the use of RNN and CNN, but as Z. Zhang et al. (2018) states; In theory, combining them should show to be more powerful than just solely based on either. In hate speech, CNN is useful to extract word or character combinations (word embeddings), and RNN will learn word or character dependencies (order information). They hypothesise that a combined structure can be more effective as it may capture co-occurring word n-grams as useful patterns for classification.

Pitsilis et al. (2018) proposed a detection scheme that is an ensemble of RNN classifiers, and it also incorporated various user-features. Their solution achieved a higher classification quality than the current state-of-the-art algorithms. One of these algorithms included Badjatiya et al. (2017). Furthermore, Fagni et al. (2019) looked into six different machine learning classification strategies, three classic and three deep learning. They compared SVM, NB and LR to CNN, GRU and ensemble and concluded that the best classification results were obtained through deep learning techniques, and ensemble in particular. H. Liu et al. (2019) proposed a fuzzy multi-step method to classify, and it was compared to SVM, CNN and LSTM beating the current state of the art. Meyer and Gambäck (2019) proposed an optimised architecture to detect hate speech by combining CNN and LSTM networks, utilising both character n-grams and word embeddings to produce the final classification. They used the Waseem and Hovy (2016) dataset and they outperformed all previous state-of-the-art approaches with an F_1-score of 0.792.

Schmidt and Wiegand (2017) stated that there does not exist comparative studies which would allow making a judgement on the most effective learning method. However, there exist several studies that compare the performance of different classification methods. Burnap and Williams (2015) did a comparative study which concluded that an ensemble method seemed most promising. Still, considering that this method could only work well for the exact dataset and features, it does not prove that it is the ideal approach for every hate speech problem.

As hate speech detection is experiencing an increase in popularity, SemEval 2019, a yearly international workshop on semantic evaluation, had two tasks regarding hate speech detection. Zampieri et al. (2019) presented the results and the main findings where the task was to identify and categorise offensive language in social media, with three sub-tasks. One hundred and four teams submitted a result for the first sub-task where the goal was to discriminate between offensive and non-offensive posts. The most popular models involved deep learning approaches, and within these, there was a variation of models used, but ensemble being the most popular. However, among the top 10 teams, seven used a variant of BERT (Devlin et al., 2018), the top non-BERT model used an ensemble of CNN with BiLSTM and BiGRU and got ranked sixth.

3.6. Hate speech detection for non-English languages

Since this thesis is concerned with the creation of a method for hate speech detection that is language-independent, it is relevant to look into existing approaches to hate speech detection for other non-English languages.

Van Hee et al. (2015) annotated their own Dutch dataset from *Ask.fm* consisting of approximately 86k posts. They used SVM in combination with BoW, word and character n-grams and semantic features when trying to detect Dutch cyberbullying. They achieved an F_1-score of 0.554, which was in line with the state-of-the-art approaches for automatic cyberbullying detection at that time. A Finnish group of data scientist from the Finnish company Futurice[5] looked into NB, RF and SVM in combination with both BoW and fastText tested on Finnish hate speech data from Facebook discussions. They achieved quite good results, where SVM with BoW outperformed the other methods. Vigna et al. (2017) reported performance for a simple LSTM classifier not better than an ordinary SVM, where SVM beat the simple LSTM on some occasions. Jaki and De Smedt (2018) trained a single-layer averaged Perceptron algorithm (Collins, 2002) in combination with character n-grams on a German right-wing hate speech dataset consisting of 100k instances. They achieved good results but state that the reliability of the model can be improved by more recent techniques such as deep learning systems with word embeddings.

Sigurbergsson and Derczynski (2019) were the first to detect hate speech for Danish and are to the best of our knowledge the only ones who have done it for Scandinavian languages. They constructed a Danish dataset containing user-generated comments from Reddit and Facebook. They split the classification into three sub-tasks in order to capture different types of offensive language. Sub-task A consists of classifying each post as either offensive or not. Sub-task B was to do an automatic categorization of the offensive language types, and for sub-task C the goal was to classify the target of the offensive language. Four automatic classification systems consisting of logistic regression and different types of BiLSTM were designed for both English and Danish. They used surface, linguistic and semantic features, as well as different types of word representations. The best performing system for Danish for sub-task A achieves an F_1-score of 0.699 with logistic regression.

Several national semantic evaluation workshops for hate speech have lately taken place, which has focused on their respective languages. Wiegand et al. (2018), Bosco et al. (2018) and Ogrodniczuk and Kobyliński (2019) consists of various research papers in German, Italian and Polish. In general, many state-of-the-art approaches were explored as well as

[5]https://www.futurice.com/blog/hate-speech-detection

different word embedding methods, with transfer learning models also rising in popularity. In addition, SemEval's 2019 (Basile et al., 2019) task 5 consisted of multilingual detection of hate speech against immigrants and women on Twitter and focused on Spanish and English messages. The first sub-task consisted of a binary classification task where the system had to predict whether a tweet in English or Spanish contained hate speech. SVM with TF-IDF vectors was used as a baseline for all the models. The best model used SVM in combination with ElMo and achieved an F_1-score of 0.651 (Indurthi et al., 2019). The other top performers used different combinations of neural networks, in particular, CNN and LSTM models with a variation of fastText and BERT models.

Alfina et al. (2017) created a dataset in Indonesian that covers hate speech in general, including hatred for religion, race, ethnicity, and gender. In addition, they conducted a preliminary study using different machine learning models such as Naïve Bayes, Support Vector Machine, Bayesian Logistic Regression, and Random Forest. Features they extracted were word n-grams, character n-grams and negative sentiment. They achieved an F-measure of 93.5% using word n-gram feature with Random Forest algorithm and found that word n-grams outperformed character n-grams.

3.7. Summary

As the above discussion illustrates, many methods have been proposed for detecting hate speech and offensive utterances, but these existing methods have issues that need to be addressed. Table 3.2 summarises the approaches directly related to hate speech detection, which was discussed in this chapter. As mentioned, there exists a large amount of research related to the detection of hateful expressions, and this chapter provided a comprehensive description of a selection of these. However, there exists much more research not included here. The research related to anomaly detection is not included in the table since this research is not directly related to hate speech detection.

[6]https://github.com/clips/pattern
[7]https://github.com/zalandoresearch/flair

Work	**Description**	**Resources used**	**Classification**	**Dataset**
Burnap and Williams (2015)	Classification and statistical modeling for detecting cyber hate speech	Simple surface features i.e. BoW and character n-grams	Bayesian LR, RFDT, SVM and ensemble	Own Twitter dataset
Waseem and Hovy (2016)	Presents a publicly available hate speech corpus	N-grams and text features	LR	Own Twitter dataset
Nobata et al. (2016)	Develop a hate speech corpus and a ML method to detect hate speech	Several different NLP features, including lexicon, n-grams, word2vec and syntactic	Skip-bigram and distributed memory model	Own Yahoo dataset (Djuric et al., 2015)
Dennis Gitari et al. (2015)	Generate a lexicon of sentiment expressions and use this to create a classifier	Sentiment analysis	SVM and Maximum Entropy	Own website dataset
Davidson et al. (2017)	Hate speech detection using classic ML methods	Surface features, linguistic features and sentiment features	LR, NB, DT, RF and SVM	Own Twitter dataset
Hossein-mardi et al. (2015)	Detect cyberbullying over images in Instagram	N-grams, image and text features	NB and SVM	Own Instagram dataset
Zhong et al. (2016)	Detect cyberbullying from Instagram images	N-grams, BoW, Word2Vec, image and text features	SVM and CNN	Own Instagram dataset
Fagni et al. (2019)	Hate speech detection with classic and deep learning	Text features, BoW, word2vec	LR, NB, SVM and CNN, GRU and Ensemble	Own Twitter dataset
Gambäck and Sikdar (2017)	Deep learning hate speech classification system	Character n-grams, one-hot encoding and word2vec	CNN	Waseem and Hovy (2016)
Founta et al. (2019)	Unified deep learning architecture for abuse detection	TF-IDF BoW and GloVe	NB as baseline, RNN, Bi-RNN, GRU and LSTM	Chatzakou et al. (2017), Waseem and Hovy (2016), Davidson et al. (2017), Rajadesingan et al. (2015)

Table 3.2.: Overview of related approaches

Work	Description	Resources used	Classification	Dataset
Mehdad and Tetreault (2016)	Analysis of word and character n-grams in abusive language detection	Word and character n-grams, BoW, Word2vec, image and text features	Distributional Representation of Comments (C2V), SVM, NB and RNN	Nobata et al. (2016)
Park and Fung (2017)	One-step and Two-step classification for abusive language	Word and character n-grams, one-hot encoding and word2vec	HybridCNN and LR	Waseem and Hovy (2016)
Z. Zhang et al. (2018)	Detecting hate speech through combining CNN and GRU based DNN	Surface, linguistic, sentiment and enhanced features, word2vec	CNN+GRU and SVM	Waseem and Hovy (2016), Davidson et al. (2017) and own Twitter dataset
Pavlopoulos et al. (2017)	Deep learning for user comment moderation	GloVe and word2vec	RNN+GRU, wordlist and CNN	Own Gazzetta dataset, Wulczyn et al. (2017)
Badjatiya et al. (2017)	Hate speech detection using different methods within deep learning	Character n-gram, TF-IDF, BoW and GloVe and semantic embeddings	LR, RF, GBDT, SVM, FastText, CNN and LSTM	Waseem and Hovy (2016)
Pitsilis et al. (2018)	Ensemble of RNN classifiers to detect offensive language focusing on the users' behaviour	Word-based frequency vectorization	RNN and LSTM	Waseem and Hovy (2016)
De Gibert et al. (2018)	Hate speech detection on white supremacy forum	BoW and randomly initialized word embeddings	SVM, CNN and LSTM	Own Stormfront dataset
Malmasi and Zampieri (2017)	Detect hate speech in social media and distinguish from profanity	Surface features, word skip-gram, Brown clusters	SVM	Davidson et al. (2017)
MacAvaney et al. (2019)	Challenges and solutions within hate speech detection	TF-IDF	BERT as baseline, multi-view SVM	De Gibert et al. (2018), Kumar et al. (2018), Davidson et al. (2017)

Work	**Description**	**Resources used**	**Classification**	**Dataset**
H. Liu et al. (2019)	Fuzzy multi-step method for detecting hate speech	DBoW and word2vec	SVM, DT, GBT and DNN as baseline, Mixed Fuzzy Rule Formation (Berthold, 2003)	Own Twitter dataset
H. Zhang et al. (2019)	Identify and categorize offensive language in social media	GloVe	classic methods baseline, CNN, Bi-LSTM, Bi-GRU	OLID dataset Zampieri et al. (2019)
Sharma et al. (2018)	Degree based classification of harmful speech in social media	TF-IDF and BoW	SVM, NB and RF	Own Twitter dataset
Sigurbergsson and Derczynski (2019)	Offensive language and hate speech detection in Danish	Surface, linguistic and sentiment features, word representations	Logistic regression and BiLSTM	Own Danish dataset from Reddit and Facebook
Meyer and Gambäck (2019)	Platform agnostic hate speech detection	Character n-gram and word embeddings	CNN and LSTM	Waseem and Hovy (2016)
Vigna et al. (2017)	Hate speech detection on Facebook for Italian	Lexical and sentiment features, POS, word2vec	SVM and LSTM	Own Italian Facebook dataset
Van Hee et al. (2015)	Dataset construction, classification and identification of cyberbullying	Pattern[6], surface and sentiment features, POS-tagging	SVM	Own Dutch Twitter dataset
Biesek (2019)	Automatic cyberbullying in Polish tweets	TF-IDF and FastText	SVM, bi-GRU, Flair[7] framework with Contextual Embeddings (Akbik et al., 2018)	Ogrodniczuk and Kobyliński (2019)
Jaki and De Smedt (2018)	Right-wing German hate speech detection	Character and word n-grams	Single-layer averaged Perceptron algorithm (Collins, 2002)	Own German Twitter dataset
Alfina et al. (2017)	Preliminary models for hate speech detection in Indonesian	Word & character n-grams and negative sentiment	NB, SVM, Bayesian LR and Random Forest	Own Indonesian Twitter dataset

4. Preparation of Data Collection

When dealing with the detection of hateful utterances using machine learning methods, the representation and amount of data are crucial for attaining adequate results. Furthermore, it is beneficial that the data is moderately representative of a real-world scenario. There is a requirement for a precisely labelled dataset in order to train the model, and the labelling determines the system performance. Hence, in order to create an effective system, it is essential that the data have appropriate labels and can be considered a generalisation of the model's input. Nevertheless, creating a labelled dataset can be a time-consuming and demanding task. Therefore, a significant part of this thesis involved the creation of a general dataset containing Norwegian Twitter messages (tweets) and social media comments from Facebook and *Resett*.[1] This dataset will hopefully function as a benchmark dataset for future research within the field of hate speech detection in the Norwegian language.

The creation of this dataset was collaborative work done by Tora Seim Gunstad, Marie Andreassen Svanes and I, and it is a comprehensive and improved dataset based on the work conducted in the specialisation project in the Fall of 2019 (Jensen et al., 2019). In the following sections, the process of collecting and annotating data are presented. In addition, the annotation procedure and guidelines are introduced, and the inter-annotator agreement is calculated and discussed.

4.1. Collecting data

The first step in the process of creating a Norwegian dataset was the collection of data. We decided that in order to obtain a generalised and high-quality dataset, it was necessary to collect data from several different sources. A significant drawback to the dataset created in Jensen et al. (2019) was that all the data came from one source; *Resett*, which was chosen because it is known for being rather xenophobic and hateful. However, it was discovered that the majority of the hateful comments were related to the immigration of Muslims in particular. As a result, all the classification models were not capable of detecting hate on other topics such as politics, feminism or just general cyberbullying. Therefore, it was decided to collect data from Twitter and Facebook, and also include a part of the data previously collected from Resett.

[1]https://www.resett.no/

This section describes the collection of data gathered from Twitter, Facebook and Resett, as well as what preprocessing steps were conducted before the dataset was annotated.

Twitter

In order to collect data from Twitter, it was necessary to utilise the Twitter API. This requires an application to Twitter, stating the purpose of the data collecting. An approved application provides a Twitter developer account that grants an authorisation token that needs to be used when calling an endpoint. Because of Twitter's restrictions, we created three developer accounts in order to gather enough data. To access the API, we used the build-in Python package `requests`, that allows for HTTP requests and collects the results as a json file. Requests include a query, which is a search word that the API uses to provide relevant tweets, and thus we had to create a list of search terms. We chose terms that most likely contained some hateful utterances, and tried to include terms with different topics. Examples of included terms are *parasitt*, *rasist* and *feminist*. The full list of search words can be found in Appendix C.1.

All the collected tweets are written in the period 03.02.2020 12:00 to 03.03.2020 12:00. Moreover, all the retweets were ignored, and the tweets were checked if they were written in Norwegian or not. The Twitter API allows for language specifications, but even though the Norwegian language is specified, not all retrieved tweets are in Norwegian. It was chosen to keep all tweets containing at least 15% Norwegian words because, in social media, many people add English words to their vocabulary when they are writing in Norwegian. It is reasonable to believe that this trend will not disappear in the future, and a system should be able to handle such behaviour. Furthermore, all comments were going to be manually annotated by at least one annotator, so if a comment included too many English words, it would be removed during the annotation process.

Facebook

The comments from Facebook were manually collected and retrieved using `fbcrawl`.[2] This is an advanced crawler for Facebook written in Python that utilises `Scrapy`,[3] an open-source framework for extracting data from websites. It is only possible to crawl specific posts or pages, and hence, we had to manually browse several Facebook pages in order to find all of the Facebook posts that we wanted to crawl. We ended up using 74 posts from 18 pages, including the pages of several Norwegian news agencies, such as *TV2* and *NRK Nyheter*. According to Veledar (2018), approximately 10% of the comments on these Facebook pages were hateful, and thus these pages was a good starting point. Furthermore, we included pages that most likely posts content that encourages discussion, such as "Norge fritt for Islam" and "Vi støtter Sylvi Listhaug, innvandrings- og integreringsminister". Posts from these pages were chosen based on the number of

[2]https://github.com/rugantio/fbcrawl
[3]https://scrapy.org/

comments on the post. The full list of included pages can be found in Appendix C.2. We only selected publicly available pages and assured that all names were removed to safeguard privacy.

Resett

The original dataset from Jensen et al. (2019) consisted of 14 620 comments from Resett. This data was collected by building a web crawler that could crawl a Disqus comment system.[4] The crawler used `Scrapy` and `Splash`[5], a headless browser designed for web scraping.

Mostly all comments from Twitter and Facebook were annotated before comments from Resett.no were included in the dataset. We observed a massive imbalance between neutral and hateful comments from the data collected from Facebook and Twitter. Therefore, it was chosen to include all comments labelled as either offensive or hateful from the Resett dataset in Jensen et al. (2019). In addition, neutral comments were added, so the total number of comments from Resett eventually became six thousand. It was chosen only to include six thousand comments because it was undesirable for the new dataset to include too many of these comments.

4.1.1. Preprocessing

The following preprocessing steps were conducted in order to create comments that were readable for a human annotator and to make all the comments anonymous.

Both the tweets and Facebook comments contain a lot of names and usernames. Since outside annotators were going to be included in the annotation process, and since the dataset are made publicly available, it was necessary to make all the comments anonymous. For the tweets, this involved replacing the usernames, e.g. @example_username, with the general user mentioning @USER. For Facebook, the process of removing names was more challenging. There is no mentioning symbol, such as the @ for a Twitter user when mentioning a person in a Facebook comment. Here, the mentioned person's full name appears anywhere in the comment. Hence, to replace all the names with NAVN (the Norwegian word for "name") we had to search through the comments and look for common Norwegian names. Two lists of common first and last names from *Statistisk Sentralbyrå (SSB)* was also used.[6] Replacing all of these names still left many names in the dataset. Hence, it was decided to scroll through the entire dataset and write down all names that were not already replaced. Merging the new list with the lists from SSB gave two new lists of first names and last names, which can be found in Appendix C.2.2. All of these names were replaced in the dataset.

[4]https://disqus.com/
[5]https://scrapinghub.com/splash
[6]https://www.ssb.no/navn

Several tweets also contained the phrase "via @example_user", which were also removed, in addition to the hashtag symbol (#). Furthermore, all URLs were removed from both of the Twitter and Facebook datasets, and all potential duplicates were removed. After analysing a part of the dataset, we found some tweets to be ads, tweets from magazines such as DN+ or tweets from police stations. Since these are not similar to a typical comment or tweet, these were removed from the dataset.

4.2. Annotation procedure

After the data was collected, it had to be manually annotated by at least one annotator. This annotation process is an essential part of text analytics. Although the annotators only work with a limited amount of the data instances, their results will have a large impact on the final classification results. Therefore, reliability of the annotators and the sufficiency of the defined labels are of significant importance in order to achieve satisfactory results. The definitions of the categories are especially important in the case of hate speech detection, due to the high chance of subjective interpretations.

For the dataset to fit the problem specifications of both theses, it was decided to categorise the data into five distinct categories, labelled 1 to 5. All comments that should be removed were marked with X. Comments with non-Norwegian language or comments consisting of only URLs are examples of this category. The definitions of the different categories are based on the definitions in Section 2.1, and can be found in Section 4.2.2. Tora, Marie and I, from now referred to as the annotators, annotated the first 2500 instances from the Twitter dataset, and several inter-annotator agreement metrics were calculated. This is presented in Section 4.2.3. The majority vote was used as the final label for these data instances, meaning that if two or more annotators agree on the category for a comment, then that label should be used. If all three annotators disagreed, the median was used. The rest of this section describes the procedure of involving user-based annotations and guidelines used for annotation, as well as the annotation agreement.

4.2.1. User-based annotation

In order to mitigate annotator bias in the dataset, it was decided to include more annotators, which from now will be referred to as *outside annotators*. Twenty people volunteered to categorise five hundred Facebook comments each. The majority of these volunteers were students at the Norwegian University of Science and Technology, studying either computer science or communication technology. The rest of the dataset, i.e. all the tweets, the comments from Resett and the remaining Facebook comments, were divided equally between the annotators. All three annotators annotated a large part of the dataset and distributed some comments to friends and family.

Waseem (2016) examined the influence of annotator knowledge of hate speech in classification results. They compared the classification results obtained from models which were trained on data labelled by expert and outside annotators and found that the outside annotators were more likely to classify offensive language as hate speech. Hence, systems that were trained on expert annotated data achieved better results, which highlights the problem of subjective interpretation of hate speech. Furthermore, as discussed in Section 3.1, both Ross et al. (2017) and Waseem and Hovy (2016) concluded that annotators should be given a more detailed definition of hate speech before annotation, in order to minimise the subjective perceptions of the content. Therefore, in order to provide an extensive and well-defined dataset containing Norwegian tweets and social media comments, it was necessary to provide clear guidelines with definitions and examples to all of our outside annotators. It was included several examples found in the first two thousand five hundred tweets for each category, in order to give a better explanation of the formal definitions.

Ross et al. (2017) show that regardless of providing annotators with a definition of hate speech, the annotators still failed to produce annotation of an acceptable level of reliability. Too much wrongfully annotated data would have a severe impact on our results, and therefore, we quality assured the annotated data. We checked what the outside annotators had categorised and changed everything we considered to be incorrectly annotated. This was particularly important for the more offensive and hateful classes, in order to create a clear line between offensive and hateful speech. The following section presents the guidelines provided to all outside annotators. These were the same guidelines we used when annotating the first two thousand five hundred comments, only differentiating in the inclusion of examples.

4.2.2. Guidelines

Both De Gibert et al. (2018) and Schmidt and Wiegand (2017) acknowledge that previous studies have not had a common annotation procedure for hate speech. Hence, we had to create our own guidelines which we based on the guidelines from De Gibert et al. (2018). The guidelines were created in order to ensure that all annotators had the same understanding of the different categories. An important part of the guidelines was to develop a line between offensive content and hateful content. It is desirable to develop a system that can successfully distinguish between these types of utterances and hence, detect the worst of the worst comments.

Since the goal of this thesis involves using anomaly detection to detect hate speech, it is only relevant to consider normal versus anomalous instances. Hence, it is undesirable to keep all the following five classes. As mentioned, the comments are categorised into five categories in order to fit both theses' problem statement. If only considering the absolute worst of the worst comments as anomalous speech, then all the rest of the categories will be considered normal instances. How the dataset is used during experiments is elaborated in Section 6.2.1.

The five different categories used to annotate the dataset are the following:

1 - Neutral: Everything not included in the other categories.
Examples of neutral comments are:

1. *Blir forbanna over at listhaug sier at spagetti er en norsk verdi eller noe sånt*
2. *Gud hjelpes for en gjeng*

2 - Provocative: An utterance is defined as provocative if it contains aggressive language used to express an opinion. This includes the use of profane words, patronizing language or ironic and sarcastic tone.
Examples of provocative comments are:

1. *Greta Thunberg er en BLØFF ! En stor løgn skapt kunstig av sosialister som blir betalt for å lyve!*
2. *Ehhh ... om du er religiøs er du hjernevasket. Islam er en eneste stor sekt, hva annet? Sekten holder 1,5+ milliarder mennesker nedtrykt i patriarkalske strukturer, underutvikling og uvitenhet, ekstrem fattigdom ... hva godt gjør d for menneskeheten å sitte å pugge koranvers?*

3 - Offensive: An utterance is defined as offensive when it contains hurtful, derogatory or obscene comments either directed towards an individual or a group based on their characteristics. The difference from the above classes is the degree of inciting actions and severity of the derogatory language.
Examples of offensive comments are:

1. *Vil bare hoppe inn her og si at du er ond og ræva, men først og fremst veldig dum, Someone20004. Dårlig fungerende hjerne! Og det må vi respektere, alle er ikke født med en ordentlig hjerne*
2. *Milan bare endra tissen min fra stiv til halvkram, litt som å pule en feit hore #fetehorer #æsj #fårdenikkeopp*

4 - Moderately hateful: Utterances which are partly or wholly motivated by hate or negative attitude towards groups or individuals based on ethnicity, religion, sexuality, gender or disabilities. The utterances do not call to action but still violates the integrity and disparages a group or individual's dignity.
Examples of moderately hateful comments are:

1. *Begriper ikke at noen kan slippe unna med et mordforsøk som dette var - eller terror.mange media er også redde for å skrive at d er innvandrere - er på en måte unødvendig da alle overfall utføres av disse jævlene... #islam #overfall #mordforsøk*
2. *Ja, de flytter hit og vi får verre liv = klankultur, misogyni, høye økte offentlige utgifter, press på skole og helsevesen, gjengproblematikk, æresrelatert vold, og aller verst - en retardert, avskyelig religion! Bruker vi din logikk bør vel alle pakistanere få komme inn og bli?*

5 - Hateful: Utterances which are partly or wholly motivated by hate or negative attitude towards groups or individuals based on ethnicity, religion, sexuality, gender or disabilities and which encourage violent actions. These utterances encourage a violation of integrity and a severe disparagement of a group or individual's dignity. Examples of hateful comments are:

1. *Alle som jobber og sliter og blir mishandlet og fornedriget av disse kriminelle monstrene er de samme som betaler regninga for deres liv i Europa. Send ALLE kriminelle tilbake til hjemlandet og FORBY ISLAM!*
2. *Send henne hjem til Somalia med hele familien!*

Furthermore, the guidelines included some additional information and tips that the annotator should consider while annotating. This includes information about comments that should be marked with X and freedom of speech. Freedom of speech is essential in Norway, and everything related to statements about politics has a higher threshold for being perceived as non-neutral in a debate. It is perfectly permissible to be strongly opposed to various political issues, but it is considered as non-neutral when it falls into the other categories mentioned above. The full guidelines provided to the annotators (in Norwegian) can be found in Appendix B.

4.2.3. Inter-annotator agreement

When several annotators annotate the same part of the dataset, it is a common practice to compare their level of agreement. The annotators used the guidelines provided in Section 4.2.2, in order to assure that the knowledge used to annotate were transferable. They worked individually and did not discuss individual data instances during the process. The goal was to determine if the annotations were reliable and hence, determine if our guidelines were adequate. There are several different metrics one can use to calculate the inter-annotator agreement, as described in Section 2.6.3.

Melzi et al. (2014) used Cohen's kappa to calculate inter-annotator agreement between annotators. They divided Spine-health forum data into six categories and used both Master's students and health professionals as annotators. They calculated kappa between the students to 0.26 and the agreement between the health professionals and the students to 0.46. Bermingham and Smeaton (2009) got on average 3.6 annotators to annotate 150 topics from 115 documents. The annotators used sentence-level annotation and categorised them into five classes. They evaluated the inter-annotator agreement using Krippendorff's alpha and achieved a score of 0.4219, which was considered to be moderate. Bobicev and Sokolova (2017) examine the inter-annotator agreement in multi-class, multi-label sentiment annotation of messages. They collected 65 discussions, each containing 10-20 posts. Three annotators annotated the posts using four distinct labels, where each post could get several labels. For evaluating their inter-annotator agreement, they calculated both percentage agreement, Cohen's kappa, Fleiss' kappa and Krippendorff's alpha.

Based on the related articles, we decided to calculate both the pairwise Cohen's kappa, Fleiss' kappa and Krippendorff's alpha, as well as percentage agreement for the 2500 comments that were annotated by all three annotators.

The actual annotation pattern was examined, where Table 4.1 shows the pairwise utterance coherence between annotators.

	1	2	3	4	5
1	2171	29	9	1	0
2	111	24	8	2	0
3	15	18	10	2	0
4	0	2	1	6	0
5	0	0	1	3	2

(a) A1 vs A2

	1	2	3	4	5
1	2254	37	6	0	0
2	38	19	16	0	0
3	8	4	14	3	0
4	0	2	3	6	3
5	0	0	0	2	4

(b) A2 vs A3

	1	2	3	4	5
1	2174	26	10	0	0
2	107	27	11	0	0
3	19	6	18	2	0
4	0	3	0	5	1
5	0	0	0	2	4

(c) A1 vs A3

Table 4.1.: Pairwise utterance coherence between the annotators. The numbers 1 to 5 is the different categories and A1, A2 and A3 represents the three annotators.

The diagonal shows all the instances where both annotators agree, and everything else is where they disagree. As can be seen from the table, all three annotators agree on most of the neutral comments (category 1), but there are some instances where the annotators disagree if a given comment is category 1 or 2. There are also a few instances where they disagree if it is category 1 or 3. Furthermore, it appears to be some disagreement about whether the hateful comments should be labelled as moderately hateful (4) or hateful (5). Because of these disagreements, it was also interesting to find the number of comments labelled in each category. Table 4.2 presents the number of comments that were labelled by at least one, two and all annotators.

Table 4.2.: The number of comments in each category labelled by at least one or two annotators and by all annotators.

Category	**At least one**	**At least two**	**All three**
Category 1	2358	2299	2149
Category 2	217	56	7
Category 3	78	28	7
Category 4	19	9	4
Category 5	7	4	2

From Table 4.2 it can be observed that seven comments were labelled as category 5 by at least one annotator. How the comments were labelled can be seen in Table 4.3. Again, from the table, it can be seen that all three annotators seem to disagree when annotating these data instances. Hence, this leads to the conclusion that it is difficult to decide if harassing comments are hateful or just moderately hateful.

In six out of seven of the cases presented in Table 4.3, at least two of the annotators

Table 4.3.: The comments where at least one annotator categorised a 5. For the sake of clarity, the comments are not presented in this table. Each row in the table corresponds to the annotators categorisation on a certain comment. A1, A2 and A3 represents the annotators, where each of these columns presents the annotator's decision for each of the comments.

Comment	**A1**	**A2**	**A3**
# 1	5	4	5
# 2	5	4	4
# 3	5	5	5
# 4	5	4	5
# 5	5	5	5
# 6	5	3	4
# 7	4	4	5

agree on the category. There are two instances where all three annotators agree. This applies to the following comments:

1. *Send henne hjem til Somalia med hele familien!*
2. *Når du trenes opp i bomber, kutter. strupen på barn og kvinner. Da er du et hjernevasket sharia dyr. Og du er etnisk pakistansk. Bør du bli sendt til pakisan med ett lite bidrag. De klarer av slikt. Send hennes influensere dit også!*

These comments are clearly related to a group of people; immigrants in Norway. The first comment implies that a person from Somalia should be returned to her home county. The comment encourages hateful actions and does not include any political opinions. Hence, this is considered hateful.

There is also one comment where all three annotators disagree:
Lite trolig, mer trolig at russerne må hjelpe til med å kaste de virkelige fiendene ut av Europa til slutt, nemlig den islamske invasjonen.

This comment expresses a negative opinion about Islam and addresses them as the enemy. Comments like these are difficult to annotate because they can be interpreted as both hateful and offensive. The comment is implying that the writer wishes to deport Muslims, but it is not explicitly stated. Further, the sentence is formulated as an assumption, where the author asserts something that he or she thinks might happen.

Observed agreement

The observed agreement (percentage agreement) is the most straightforward and easiest measure, and it was calculated for both the entire dataset and for each category. It was found that 89.8% of all comments were classified equally by all three annotators and that 99.2% of all comments were classified equally by at least two annotators. The percentage agreement was calculated between each pair of annotators, and the results are reported in Table 4.4.

Table 4.4.: Percentage of inter-annotation agreement between each pair of annotators, both for the total dataset and for each category. C1 to C5 represents the five different categories. A1, A2 and A3 are the annotators. Total is the entire dataset.

Annotator pair	**C1**	**C2**	**C3**	**C4**	**C5**	**Total**
A1 vs A2	92.9%	12.4%	15.6%	35.3%	33.3%	91.6%
A2 vs A3	96.2%	16.4%	25.9%	35.3%	40.0%	95.0%
A1 vs A3	93.0%	15.0%	27.3%	38.5%	57.1%	92.2%
Average	94.0%	14.6%	22.9%	36.4%	43.5%	92.9%

The percentage agreement calculations in each category are based on the number of comments labelled as this category by either one of the annotators in the annotator pair. It is worth noting that since there are so much fewer comments containing hate, the percentage values drop significantly when the annotators disagree on a few comments.

To check how good the annotators were at separating between the neutral comments and everything else, all categories from 2-5 were merged into a new category. When calculating agreement on this data, it was found that all three annotators categorised similarly in 91.4% of the cases (2207 out of 2415 comments). Here, 2150 comments were of class 1 and 57 were of the other class. If looking at all the comments where at least two are agreeing, then the annotators agree in 100% of the cases. This indicates that they mostly agree on the neutral comments, but tend to disagree more once the comment contains provocative, offensive or hateful content.

Kappa and alpha

Cohen's kappa, Fleiss' kappa and Krippendorff's alpha were calculated for each annotator pair and all three annotators. Since Cohen's kappa can only be applied to two annotators, it is only calculated for the annotator pairs. The results are presented in Table 4.5. It was found that Fleiss' kappa was equal to Cohen's kappa for all the annotator pairs. Therefore, there is only one row for both Cohen's kappa and Fleiss' kappa.

Table 4.5.: Calculated inter-annotator metrics. The row for kappa presents the Cohen's kappa for each pair of annotators, and Fleiss' kappa is the calculated value for all three annotators. Fleiss' kappa is equal to Cohen's kappa when the number of annotators is two.

Metrics	**A1 vs A2**	**A2 vs A3**	**A1 vs A3**	**All three**	**Average**
Cohen's/Fleiss' κ	0.3442	0.4666	0.3884	0.3931	0.3981
Krippendorff's α	0.3414	0.4667	0.3854	0.3918	0.3963

As can be observed from the table, the overall inter annotator-agreement using all three metrics are relatively low. Based on the interpretation of kappa in Section 2.6.3, the average kappa score is considered to be a fair agreement. A possible reason why the metrics are low is because of the expected agreement. With only five classes and 2500 comments, the expected agreement is anticipated to be high. Furthermore, as already stated, both Ross et al. (2017) and Schmidt and Wiegand (2017) emphasised that even though annotators have common annotation guidelines, the agreement score amongst the annotators are often deficient. This is in line with the obtained scores for this dataset.

In the case where the dataset is only split between normal and anomalous data (category 1, 2 and 3 versus 4 and 5), as is relevant in this thesis, the metrics are presented in Table 4.6.

Table 4.6.: Calculated inter-annotator metrics when only separating between normal and anomalous samples. The row for kappa presents the Cohen's kappa for each pair of annotators, and Fleiss' kappa is the calculated value for all three annotators. Fleiss' kappa is equal to Cohen's kappa when the number of annotators is two.

Metrics	A1 vs A2	A2 vs A3	A1 vs A3	All three	Average
Cohen's/Fleiss' κ	0.7078	0.7317	0.8265	0.7540	0.7550
Krippendorff's α	0.7079	0.7317	0.8266	0.7541	0.7551

The average metrics presented in the table above are categorised as substantial performance and is considerably better than the results when separating between all five classes. Hence, the inter-annotator agreement should probably not adversely affect the results in this thesis.

Reliable annotation is a desirable goal, but that is often difficult to attain in linguistic annotation tasks. Nevertheless, it is possible that annotation that is less than reliable also contains enough sufficient information to allow inference of the correct labels by a learning model (Artstein, 2017). So even though several of the agreement metrics were relatively low overall, it was concluded that for the rest of the dataset, only one person could annotate each comment. The reasoning behind this conclusion is based on both the percentage agreement and the restricted time and financial situation. Since the dataset had to be completed within a strict time frame, there was not enough capacity to annotate all the comments several times.

4.3. Challenges

The subjective interpretation of hate speech and offensive language creates challenges related to the creation of a dataset. Furthermore, the difficulty of annotation and challenges related to language are problems that need to be addressed. A substantial part of the section discussing language issues was written as a part of the specialisation project (Jensen et al., 2019), but some additional information has been added.

4.3.1. Annotation

One common challenge with annotated datasets is biasing. In this thesis, a part of the dataset was annotated and tested by the annotators, resulting in annotation bias. Presumably, the dataset is also biased because of the user-based annotation. Even though guidelines were provided, there was no insurance that the outside annotators actually followed these. The categorisations were checked, and everything the expert annotators considered to be incorrectly categorised was changed. However, since there were many comments, there is a chance that some comments are still miscategorised. Due to the lack of time, the challenge of annotation bias has not been handled further during this thesis.

Furthermore, another challenge related to the user-based annotation is that almost all of the annotators are approximately the same age and has the same education. There is a possibility that the dataset would have been annotated differently if a more representative sample of the population carried out the annotation.

All the expert annotators have achieved a substantial amount of insights on the topic and had a clear definition of hate speech and offensive language. However, the overall inter-annotator agreement metrics kappa and alpha was low. When the expert annotators do not agree on the categorisation of the data, it can be expected that the user-based annotation varies even more. Another challenge of annotation was context. Not having the context of the comments made the annotation more open for subjective opinions. A possible solution is to cluster comments into conversation threads and thus provide entire conversations to the annotators.

4.3.2. Language issues

English language, being one of the most used languages on the Internet today, is known as relatively easy to preprocess with high accuracy within fields such as part-of-speech tagging, named entity recognition and automatic text summarization. However, languages differ in how challenging they are to preprocess. When studying a language with richer morphology, more flexible word order and distinctive linguistic characteristics, more preprocessing are needed.

Compound words are a common part of Norwegian vocabulary. The Norwegian language often constructs compound words whereas in English two words would be written. Examples of this can be "ferietur" in Norwegian and "holiday trip" in English. Compound words can be a challenge because there is no guarantee that commentators write grammatically correct, resulting in one word being written as two, thus possibly changing the meaning of the sentence and confusing the model.

Another challenge with text classification for Norwegian is the fact that Norway has three official languages: Bokmål, Nynorsk and Sami. The Sami language is beyond the scope of this work. However, the difference in Bokmål and Nynorsk is a challenge. For instance, different words that have the same meaning such as "Kjærleik" and "Kjærlighet" will lead to an even higher dimensionality because the words are interpreted as different words but have the same meaning. This is also the case if using word embeddings. A few pre-trained word embeddings are available for the Norwegian language, but they differentiate between Bokmål and Nynorsk. This causes the challenge that the pre-trained model might not represent a significant amount of the words. The same issue can also be found when commentators write in dialect. For human annotators, it will most likely be possible to understand the meaning of the comment, but for a model, it might be more challenging with words such as "jøtt" and "gæli". In addition, English has a rather strong position in Norway and a lot of Norwegians, especially youth, write and use English as a natural part of their everyday language. This will also potentially create a more sparse matrix.

5. The ADAHS Approach

This chapter presents our hate speech detection approach called ADAHS; a deep semi-supervised Anomaly Detection Approach to Hate Speech detection. The model is based on and extends the principles and implementation of Deep SAD, a method for general semi-supervised anomaly detection, proposed by Ruff et al. (2020), as well as the implementation of Context Vector Data Description (CVDD) by Ruff et al. (2019). The method builds upon the idea that normal data instances (neutral speech) often resemble each other, while this is not necessarily the case for anomalies (hate speech). Throughout this thesis, normal data instances include both neutral and offensive speech, whereas anomalous instances represent hate speech unless stated otherwise. ADAHS uses semi-supervised learning and consists of pre-trained word embeddings and a convolutional neural network (CNN) that are implemented using `PyTorch` in `Python`. This chapter starts by presenting the text preprocessing steps conducted before the architecture is described in great detail. Lastly, the system's functionality, including learning objective, optimisation and regularisation, is outlined.

5.1. Text preprocessing

It is challenging for a computer system to be able to interpret ambiguous and unstructured language correctly. Deep neural networks methods aim at overcoming the need for manual feature engineering by being able to learn the relevant features from the data automatically. Even though they do not require the amount of feature extraction necessary for shallow approaches, they are still not capable of handling raw text as input. Therefore, for the models to be able to understand natural language, the text needs to be preprocessed. This is an essential but difficult step in NLP, especially when handling user-generated content that contains much noise.

In this thesis, two datasets are used; one in the Norwegian language and one in the English language, and these require customised text cleaning. On both datasets, the text is always lowercased, and '\n' is replaced with whitespace. Further, the text is stripped for whitespaces before and after the text. As a part of the manual annotation process of the Norwegian dataset, it was observed that many comments did not include whitespace after punctuation marks. Hence, if these punctuation marks are removed, the preceding and following words would become one word. For instance, in the following case *[...] her.dette [...]*, one would be left with the new word *herdette* if the punctuation mark is removed. Hence, whitespace is added after all punctuation marks to avoid this issue. Furthermore, for the English dataset, CSS phrases and specific tokens used in Wikipedia are removed. This includes image names and templates such as *wikipedia:*, *user:* and *category:*. For the Norwegian dataset, tokens referring to a user are removed. This includes both *@user* and *navn* which were used instead of names and usernames when the dataset was created.

Moreover, for both datasets, the text is stripped from punctuation, numbers and special characters, by creating a plain ASCII string from the text. Here, the additional letters *æ*, *ø* and *å* from the Norwegian alphabet are added to the list of ASCII letters from the English alphabet. Finally, all redundant whitespaces are removed.

Since pre-trained word embeddings are utilised, it was desirable to get the vocabulary as close to the embeddings as possible. Therefore, common misspellings were replaced. Two dictionaries of common misspellings in English and Norwegian were created, containing both the incorrect and correct spelling. These contain regular common misspellings, as well as misspellings specific for the datasets. The specific misspellings were found by analysing the Out of Vocabulary (OOV) words obtained by comparing the datasets and the pre-trained word embeddings. OOV words are all the words that are present in the datasets vocabulary, but not present in the pre-trained model's vocabulary. The pre-trained embeddings used for the comparisons with the English dataset was GloVe 5B, while fastText with Norwegian Bokmål tokens was used with the Norwegian dataset.

After the text was cleaned, the semi-supervised setting was created. This setting is described further in the following subsection. Furthermore, a text vocabulary was built to later convert the tokens into integer numbers. The vocabulary is constructed based on all words present in the entire dataset. Following, the text was tokenised using the `SpacyEncoder`[1] from `PyTorch-NLP`, that encodes the text using `spaCy`'s[2] tokeniser. This tokeniser is language-specific but has support for both Norwegian and English. In Section 2.5, a common preprocessing pipeline is described. This includes tokenisation, but also the removal of stopwords and stemming/lemmatisation. When using pre-trained word embeddings, these two last steps should be avoided. The reason is that valuable information is lost, which could help the neural network discover patterns. Hence, stopword removal and stemming/lemmatisation were not included in the preprocessing.

Before the data are used in training, validation and testing, the data is divided into batches. Generally, each batch contains 64 samples. The `BucketBatchSampler` from `PyTorch-NLP` was used to pool together samples from the data with similar length to minimise the amount of padding needed, while still maintaining some noise through bucketing. The last batch was dropped if its size was smaller than the batch size to assure that all batches were of the same size. The data in each generated batch were then stacked, and the text field was zero-padded to achieve samples of equal length. Further, the text was transposed in order to achieve speed and integration with CUDA.

[1]https://pytorchnlp.readthedocs.io/en/latest/source/torchnlp.encoders.html
[2]https://spacy.io/models

5.1.1. Semi-supervised setting

The model uses semi-supervised learning, which leads to a semi-supervised anomaly detection setting: The data consists of n unlabelled samples x_1, ... , x_n, which is mostly or only normal data, and m labelled samples $(\tilde{x}_1, \tilde{y}_1)$, ... ,$(\tilde{x}_m, \tilde{y}_m)$, where $\tilde{y} = +1$ denotes normal data and $\tilde{y} = -1$ denotes anomalous data.

In our case, we have two datasets that are completely labelled. In order to use semi-supervised learning, the semi-supervised setting had to be created. This involves choosing a ratio of labelled normal and anomalous data to include. This ratio changes during the experiments. Given a desired ratio of labelled data, the returned dataset includes semi-labels where the normal labelled data have labels $\tilde{y} = +1$, the labelled anomalies have labels $\tilde{y} = -1$ and the unlabelled part of the dataset has $\tilde{y} = 0$. The ground truth labels of all data samples are kept in order to quantitatively measure performance during testing. The exact semi-supervised setup for both datasets are further described in Section 6.2.1.

5.2. Model architecture

Roughly, the model consists of a word embeddings part and a convolutional neural network (CNN) part. An overview of the model's architecture is presented in Figure 5.1. The figure does not include the cleaning and text preprocessing, only the architecture of the implemented model. The input is the cleaned, preprocessed and batched text, as described in Section 5.1.

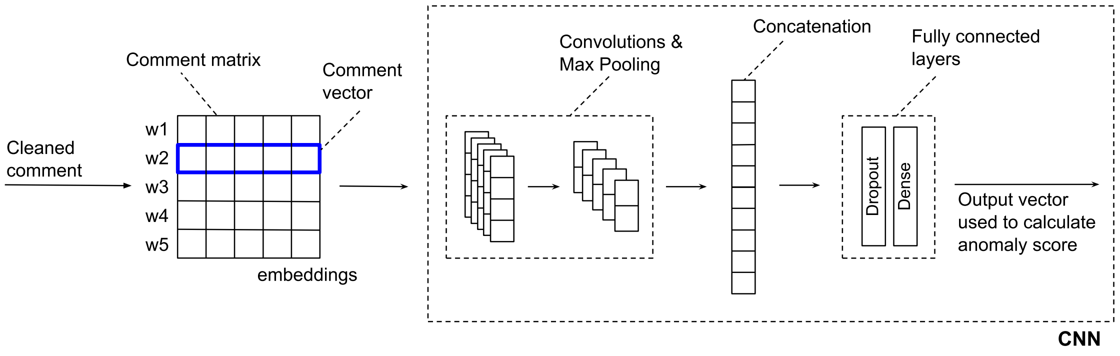


Figure 5.1.: An overview of the system architecture

Proper text representation is crucial for designing well-performing machine learning algorithms. As explained in Chapter 3, several existing methods within the field of hate speech detection utilises pre-trained word embeddings, but only the work by Ruff et al. (2019) utilises it in combination with anomaly detection on textual data. Hence, their work was used as a starting point for the inclusion of pre-trained word embeddings in this system. The system can handle the use of several different pre-trained models. This includes all the available GloVe models (English language only); 6B tokens with dimensions 50, 100, 200 or 300 trained on Wikipedia and Gigaword, 42B and 840B tokens with dimension 300 trained on Common Crawl and 27B tokens with dimensions 25, 50, 100 or 200 trained on Twitter data.[3] Furthermore, the model can also use fastText with dimensions 300 for both English and Norwegian Bokmål.[4]

It was conducted research on how to handle Out of Vocabulary (OOV) words. Probably, the best way to handle these instances is by creating a language model built to produce embeddings for OOV words depending on their context, as done by Kandi (2018). Nevertheless, due to the limited time and resources, this was considered unnecessary for the purpose of this thesis. Therefore, the OOV words were initiated as zero-vectors with the same dimension as the pre-trained vectors. However, a drawback to this approach is that the model cannot find relationships between these OOV words the other words. Another common way of initialising these vectors are by randomly assigning vectors with the same dimension. A significant drawback to this approach is that the randomly generated vector might be similar to other pre-trained word vectors. As a result, the model might believe that two words are of similar meaning, even though they are not.

The architectural overview in Figure 5.1 shows that the CNN consists of convolutions and max pooling, as well as a dropout layer and a dense/linear layer. Convolutions are sliding window functions applied to a matrix. This sliding window is often called a kernel or a filter, and it can have variable sizes. CNNs are most commonly used on images, which is just a matrix of pixels. When that is the case, the filters may have different height and width, and the final convolutions are found by sliding the filters over the entire matrix. On the other hand, when convolutions are used in NLP, the filters are applied to a matrix of stacked word embedding vectors. We denote the dimensionality of the word vectors by d. If the length of a given padded comment is s, then the dimensionality of the matrix is $s \times d$. This matrix is the leftmost part of the figure. In NLP, the filters usually have the same width as the length of the word embeddings. Hence, the filters slide over the word embeddings, which is matrix rows. The filter height (also called region size) varies and corresponds to the number of adjacent rows considered jointly, i.e. the n-grams (a 1xn filter) within the text.

In our model, multiple filters for the same region size are used in order to learn complementary features from the same regions. A figure displaying all the layers in the model is presented in Figure 5.2. This CNN architecture is based on the model proposed by Y. Zhang and Wallace (2016), which uses a CNN for sentence classification.

As can be seen from the figure, the model's input first passes through the embeddings layer. This is where the text is converted into word vectors based on the pre-trained vectors. After that, we depict five convolutional layers with filter region sizes varying from 1 to 5, each with 100 filters. The filters perform convolutions on the comment matrix and generate feature maps of variable length. This corresponds to finding the one to five

[3] 6B, 42B etc. (B = billion) refer to the number of tokens in the model's vocabulary. All GloVe pre-trained models can be found here: https://nlp.stanford.edu/projects/glove/

[4] Both fastText pre-trained models can be found here: https://fasttext.cc/

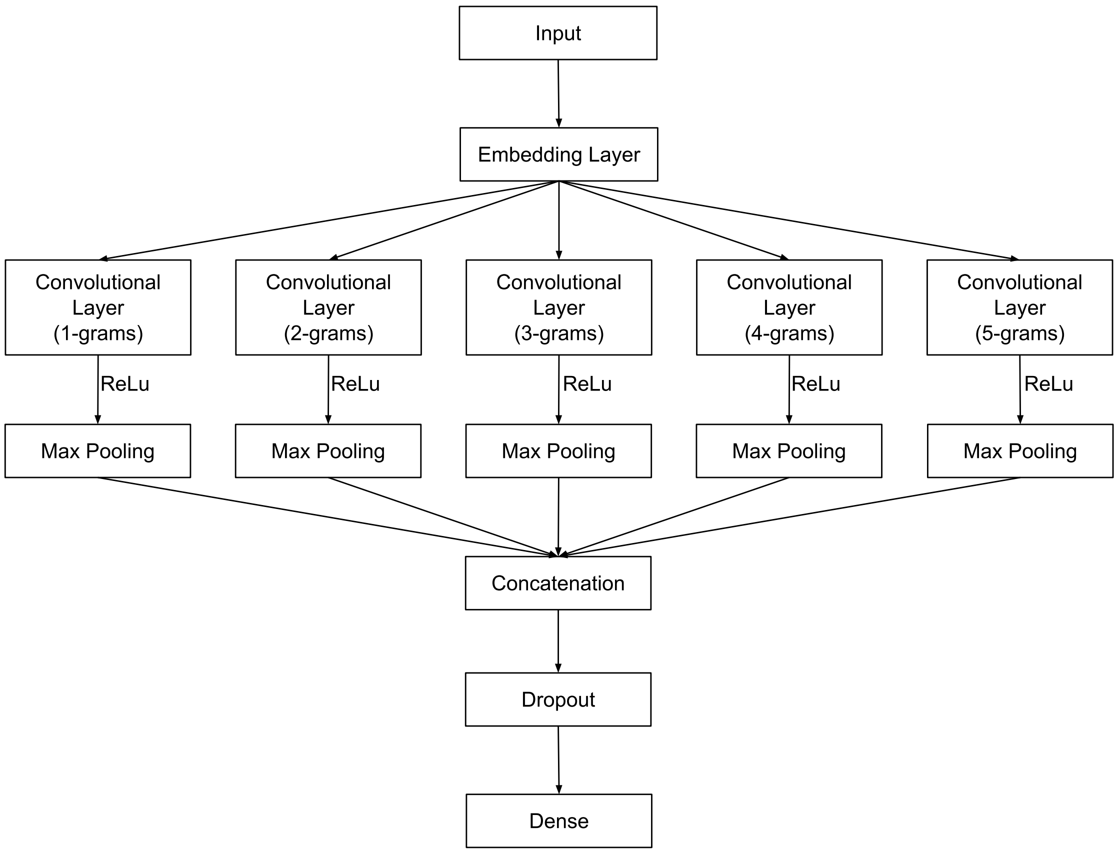

Input
Embedding Layer
Convolutional Layer (1-grams)
Convolutional Layer (2-grams)
Convolutional Layer (3-grams)
Convolutional Layer (4-grams)
Convolutional Layer (5-grams)
ReLu
ReLu
ReLu
ReLu
ReLu
Max Pooling
Max Pooling
Max Pooling
Max Pooling
Max Pooling
Concatenation
Dropout
Dense

Figure 5.2.: All the layers in the neural network model

n-grams of the text, meaning the one to five consecutive words. The purpose of this part is to capture the spatial relationship in text, which makes it easier for understanding the meaning of the comments. The rectified linear unit (ReLu) activation function is applied to the output of the convolutional layers before max-pooling is applied. The idea behind the max-pooling is to find the maximum valued feature because this is presumably the most important feature, or in this case, the most important n-gram. As our model has 100 filters of five different region sizes, that means we have 500 different n-grams the model thinks are essential. The outputs of the max-pooling layers are then concatenated to form a feature vector for the penultimate layer. Further, dropout is applied to prevent the model from overfitting and increase robustness by randomly dropping nodes during training with a probability equal to 0.5. Lastly, the final fully-connected dense/linear layer receives this feature vector as input and uses it to create an output/latent vector with a representation dimension $d = 32$. This output vector is used to calculate a comment's anomaly score. A detailed view of the described architecture is presented in Figure 5.3.

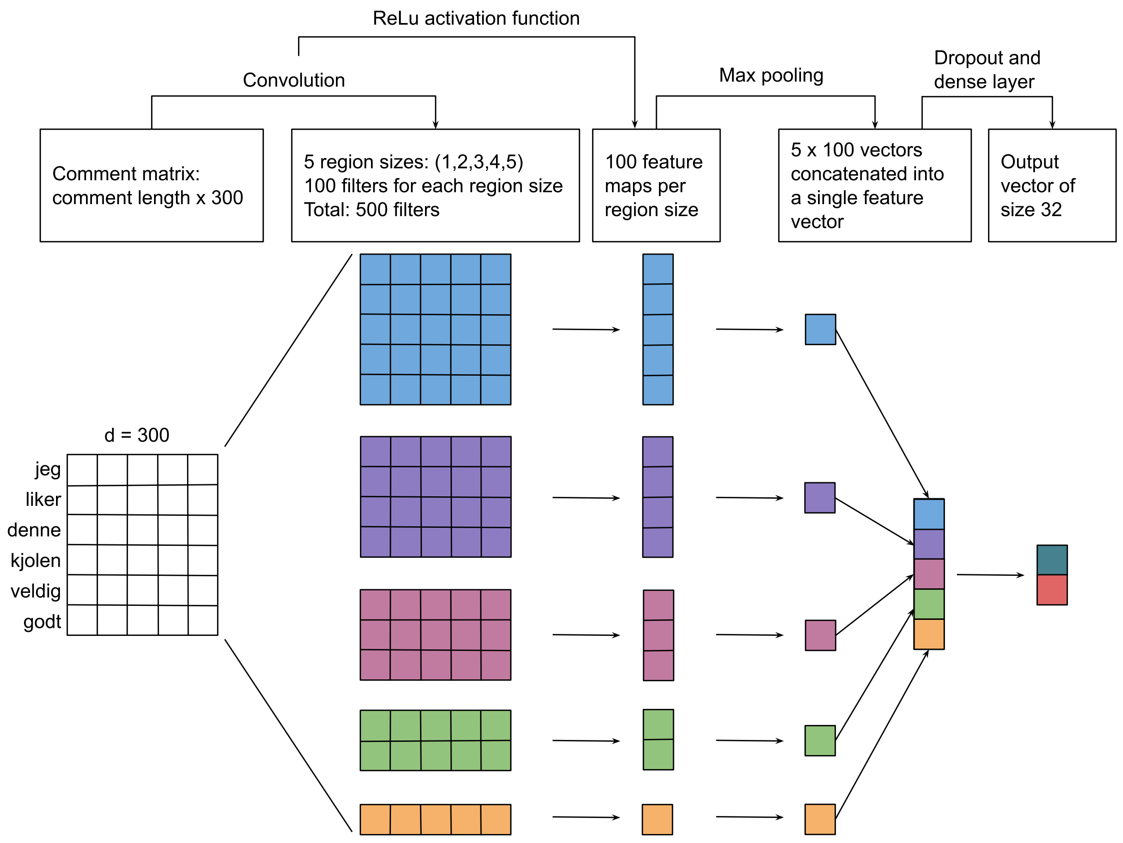

ReLu activation function
Convolution
Max pooling
Dropout and dense layer
Comment matrix: comment length x 300
5 region sizes: (1,2,3,4,5)
100 filters for each region size
Total: 500 filters
100 feature maps per region size
5 x 100 vectors concatenated into a single feature vector
Output vector of size 32
d = 300
jeg
liker
denne
kjolen
veldig
godt

Figure 5.3.: A detailed illustration of the CNN architecture. The figure is based on the figure by Y. Zhang and Wallace (2016).

5.3. System functionality

The system aims at solving the semi-supervised anomaly detection problem, as presented in Section 5.1.1, where the goal is to learn a model that characterises the "normal class". The system builds upon the principle that normal data instances often resemble each other, whereas this is not the case for anomalies. In this way, the normal data obtains a latent distribution with *low entropy*, and the anomalous data obtain a latent distribution with *high entropy*. Hence, the model's learning objective may be interpreted as modelling the latent distribution/output space, Z, of normal data, $Z+ = Z|\{Y = +1\}$, to have a low entropy, and the latent distribution of anomalies, $Z- = Z|\{Y = -1\}$, to have high entropy. In order to achieve a latent distribution that concentrates the normal data around centre c, a quadratic loss is imposed on the distance between the labelled normal samples ($y = +1$) and c. In contrast, for the labelled anomalies ($y = -1$) the loss is calculated based on the inverse of the quadratic distance between the samples and c. As a result, these samples are mapped further away from c. An epsilon (`eps` ~10^{-6}) is added to the denominator of the inverse to ensure numerical stability (Ruff et al., 2020). This is in line with the common assumption that anomalies are not concentrated because they are not similar. Ruff et al. (2020) argue that such a model better captures the nature of anomalies, as they can be generated from a mixture of different distributions dissimilar from the normal data distribution. It is worth noting that this method does not impose any cluster assumption on the distribution of the anomalies, i.e. it does not imply that anomalies must be a part of the same class, just that they are not a part of the normal class.

It is important to choose a proper weight initialisation strategy to maximise performance, and the choice of strategy depends on the activation functions used in the model. Therefore, the network weights W are initialised using *Kaiming He* initialisation (He et al., 2015). Kaiming He initialisation strategy brings the variance of the outputs to approximately one when the ReLu activation function is used. After the network weights W are initialised, the hypersphere centre c is calculated as the mean of the network outputs obtained from an initial forward pass of the data. This was based on the findings from Ruff et al. (2020) that observed a faster network convergence by fixing c in the neighbourhood of the initial data instances. Once the network is trained, the anomaly score for a given unlabelled test instance, x, is calculated as the distance from x to c. For the labelled data instances, a new loss term is proposed. This loss term introduces a hyperparameter $\eta > 0$ used for controlling the balance between labelled and unlabelled data. If $\eta > 1$, it puts more emphasis on the labelled data, whereas an $\eta < 1$ puts more emphasis the unlabelled data.

The system's functionality and learning objective are similar to those of the model presented by Ruff et al. (2020). The difference is that they use an autoencoder network to pre-train their model's weights instead of using Kaiming He initialisation. Furthermore, their system does not handle textual data.

5.3.1. Optimisation and regularisation

For a computationally efficient optimisation, the system can use either Adam or SGD to optimise the network weights using backpropagation. Adam was used in all experiments conducted in this thesis. A two-phase learning rate schedule is employed for "searching" and "fine-tuning", respectively. This involves changing the learning rate after a given number of epochs. In order to avoid overfitting, it is necessary to apply some regularisation to the system. As described, the primary means of regularisation is dropout, which is applied after the concatenation with a probability of 0.5. Furthermore, an L^2 weight decay regularisation, λ, with $\lambda > 0$ are added for improved generalisation. λ is a parameter that is set at system initialisation and is used when the optimiser is initialised. The specific parameters used in the experiments conducted in this thesis are described in Section 6.2.

6. Experiments and Results

In order to investigate how to detect hate speech using anomaly detection techniques and to answer the research questions stated in Section 1.2, a series of experiments were carried out. This chapter presents the experiments conducted to test the implemented architecture from Chapter 5, and determine if there is a potential for using anomaly detection to detect hateful utterances. Since the purpose of this thesis was to investigate whether or not the problem can rightfully be considered an anomaly detection problem, the focus of the experimental part has not been to find the best configurations for the system. These configurations involve, for instance, the number of layers, filter sizes and other hyperparameters. Instead, the main focus of the experiments was to check for potential and discover how the amount of labelled training data and pollution in the training data, affect performance. Also, an important part of the experimental research was to test if the implemented model was language independent, as proposed in Research question 3. First, Section 6.1 provides a description of which experiments were planned and conducted, as well as what questions the experiments aim to answer. Further, Section 6.2 presents the datasets used in the experiments and the methodology for the experimental evaluation. This section also includes a detailed description of the parameters and settings used during the experiments and a description of baseline methods. Finally, Section 6.3 presents the experimental results.

6.1. Experimental plan

An experimental plan is crucial for keeping a series of experiments effective and structured, and the experimental plan for this research consists of three main parts as described in the rest of this section.

Even though the primary purpose of these experiments was not to find the best system configurations, it was desirable to test two of the model's parameters using different values. This was the first part of the experimental conduction and included the following parameters:

Hyperparameter η: Ruff et al. (2020) did a sensitivity analysis of the model's hyperparameter η and found that setting $\eta = 1$ gave a substantial performance improvement. However, they conducted experiments on images, and it was thus determined to test different values for η. Since this was not the main purpose of the experiments, the analysis is not as thorough as the one presented by Ruff et al. (2020). It was decided to test five values for η, only varying this hyperparameter.

L^2 weight decay, λ: It was also desirable to test different values for λ, only varying this parameter. Again, this testing was not comprehensive and only included testing three different values.

The second part of the experimental phase serves to answer the research questions and the thesis' overall goal. This includes identifying how the amount of labelled training data and pollution affect performance on two datasets of different language. It involves the examination of two experimental tests, in which the following experimental parameters are varied: (1) the ratio of labelled training data γ_l; both normal, γ_n, and anomalous, γ_a samples, and (2) the ratio of pollution, γ_p, with anomalous samples in the unlabelled part of the dataset.

Test (1) Adding labelled training data: The purpose of this test is to investigate the effect of labelled samples on system performance. Additionally, it is conducted to test if satisfactory performance can be achieved even though one might not have access to a substantial amount of labelled data. These experiments also include the unsupervised setting where no labelled data is added. To conduct these experiments, the ratio of labelled training data, $\gamma_l = m/(n+m)$ is increased by adding more labelled samples. In the experiments, both the ratio of normal data, γ_n, and anomalous data, γ_a, are varied, and the labelled samples are drawn randomly from a pool of normal data samples and anomalous data samples. As previously stated, the main issue with existing hate speech detection methods is that they have poor performance when they are trained on one dataset and tested against another, which indicates that they struggle with the detection of novelties. Therefore, these experiments are also conducted to test the model's ability to handle novelties and to simulate the unpredictable nature of anomalies. This is done by only sampling labelled anomalies from a specific class of anomalies. Hence, this is only relevant when a dataset has different anomaly classes. In this test, all the unlabelled data in the training set is normal samples; hence, the ratio of pollution is set to zero.

Test (2) Polluted training data: In the ideal case, the unlabelled part of the training dataset only consists of normal data samples, which is a common assumption used in anomaly detection methods. However, in a real-life scenario where one may have a large proportion of unlabelled samples, it is difficult to determine if all of these samples are, in fact, normal samples. There is a chance that some of the samples are anomalous. Therefore, in this test, the model's robustness to increasing pollution of unlabelled anomalous samples in the training dataset is investigated. The unlabelled part of the dataset is polluted with a ratio of anomalous samples, γ_p. These samples are drawn randomly from either a specific anomaly class or from a pool of all anomalous samples.

The third experimental part involves testing both datasets with two baseline models; the one-class SVM (Schölkopf et al., 2001) and the Context Vector Data Description (CVDD) model (Ruff et al., 2019). This is further elaborated in Section 6.2.4.

6.2. Experimental setup

This section presents the data, hyperparameters and other configurations necessary to reproduce the experiments. It starts with a description of the English and Norwegian dataset used in the experiments, followed by a presentation of the semi-supervised setup for both datasets. Furthermore, configurations and hyperparameter values are presented. Finally, the evaluation methodology is explored, which involves an overview of evaluation metrics and setup of the baseline methods.

6.2.1. Datasets

To perform the experiments with semi-supervised detection of hate speech and to check the feasibility of the approach, two large datasets were used; one in the English language and one in the Norwegian language. For both datasets, the text preprocessing steps described in Section 5.1 were applied.

English (Jigsaw) dataset

The English dataset used in the experiments was the *Jigsaw* dataset from a *Toxic Comment Classification Challenge*, which is available on Kaggle.[1] The dataset consists of a number of comments and their respective set of labels. There are six categories in the dataset; toxic, severe toxic, obscene, threat, insult and identity hate. Two random samples from the dataset looks like this:

Table 6.1.: Two random samples from the Jigsaw dataset showing how the data is represented. Each row also contains an id which is not included here.

comment text	**toxic**	**severe toxic**	**obscene**	**threat**	**insult**	**identity hate**
FUCK YOUR FILTHY MOTHER IN THE ASS, DRY!	1	0	1	0	1	0
You, sir, are my hero. Any chance you remember what page that's on?	0	0	0	0	0	0

As can be seen from Table 6.1, a comment may have several labels, exactly one label or no labels. The comment is neutral if does not have any labels, i.e. none of its labels are set to one. The dataset consists of one training and test set, where some of the labels in the test set are set to -1. After all of these labels were removed, the dataset consisted of

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

223 549 comments in total, where 159 571 are in the train set and 63 978 are in the test set. Hence, the test set consists of 28.6% of all comments. Table 6.2 shows the number of comments that are labelled in each category.

Table 6.2.: The number of comments in each category in the English dataset

Label	**Train set**	**Test set**
Toxic	15294	6090
Severe toxic	1595	367
Obscene	8449	3691
Threat	478	211
Insult	7877	3427
Identity hate	1405	712

Since many comments are labelled with more than one label, the total number of hateful comments is less than the sum of the comments presented above. Table 6.3 shows the distribution between hateful and neutral in both the training and test set. The number of hateful comments is calculated by summing all comments that contains at least one hateful label. The number of neutral comments is the sum of all comments without any hateful label.

Table 6.3.: The number of comments categorised as neutral and hateful in the English dataset.

Dataset	**Neutral**	**Hateful**	**Total**
Train	143346	16225	159571
Test	57735	6243	63978

Based on the amount of neutral and hateful comments presented in the table above, one can find that both datasets contains approximately 90% neutral comments.

Since the test set contains approximately 30% of all the data, one third of this data are used for validation instead of testing. This results in approximately 70% of the data used for training, 10% for validation and 20% for testing.

Norwegian dataset

After the entire dataset was annotated and all comments labelled with X were removed, the dataset was complete. Every row in the dataset is on the format 'id, label, text'. The dataset contains the number of comments presented in Table 6.4.

Table 6.4.: The number of comments and percentage of total in the annotated Norwegian dataset

Category	Number of comments	Percentage of total
1 - Neutral	34083	82.8%
2 - Provocative	4734	11.5%
3 - Offensive	1563	3.8%
4 - Moderately hateful	509	1.2%
5 - Hateful	250	0.6%
Total	41139	100%

The distribution of comments in the five distinct categories is also displayed in Figure 6.1.

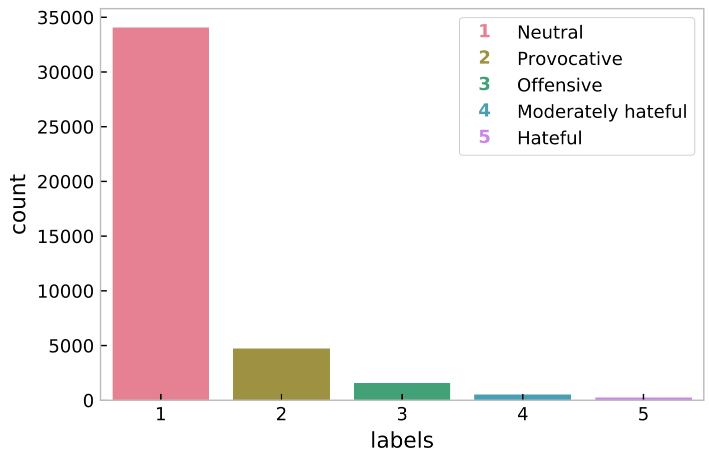


Figure 6.1.: The distribution of comments in each category in the Norwegian dataset

To fit the problem statement of this thesis, it was necessary to only separate between hateful/anomalous and everything else. The inter-annotator agreement calculations in Section 4.2.3 indicated that even the expert annotators struggled to agree on the annotation of the comments in category 4 or 5. Based on this and the definition in Section 2.1, it was chosen to include both the moderately hateful and hateful comments as the anomaly class, whereas category 1 to 3 was the normal class. But since many hate speech detection methods struggle with the separation of offensive and hateful utterances, it was decided to compare the performance of the model with and without the inclusion of the offensive class as anomalies. This involves the investigation of two cases: (1) class 1, 2 and 3 are normal samples and class 4 and 5 are anomalous, (2) class 1 and 2 are normal samples and class 3, 4 and 5 are anomalous. Both experimental tests from Section 6.1 were conducted using these two cases. Table 6.5 displays the number of comments and percentage of total when only separating between normal and anomalous in the two cases explained above.

Table 6.5.: Preprocessed combined annotated Norwegian dataset with and without the inclusion of the offensive class as anomalies.

	Case 1: 4+5 are anomalies		**Case 2: 3+4+5 are anomalies**	
Category	**#comments**	**% of total**	**#comments**	**% of total**
Normal	40880	98.2%	38817	94.4%
Anomalies	759	1.8%	2322	5.6%
Total	41139	100%	41139	100%

The dataset is separated into a training, test and validation set using stratified splits in the beginning of every experiment. In other words, the data were divided so that each set had approximately the same distribution between the different classes.

6.2.2. Semi-supervised setup

The semi-supervised setup involves choosing a ratio of labelled and unlabelled normal and anomalous data to include during training, as described in Section 5.1.1.

The Jigsaw dataset contains seven categories; one neutral and six hateful. The neutral category is always chosen as the normal class, while the rest of the categories represent anomalies. The desired ratio of labelled normal samples, γ_n, are drawn randomly from the normal class. The remaining normal samples are included as the unlabelled part of the training set. This setting fulfils the assumption that most (in this case all) of the unlabelled samples in the training set are normal. The training data of the six anomaly classes form the data pool from which anomalies are drawn to achieve the experimental tests presented in Section 6.1. Both of the tests involve varying the ratio of normal samples, γ_n, anomalous samples, γ_a, and pollution, γ_p. In the first test, the semi-supervised setup involves setting $\gamma_p = 0.00$, while $(\gamma_n, \gamma_a) \in \{(0.00, 0.00), (0.10, 0.00), (0.00, 0.05), (0.00, 0.10), (0.05, 0.05), (0.10, 0.10)\}$. The cases where $(\gamma_n, \gamma_a) \in \{(0.00, 0.00), (0.10, 0.00)\}$ involves not adding any labelled anomalies, and will from now be referred to as the unsupervised and normal setting, respectively. For the second test, the setup involves setting $\gamma_n = \gamma_a = 0.05$ and $\gamma_p \in \{0.01, 0.05\}$. In both tests, the labelled anomalies are either sampled from one of the six anomaly classes, or a data pool consisting of all the six classes. Hence, it involves seven experiments per semi-supervised setup. During testing, all six hateful classes are considered anomalies, i.e. there are five novel classes at testing time.

On the other hand, the Norwegian dataset contains five categories. As explained in Section 6.2.1, two cases are considered, where these five categories are separated into a normal class and an anomalous class. In both cases, either category {1, 2, 3} or {1, 2} constitute the normal class, and hence, the labelled normal samples are drawn randomly from this concatenated class. The remaining normal samples are included as the unlabelled part of the training set. Either category {4, 5} or category {3, 4, 5} represents the anomalies, and the labelled anomalies are drawn from this class. The semi-supervised setup for the two tests are similar to the setup for the English dataset, but in this case, there are only two experiments per setup: case 1 and 2.

6.2.3. Configurations and hyperparameters

Pre-trained word embeddings from GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017) were employed in the experiments. For GloVe the 6B tokens vector embeddings of $p = 300$ dimensions that have been trained on the Wikipedia and Gigaword 5 corpora were used. GloVe only have pre-trained embedding for English. For fastText, the English and Norwegian Bokmål word vectors of size $p = 300$ dimensions are considered, which have been trained on the Wikipedia corpora and English and Norwegian web crawl.

In all the experiments, the seed was set to 1. This involves setting the random seed of build-in modules in order to make the experiments reproducable. Furthermore, the Adam optimiser was used with default hyperparameters as recommended by Kingma and Ba (2015). For the employed two-phase learning rate schedule, the searching phase uses learning rate eps $= 10^{-4}$ for 50 epochs, while in the fine-tuning phase uses eps $= 10^{-5}$ for another 50 epochs. Furthermore, all experiments are run using a batch size of 64. In accordance with the suggestions made by Srivastava et al. (2014), the probability of losing nodes due to dropout was set to 0.5.

The hyperparameters η and the L^2 weight decay regularisation, λ were fine-tuned in the first part of the experimental phase. The hyperparameters $\eta \in \{0.1, 1, 5, 10\}$ were tested using both the English and Norwegian dataset. When using the English dataset, only the GloVe word embeddings were used, while the Norwegian vectors from fastText were employed when experimenting with the Norwegian dataset. All other hyperparameters were set as described above, and λ was set to 0.5×10^{-6} in both sets of experiments. These tests showed that the greatest results were achieved when using $\eta = 10$ for both datasets. $\eta = 10$ was used in the following experiments with the weight decay regularisation. When experimenting with different values for λ, only the English dataset were used, and GloVe word embeddings were employed. All hyperparameters were set as described above, and the values considered were $\lambda \in \{0.5 \times 10^{-4}, 0.5 \times 10^{-5}, 0.5 \times 10^{-6}\}$. Both the experiments with η and λ used a ratio of labelled normal samples and labelled anomalous samples equal 0.05. The ratio of pollution was 0.00. The results showed that λ should be set to 0.5×10^{-6}, and hence this value was used for all the remaining experiments.

6.2.4. Evaluation methodology

This section briefly explains how the experiments are evaluated. First, the applied evaluation metrics are described. Furthermore, the chosen baseline methods are presented.

Evaluation metrics

To evaluate the model's ability to separate between normal and anomalous data, the AUC metric, as explained in Section 2.6.2, is computed using ground-truth labels. Even though the test data does not include labels when tested, the ground truth labels of all data samples are kept in order to measure performance quantitatively. Furthermore, ROC curves and precision-recall curves are used to visualise the performance. Average precision is also calculated by summarising the precision-recall curve as the weighted mean of precisions achieved at each threshold. Here, the increase in the recall is used as the weight. Additionally, a histogram was created based on the anomaly scores, which was used to set a threshold so that the scores could be converted into a binary labels. These labels are used to calculate precision and recall. Note that this is not conducted for every experimental setup, but only for those achieving the highest AUC.

Baseline methods

To assess the effectiveness of the proposed semi-supervised model, the model's results are compared with the results of two baseline methods. The considered baselines are the One-Class SVM (Schölkopf et al., 2001) and the Context Vector Data Description (CVDD) as proposed by Ruff et al. (2019). Both baseline methods use unsupervised learning, and are thus not tested on the same semi-supervised settings as ADAHS. Both of the baseline models use the same pre-trained embeddings as described above, and only consider class 4 and 5 as anomalies for the Norwegian dataset.

The OC-SVM model is implemented using the `sklearn` framework. It uses a radial basis function (rbf) kernel and the *gamma* parameter set to *scale*. Three baselines for aggregating the word vector embeddings to fixed-length sentence representations are considered: mean, tf-idf weighted mean, and max-pooling. The mean is a simple average sentence embedding, the tf-idf weighted mean includes document-to-term co-occurrence statistics, and max-pooling uses the maximum value of the word embedding dimensions. The model is tested for hyperparameters $\nu \in \{0.05, 0.1, 0.2, 0.5\}$. ν is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors relative to the number of training samples (Schölkopf et al., 2001).

The CVDD model (presented in Section 3.3) was cloned from GitHub.[2] The model had to be adapted in order to function with the two hate speech datasets, which involved adding support for handling these datasets and for Norwegian preprocessing. The experiments are conducted employing a self-attention module with dimensionality $d = 150$ and results are presented for $r \in \{3,\ 5,\ 10\}$ number of attention heads/contexts. CVDD uses the Adam optimiser with default parameters, a weight decay λ set to 0.5×10^{-6} and a batch size of 64. It employs a learning rate similar to the one explained in Section 5.3.1, using learning rate 0.01 for 50 epochs and 0.001 for another 50 epochs. Furthermore, for weighting contexts, a logarithmic annealing strategy for hyperparameter $\alpha \in \{0,\ 10^{-4},\ 10^{-3},\ 10^{-2},\ 10^{-1}\}$ is used, where α is updated every twentieth epoch. The hyperparameter for context vector orthogonality regularisation P is set to 1.0. Please refer to the work by Ruff et al. (2019) to get a description of all the presented parameters.

6.3. Experimental results

This section presents the experimental results achieved when performing the experiments presented in the previous two sections. The results from ADAHS using the two datasets are first presented separately before the results from the baseline methods are presented in its own section.

In both datasets, there was a drastic imbalance between the two classes, where the hateful instances were sporadic compared to the neutral instances. Therefore, a problem accounted was that not all anomaly classes contained enough data samples to add 5% or 10% labelled anomalous samples to the training set. An example applies to the "Threat" class (class 4) in the English dataset, which only consists of 478 samples. Two solutions to this problem were tested. The first solution included only adding the available amount of data as labelled anomalous samples. This solution involves adding all the normal data, and potential unlabelled anomalous samples, and hence, resulting in a smaller ratio of labelled anomalies. The second solution involved scaling the dataset in order to maintain the desired ratio of labelled anomalies. This approach includes adding all possible labelled anomalies from the respective class and decrease the amount of normal and unlabelled anomalous data in order to obtain the correct ratio.

Several of the classes containing too little data samples were tested using both solutions. One could observe that most of the AUC scores increased significantly for the English dataset by using the second approach. Even though this approach involves decreasing the total amount of training data, and thus also decreased performance compared to those classes containing enough anomalous samples, this approach still increased performance for these classes by an average of 1.1% AUC. On the other hand, the first approach was superior when using the Norwegian dataset, and improved AUC score by an average of 7.0%. Only the results using the preferable approach for each dataset are included in this section, but the results for the respective other approaches, are presented in Appendix A.

[2]https://github.com/lukasruff/CVDD-PyTorch

6.3.1. Results using the English dataset

In all tables in this section, the following numbers refer to the respective anomaly class:

1:	Toxic
2:	Severe toxic
3:	Obscene
4:	Threat
5:	Insult
6:	Identity hate

The first part of the experiments involved testing the hyperparameters η and λ. Table 6.6 presents the achieved results using the English dataset. As stated in Section 6.2, all these experiments are conducted using $\gamma_n = \gamma_a = 0.05$ and $\gamma_p = 0.00$. Only the GloVe 5B word embeddings were used and λ was set to 0.5×10^{-6} when experimenting with η. The experiments with λ were conducted after it was found that $\eta = 10$ achieved the best results. Hence, $\eta = 10$ was used in all of these experiments. That is why the AUC values for $\eta = 10$ and $\lambda = 0.5 \times 10^{-6}$ in the two tables are equal.

Table 6.6.: AUC scores (in %) of the first part of the experiments; testing hyperparameter values.

(a) AUC values for different values of η.

η	AUC
0.1	83.0
1	90.1
5	92.3
10	93.4

(b) AUC values for different values of λ.

λ	AUC
0.6×10^{-5}	93.4
0.5×10^{-5}	93.0
0.4×10^{-5}	93.4

These tests showed that the greatest results were achieved when using $\eta = 10$ and $\lambda = 0.5 \times 10^{-6}$. From the table, it can be observed that the AUC score when λ is 0.5×10^{-4} is equally good, but this is only due to rounding.

Test 1

The first test includes adding labelled data samples to the training data. It is conducted by adding either normal samples, anomalous samples or both. Table 6.8 presents the results for these setups. As previously mentioned, γ_l represents labelled samples (both normal and anomalous). In the table, when γ_l is for instance 0.05, then both the ratio of normal and anomalous samples is equal to 0.05.

In the unsupervised and normal setting, no experiments related to the different anomaly classes are conducted. The reason is simply that the anomalous samples are not included in the training dataset. Hence, these results are presented in the ‘All’ row.

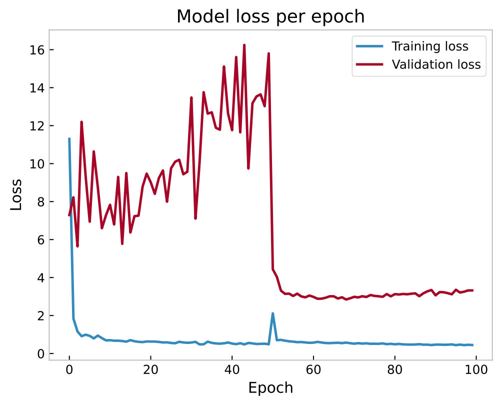
Table 6.8.: AUC scores (in %) of the experiments with ADAHS on test 1 with the English dataset. The scores for all anomaly classes and using both GloVe and fastText vectors are presented. AC = anomaly class. UN = unsupervised setting (no labelled samples, $\gamma_l = 0.00$). The normal setting is when $\gamma_n = 0.10$.

English dataset: Adding labelled training data												
	GloVe						**fastText**					
	UN	γ_n	γ_a		γ_l		**UN**	γ_n	γ_a		γ_l	
AC	0.00	0.10	0.05	0.10	0.05	0.10	0.00	0.10	0.05	0.10	0.05	0.10
1			92.7	93.2	93.0	93.5			95.2	94.7	**95.2**	95.1
2			87.8	88.2	87.3	86.6			88.0	88.2	87.5	**88.6**
3			91.6	92.4	92.3	92.4			93.2	**94.2**	94.1	93.9
4			80.2	73.3	78.7	59.6			**87.7**	82.9	83.8	82.7
5			92.9	93.3	93.2	93.3			94.7	**94.9**	94.7	94.8
6			89.1	88.4	88.9	87.8			89.4	90.3	89.7	**90.9**
All	60.9	62.6	93.4	93.5	93.4	93.1	53.5	50.0	94.7	95.1	95.1	**95.2**

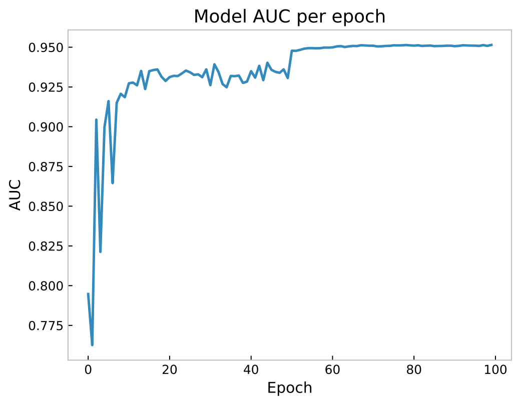
The highest AUC values for each anomaly class are highlighted. As can be observed from Table 6.8, all the best performing configurations were achieved using fastText. Furthermore, the model performs very poorly with no added labelled anomalous data, with the worst result corresponds to random guessing. However, the system performs significantly better by only adding little labelled anomalous data. One can also notice a decrease in performance for those anomaly classes that do not contain enough labelled samples, which is the case for class 2: severe toxic, 4: threat and 6: identity hate.

The overall best performing configuration was to include 5% labelled anomalous and normal samples ($\gamma_l = 0.05$), and only drawing anomalies from the toxic class (AC = 1) when using vectors from fastText. Then, the achieved AUC score was 95.21%. Two other configuration achieved almost identical score: when $\gamma_l = 0.10$ and AC = all, the achieved AUC score was 95.19%, and when $\gamma_a = 0.05$ and AC = 1 the AUC was 95.18%. For the best configuration, the training and validation loss, as well as validation AUC is presented in Figure 6.2. Usually, validation accuracy is calculated during training, but in this case it was replaced by the AUC score, because calculating accuracy involves setting a specific threshold for separating normal and anomalous samples.

As can be seen from Figure 6.2a, the training loss quickly reaches approximately one and continues to decrease slightly for the rest of the epochs. The only exception is the spike around epoch fifty. The validation loss on the other hand, increases and varies significantly for the first fifty epochs, before it stabilises at approximately three for the remaining epochs. It is worth noting that these losses are calculated based on the distance to the hypersphere centre, so a loss close to zero indicates that most of the normal samples are close to centre c, while the anomalous samples are not. The model's validation AUC



(a) Training and validation loss

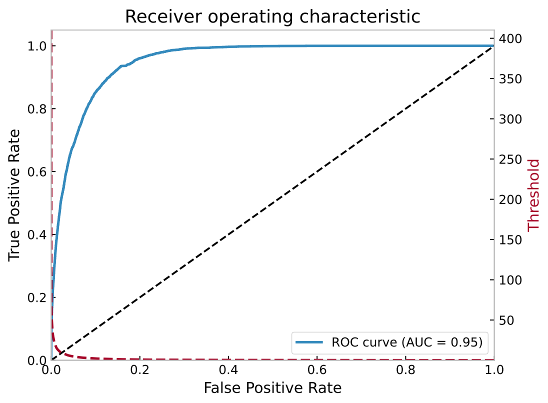


(b) Validation AUC

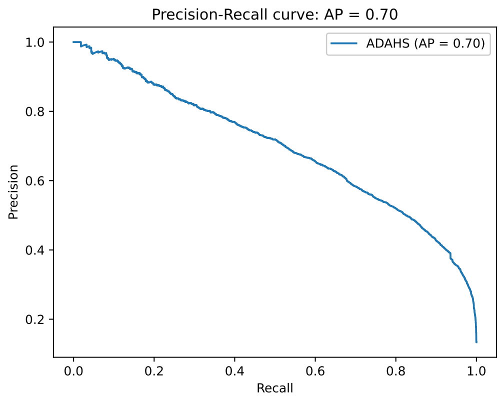
Figure 6.2.: Training and validation loss, as well as validation AUC for the English dataset.

values, showed in Figure 6.2b, increases drastically for the first ten epochs and continues to vary slightly until nearly epoch fifty five. After that the validation AUC stabilises at roughly 0.95 for the remaining epochs.

The ROC curve and P-R curve for the configuration are shown in Figure 6.3.



(a) ROC curve

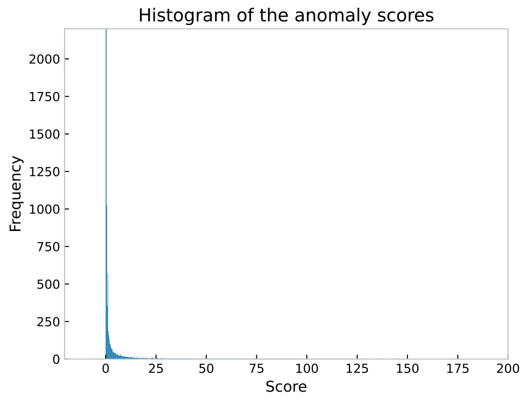


(b) Precision-recall curve and average precision

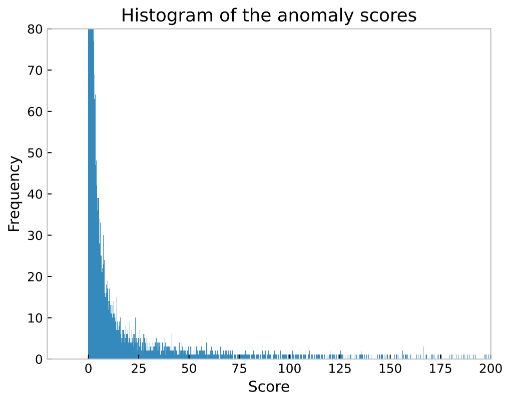
Figure 6.3.: The ROC curve and Precision-Recall curve with average precision (AP) for the English dataset.

Figure 6.3a shows the ROC curve with the AUC value equal to 0.95 and the thresholds used to calculate the true positive rates and false positive rates. The black dotted line is the baseline corresponding to random guesses. Figure 6.3b shows the P-R curve and the calculated average precision equal to 0.70. This value is the average of calculated precision values at different thresholds. As can be observed, the precision decreases with increased recall.

The anomaly score for each comment in the English test set was used to create histograms displaying the results. These can be found in Figure 6.4. Figure 6.4a shows the scores sorted in bins and the respective frequency of each bin, while Figure 6.4b presents a zoomed version of the same histogram.



(a) Histogram



(b) Zoomed histogram

Figure 6.4.: Histograms of the anomaly scores for the English dataset

As can be observed, most of the scores are close to 0. This is expected, because the majority of the data is normal data and model concentrates the normal data around centre c, and hence these data samples achieve a low anomaly score. In contrast, the data samples believed to be anomalous achieve a higher score. Based on the histogram, different frequency values were chosen and their corresponding scores were found. These scores represents different thresholds used to separate normal samples from anomalous samples. Based on the zoomed histogram in Figure 6.4b, four different frequency limits were set; limit $\in \{10, 20, 30, 40\}$. The anomaly score of the first bin with frequency lower than the limit is used as the threshold. Hence, all samples with score above this is deemed anomalous. Precision, recall, F_1-score and accuracy was calculated for each threshold. Table 6.9 displays the results for the thresholds corresponding to the limits, in an increasing order.

From Table 6.9 it can be observed that both precision, recall and F_1-score is generally very high for the neutral class, but lower for the hateful class. Whether recall or precision is higher depends on the threshold. Precision is better when $t = 8.243$, while recall is better for the other three thresholds. The highest recall is 78%, which means that a large part of the hateful comments are detected by the model. A smaller threshold involves deeming more comments to be anomalous, and hence, the chance of discovering an actual hateful comment is bigger.

Table 6.9.: For different thresholds (t), precision, recall and F_1-score are calculated for both the normal and anomalous class using the best performing configuration with the English dataset. Furthermore, accuracy for the entire dataset is calculated using the same thresholds.

Metrics		**Thresholds**			
		t = 8.243	t = 5.308	t = 4.330	t = 3.042
Recall	Neutral	0.97	0.96	0.95	0.93
	Hateful	0.55	0.66	0.70	0.78
Precision	Neutral	0.95	0.96	0.97	0.98
	Hateful	0.68	0.62	0.58	0.53
F_1-score	Neutral	0.96	0.96	0.96	0.95
	Hateful	0.61	0.64	0.64	0.63
Accuracy		0.93	0.93	0.92	0.91

Test 2

In the second test, the ratio of pollution in the training set, γ_p, is tested. All the experiments are conducted using $\gamma_n = \gamma_a = 0.05$. Table 6.10 presents the obtained results.

Table 6.10.: AUC scores (in %) of the experiments with ADAHS on test 2 with the English dataset. The scores for all anomaly classes and using both GloVe and fastText vectors are presented. AC = anomaly class.

English dataset: Polluted training data						
	GloVe			**fastText**		
AC	$\gamma_p = 0.00$	$\gamma_p = 0.01$	$\gamma_p = 0.05$	$\gamma_p = 0.00$	$\gamma_p = 0.01$	$\gamma_p = 0.05$
1	93.0	92.6	92.4	95.2	94.9	94.4
2	87.3	87.7	85.0	87.5	87.9	85.5
3	92.3	92.5	92.2	94.1	94.0	94.0
4	78.7	75.7	69.4	83.8	80.4	73.3
5	93.2	92.8	92.1	94.7	94.5	94.3
6	88.9	87.5	82.6	89.7	88.1	84.6
All	93.4	92.7	92.7	95.1	94.7	94.4

As can be observed from the table above, for almost every anomaly class, the model's performance has a tendency to decrease with increasing degree of pollution. The only exception is a small increase for AC = 2 and AC = 3 (with GloVe vectors) when 1% anomalies are added.

6.3.2. Results using the Norwegian dataset

The dataset is first split into a training, validation and test set. All the presented results are achieved during testing, on the test set that contains 8042 normal comments and 150 anomalous comments if the anomalies are drawn from {4, 5}. If the comments are drawn from {3, 4, 5} it contains 7732 normal and 460 anomalies. The first part of the experiments involved testing the hyperparameter η. Table 6.11 presents the achieved results from ADAHS using the Norwegian dataset. The experiments were conducted using $\gamma_n = \gamma_a = 0.05$ and $\gamma_p = 0.00$, only drawing anomalies from category 4 and 5. The fastText vectors with Norwegian Bokmål tokens were used and λ was set to 0.5×10^{-6} when experimenting with η. These experiments showed that the greatest result was achieved using $\eta = 10$.

Table 6.11.: AUC scores (in %) for different values of η using the Norwegian dataset.

η	AUC
0.1	61.6
1	67.1
5	70.4
10	71.3

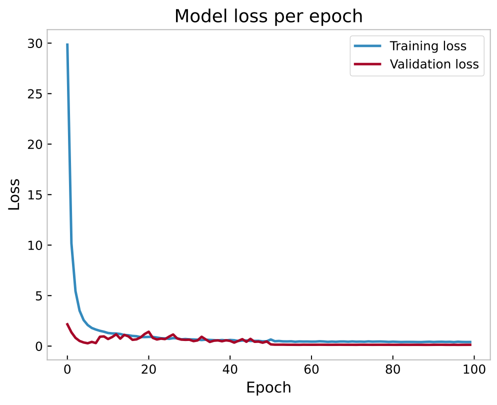
Test 1

The first test involves adding labelled samples. Since this dataset contains very little anomalous data, there is seldom enough samples to add 5% or 10% labelled anomalies. As explained, all anomalous samples are added, and the original amount of normal data is included. The result is thus using a smaller ratio of labelled anomalies. For this dataset, the total amount of hateful comments in category {4, 5} is 1.84%. Hence, this is the actual ratio of labelled anomalies added in each setup. When the anomalies consists of classes {3, 4, 5}, then 5.65% anomalies are added when the desired ratio is 10%, and 5% is added when the desired ratio is 5%. Table 6.12 presents the results for these setups. Again, γ_l represents labelled samples (both normal and anomalous).

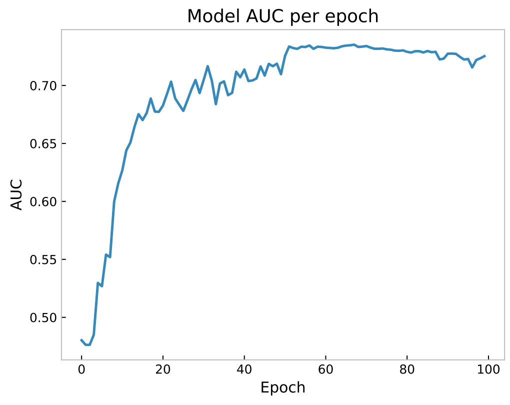
Table 6.12.: AUC scores (in %) of the experiments with ADAHS on test 1 with the Norwegian dataset. AC = anomaly classes. UN = unsupervised setting ($\gamma_l = 0.00$)

Norwegian dataset: Adding labelled training data						
	UN	γ_n	γ_a		γ_l	
AC	0.00	0.10	0.05	0.10	0.05	0.10
{4, 5}			74.4	74.4	73.7	**75.3**
{3, 4, 5}			76.8	**77.3**	75.6	75.8
None	51.2	54.5				

The highest AUC values are highlighted for each set of anomaly classes. As can be seen from Table 6.12, the model performs poorly when no labelled anomalous data is added. The method's best performing configuration with anomaly class {4, 5} is when the desired ratio of labelled neutral and hateful data samples are 10%, and the actual ratio of the anomalous samples is 1.84%. Then an AUC value of 75.3% was achieved. For this configuration the training and validation loss, as well as validation AUC is presented in Figure 6.5.



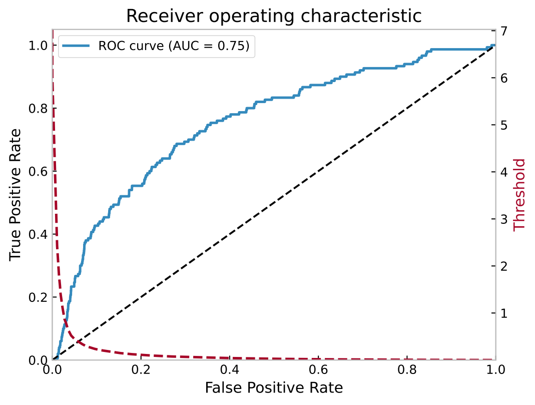
(a) Training and validation loss



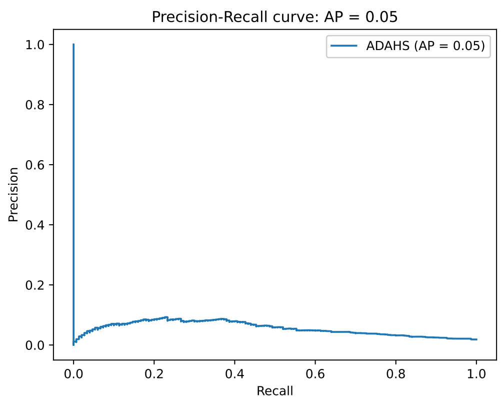
(b) Validation AUC

Figure 6.5.: Training and validation loss, as well as validation AUC for the Norwegian dataset.

As can be seen from Figure 6.5a, the training loss quickly reaches close to zero and the validation loss varies slightly before it stabilises close to zero. The model's validation AUCs, showed in Figure 6.5b, increases and reaches a top at approximately epoch sixty, before it slightly decreases. The ROC curve and P-R curve for the configuration are shown in Figure 6.6.



(a) ROC curve

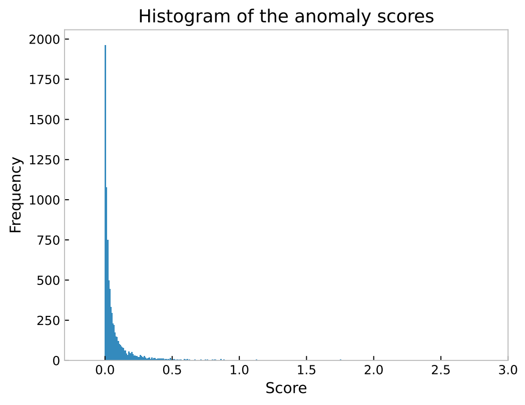


(b) Precision-recall curve and average precision

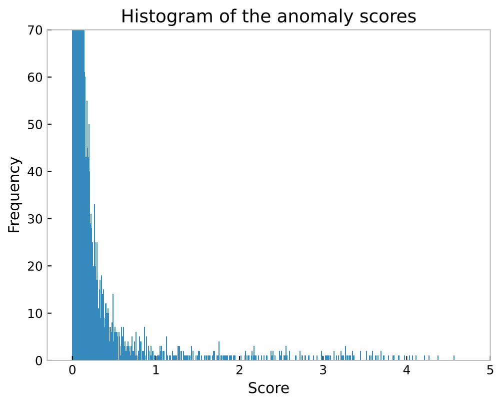
Figure 6.6.: The ROC curve and Precision-Recall curve with average precision (AP) for the Norwegian dataset.

Figure 6.6a shows the ROC curve with the AUC value equal to 0.75 and the thresholds used to calculate the true positive rates and false positive rates. Figure 6.6b shows the P-R curve and the calculated average precision. As can be observed, the precision drops instantly with increased recall, and continues to stay relatively low for all recall values.

The anomaly score for each comment in the Norwegian test set was used to create histograms displaying the results. These can be found in Figure 6.7. Figure 6.7a shows the scores sorted in bins and the respective frequency of each bin, while Figure 6.7b presents a zoomed version of the same histogram.



(a) Histogram



(b) Zoomed histogram

Figure 6.7.: Histograms of the anomaly scores for the Norwegian dataset

As can be observed, most of the scores are close to 0 because the majority of the samples are normal. Again, different frequency values were chosen, and their corresponding scores were found. Based on the zoomed histogram in Figure 6.7b, four different frequency limits were set; limit $\in \{20, 30, 40, 50\}$. Precision, recall, F_1-score and accuracy were calculated for each threshold. Table 6.13 displays the results for the thresholds corresponding to the limits, in increasing order.

From Table 6.13 it can be observed that both precision, recall and F_1-score is high for the neutral class, but low for the hateful class. However, the recall is significantly higher than precision for the hateful class. The highest recall is 49%, which means that almost half of the hateful comments are detected by the model.

Table 6.13.: For different thresholds (t), precision, recall and F$_1$-score are calculated for both the normal and anomalous class using the best performing configuration with the Norwegian dataset. Furthermore, accuracy for the entire dataset is calculated using the same thresholds.

		Thresholds			
Metrics		t = 0.242	t = 0.211	t = 0.164	t = 0.157
Recall	Neutral	0.91	0.89	0.86	0.85
	Hateful	0.42	0.44	0.49	0.49
Precision	Neutral	0.99	0.99	0.99	0.99
	Hateful	0.08	0.07	0.06	0.06
F$_1$-score	Neutral	0.94	0.94	0.92	0.92
	Hateful	0.13	0.12	0.11	0.11
Accuracy		0.90	0.88	0.85	0.85

Test 2

The ratio of pollution in the training set, γ_p, was tested. As for the English dataset, all the experiments are conducted using $\gamma_n = \gamma_a = 0.05$. Table 6.14 presents the obtained results.

Table 6.14.: AUC scores (in %) of the experiments with ADAHS on test 2 with the Norwegian dataset. AC = anomaly class.

Norwegian dataset: Polluted training data			
AC	$\gamma_p = 0.00$	$\gamma_p = 0.01$	$\gamma_p = 0.05$
{4, 5}	73.7	73.6	74.9
{3, 4, 5}	75.6	74.7	75.4

As can be observed from the table above, the model's performance decreases slightly when 1% pollution is added, but the performance increases with 5% pollution. When the anomalies are drawn from class 4 and 5, the increase in performance when adding 5% pollution surpasses the performance with no pollution. This is not the case when the anomalous samples includes the offensive comments, where the model still obtains the best results with no added pollution. However, both the decrease and increase are minimal, and the model obtains decent results in all setups.

6.3.3. Results from the baseline methods

Both baseline methods use a one-class classification setup with unsupervised learning. The neutral class is used as the normal class when experimenting with the English dataset, while category 1, 2 and 3 from the Norwegian dataset constitute the normal class when conducting experiments with this dataset. The remaining classes in both datasets are considered anomalous. Training is always performed using only the training data from the respective normal class. Following, testing is performed on the data samples from all classes, where samples from the normal class are labelled y = 0 and samples from all the remaining classes are labelled y = 1 for determining the AUC.

Table 6.15 presents the obtained results from experimenting with the OC-SVM method using both datasets. The highest achieved AUC score for both datasets is highlighted.

Table 6.15.: AUC scores (in %) for the experiments with OC-SVM on both the English and Norwegian dataset.

OC-SVM

	English dataset						**Norwegian dataset**		
	GloVe			**fastText**			**fastText**		
ν	**mean**	**tf-idf**	**max**	**mean**	**tf-idf**	**max**	**mean**	**tf-idf**	**max**
0.05	65.5	**67.6**	58.9	64.4	63.5	55.7	47.8	46.8	**50.4**
0.1	62.3	64.3	60.3	60.6	61.2	56.0	48.0	47.2	49.6
0.2	58.3	60.8	61.4	56.4	57.4	56.3	48.2	47.5	48.8
0.5	51.9	55.3	62.0	50.3	51.2	59.9	48.3	47.7	47.2

As can be seen from the table above, the highest achieved AUC score was 67.6% using the English dataset and 50.4% using the Norwegian dataset. For the best performing setup with the English dataset, tf-idf weighted mean was used for aggregating the word vector embeddings and hyperparameter $\nu = 0.05$. However, for the Norwegian dataset the best performing setup involved using max-pooling and $\nu = 0.05$. Overall, the results using the OC-SVM model are poor for both datasets.

Table 6.16 shows the results from the experiments with CVDD. The highest achieved AUC score for both datasets is highlighted.

Table 6.16.: AUC scores (in %) for the experiments with the CVDD model using both the English and Norwegian dataset.

CVDD			
Dataset	**Embedding**	**Attention heads**	**AUC**
English	GloVe	3	68.1
		5	**70.9**
		10	67.4
	fastText	3	69.5
		5	68.7
		10	66.9
Norwegian	fastText	3	**55.2**
		5	54.3
		10	50.7

As can be observed from Table 6.16, the best results are obtained using GloVe word embeddings with 5 attention heads/contexts for the English dataset, and using 3 attention heads for the Norwegian dataset. The results are then 70.9% and 55.2%, respectively. The results from CVDD are, on average, slightly better than those achieved using OC-SVM. However, they are still undoubtedly worse than the results achieved by ADAHS.

7. Evaluation and Discussion

The first part of this chapter interprets and evaluates the experiments and results presented in Chapter 6. The second part discusses the evaluated findings in relation to the research questions presented in Section 1.2.

7.1. Evaluation

The conducted experiments in this thesis were divided into three parts as presented in Section 6.1, and all the experiments were carried out by following the experimental plan and setup presented in Section 6.1 and 6.2. This section evaluates several aspects of the experiments and the results obtained. The evaluation starts with discussing general trends and observations before the experimental results for both datasets are examined.

7.1.1. General trends and observations

The first test in the second experimental part aims at investigating the effect of labelled samples on system performance. Furthermore, these experiments are also conducted to test the model's ability to handle novelties by only sampling labelled anomalies from a specific class. Generally, the system performs well on all anomaly classes, but the performance is lower for the classes that initially contained too few anomalies to add the 5% or 10% labelled anomalous data. A significant drawback to these experiments is that it is overlap between the anomaly classes, so even though only one class is included during training, the result is that comments which are also categorised as another class is added. This makes it difficult to determine if the system can, in fact, effectively handle novelties. More research would have to be conducted to determine if the system has this ability.

Neither of the experiments with either dataset are conducted using k-fold cross-validation, because of the amount of time each experiment would require. One experiment with the English dataset could take between four and ten hours, so using k-fold cross validation would nearly multiply this time with k. This thesis was interested in discovering a potential for using anomaly detection, and since the goal did not involve fine-tuning a model to compete with state-of-the-art solutions, using cross-validation was discarded.

Datasets

Often, it might provide valuable insight to compare the results when experimenting with two datasets. Nevertheless, in this case, when the datasets represent two different languages, there is possibly less information to draw from such a comparison. However, it might be possible to discover some similarities and trends, and additionally, it is most likely a rationale for differences. A machine learning model's performance is profoundly affected by both the amount and type of data.

The datasets differ in many ways, but an essential difference is their language. Having to handle two languages heavily affects the preprocessing step. For instance, dataset-specific tokens were removed from the respective dataset, and two dictionaries of common misspellings were created. Several of the misspellings were not just language-specific but also distinct for each dataset and was found by comparing the dataset's vocabulary and the word embeddings' vocabulary. After the preprocessing steps were applied, it was only discovered embeddings for 36.2% of the vocabulary for the English dataset, while for the Norwegian dataset it was found for 63.0%. However, this corresponds to 97.3% and 96.4% of all the text, respectively. This means that there are many words in both datasets that are very infrequent and does not match the terms in the pre-trained word embeddings' vocabulary. Examples are "dickhead", "noobs" and "omfg" for the English dataset and "resett", "svåret" and "forsvåret" for the Norwegian dataset.

Another notable difference is the dataset size. The English dataset consisted of 223 549 comments, while the Norwegian dataset consisted of only 41 139 comments. As described in Section 2.4, one of the main challenges with deep learning models is that they generally require a large amount of data to perform well. Hence, if ignoring all other differences between the datasets, better performance can still be expected when using the English dataset. Furthermore, another striking distinction between the datasets is their content. In the Norwegian dataset, the comments typically discuss the same topics, resulting in many similar words used in both the hateful and neutral comments. On the other hand, this is not the case for the English dataset. Here, the abusive comments contain more direct hate, profanities and cursing.

Results

Generally, the results using both datasets indicate that it is challenging to separate hateful language from neutral language. These difficulties were also remarked by several others that experimented with hate speech detection, such as Davidson et al. (2017) and Malmasi and Zampieri (2017).

The system achieves poor performance when not adding any labelled anomalies, using both datasets. As can be seen from the validation AUC plots from these experiments, presented in Appendix A, the AUC score varies extensively throughout all of the epochs and appears to be affected by the number of epochs only, leading to a random performance. However, the system experiences a significant increase in performance when adding a small amount of labelled anomalous data to the training set. The rest of this section evaluates the experimental results presented in Section 6.3 and is divided based on the two datasets to evaluate the results separately.

7.1.2. Results using the English dataset

This section evaluates the presented results from ADAHS using the English dataset.

Test 1

In Table 6.8, the results from test 1 are presented. The table presents the results when the number of normal samples is decreased to obtain the desired ratio of anomalous samples in each experiment. The results from the alternative approach, which involves not decreasing the number of normal samples, are presented in Appendix A. Table A.2 shows a comparison between the two approaches, and one can observe that most of the setups experience an increased performance by decreasing the number of normal samples. However, this is not the case for all setups. For instance, when using fastText and $\gamma_l = 0.05$ for anomaly class 2, the performance decreased from 88.2% to 87.5%. Furthermore, when using GloVe and anomalies from class 4, the performance decrease with 6.5% when normal samples are deducted. However, on average, for all three classes (2, 4 and 6), the performance increased with 1.1%. The increase in performance is particularly prominent when using fastText and class 4 anomalies, which experiences an average increase of 11.7%. Furthermore, in addition to the increased performance, decreasing normal samples also reduce the system's runtime, which is favourable.

Typically, the trend is that the performance increases with the amount of labelled anomalous data, but with some exceptions. The results also indicate that there is no or little effect of adding labelled normal samples. This can, for instance, be observed when considering the difference in performance between $\gamma_a = 0.10$ and $\gamma_l = 0.10$. In this case, the only difference is the added normal samples. In some of the cases, adding labelled normal data leads to a minor increase, whereas in other cases the results are equal or even lower. There is no obvious reason why the performance decrease when adding normal samples, but it is possibly due to a correspondence between the labelled normal comments and the hateful comments. These similarities can, for instance, be a discussion of the same topic. If this is the case, it becomes more demanding for the model to determine if new data instances related to this topic is neutral or hateful. Another possibility is that it caused by the random drawing of samples, and thus, different results could be obtained by changing the dataset splitting and random seed.

Furthermore, it can be observed an increasing performance when the anomalies are drawn from a pool containing all the hateful comments. This is expected because it eliminates the case of detecting novelties. As we can see, the results when drawing anomalies from class 1 (toxic) only, are almost identical to the results obtained when the anomalies are drawn from a pool of all samples. Class 1 is the largest class of anomalies with 15 294 samples in the training set. As pointed out earlier, there is an overlap between the anomaly classes, and it can thus be expected that this class contains overlap to all of the other classes since it is the biggest. As discussed, the overlap between different anomaly classes makes it difficult to determine if the system can effectively handle novelties. Hence, it was determined to conduct test 1 on the system without any overlap between the classes. This experiment was carried out by isolating all the comments that were only labelled with one anomaly class. The number of non-overlapping samples in class 1 to 6 was 5666, 0, 317, 22, 301, 54, respectively. Thus, only class 1 (toxic) had a sufficient amount of samples and was the only class used in the experiment. These 5666 comments were added as labelled anomalies, and the number of normal samples was decreased to obtain $\gamma_a = 0.05$. In this case, the system is tested on all anomaly classes, but only non-overlapping samples from class 1 are included during training. Hence, all anomalies from class 2 to 6 are novelties. The results showed an AUC score of 94.0%, which is only 1.2% lower than when the overlapping samples are included. This shows that the system achieves good performance even when tested on novel samples. Nevertheless, this is only tested using one anomaly class, and might not provide similar results when tested on other novelties.

All the best performing configurations for each anomaly class are achieved when using the pre-trained embeddings from fastText . This might be caused by the number of words in the vocabularies because fastText contains vectors for more words than GloVe 6B, which might have spiked performance. It was desired to test this theory, and thus the dataset's vocabulary was compared to fastText's vocabulary to check if the coverage was bigger than with GloVe. Only 36.2% of the vocabulary matched pre-trained vectors in GloVe 6B, but 55.9% matched vectors in fastText, which is an increase of 19.7%. Hence, the size of the pre-trained vectors appears to have a large effect, so utilising GloVe 840B instead of 6B will most likely lead to an even more considerable increase.

The most noticeable result is when GloVe vectors are used, and 10% normal and anomalous data are added from anomaly class 4. In this case, the method only achieves an AUC score of 59.6%, which is much lower than for all other configurations. When the amount of normal data is not deducted, the same configuration achieved an AUC score of 80.9%, which is an increase of 21.3%. Class 4 only contains 478 anomalous samples, and thus to obtain the correct ratio, approximately 4.3k unlabelled neutral comment can be added, along with another 478 labelled neutral samples. Since only a small amount of data are used in this case and deep learning models usually require a large amount of data to perform well, a lower performance can be expected. However, the results for this configuration are significantly lower than the results obtained when only 10% anomalous data was added. The only difference between these two setups is the addition of 10% labelled normal samples, and there is no obvious reason why this would result in such a bad performance. Furthermore, when using fastText with the same configuration, the results are 23.1% better. Hence, as described earlier, the poor performance might be caused by the random drawing of samples.

Loss and AUC plots

Figure 6.2 shows the training and validation loss, as well as the validation AUC scores for each training epoch. As can be seen from Figure 6.2a, the training loss reaches close to one at approximately epoch two. On the other hand, the validation loss varies drastically in the "searching"-phase (first fifty epochs) but then stabilises close to three for the remaining epochs, i.e. in the "fine-tuning" phase. As mentioned, it is expected to have a loss close to zero. Here, the validation loss is consistently higher than the training loss for all epochs. This can indicate that the model have learned some patterns in the training data, that is not present in the validation data. A validation loss that is significantly larger than the training loss often means that the model is overfitting. However, the difference between the losses is relatively small, which does not necessarily mean overfitting.

The model's validation AUC scores showed in Figure 6.2b increases drastically and reaches a top at approximately epoch fifty-five. This behaviour is expected since the model improves during training. The first fifty epochs are the "searching"-phase where the learning rate is higher in order to traverse quickly from the initial parameters to a range of more decent parameter values. The last fifty epochs are the "fine-tuning"-phase where the learning rate is lower to explore the deeper parts of the loss function. This can be observed from the figure since the AUC change significantly more in the searching phase than in the fine-tuning phase.

ROC and Precision-Recall curves

Figure 6.3a shows the ROC curve with the corresponding threshold curve. The AUC value of 0.95 indicates a good degree of separability between the normal and anomalous classes, i.e. the model is capable of distinguishing between classes to a great extent. Figure 6.3b shows the Precision-Recall curve (PRC) and the calculated average precision. As can be observed, the precision decreases evenly with increased recall. For a non-perfect model, this is expected because as recall increases and more samples are deemed anomalous, more normal samples are also misclassified, leading to lower precision. While the baseline is fixed with ROC, the baseline of PRC is determined by the ratio of positives (P) and negatives (N) as $y = P / (P + N)$, to account for imbalanced datasets. In this case, the baseline is $y = 4158 /(4158 + 38\,466) = 0.0976$. This value is so low because the dataset is very imbalanced. The average precision is usually equivalent to the area under the P-R curve, and they are identical in this experiment, both equal to 0.70. Hence, the average precision is substantially better than the baseline.

Classification results

Table 6.9 displays the calculated precision, recall and F_1-scores for different thresholds. Generally, all scores are very high for the neutral class but lower for the hateful class. The recall varies between 0.55 and 0.78, while precision varies between 0.53 and 0.68, depending on the chosen threshold. The trend is that recall for the hateful class increases with decreasing threshold, while precision increases with an increasing threshold. This is expected because lowering the threshold involves including more samples to the predicted anomaly class. Higher and lower scores for both precision and recall may be achieved by setting the threshold higher or lower than the four values present as a part of the experimental results. The best recall value is 78%, which means that the model detects a significant part of the hateful comments. To further increase recall for the hateful class, the threshold has to be set even lower.

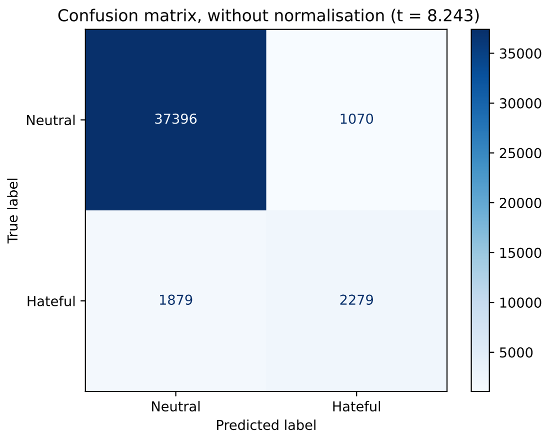
The confusion matrices for the highest and lowest threshold used for the English dataset are presented in Figure 7.1. The remaining two confusion matrices for limit 20 and 30 can be found in Figure A.3 in Appendix A.

Based on the confusion matrices, it can be seen that the model misclassifies a few samples, both hateful and neutral. The number of misclassified comments depends on the chosen threshold, where setting a lower threshold involves correctly classifying more hateful comments, but also misclassify more normal samples. In the case where t = 3.042, the model correctly classifies 3247 hateful samples, which is 23.3% more than when t = 8.243. This happens at the expense of the number of correctly classified normal samples, which has decreased by 0.05%. This percentage decrease is so little because there as so much more normal than anomalous data.

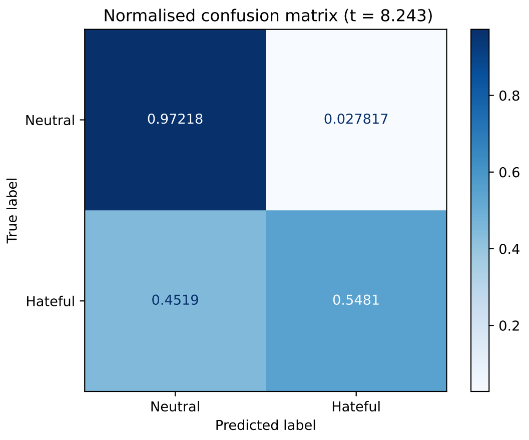
For each setup, the fifty comments with highest and lowest anomaly score are found. In this case, all the fifty comments with the highest anomaly score are actual hateful comments, and all the comments with the lowest anomaly score are normal. This means that at least the top and bottom fifty are classified correctly. Out of all the top fifty comments, there is not one comment that does not contain profanities. Words such as "fuck", "asshole", "slut" and "cunt" are used in almost every comment. Following are three of the top fifty anomalous comments:

1. go suck a dick you faggot ass lame pussy fuck wikipedia
2. fuck you fuck you i hope your family dies and you die and your brother and sisters die and i hope you die i hope you get aids and die and get your ass fucked i hope you die again and sucks satans cock you dirty bitch nigga nigger ufck you cocksucker fuck you fucking bitch fuck you fuck you from someone you fucked you dirty piece of shit
3. you are a huge massive rapist cock sucking faggot

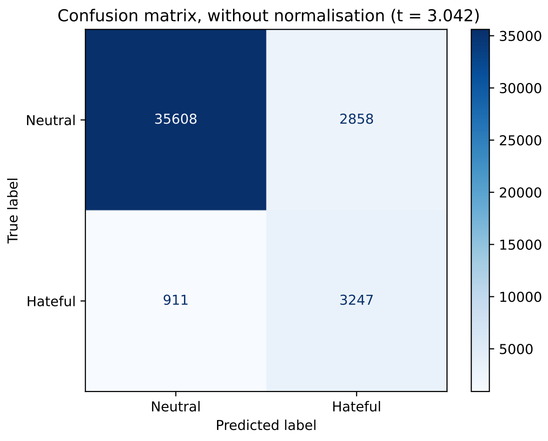
As can be observed, these comments all contain much obscenity and swearing.



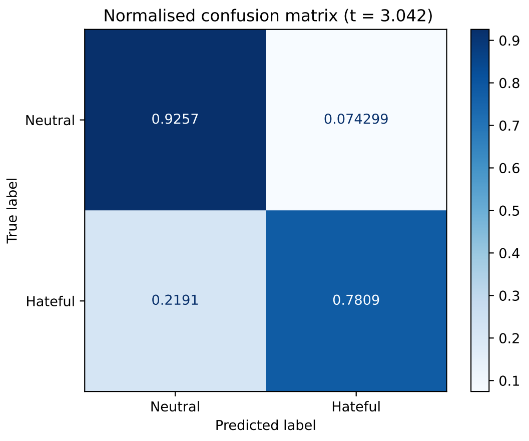
(a) Confusion matrix with t = 8.243



(b) Normalised confusion matrix with t = 8.243



(c) Confusion matrix with t = 3.042



(d) Normalised confusion matrix with t = 3.042

Figure 7.1.: Confusion matrices for the English dataset. (a) and (b) are found by setting the frequency limit to 10, and hence, threshold t = 8.243. On the other hand, (c) and (d) are found by setting the frequency limit to 40, and hence threshold t = 3.042. (b) and (d) contains normalised predictions.

Furthermore, it was found that the first comment the model misclassified is ranked as the 77th most anomalous comment. The comment is very long, so only an excerpt is presented here. The comment is as follows:
"Drew Pickles grew up as a gay child. When he was just born his cock was five feet long it grew much bigger every second. When he was very young Drew liked to masturbate to the wiggles, spongebob and dragon tales. His dick got very hard and much longer. [...] Drew skipped lots of school because he wanted to stick dildos up his ass and poop on his grandpa all day. [...]"

The comment is not considered to be hateful but can be perceived as offensive or provocative. It contains many words that are often used in the hateful comments, such as "gay", "cock" and "dick", which might be the reason why the model believes it to be hateful.

Test 2

Table 6.10 shows the results from the second test, which was adding unlabelled anomalous samples to the training data, i.e. increasing level of pollution. As stated, the model's performance has a tendency to decrease with increasing degree of pollution. The only exception is a small increase for AC = 2 and AC = 3 (using GloVe) when 1% anomalies are added. Even though the majority decrease when anomalies are added, the system still obtains adequate results and the difference in performance is almost insignificant. The AUC values for anomaly classes 2, 4 and 6, which contains a decreased amount of normal data, decrease more than the other anomaly classes. Hence, it appears as if the system is more prone to pollution when a smaller amount of data is added. However, adding unlabelled anomalies only has a minor effect on performance and indicates that the model is reasonably robust against pollution.

Comparison with baseline methods

The best performing setup with OC-SVM was using tf-idf weighted mean for aggregating the word vector embeddings and setting hyperparameter $\nu = 0.05$. Then an AUC score of 67.6% was achieved, as presented in Table 6.15, which is considered a relatively poor or decent performance. Using the CVDD model, the maximum achieved AUC score was 70.9%, as found in Table 6.16. This was attained when using five attention heads/contexts and the GloVe word embeddings. The results are slightly better than the results achieved by the OC-SVM but are still significantly worse than the best performing setup with ADAHS, which attained an AUC score of 95.2%. A possible reason why the baseline methods have poor performance is that their parameters are not optimised. According to Aggarwal (2017), the main challenge associated with support-vector machines is that they can be sensitive to the choice of the kernels and the method's many hidden parameters. Furthermore, they state that different detectors require the identification of different hyperparameters. This may lead to challenges for comparison purposes. The range of parameters used in the OC-SVM is very different from the parameters used in CVDD. Furthermore, they both differ from the parameters in ADAHS. Achieving maximum performance for all the models requires that all their parameters must be optimised. Since this is not conducted, it can further complicate a proper understanding of their relative performance. Nevertheless, the results obtained from ADAHS are considerably better than the results from the baseline methods, which may indicate that ADAHS, despite optimisation, will still be the best performing method.

7.1.3. Results using the Norwegian dataset

This section evaluates the presented results from ADAHS using the Norwegian dataset.

Test 1

In Table 6.12, the results from test 1 are presented. As previously stated, the actual ratio of labelled anomalies in the experiments with anomalies from {4, 5} (case 1) is 1.84%. As a result, the only difference between the configurations is the number of labelled normal samples. This involves that the two setups, where only anomalous data is added ($\gamma_a =$ 0.05 or 0.10), are equal in this case.

Moreover, the method performs poorly using this dataset when no anomalous data is added but experiences an increased performance when adding a small number of labelled anomalies. Similarly to when the English dataset is used, the overall trend is that the performance increases with increasing ratio of labelled anomalies. The results also indicate that there is a minor effect of adding labelled normal samples. For instance, when adding anomalies from {4, 5}, the performance increase with 1.6% when increasing the amount of labelled normal data from 5% to 10%. This increase is independent of labelled anomalies because both configurations contain the same amount of labelled anomalous data, as described earlier.

It can also be observed an increasing performance when the offensive comments (category 3) is added to the set of anomalous samples. This is expected because adding the offensive comments as anomalies involves not having to separate between offensive and hateful utterances. As discussed in Section 2.1, it is often challenging to distinguish between offensive and hateful content, and as briefly described in Section 3.2, several existing methods struggle with this distinction. Nevertheless, the increase in performance is less than expected, with an increase of only 2% for the best configurations in the two cases. This suggests that the system is either bad at separating offensive and hateful content from neutral and provocative content (normal class), or relatively good at distinguishing hateful from all other content. However, the achieved results are decent but show that the method struggles to separate the normal and anomalous content. Hence, this supports the first suggestion. Furthermore, it is difficult to determine if this increase is just a result of adding more labelled samples. In the case where class 3 is considered an anomalous class, 5.65% anomalous samples are added, in contrast to 1.84% for the case where this class is not considered anomalous.

Interestingly, for case 2, where the anomalies are drawn from {3, 4, 5}, it can be observed from Table 6.12 that the system performs better when only anomalous data is added. When the desired ratio of anomalies is 10%, and the actual ratio is 5.65%, the system achieves an AUC score of 77.3% when $\gamma_a = 0.10$ and 75.8% when $\gamma_l = 0.10$. As also discussed for the results using the English dataset, there is no apparent reason why this is the case, but it might be due to similarities between the labelled normal and hateful data, such as the discussion of related topics.

As discussed, the presented results involve adding a smaller ratio of labelled samples than desired, because of too few hateful samples in the dataset. The results from test 1 and 2 presented in Table 6.12 and Table 6.14 are compared to the results when using the alternative approach (decreasing amount of normal data), which is presented in Table A.4 and Table A.5 in Appendix A. Table A.6 shows a comparison of the results from both approaches. From this table, it becomes clear that the performance for all configurations increase when the original amount of neutral samples are included. Furthermore, these results are, on average, 7% better than if the number of normal samples is decreased. The increase in performance is particularly prominent for the setup where it is desirable to add 10% labelled anomalies from {4, 5} and normal data. Furthermore, the two unquestionably largest increases were when 5% pollution was added in test 2, where the performance is 24.3% and 21.9% better when including all normal data. This indicates that the method is much more prone to pollution when only a small amount of data is used. For the rest of the configurations, the increase is much smaller.

Loss and AUC plots

Figure 6.5 shows the training and validation loss, as well as the validation AUC scores for each training epoch. As can be seen from Figure 6.5a, the training loss reaches close to zero at approximately epoch ten. On the other hand, the validation loss varies slightly in the searching-phase but then stabilises close to zero in the fine-tuning phase. The validation loss is consistently lower than the training loss for the last fifty epochs. The experiments with this dataset should thus also be conducted with more training because as long as validation loss is lower than or even equal to training loss one should keep doing more training. When the validation loss is slightly higher than the training loss, the training should be completed. Furthermore, there are several possible reasons why the validation loss is slightly lower than the training loss. One possible reason is that regularisation (dropout) is applied during training, but not during validation and testing. Another possible reason is that the validation set may be "easier" than the training set, which can happen by chance if the validation set is too small. Since the validation loss is not smaller than the training loss when using the English dataset, there is a possibility that it is caused by the small size of the Norwegian dataset.

The method's validation AUC scores showed in Figure 6.5b, increases and reaches a top at approximately epoch sixty, before it slightly decreases. The AUC varies from approximately epoch twenty to fifty-five before it becomes relatively stable with a small decreasing tendency from about 0.72 to 0.70 in AUC score.

ROC and Precision-Recall curves

Figure 6.6a shows the ROC curve with the corresponding threshold curve. The AUC value of 0.75 indicates a decent degree of separability between the neutral and anomalous classes, i.e. the method is capable of distinguishing between classes to some extent. Figure

6.6b shows the Precision-Recall curve (PRC) and the calculated average precision. As can be observed, the precision drops instantly with increased recall and continues to stay relatively low for all recall values. This is presumably because the system misinterprets many neutral comments for being anomalous. In this case, where the anomalies are drawn from class 4 and 5, the baseline is y = 150 /(150 + 8042) = 0.018. The average precision is equivalent to the area under the P-R curve, both equal to 0.05. So even though the average precision is very low (0.05), it is still better than the baseline. However, based on this plot, it appears as if the system is a weak classifier because it has to take a substantial hit in precision to get a higher recall.

Classification results

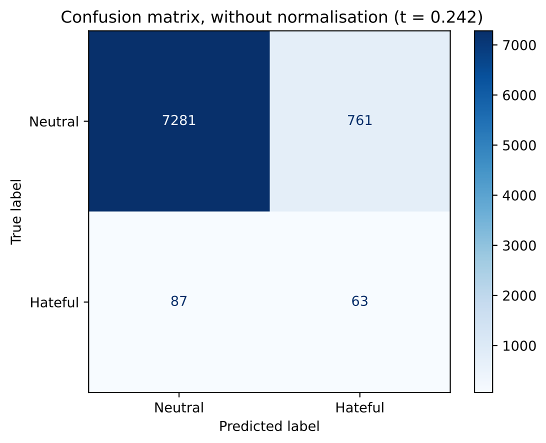
Table 6.13 displays the calculated precision, recall and F_1-scores for different thresholds. All scores for the hateful class is considered relatively poor, but the recall is significantly higher than precision. The highest recall is 49%, which means that the system detects almost half of the hateful comments. A way to increase recall for the hateful class is to set the threshold even lower. As a consequence, precision will decrease because even more normal samples would be deemed anomalous. If it is desirable to increase prediction, the threshold would have to be increased.

The confusion matrices for the highest and lowest threshold used for the Norwegian dataset are presented in Figure 7.2. The remaining two confusion matrices for limit 30 and 40 can be found in Figure A.4 in Appendix A.

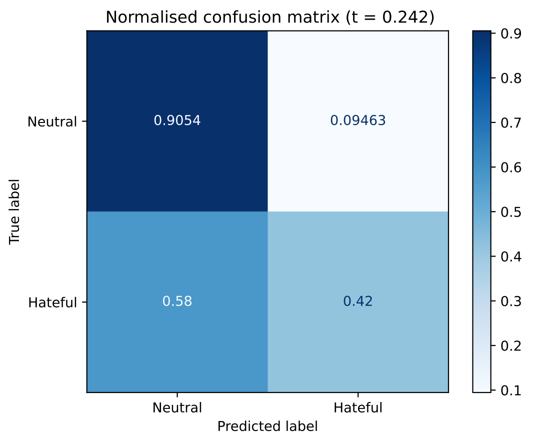
Based on the confusion matrices, it can be determined that the model misclassifies a significant number of samples, both hateful and neutral. For each setup, the fifty comments with the highest anomaly score are found. Following are some of the comments the model have given high anomaly score, and their actual label:

1. Enig. Ta en dna test og send dem på huet og ræva tilbake det de hører hjemme
   Actual label: 5
2. Og bra er det. Mer falske og dobbelmoralske mennesker enn kristne finnes ikke! De kristne kan ta med seg muslimene og reise hjem til midtøsten der de hører hjemme
   Actual label: 3
3. I dommen mot sian nå så spiller det visst nok ikke noen rolle om det er mot muslimen, enkeltmenneske eller voldsideologien islam
   Actual label: 2
4. Snart klart for yrkesjobbmesse nav jobbmesse mosjøenvgs nordland mosjøen utdanning velgyrkesfag yrkesfag sensio smarthus googlehome eaton appstyring stemmestyring
   Actual label: 1

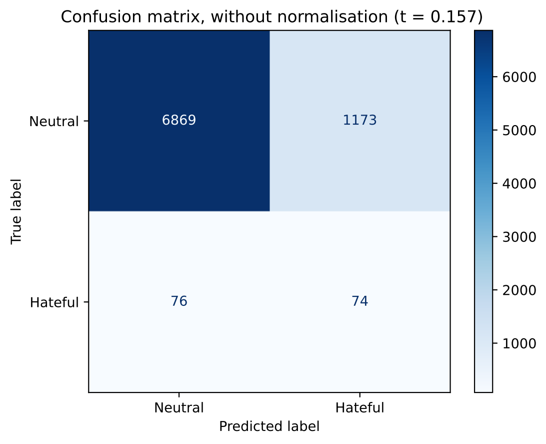
The model successfully categorises comment (1), even though the comment does not explicitly state who the subject is. Evaluating comment (2) shows that the subjective



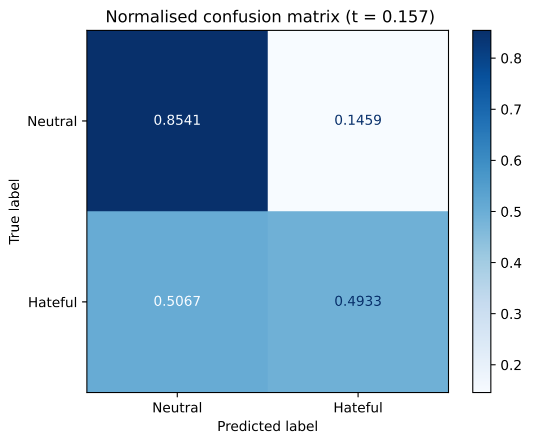
(a) Confusion matrix with t = 0.242



(b) Normalised confusion matrix with t = 0.242



(c) Confusion matrix with t = 0.157



(d) Normalised confusion matrix with t = 0.157

Figure 7.2.: Confusion matrices for the Norwegian dataset. **(a)** and **(b)** are found by setting the frequency limit to 20, and hence, threshold t = 0.242. On the other hand, **(c)** and **(d)** are found by setting the frequency limit to 50, and hence threshold t = 0.157. **(b)** and **(d)** contains normalised predictions.

interpretation of hate speech is a significant issue. This comment has many similarities to comment (1), where both statements might be perceived as both stigmatising and derogatory by affecting the dignity and status of a specific group based on various characteristics, which is a part of the hate speech definition used in this thesis, as presented in Section 2.1. Hence, both comments fit the description of hate speech, but the latter was only regarded as offensive by the person annotating it. In comment (3), the terms "muslimen" and "voldsideologien islam" are mentioned. Many of the hateful comments contain hatred towards Muslims, and since this comment mentions Muslims and refers to Islam as an ideology based on violence, it is understandable that the system believes the statement to be hateful, even though it is not. Comment (4) contains many words that are most likely hashtags. Most of the words are related explicitly to job fairs and education, and these words are probably neither well represented in the dataset nor included in the pre-trained embeddings' vocabulary. Most likely, this comment is not similar to other comments in the dataset, and many of its words do not have pre-trained embeddings. Thus, it can be expected that the model fails to consider this as a normal data sample. This indicates that it is crucial to include enough normal data of various topics in order to successfully build a comprehensive model for data normality.

For each setup, the samples considered to be most normal are also found. A wrongfully predicted comment is "ville kanskje vurdert å sterilisere meg snart om jeg var deg, men det er jeg jo ikke" which has 5 as the actual label. First of all, one may argue that this comment does not provide enough information to be categorised as hateful because it does not state why the statement's receiver should sterilise. Second, if the comment is considered hateful, this type of hate is difficult for the model to identify because it does not contain any words that are explicitly hateful or offensive.

By evaluating all the most anomalous and most normal samples provided by the model, it was found that most of the returned anomalous samples were actually normal. This is very different from using the English dataset, where all top fifty anomalous samples were, in fact, hateful. The model performs considerably worst when tested on this dataset.

Analysing the misclassifications provided insights into what the model has a tendency to misinterpret and flaws in the annotated dataset. To compare, the following comments are misclassified by the model when offensive comments also are considered anomalous:

1. Fuck jøder er heilt bort i natta å seie men straffbart, trur ikke det. Hadde nokon derimot kalt ein gitt jøde jævla kakerlakk hadde det vore annleis. Det var uttrykk hitler brukte om jødar, han kalla dei skadedyr som måtte utryddast. Det er for meg over streken.
   Actual label: 1
2. Kom deg av doen din fitte
   Actual label: 2
3. Noe som er veldig rart er at om du står imot totalitære regimer som kalles religion islam, deres dreping, vold, voldtekter, pedofili, jødehat osv, så blir du kalt høyreekstrem eller rasist selv av pst. Der må det arbeide mange uvitende mennesker.
   Actual label: 2

Comment (1) contains several terms that are used in many of the hateful and offensive comments, such as "fuck" and "jævla". There are also several combinations of words (captured by the n-grams) that appears hateful, such as "Fuck jøder" and "jøde jævla kakkerlakk" that does most likely not fit into the model's learned normal representation. Furthermore, it can be observed that the comment is written in Norwegian Nynorsk, which involves using many words not included in the pre-trained word embeddings' vocabulary. All of this creates probable cause for the model to believe this comment to be anomalous. In comment (2) the use of the word "fitte" is probably why the model believes it to be hateful. However, even though the comment contains a profane word, it does not make it hateful. Again, weaknesses with the dataset annotation are shown when evaluating comment (3). It is difficult to determine whether or not this comment is hateful since the purpose of the comment is to downplay PST (the Norwegian police security service) and their actions. However, to express this opinion, the author utters a hatred towards Muslims by announcing that they are committing murder, rape and paedophilia. The person who has annotated this comment did not consider this to be categorised as hateful, probably because the hatred is not entirely explicit.

The misclassifications when the offensive comments are considered hateful looks similar to the misclassifications presented earlier. It looks like the model wrongfully predicts the same type of comments in both cases.

Test 2

Table 6.14 shows the results from the second test, which was adding unlabelled anomalous samples to the training data, i.e. increasing the level of pollution. As stated, the model's performance decreases slightly when 1% pollution is added but increases with 5% pollution. There is no apparent reason why the model experiences this increase, and it might be caused by the random drawing of anomalies. The random seed is equal in all the experiments, and changing this seed, which will create a different dataset split and a different random drawing of anomalous samples, might provide different results. However, the increasing and decreasing performance is relatively insufficient and indicates that the method is reasonably robust against pollution.

Comparison with baseline methods

The best performing setup with OC-SVM was using max-pooling for aggregating the word vector embeddings and setting hyperparameter $\nu = 0.05$. Then, an AUC score of 50.4% was achieved, as presented in Table 6.15. This is a poor performance, and it is approximately equal to a random guess. Using the CVDD model, the maximum achieved AUC score was 55.2%, as found in Table 6.16. This was attained when using three attention heads/contexts. The results are slightly better than the results achieved by the OC-SVM, but they are still significantly worse than the best performing setup with ADAHS, which attained an AUC score of 75.3%. As explained in Section 7.1.2, the baseline's parameters were not optimised, which have probably lead to worse performance.

7.2. Discussion

In this section, the findings of this thesis are discussed in relation to the research questions and overall goal presented in Section 1.2. The experimental results were thoroughly reviewed in Section 7.1 and are thus not revisited in this section. However, the overall performance of the system is discussed further. Since several aspects of the implemented method must be considered and discussed to determine if the proposed method can be useful in detecting hate speech, this section includes a discussion of the advantages, disadvantages and challenges of such a method compared to conventional classification methods, as well as possible improvements. The last section discusses the findings in terms of the proposed research questions.

7.2.1. Overall performance

When the experimental results were evaluated, it was stated that the method achieves poor performance using both datasets when no labelled anomalous data is included in the training set. Hence, the method is unsuited to handle unsupervised learning. In contrast, Ruff et al. (2020) still achieves good results when no anomalous data is added. The difference in performance with and without the inclusion of anomalous data is significantly smaller than for the experimental results obtained in this thesis. A possible reason is due to their use of an autoencoder to pre-train their neural network weights. They state that this pre-training follows the Infomax principle, because autoencoders implicitly maximise the mutual information, and thus minimise the entropy in the latent distribution. On the other hand, the implemented method in this thesis uses *Kaiming He* initialisation (He et al., 2015) on the network weights, which helps in attaining a global minimum of the cost function more efficiently. Both of these initialisation methods prevents the vanishing and exploding gradient problem and results in faster convergence. Using pre-training is not explored as a part of this implementation, but might improve performance and should thus be tested.

Another potential cause of poor performance in the unsupervised and normal setting is the use of network bias terms. When the network includes bias terms, it is for the unsupervised and normal setting possible to achieve a trivial, uninformative solution which involves that the neural network converges to a constant function (Ruff et al., 2020). The reason is that bias terms can learn a constant function that is independent of the input, which can map directly to the hypersphere centre and thus create a trivial solution. However, this does not happen for the semi-supervised setting as long as there are sufficiently many labelled anomalies, which is due to the difference in the loss calculations. Hence, the poor and presumably random performance in these settings can be caused by network bias terms. To test if this was the actual cause, the experiments for the unsupervised and normal setting was conducted again, this time without network biases. The resulting AUC scores using both GloVe and fastText vectors are presented in Table A.3 in Appendix A.1. The AUC scores for the unsupervised setting was 57.1% and 55.9%, and the scores for the normal setting was 56.5% and 53.7%. This means that both scores decreased when bias terms were removed using GloVe, whereas both scores increased when using fastText. In both cases, the results are close to the achieved performance when using the OC-SVM baseline model. The validation AUC plots that can also be found in Appendix A.1, reveals that the AUC is significantly more stable throughout all of the epochs when bias terms were removed. On the other hand, when bias terms were included, the performance appeared random depending on the number of epochs only. Either way, the results are insufficient, and the method can still not correctly handle the unsupervised and normal setting.

It is interesting to notice that decreasing the amount of normal data in the English dataset on average increased performance, but decreased performance when conducted on the Norwegian dataset. A possible reason is that the Norwegian dataset is much smaller. When adding 10% anomalous samples, this is equivalent to only adding 522 labelled anomalous samples. If one were to decrease the number of normal samples and obtain the correct ratio, only approximately 4.1k unlabelled neutral comment can be added, along with another 522 labelled neutral samples. The amount of available training data for the model is in this setup significantly decreased, and it appears as if this amount is not enough for the model to learn trends and patterns. This is in line with the results obtained for anomaly class 4 in the English dataset, which only contains 478 anomalous samples. In this case, decreasing the amount of normal data also decreased performance. However, the other anomaly classes in the English dataset that contained a small number of anomalies experienced increased performance when normal samples were deducted. It is thus difficult to determine why the performance for the English dataset is so much higher in these cases, but it might be due to the dataset differences, such as language and content.

After reviewing many comments from both datasets, it becomes clear that the English dataset contains a lot more profanities. The hateful comments encompass a large variety of abuse words that are barely or never present in the neutral comments. On the other hand, both the neutral and hateful Norwegian comments discuss the same topics and use many similar words. Furthermore, there are a lot less offensive terms used in the Norwegian dataset, which is possibly a part of the reason why the model performs so much better on the English dataset. It is easier to correctly determine the anomalies when the model can rely on the frequency of offensive or profane words to separate these instances from the normal class.

Generally, the system's performance on the Norwegian dataset is much lower than on the English dataset, and the method does not currently perform sufficiently to be considered a viable detection method using Norwegian comments. It shows potential, and with further research and fine-tuning, it may be possible to achieve a system that can at least find the major part of the hateful comments. Furthermore, it is reasonable to believe that the system will perform better with a larger dataset. In this way, the model may be able to better capture the nature of normal samples and be better equipped to handle the issue of representing normality. The applied preprocessing steps account for the Norwegian language, but does not explicitly handle challenges such as compound words and Norwegian Nynorsk. For a system to be able to achieve satisfactory performance in the Norwegian language, more research on how to most effectively conduct the preprocessing must be completed.

The performance of the method is mainly evaluated using the AUC score from the ROC curve. For a hate speech detection system, one would first of all like to know the sensitivity (= recall) to be sure that the test identifies the vast majority of hateful comments as hateful. Both the ROC curve and the P-R curve involves recall, but the ROC curve also includes specificity (FPR), while the P-R curve includes precision. Both plots are included to provide two different views of the experimental results. Saito and Rehmsmeier (2015) found that the ROC curves are the most commonly used evaluation method with imbalanced data. However, the authors concluded that changing the primary evaluation method from ROC to PRC may influence many studies. The reason is that a ROC curve might provide an overly optimistic view of an imbalanced dataset. In this case, if one is adding many negatives (neutral comments), which achieves a low anomaly score, it may result in a significant improvement of the ROC curve without improving the sensitivity. On the other hand, a PRC is not impacted by this addition. Therefore, since all the experiments are evaluated using AUC from the ROC curve, it is not guaranteed that the results do not appear better than they are. Hence, the PRC should be evaluated for all test results as a supplement to get the full picture. The early retrieval area (first 25% of the FPR axis), which is a region with higher specificity values, may also be used because this is less affected by the imbalance. One can, for instance, calculate the area under the curve for only this region, which is especially useful for comparison reasons.

The ultimate goal of automatic hate speech detection is to be able to implement a system that can find all hateful comments, without misjudging neutral data for being hateful. Hence, such a system would achieve both precision and recall equal to 1. This is a complicated task, as previously discussed. Since this might be impossible, other approaches to automatic detection systems may be explored. The method implemented in this thesis is probably best suited to flag hateful comments for manual moderation because it does not achieve satisfactory precision, regardless of threshold. However, if the threshold on the anomaly score is set relatively low, the majority of the hateful comments are retrieved. The drawback is that many of the neutral comments are also flagged. If a significant part of the neutral comments can avoid moderation, social media platforms can save a considerable amount of time and resources. According to Van Royen et al. (2014), expert moderators favour automatic monitoring, but with some conditions, such as including effective follow-up strategies and protecting both the commentators' privacy and their self-reliance. By adopting this approach, automatic monitoring is applied, and follow-up strategies to handle the flagged data can be included.

7.2.2. Advantages

There are several advantages of using a semi-supervised anomaly detection approach to detect hateful content, as described earlier. One of the significant advantages using anomaly detection is that these systems do not try to find similarities between the hateful comments. A supervised classification approach would, however, only learn to recognise hateful comments similar to those seen during training. An issue with this approach is that these utterances might not resemble each other. For instance, both racism and sexism are considered hate speech, but these utterances are not necessarily similar. Furthermore, this makes the methods unequipped to handle hateful expressions that have not yet been seen, i.e. novelties. On the other hand, *anything* not normal is, by definition, an anomaly in an anomaly detection system. The hateful expressions are considered an anomalous variant of ordinary speech, and thus the hateful utterances do not have to be similar.

Another advantage of using anomaly detection to distinguish neutral and hateful content is that they are more capable of handling evolving language. As presented, Nobata et al. (2016) stated that abusive language evolves with time since people create new slurs and inventive ways to avoid being detected. This can cause the hateful comments to change significantly, and thus for a regular classification model, this involves that they no longer fit into their designated class. On the other hand, with anomaly detection approaches, which does not assume similarities between hateful expressions, the comments are still not normal (neutral), and hence, they are considered anomalous. Of course, this presupposes that the evolved comments do not resemble the normal comments.

Applying pre-trained word embeddings have shown excellent results when detecting hateful utterances, but as stated by Pitsilis et al. (2018), they make the detection of hateful statements unfeasible when the author has deliberately replaced the offensive terms with short slang words. These words are probably not included in the pre-trained model's vocabulary, because they are trained on more formal and correctly written text, such as Wikipedia articles. As a result, the semantic meaning of slang words is not represented. However, when using AD approaches, these words are typically regarded as anomalous, as seen when evaluating the experimental results of the Norwegian dataset. For instance, the comment "Snart klart for yrkesjobbmesse nav jobbmesse mosjøenvgs nordland mosjøen utdanning velgyrkesfag yrkesfag sensio smarthus googlehome eaton appstyring stemmestyring" was believably deemed anomalous because it contains so many words that are not contained in the embeddings' vocabulary. For handling slang in hateful comments, this is a potential advantage of using AD techniques over conventional classification. However, this substantiates the claim that a vital challenge in anomaly detection is to avoid that unknown neutral terms are considered anomalous, i.e. to be able to correctly model all normal data.

In a hate speech detection problem, one can separate the comments into two classes; neutral and hateful. Hence, it is possible to address this problem using supervised classification approaches, such as the majority of previous work in the research field. In a real-life scenario, the majority of utterances are neutral, which leads to a very imbalanced distribution between neutral and hateful content. Due to this drastic imbalance between the two classes, using regular classification are rarely successful and can result in too many false negatives (Mehrotra et al., 2017). In contrast, an anomaly detection system is created to handle this imbalance, and are thus better suited to function efficiently on a dataset that represents real-life data.

The developed system uses semi-supervised learning, and it can consequently be argued that it is more usable than a supervised method because it utilises large amounts of unlabelled data. As discussed in Section 3.2, there is still not one commonly accepted hate speech corpus, which makes it difficult for various platforms to train a model based on supervised learning. Furthermore, as Gröndahl et al. (2018) discovered, hate speech detection systems tend to achieve poor performance when they are tested on another dataset than they were trained on. Hence, if an online platform trains a detection method on an already existing dataset, they will most likely achieve poor performance when the method is tested against their own data. The type of comments and users often varies at different platforms, so they would have to train the model on their own data to achieve adequate results. Moreover, it is a time-consuming process to label a large dataset. Online services and platforms might not have the resources, time or capacity to label a large enough dataset to utilise supervised approaches with their own data. However, they often have access to large amounts of unlabelled data, where the majority is neutral content. Fortunately, only a small part of the data has to be annotated to use this semi-supervised approach. The experimental results from this thesis show that the proposed method achieved an AUC above 80% on the English dataset with only 0.33% labelled anomalies (for anomaly class 4). This indicates that the method can achieve relatively good performance with only a small fraction of labelled anomalies. Furthermore, the results show that the proposed method is relatively robust against pollution of anomalous samples in the dataset, which means that the entire dataset does not have to be revised. This makes the system easier to utilise for several platforms.

Utilising user-oriented behavioural data is feasible because such information is available and retrievable for many platforms like Facebook and Twitter. However, for a hate speech detection system to be applicable to a multitude of services and platforms, it is advantageous that the system is independent of platform-specific information, such as user-oriented data. The implemented system only depends on the actual comments and the language of the comments, which makes it easy to adapt without much change.

The proposed system induces an anomaly score instead of giving a distinct evaluation as either neutral or hateful. An advantage of having such a system is that it allows choosing the most suitable threshold based on application purposes. This makes it possible for the various services to determine the threshold based on their guidelines and restrictions. Typically, for a hate speech detection system, it will most likely be more advantageous to flag too many comments as hateful and then use manual moderation to find the actual hateful comments among the flagged ones. The alternative is to detect too few samples, but with higher precision. Hence, setting a low threshold is probably favourable for this application purpose.

7.2.3. Disadvantages and challenges

The model builds on the assumption that normal data instances are similar to each other, and hence, the latent distribution of normal and anomalous data should have low and high entropy, respectively. In many cases, this is a valid assumption. However, when using textual data, this is not necessarily a legitimate assumption, due to the ambiguous and unstructured property of language. Unquestionably, there might be significant variations in both content and structure of the neutral comments. On the other hand, conventional supervised classifiers assume similarities between the data in each class, thus also between the normal data. Accordingly, this assumption is adapted by both approaches and is hence a challenge regardless of the chosen approach. As discussed in Section 2.3.4, another critical challenge that may arise, is that building a comprehensive model for data normality is challenging, and might even be impossible. The reason is that it is often difficult to find all normal behaviours in a system. When working with natural language, there are possibly an endless amount of normal data instances that the model must account for to create a complete representation of the normal class. If the model has not been exposed to a particular form of neutral comments, for example, related to a specific topic, the model is not able to recognise these as normal.

As for all anomaly detection approaches, the developed method is based on the assumption that normal data is stationary, which means that the underlying processes do not change significantly over time. Hence, the method assumes that statistics characterising the system in the past will continue to characterise the system in the future (Mehrotra et al., 2017). Language change over time and this is an issue all models that aim at detecting hate speech must consider. Even though this method works reasonably well now, it does not mean it will continue to work in the future. If neutral speech change significantly, as can be expected long-term, the method will no longer understand that this is considered normal data.

As discussed in Section 2.3.4, it is crucial to determine the correct similarity measure for an anomaly detection algorithm to perform sufficiently. The implemented system uses the calculated distance to a hypersphere centre as the similarity measure. The method achieves relatively good results, but it is still challenging to determine if this measure is optimal. Choosing the optimal measure is a challenge that does not have to be taken into account if the problem is considered a classification problem.

A drawback to using anomaly detection is that one can only separate between hateful and not hateful and it is thus a binary detection problem, due to the specifications of an AD system. Malmasi and Zampieri (2017) found that only distinguishing hate from no-hate using binary classification has previously been the most commonly used approach to detect hate speech. However, as noted by Dinakar et al. (2012), models trained on such data often rely on the frequency of offensive or profane words to distinguish between the classes. Therefore, multi-label classification has become more popular. Using a one-class

AD system have the limitation that it can only learn to represent the normal data, and it can hence not differentiate between various types of anomalies, such as variations of hate speech like racism and sexism. If the system's goal is to detect all hateful content, then this is not an issue. However, if it is desirable to determine the type of hatred that is often uttered by a specific user, an anomaly detection system would provide unsatisfactory results.

As expressed in Section 2.3.4, handling sarcastic irony is a difficult, but essential task. It should be addressed in order to create a system that can capture all hateful content. Due to the challenges related to this research field, the implemented system is not capable of handling sarcastic irony. This involves that the method is wrongfully determining sarcastic comments to be neutral speech.

The method is trained using semi-supervised learning, which consists of mostly unlabelled data. However, to test the method's performance, datasets containing ground-truth labels are used. Furthermore, the labels of all data samples are kept to quantitatively measure performance during testing. This is beneficial for research purposes because evaluating the approach without access to ground-truth labels is challenging. Nevertheless, if other online services or platforms utilise the system, they are not capable of measuring performance without labelling an entire test set. Hence, if the approach is used, the anomaly score for each tested comment must be manually evaluated to test its achievements. Consequently, adopting the approach is more manageable because unlabelled data are utilised, but testing its performance on the unlabelled dataset is more cumbersome.

The experimental results and evaluation suggest that the developed method struggles with at least three specific types of comments. The first case is the neutral comments that objectively discuss topics and include words that are often used in the hateful comments. This applies to comments such as "Fuck jøder er heilt bort i natta å seie men straffbart, trur ikke det. Hadde nokon derimot kalt ein gitt jøde jævla kakerlakk hadde det vore annleis. Det var uttrykk hitler brukte om jødar, han kalla dei skadedyr som måtte utryddast. Det er for meg over streken". The second case involves hateful comments that do not include any of the usual hateful terms, such as "han er en radikal muslimut med det greiene der". This comment was annotated as hateful (category 5) because it expresses the desire to deport an individual based on his or her religion. Since the author did not include whitespace between the words "muslim" and "ut", the model does not understand the meaning of the sentence. The third case involves neutral comments that contain a significant amount of words that are neither well represented in the dataset nor included in the pre-trained embeddings' vocabulary. These comments contain many words that are not similar to other known words and are thus evaluated as anomalous by the model. An example of such a comment is "Snart klart for yrkesjobbmesse nav jobbmesse mosjøenvgs nordland mosjøen utdanning velgyrkesfag yrkesfag sensio smarthus googlehome eaton appstyring stemmestyring".

7.2.4. Improvements

Even though the proposed system performs relatively well, there are many ways to improve even further. Since thorough optimisation was not a priority in this research, it can be expected an increased performance by discovering the system's optimal configurations. Probably, one way to boost system performance is applying grid search for hyperparameter optimisation. The system contains several parameters that should be tested, including η, weight decay λ, learning rate schedule, batch size and optimiser. In this research, only five values for η and three values for λ was tested, and both were tested by setting a static value for the other parameter. Hence, a non-optimal solution was possibly found since all five values for η were not tested against all three values for λ. Furthermore, many design-related choices have not been tested. This includes the representation dimension of the output space d, which was set to 32 based on findings from Ruff et al. (2020). This dimension represents the output from the model's last layer and is used to calculate the distance to the hypersphere centre in the 32-dimensional space. A few experiments were conducted setting $d = 300$, which is equivalent to the dimension of the word embeddings. It was discovered that the model had to train for many more epochs before achieving correspondingly good results, and thus this was not desired. However, a range of other dimensions should be explored. Other design choices include filter sizes and the number of filters, i.e. how many n-grams to consider and how many filters to use for the same region size. It is also possible to change the number of layers in the model, by adding more layers such as dropout after embeddings layer or additional fully connected layers. Furthermore, batch normalisation should be explored, and leaky ReLu can be tested instead of regular ReLu. K-fold cross-validation should also be included to minimise bias caused by the dataset splitting. Moreover, implementing *early stopping* should be employed, to stop the training process if the validation AUC continues to decrease for a chosen number of epochs.

As explained in Chapter 5, out of vocabulary words were initiated as zero-vectors with the same dimension as the pre-trained vectors, which has the drawback that the model cannot find relationships between these OOV words the other words. A way to improve the achievements of this system would be to handle these instances by creating a language model built to produce embeddings for OOV words depending on their context, as done by Kandi (2018). As presented, it was only found embeddings for 36.2% (GloVe) and 55.9% (fastText) of the vocabulary for the English dataset and 63.0% for the Norwegian dataset. By creating embeddings for these words, an escalation in performance may be expected.

Another possible way to improve performance is to add the context of the comments. When only considering a comment, it can in many circumstances, be challenging to determine if the statement is hateful or not. Many comments can be considered hateful in certain contexts, but not in others. For instance, comment (2) and (3) presented as examples of hate speech in Section 2.1, was only found to be hateful based on their context. The Supreme Court found that the first comment was related to Muslims, while the second comment was aimed towards dark-skinned people. The comments could not have been considered hate speech if the context was unknown, which causes an issue for hate speech detection systems that do not include context, such as this developed system. A possible way of including the context could be to provide a dataset which contains the comments and the text from the original post or the previous comment in the same comment thread. By using this information, the system can determine if a comment is hateful based on the content it is referring to. The importance of context is further discussed in Section 7.2.6.

Text representations are genetically noisy because text often involves synonymy and polysemy, which is when the same concept may be represented with multiple words, and the same word can have multiple meanings (Aggarwal, 2017). Using pre-trained word embedding makes it possible to handle synonymy to some degree because they capture semantic similarities. Nevertheless, they do not handle polysemy because a word's pre-trained vector is always the same regardless of the context where it occurs. This might lead to misinterpretation, and in this case, it would be preferable to use BERT, as described in Section 2.5.2, because it can capture the context of a word. However, Ruff et al. (2019) tested both GloVe and fastText embeddings and BERT language model, and found that the improvements using BERT were insufficient and did not justify the increased computational cost. However, as described in Section 3.5, seven out of the top ten best hate speech models presented at SemEval 2019 used BERT. Thus there is a possibility that utilising a language model can improve the performance of this hate speech system. BERT provides a multilingual model that includes support for the Norwegian language, so using BERT can be tested for both datasets.

7.2.5. Language independence

The developed system itself does not depend on language. However, one of its components are language-specific, and it can thus be argued that the system is restricted by language. The input to the method is social media comments, i.e. sentences or paragraphs, which is first preprocessed. Having to handle different languages heavily affects the preprocessing step, and thus this step depends on the language. Different languages might require different preprocessing steps because there are challenges related to each language that should be addressed. For instance, in the Norwegian language challenges include compound words and handling Norwegian Nynorsk. A drawback to the preprocessing step in this method is that these challenges are not handled, because of the lack of research using text in Norwegian. For this system, two dictionaries of common misspellings had to be created; one for each language. This step is also dataset-specific because specific tokens were removed from the respective dataset, such as the token "navn" that were removed from the Norwegian dataset.

As presented in Chapter 5, the developed method roughly consists of a word embeddings part and a convolutional neural network part. The preprocessed textual input first passes through the embeddings layer, which is where the text is converted into word vectors. After that, convolutions and max-pooling are applied before the outputs are concatenated to form a feature vector. Further, dropout is applied, and the feature vector is used as input to the final fully-connected linear layer. The only step in this process that is language dependant is the use of pre-trained word embeddings. These vectors are language dependant since they are representations of a specific language, trained on a corpus consisting of text in this language. Hence, the language of the embeddings restricts the system, but it works similarly well when trained on embeddings from another language, as long as the input text is of the same language. This is exemplified by using two datasets with different languages, and the method is able to handle both. The method can also handle other languages, as long as support for handling preprocessing and pre-trained embeddings for the specific language is added.

7.2.6. Dataset annotation

In Chapter 4, the construction of a dataset consisting of Norwegian social media comments and tweets from Facebook, Twitter and Resett was described. In order to mitigate annotator bias in this dataset, several annotators were included in the annotation process, which was referred to as user-based annotation. An important challenge when annotating a dataset is biasing. A part of the dataset was annotated and tested by the annotators, resulting in annotation bias. Presumably, the dataset is also biased because of the user-based annotation, as discussed in Section 4.3.

In hate speech detection, there is not a clear cut between what is normal and what is not. As discussed in Section 2.1 and 3.1 the subjective interpretation of hate speech is a major issue. As Schmidt and Wiegand (2017) stated, hateful utterance might be influenced by several aspects such as the domain of an utterance, its discourse context, time of posting and identity of author and target recipient. As pointed out by Davidson et al. (2017), an utterance may be perceived as offensive or not depending on one's cultural background. This is an important challenge for all hate speech detection systems and has a considerable impact on the annotated datasets. When evaluating the system's performance on the Norwegian dataset, it could be observed that this caused issues. Several comments misclassified by the method was difficult to annotate even for human annotators. Comments such as "Og bra er det. Mer falske og dobbelmoralske mennesker enn kristne finnes ikke! De kristne kan ta med seg muslimene og reise hjem til midtøsten der de hører hjemme", fit the description of hate speech, but was only regarded as offensive by the person annotating it. Hence, this underlines that although the annotators only work with a limited amount of the data instances, their results will have a large impact on the final results.

Veledar (2018) stated that what drives the sender is not decisive for an utterance to be hateful. What is decisive is how the ordinary audience perceives the utterance, given the context in which it is presented. This thesis has not accounted for the context of a comment, neither in the annotation process nor in the implementation. This missing context increases the difficulty of annotation, because not having the context made the annotation more open for subjective opinions. Several of the annotators gave feedback on the provided data samples, stating that some of the comments were arduous to annotate without knowing the explicit context for when it was written. They stated that they would be more certain about their decisions if they knew these circumstances. For instance, it is problematic to determine if the comment “Nei, det gjør de slett ikke. Og gjør de det blir de ikke savnet. Bare reis, forsvinn, sier jeg.” is hateful, based on the text alone. Comments that refer to an action or a group of people mentioned in the main post or a previous comment are hard to annotate. Often, these comments do not explicitly contain hateful language, but the purpose of the comment might be interpreted as hateful in specific contexts. A possible solution is to cluster comments into conversation threads and hence provide entire conversations to the annotators. In this way, the dataset will be more reliable, which will further improve the results when used with a machine learning model.

7.2.7. Revisiting the research questions

The goal of this thesis was to investigate how to accurately detect hate speech in text using anomaly detection techniques. One main research question was formulated for reaching this goal, which was “How can effective hate speech detection be achieved by applying anomaly detection?”. Three sub-questions were defined to answer this main question. The findings in relation to these questions are addressed in this section.

Research question 1 Which principles and models are effective when using anomaly detection on textual data?

Our study of the state of the art has shown that there are a limited amount of works that address anomaly detection on text data, as described in Section 3.3. Since there has been a lack of research in the field, it is difficult to determine which principles and models that are superior. However, studying the different approaches is essential to gain insights into the possible techniques and approaches that can be utilised in the implementation of an anomaly detection system to detect hateful expressions. Furthermore, it is also interesting to explore and include principles previously used to detect hate speech.

In Section 3.3, the current state of the art within anomaly detection was presented. Even though there are a limited amount of works that use textual data, there are several possible applications of AD techniques on text due to the ubiquity of text in user-generated data online. Previous works include methods such as naïve Bayes and one-class classifiers, but in recent years, there has been an increasing interest in deep anomaly detection algorithms as suggested by more recent papers. These papers include neural networks such as CNNs and autoencoders, the utilisation of pre-trained word vectors and state-of-the-art techniques such as the attention mechanism. As of today, no previous works have exploited deep anomaly detection approaches to detect hateful expressions.

As previously pointed out, hate speech detection is a popular topic, and the amount of research in the field is increasing correspondingly. The literature review also considered research related to hate speech detection focusing on exploring the existing available solutions to automatic hate speech detection, as well as hate speech datasets, relevant features and text representations and hate speech detection for non-English languages. Table 3.2 in Section 3.7 presents an overview of the state of the art within the field of hate speech detection. Chapter 3 also presents popular ways of handling text representation. Recent work has found that proper text representation is crucial for designing well-performing machine learning algorithms. As presented in Chapter 3, several existing methods that use textual data utilises unsupervised pre-trained word models, like word embeddings. There exist several popular word embedding models including Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017). However, recently language models such as BERT Devlin et al. (2018) have outperformed the current state of the art. Another important finding was that the vast majority of existing methods were based on text classification approaches, where neural networks and deep learning methods tend to outperform classical NLP methods. However, Gröndahl et al. (2018) argue that the labelling and type of data are more important than the model architecture and that the lack of a standard hate speech definition and corpus causes several issues.

Based on the findings discussed above and in Chapter 3, it was decided it build on and extend the anomaly detection system provided by Ruff et al. (2020), which involves semi-supervised deep anomaly detection. They argued that their method outperforms all competitors, and can thus be considered state of the art within anomaly detection. However, their approach did not involve textual data and had to be extended to fit the problem specifications of this thesis. Furthermore, it was decided to include pre-trained word embeddings as a part of the developed solution because it has also shown excellent results in previous work.

Research question 2 Would a semi-supervised deep learning model for anomaly detection be effective at correctly determining hateful social media comments?

To answer this research question, a deep anomaly detection method which employs semi-supervised learning was implemented, and an extensive set of experiments were conducted to test its usability. The proposed system is described in Chapter 5 and the experiments and obtained results are presented in Chapter 6. Several aspects of the implemented method have been considered and discussed earlier in this chapter, in order to determine if such a method can be useful in detecting hate speech.

The system's effectiveness has been questioned by discussing several advantages and related issues in the above sections. The experiments showed that the performance decreases when a small dataset is provided, but that the method overall achieved good performance. Moreover, it generally functions more efficiently when using the English dataset. Furthermore, the system is robust against pollution in the training dataset, and it is relatively good at handling novelties. Using semi-supervised learning provides the opportunity to utilise large amounts of unlabelled data, which is a significant advantage compared to using only labelled data. Furthermore, using anomaly detection approaches provides several other advantages over conventional classification methods, such as being more suited for handling a real-scenario distribution between neutral and hateful content, and not assuming any similarities between the hateful comments. Nevertheless, the system faces several issues, such as creating a sufficient representation of data normality and differentiating hateful and neutral content that discusses the same topics. The system is per now not sufficiently effective at distinguishing hateful and neutral content, and there are still many challenges and possible improvements that should be addressed. However, overall, the system's achievements suggest that with further research and fine-tuning, it might be possible to utilise anomaly detection techniques and partially unlabelled data to detect hateful utterances.

Research question 3 How to develop a method for hate speech detection based on anomaly detection that is language independent?

This research question is closely related to Research Question 2 and aims at investigating the developed method's ability to handle a non-English dataset. To test the method with a dataset of another language, a dataset consisting of Norwegian social media comments was collected and annotated. The model's performance was tested using this created dataset, as well as an English dataset. In order to answer this research question, the system was developed to be language independent.

Chapter 4 outlines the process of collecting and annotating the Norwegian dataset, while 7.2.6 discusses some related issues. It was found that not including the context of the comments when annotating caused issues for several of the outside annotators, leaving room for subjective interpretations. It was observed that this heavily affected the method's achievements when evaluating the method using this dataset.

In the discussion in Section 7.2.5, it was stated that the developed system itself is language independent but that it contains one component restricted by language. The conducted preprocessing is dependent on the dataset since it accounts for language and specific terms to remove from each dataset. These implemented preprocessing steps are described in Section 5.1. This is the only part of the method that is concerned with language because the rest of the method only handles encoded word vectors and feature vectors. Even though the created method contains a preprocessing step that is dataset-specific, it is still easy to utilise for several platforms, independently of language. The implemented system can function with all languages as long as support for handling preprocessing and pre-trained embeddings for the specific language is added.

Creating a method that is entirely independent of language involves not having any components that are restricted by language. It is not necessarily beneficial to implement this type of system because it involves not using pre-trained word embedding, which often leads to increased performance. The alternative would be to use another way of representing text or create custom embeddings based on the available data, by training the embeddings with respect to the problem at hand. This solution works well if using a large dataset, but the embeddings created based on a small dataset are presumably not carrying much semantic information. However, comparable results can be expected if one is using a large dataset, and thus, this can be an alternative approach to achieve a system that by no means is restricted by language.

8. Conclusion and Future Work

This chapter provides a conclusion to the work conducted in this thesis and discusses how the research and findings have contributed to the field of hate speech detection. The chapter finishes with the description of possible improvements and other research ideas related to the utilisation of anomaly detection approaches to hate speech detection.

8.1. Conclusion

The debate about hate speech has been central in recent years, and an abundance of research has been conducted aiming at the creation of systems that can automatically detect hate speech. However, existing solutions are not efficient enough and suffer several significant issues. This thesis re-conceptualised hate speech detection as anomaly detection and explored the effects of utilising semi-supervised learning through a series of experiments. Furthermore, there is a lack of research concerned with the discovery of hateful utterances in non-English languages. Thus, the exploratory study conducted in this thesis also included a collection and annotation of a Norwegian dataset. Moreover, a thorough literature review of research related to hate speech detection and anomaly detection was conducted to attain valuable insights. Due to the limited amount of research that addresses anomaly detection on text data, it could not be decided with certainty which principles and models that would be most effective at detecting hateful content. However, Ruff et al. (2020) achieved great results using a method based on entropy minimisation and semi-supervised learning to detect anomalies, so it was decided to extend these ideas to detect hate speech. The thesis also presents the process of collecting and annotating a Norwegian hate speech corpus consisting of more than 41k comments and tweets. It was discovered that even though the annotators were provided common guidelines, there was still a significant level of disagreement. This underlines some of the vital challenges in the research field, which includes the subjective interpretation of hate speech and the difficulty of annotating data.

The effectiveness of the developed semi-supervised anomaly detection method was tested with an extensive set of experiments. It was found that the system is reasonably good at handling novelties and relatively robust against pollution. Moreover, it was determined that the performance decrease when the number of samples in the dataset decrease, stressing the need for extensive data amounts to sufficiently represent data normality. Furthermore, it was discovered that the method is incapable of handling an unsupervised problem, and that, in general, the system performed significantly better using the English dataset than the Norwegian. Overall, the system achieved decent results and outperformed the baseline methods. Anomaly detection systems have several advantages over regular classification algorithms, such as being more suited for handling a real-scenario distribution between neutral and hateful content and not assuming any similarities between the hateful comments. Based on the experiments, it can be determined that there exists a potential for utilising anomaly detection approaches to solve the problem of hate speech detection due to many advantages over classification methods. Nevertheless, the system is not yet practically usable or reliable at distinguishing hateful and neutral content, and there are still many challenges and possible improvements that should be addressed. However, with further fine-tuning, research and optimisation, an anomaly detection system may hopefully be able to achieve state-of-the-art results.

8.2. Contributions

This thesis contributes to the area of hate speech detection by encouraging further exploration of the potential of utilising anomaly detection approaches in NLP. An extensive literature review of related research was carried out, which provides information applicable to this and, potential future work. A comprehensive overview of the existing literature was created and is presented in Section 3.7. This overview may be used as a starting point for future research within the field. Furthermore, the work of this thesis has contributed to the creation of a Norwegian hate speech dataset, consisting of more than 41k comments collected from Facebook, Twitter and Resett. This dataset is a significant contribution to the field of hate speech detection in Norwegian since an annotated baseline dataset did not previously exist. Possibly, the most important contribution is thus the development and experimentation with an anomaly detection system to detect hateful utterances. The results from these experiments clearly show that there is a potential for utilising anomaly detection techniques on this problem. Furthermore, this involves creating more opportunities since large amounts of unlabelled data can be used when developing these systems.

The overall goal of this thesis was to investigate how to accurately detect hate speech using anomaly detection techniques. As stated, the developed method showed a potential for utilising AD techniques but was not capable of accurately detecting the hateful utterances and is currently not practically usable. Nevertheless, relevant research has been provided, and potential improvements have been suggested in order to develop the proposed method further and thus achieve better performance.

8.3. Future Work

Regardless of the research topic, it is advantageous to make improvements to existing work. In recent years, several studies have focused on hate speech detection, yet it is still challenging to achieve satisfactory results. There is also a lack of research related to anomaly detection, and thus more research should be conducted in both research fields. This section provides suggestions for how the research conducted in this thesis can be further extended and improved. In addition to these concrete suggestions, ideas on potential research that may be beneficial for the field of hate speech detection in general, are presented.

Optimise hyperparameters and handle out-of-vocabulary words

As already stated in Section 7.2, it can be expected an increased performance by discovering the system's optimal configurations. For instance, grid search can be applied for hyperparameter optimisation. Furthermore, the system contains several parameters that should be tested, including η, weight decay λ, learning rate schedule, batch size and optimiser. There are also many design-related choices have not been tested, including the representation dimension of the output space, filter sizes and the number of filters. It is also possible to change the number of layers in the model, add batch normalisation and leaky ReLu. K-fold cross-validation and *early stopping* should also be included. Moreover, proper handling of out of vocabulary words should be explored, for instance, by creating a language model built to produce embeddings for OOV words depending on their context.

Change learning model

The system developed in this thesis includes the use of a convolutional neural network (CNN). Other potential networks have not been tested, and thus for future research, the method should be tested with the use of another network or learning model. Models to experiment with can, for instance, be a HybridCNN that uses both words and characters to classify, as done by Gambäck and Sikdar (2017) and Park and Fung (2017), a combination of CNN and GRU (Z. Zhang et al., 2018) or an RNN (Founta et al., 2019; Mehdad and Tetreault, 2016; Pitsilis et al., 2018). Additionally, the use of attention, which was described in Section 2.4.5, is considered state of the art within NLP tasks. Possibly, the model can achieve both increased performance and interpretability by utilising this technique. Furthermore, it would be advantageous to add support for autoencoder pre-training of network weights as conducted by Ruff et al. (2020). Using pre-training is not explored as a part of this implementation, but might improve performance and should thus be tested.

Use common hate speech datasets

Schmidt and Wiegand (2017) stated that there does not exist comparative studies which would allow making a judgement on the most effective learning method. Because of the lack of a benchmark dataset, a lot of the existing studies use a variety of different annotations and data, making it harder to compare methods and results. However, there exist several studies that compare the performance of different methods, and it would be beneficial to compare the results obtained when using anomaly detection to previous solutions. To be able to compare the results, the method should be tested on one of the hate speech datasets commonly used in other related research. This can for instance be the datasets by Waseem and Hovy (2016), Davidson et al. (2017) or Founta et al. (2018).

Experiment with the detection of novelties

As previously mentioned, Gröndahl et al. (2018) compared five state-of-the-art hate speech models and found that all of the models had poor performance when they were trained on one dataset and tested against another. In other words, this means that the models are bad at handling hateful content that does not look similar to previously seen data (novelties). The use of anomaly detection techniques was considered a potential solution to this problem since they do not assume similarities between the hateful statements. Some of the conducted experiments described in Chapter 6 aims at discovering the methods ability to handle novelties by only adding hateful comments from a particular hateful class. However, due to the overlap between the anomalous classes, and that only one class was tested without overlap, more research would have to be conducted to determine if the system does possess this ability. A possible experiment could use the dataset from Waseem and Hovy (2016) that separates between racism, sexism and neither or the dataset from Chatzakou et al. (2017) that distinguishes between bully, aggressive and normal. Then one of the hateful classes could be added as labelled anomalies, and the model could be tested to determine its performance at handling anomalies from the other hateful category. Furthermore, the method should also be tested on another dataset than what it was trained on to compare the relative performance.

Utilising a language model

The work of this thesis explores the use of both GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2017) pre-trained word embeddings. Using these vectors makes it possible to handle synonymy to some degree because they capture semantic similarities. Nevertheless, they do not handle polysemy because a word's pre-trained vector is always the same regardless of the context where it occurs. As described, it would, therefore, be preferable instead to use a language model such as BERT or ElMo, because they can capture the context of a word. There is a possibility that utilising a language model can improve the performance of this hate speech system, and it should thus be tested. BERT provides a multilingual model that includes support for the Norwegian language, so using BERT can be tested for both datasets.

Including the context of comments

Investigating the effects of including the context of the comments to improve the detection rate, should be explored. As discussed, it can be challenging to decide whether something is hateful or not, based only on a short text, especially since a considerable number of comments are replied to other comments. For example, the comment "De omringer oss!" is not necessarily offensive, but in the context of the comment "Jeg har en mørkhudet nabo" it is clearly offensive. Information about the news article that is being discussed can also be valuable.

Another advantage by including information about context is to make it easier to distinguish offensive and hateful utterances. Hateful utterances are in some countries considered illegal, while offensive comments are often just hurtful but still legal. Thus, it is a valuable contribution to be able to distinguish the two categories correctly.

Improve the created Norwegian dataset

Challenges related to the annotation of datasets have been discussed in several sections, including Section 2.1, 3.1 and 4.2. Due to the lack of resources, the majority of all comments were only annotated by one annotator. This causes the possibilities of bias. It would thus be preferable to employ external annotators, where at least two annotators are annotating the same chunk of data. Inter-annotator agreement metrics could then be calculated, and the majority vote could be used to decide the final label of each comment. This will most likely improve the dataset quality significantly. Besides, the dataset should be further extended because as found during the experiments, the method performs better on a larger dataset. Furthermore, for the method to be more capable of creating a sufficient representation of normality, the dataset should contain more comments related to several topics and thus be more generalised. Hence, a more substantial amount of data will most likely improve the system's performance on the Norwegian dataset.

Preprocessing of the Norwegian language

As already stated, languages differ in how challenging they are to preprocess. The English language is known as relatively easy to preprocess, but when studying a language with richer morphology, more flexible word order and distinctive linguistic characteristics, more preprocessing is needed. There have only been a small amount of research within NLP that uses the Norwegian language, which means that there are many challenges that have not been addressed. For example, compound words are a common part of Norwegian vocabulary, which can be particularly challenging when handling user-generated data because there is no guarantee that commentators write grammatically correct. Another challenge is that there are two commonly used languages in Norway: Bokmål and Nynorsk. The same issue can also be found when commentators write in dialect. In order to achieve better performance when hate speech detection is using in Norwegian, more research would have to be conducted on how to handle these challenges.

A specific preprocessing step that would be advantageous to improve is the pre-trained word embeddings for the Norwegian language. In order to obtain state-of-the-art results for hate speech detection in Norwegian, it is crucial to have access to embeddings trained on a larger dataset, containing more words. Currently, the word embeddings provided by fastText includes a vocabulary that is significantly smaller than for the English language. Additionally, the embeddings are specific for either Norwegian Bokmål or Nynorsk. When dealing with user-generated content, where both language variations are used, it is necessary to have embeddings that have been trained on both. In this way, it would be possible to determine that, for instance, the words "kjærlighet" and "kjærleik" are of the same meaning, which is not possible with the current solution because the words are interpreted as unrelated words.

Determining what part of a statement is hateful

A goal within the research field is to be able to create a system that can automatically determine if a comment is hateful. Such a system can, for instance, be used to guide users when writing comments online, by providing "pop-up" messages if a user violates the terms and guidelines. In this case, it would be advantageous for the system to supply information about which part of the statement is considered hateful. When the user has written a comment and wishes to post, the user gets a notification if the underlying hate speech detection system thinks that the user is about to post a comment with degrading content. To enhance the system's usability, it should instruct the user on which part of the comment that might appear stigmatising or derogatory. This is particularly useful if the user writes long comments containing several sentences. For this approach to be possible, the hate speech detection system must consider each sentence individually. This is not a necessity, but it would be favourable if the goal is to guide the users.

Bibliography

Abati, D., Porrello, A., Calderara, S. & Cucchiara, R. (2019). *Latent Space Autoregression for Novelty Detection.*

Aggarwal, C. C. (2017). *Outlier Analysis* (2nd ed.).

Akbik, A., Blythe, D. & Vollgraf, R. (2018). *Contextual String Embeddings for Sequence Labeling.*

Akcay, S., Atapour-Abarghouei, A. & Breckon, T. P. (2018). GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11363 LNCS*, 622–637.

Alfina, I., Mulia, R., Fanany, M. I. & Ekanata, Y. (2017). Hate speech detection in the Indonesian language: A dataset and preliminary study. In *2017 international conference on advanced computer science and information systems (icacsis)* (Vol. 2018-January, pp. 233–238).

Alla, S. & Adari, S. K. (2019). *Beginning Anomaly Detection Using Python-Based Deep Learning.*

Artstein, R. (2017). Inter-annotator Agreement. In *Handbook of linguistic annotation* (pp. 297–313).

Badjatiya, P., Gupta, S., Gupta, M. & Varma, V. (2017). Deep Learning for Hate Speech Detection in Tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW 17 Companion.*

Baeza-Yates, R. & Ribeiro-Neto, B. (2011). *Modern information retrieval: the concepts and technology behind search* (Second). Harlow, England: Addison-Wesley.

Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *Iclr.*

Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Rangel, F., Rosso, P. & Sanguinetti, M. (2019). *SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter.*

Bayerl, P. S. & Paul, K. I. (2011). What determines inter-coder agreement in manual annotations? Ameta-analytic investigation. *Computational Linguistics*, *37*(4), 699–725.

Bayes, T. (1763). LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions Royal Society*, *53*, 370–418.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., Ca, J. U., Kandola, J., Hofmann, T., Poggio, T. & Shawe-Taylor, J. (2003). *A Neural Probabilistic Language Model.*

Bermingham, A. & Smeaton, A. F. (2009). A study of inter-annotator agreement for opinion retrieval. In *Proceedings - 32nd annual international acm sigir conference on research and development in information retrieval, sigir 2009* (pp. 784–785).

Berthold, M. R. (2003). *Mixed fuzzy rule formation.*

Biesek, M. (2019). Comparison of Traditional Machine Learning Approach and Deep Learning Models in Automatic Cyberbullying Detection for Polish Language. *Proceedings of the PolEval 2019 Workshop*, 121–126.

Blanchard, G., Lee, G. & Scott, C. (2010). Semi-Supervised Novelty Detection. *Journal of Machine Learning Research*, *11*, 2973–3009.

Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bobicev, V. & Sokolova, M. (2017). Inter-Annotator Agreement in Sentiment Analysis: Machine Learning Perspective. In *Ranlp 2017 - recent advances in natural language processing meet deep learning* (pp. 97–102).

Bosco, C., Dell'orletta, F., Poletto, F., Sanguinetti, M. & Tesconi, M. (2018). *Overview of the EVALITA 2018 Hate Speech Detection Task.*

Brown, P. F., DeSouza, P. V., Mercer, R. L., Della Pietra, V. J. & Lai, J. C. (1992). *Class-Based n-gram Models of Natural Language.*

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

Büttcher, S., Clarke, C. & Cormack, G. (2016). *Information Retrieval implementing and evaluating search engines.* Cambridge, MA: The MIT Press.

Chalapathy, R. & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. *CoRR.*

Chatzakou, D., Kourtellis, N., Blackburn, J., De Cristofaro, E., Stringhini, G. & Vakali, A. (2017). *Mean Birds: Detecting Aggression and Bullying on Twitter.*

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37–46.

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the 2002 conference on empirical methods in natural language processing ({emnlp} 2002)*, Association for Computational Linguistics.

Cortes, C. & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297.

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, *20*, 215–242.

D.Manning, C., Rahavan, P. & Schütze, H. (2009). *An Introduction to Information Retrieval.* Cambridge, England: Cambridge University Press.

Davidson, T., Warmsley, D., Macy, M. & Weber, I. (2017). *Automated Hate Speech Detection and the Problem of Offensive Language.*

De Gibert, O., Perez, N., García-Pablos, A. & Cuadros, M. (2018). *Hate Speech Dataset from a White Supremacy Forum.*

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

Devlin, J., Chang, M.-W., Lee, K., Google, K. T. & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

Dinakar, K., Jones, B., Havasi, C., Lieberman, H. & Picard, R. (2012). Common Sense Reasoning for Detection, Prevention, and Mitigation of Cyberbullying. *ACM Transactions on Interactive Intelligent Systems*, *2*(3).

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V. & Bhamidipati, N. (2015). Hate Speech Detection with Comment Embeddings. *Proceedings of the 24th International Conference on World Wide Web - WWW 15 Companion.*

Elden, J. C., Gisle, J. & Kierulf, A. (2018). Ytringsfrihet. *Store norske leksikon.*

Erfani, S. M., Rajasegarar, S., Karunasekera, S. & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, *58*, 121–134.

Ergen, T., Mirza, A. H. & Kozat, S. S. (2017). Unsupervised and Semi-supervised Anomaly Detection with LSTM Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Fagni, T., Nizzoli, L., Petrocchi, M. & Tesconi, M. (2019). *Six Things I Hate About You (in Italian) and Six Classification Strategies to More and More Effectively Find Them.*

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, *76*(5), 378–382.

Fortuna, P. (2018). A Survey on Automatic Detection of Hate Speech in Text. *ACM Com-put. Surv*, *51.*

Fortuna, P., Soler-Company, J. & Nunes, S. (2019). *Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible?*

Founta, A.-M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A. & Leontiadis, I. (2019). *A Unified Deep Learning Architecture for Abuse Detection.*

Founta, A.-M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., Vakali, A., Sirivianos, M. & Kourtellis, N. (2018). *Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior.*

Frenda, S. (2018). The Role of Sarcasm in Hate Speech. A Multilingual Perspective. In *Proceedings of the doctoral symposium of the xxxiv international conference of the spanish society for natural language processing (sepln 2018)* (pp. 13–17). Sevilla, Spain.

Gambäck, B. & Sikdar, U. K. (2017). *Using Convolutional Neural Networks to Classify Hate-Speech.*

Gaydhani, A., Doma, V., Kendre, S. & Bhagwat, L. (2018). *Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach.*

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning.* MIT Press.

Görnitz, N., Rieck, K., Brefeld, U. & Kloft, M. (2013). *Toward Supervised Anomaly Detection.*

Gorokhov, O., Petrovskiy, M. & Mashechkin, I. (2017). Convolutional neural networks for unsupervised anomaly detection in text data. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 10585 LNCS, pp. 500–507).

Gröndahl, T., Juuti, M., Conti, M. & Asokan, N. (2018). *All You Need is "Love": Evading Hate Speech Detection.*

Han, J., Kamber, M. & Pei, J. (2012). *Data Mining Concepts and Techniques* (3rd ed.). Waltham: Morgan Kaufmann Publishers.

He, K., Zhang, X., Ren, S. & Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *The ieee international conference on computer vision (iccv)* (pp. 1026–1034).

Hendrycks, D., Mazeika, M. & Dietterich, T. (2018). Deep Anomaly Detection with Outlier Exposure. *7th International Conference on Learning Representations, ICLR 2019.*

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.

Hosseinmardi, H., Mattson, S. A., Rafiq, I., Han, R., Lv, Q. & Mishra, S. (2015). *Detection of Cyberbullying Incidents on the Instagram Social Network.*

Howard, J. & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification.*

Ikonomakis, E. K., Kotsiantis, S., Ikonomakis, M., Kotsiantis, S. & Tampakas, V. (2005). Text Classification Using Machine Learning Techniques. *WSEAS TRANSACTIONS on COMPUTERS*, *4*(8), 966–974.

Indira Gandhi, S. K., Zareapoor, M. & R, S. K. (2015). Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection. *Information Engineering and Electronic Business*, *2*, 60–65.

Indurthi, V., Syed, B., Shrivastava, M., Gupta, M. & Varma, V. (2019). *Fermi at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media using Sentence Embeddings.*

Isaksen, V. (2019). *Detecting Hateful and Offensive Language with Transfer-Learned Models.* Norwegian University of Science and Technology. Trondheim.

Jaki, S. & De Smedt, T. (2018). *Right-wing German Hate Speech on Twitter: Analysis and Automatic Detection.*

Jensen, M. H., Gunstad, T. S. & Svanes, M. A. (2019). *Detecting offensive utterances in the Norwegian language.* Project report in TDT4501. Department of Computer Science, Norwegian University of Science and Technology. Trondheim.

Kandi, S. M. (2018). *Language Modelling for Handling Out-of-Vocabulary Words in Natural Language Processing* (Doctoral dissertation).

Kannan, R., Woo, H., Aggarwal, C. C. & Park, H. (2017). Outlier Detection for Text Data : An Extended Version.

Kim, J. & Scott, C. D. (2012). *Robust Kernel Density Estimation.*

Kingma, D. P. & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, iclr 2015 - conference track proceedings*, International Conference on Learning Representations, ICLR.

Kumar, R., Ojha, A. K., Malmasi, S. & Zampieri, M. (2018). *Benchmarking Aggression Identification in Social Media* (tech. rep. No. 1).

Lee, Y., Yoon, S. & Jung, K. (2018). *Comparative Studies of Detecting Abusive Language on Twitter.*

Lewis, D. D., Yang, Y., Rose, T. G. & Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, *5*, 361–397.

Liu, F. T., Ting, K. M. & Zhou, Z.-H. (2008). Isolation Forest. *ICDM*, 413–422.

Liu, H., Burnap, P., Alorainy, W. & Williams, M. L. (2019). Fuzzy Multi-task Learning for Hate Speech Type Identification. *The World Wide Web Conference on - WWW 19.*

Luong, M.-T., Pham, H. & Manning, C. D. (2015). *Effective Approaches to Attention-based Neural Machine Translation.* Association for Computational Linguistics.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N. & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *Plos One.*

Mahapatra, A., Srivastava, N. & Srivastava, J. (2012). Contextual anomaly detection in text data. *Algorithms*, *5*(4), 469–489.

Malmasi, S. & Zampieri, M. (2017). *Detecting Hate Speech in Social Media.*

Manevitz, L. M., Yousef, M., Cristianini, N., Shawe-Taylor, J. & Williamson, B. (2001). *One-Class SVMs for Document Classification.*

Manevitz, L. & Yousef, M. (2007). One-class document classification via Neural Networks. *Neurocomputing*, *70*(7-9), 1466–1481.

Mehdad, Y. & Tetreault, J. (2016). *Do Characters Abuse More Than Words?*

Mehrotra, K. G., Mohan, C. K. & Huang, H. (2017). *Anomaly Detection Principles and Algorithms.*

Melzi, S., Abdaoui, A. & Azé, J. (2014). *Patient's rationale: Patient Knowledge retrieval from health forums.*

Meyer, J. S. & Gambäck, B. (2019). *A Platform Agnostic Dual-Strand Hate Speech Detector.*

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality.*

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C. & Joulin, A. (2017). *Advances in Pre-Training Distributed Word Representations.*

Moya, M. M., Koch, M. W. & Hostetler, L. D. (1993). One-class classifier networks for target recognition applications. *Proceedings World Congress on Neural Networks*, 797–801.

Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y. & Chang, Y. (2016). Abusive Language Detection in Online User Content. *Proceedings of the 25th International Conference on World Wide Web - WWW 16*, 145–153.

Nockleby, J. T. (2000). *Encyclopedia of the American Constitution.* Macmillan Reference USA.

Ogrodniczuk, M. & Kobyliński, Ł. (2019). *Proceedings of the PolEval 2019 Workshop.*

Pang, G., Shen, C. & Hengel, A. v. d. (2019). Deep Anomaly Detection with Deviation Networks. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 353–362.

Park, J. H. & Fung, P. (2017). *One-step and Two-step Classification for Abusive Language Detection on Twitter.*

Pavlopoulos, J., Malakasiotis, P. & Androutsopoulos, I. (2017). *Deep Learning for User Comment Moderation.*

Pennington, J., Socher, R. & Manning, C. D. (2014). *GloVe: Global Vectors for Word Representation.*

Peters, M. E., Neumann, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). *Deep contextualized word representations.*

Pitsilis, G. K., Ramampiaro, H. & Langseth, H. (2018). *Detecting Offensive Language in Tweets Using Deep Learning.* Norwegian University of Science and Technology.

Rajadesingan, A., Zafarani, R. & Liu, H. (2015). Sarcasm Detection on Twitter: A Behavioral Modeling Approach. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM 15.*

Rebala, G., Ravi, A. & Churiwala, S. (2019). *An Introduction to Machine Learning.*

Robinson, D., Zhang, Z. & Tepper, J. (2018). Hate speech detection on twitter: Feature engineering v.s. feature selection. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 11155 LNCS, pp. 46–49).

Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N. & Wojatzki, M. (2017). *Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis.*

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R. & Kloft, M. (2020). Deep Semi-Supervised Anomaly Detection. In *Iclr.*

Ruff, L., Zemlyanskiy, Y., Vandermeulen, R., Schnake, T. & Kloft, M. (2019). Self-Attentive, Multi-Context One-Class Classification for Unsupervised Anomaly Detection on Text. (pp. 4061–4071).

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (Third Edit). New Jersey: Pearson Education, Inc.

Saito, T. & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, *10*(3), e0118432.

Salton, G. & Yang, C. S. (1973). On the specification of term values in automatic text analysis. *Journal of Documentation*, *29*, 351–372.

Schmidt, A. & Wiegand, M. (2017). *A Survey on Hate Speech Detection using Natural Language Processing.* Valencia, Spain: Association for Computational Linguistics.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Holloway, R., Smola, A. J. & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural computation*, *13*(7), 1443–1471.

Sharma, S., Agrawal, S. & Shrivastava, M. (2018). *Degree based Classification of Harmful Speech using Twitter Data.*

Sigurbergsson, G. I. & Derczynski, L. (2019). *Offensive Language and Hate Speech Detection for Danish.*

Silva, L., Mondal, M., Correa, D. & Weber, I. (2016). *Analyzing the Targets of Hate in Online Social Media.*

Srivastava, N., Hinton, G., Krizhevsky, A. & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.*

Steyn, C. & De Waal, A. (2016). Semi-supervised machine learning for textual anomaly detection. In *Pattern recognition association of south africa and robotics and mechatronics international conference, prasa-robmech 2016.*

Tax, D. M. & Duin, R. P. (2004). Support Vector Data Description. *Machine Learning*, *54*(1), 45–66.

Unsvåg, E. F. (2018). *Investigating the Effects of User Features in Hate Speech Detection on Twitter.* Norwegian University of Science and Technology. Trondheim.

Van Hee, C., Lefever, E., Verhoeven, B., Mennes, J., Desmet, B., De Pauw, G., Daelemans, W. & Hoste, V. (2015). *Automatic Detection and Prevention of Cyberbullying.*

Van Royen, K., Poels, K., Daelemans, W. & Vandebosch, H. (2014). Automatic monitoring of cyberbullying on social networking sites: From technological feasibility to desirability. *Telematics and Informatics*, *32.*

Vandermeulen, R. A. & Scott, C. D. (2013). *Consistency of Robust Kernel Density Estimators.*

Veledar, A. (2018). *Hatefulle ytringer i offentlig debatt på nett.* Likestillings- og diskrimineringsombudet.

Viera, A. J. & Garrett, J. M. (2005). Understanding interobserver agreement: The kappa statistic. *Family Medicine*, *37*(5), 360–363.

Vigna, F. D., Cimino, A., Dell'orletta, F., Petrocchi, M. & Tesconi, M. (2017). *Hate me, hate me not: Hate speech detection on Facebook.*

Wang, J.-H., Liu, T.-W., Luo, X. & Wang, L. (2018). *An LSTM Approach to Short Text Sentiment Classification with Word Embeddings.*

Warner, W. & Hirschberg, J. (2012). *Detecting Hate Speech on the World Wide Web.*

Waseem, Z. (2016). Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Emnlp workshop on natural language processing and computational social science* (pp. 138–142). Austin.

Waseem, Z. & Hovy, D. (2016). *Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter.*

Wiegand, M., Siegel, M., Ruppenhofer, J. & Klenner, M. (2018). *4 Saarland University's Participation in the GermEval Task.* UdSW.

Wulczyn, E., Thain, N. & Dixon Jigsaw, L. (2017). Ex Machina: Personal Attacks Seen at Scale. *Proceedings of the 26th International Conference on World Wide Web - WWW 17.*

Xiang, G., Fan, B., Wang, L., Hong, J. I. & Rose, C. P. (2012). *Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus.*

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N. & Kumar, R. (2019). *SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval).*

Zhang, H., Mahata, D., Shahid, S., Mehnaz, L., Anand, S., Kumar, Y., Ratn Shah, R. & Uppal, K. (2019). *MIDAS at SemEval-2019 Task 6: Identifying Offensive Posts and Targeted Offense from Twitter.*

Zhang, Y. & Wallace, B. C. (2016). *A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification.*

Zhang, Z., Robinson, D. & Tepper, J. (2018). Detecting hate speech on Twitter using a convolution-GRU based deep neural network. *Lecture Notes in Computer Science*, 745–760.

Zhong, H., Li, H., Squicciarini, A., Rajtmajer, S., Griffin, C., Miller, D. & Caragea, C. (2016). *Content-Driven Detection of Cyberbullying on the Instagram Social Network.*

Zhou, L., Li, F. & Yang, Y. (2008). Path Algorithms for One-Class SVM. In *Advances in neural networks - isnn 2008* (pp. 645–654).

A. Additional Experimental Results

The experimental plan and setup described in Section 6.1 and 6.2 involved a large number of experiments using different configurations/setups. For all experiments, the AUC score was calculated, and the majority of all scores are presented in Section 6.3. Furthermore, for every experiment, plots of the training and validation loss, validation AUC, P-R curves, ROC curves and anomaly score histograms were created. These plots are only presented for the best performing configuration for practical reasons. Furthermore, only two confusion matrices per dataset were presented in the evaluation, so the remaining two are presented here. For some of the configurations, the experiments were conducted more than once. This involves the experiments with and without deducting the number of normal samples, as described in Section 6.3. Only the results using the preferable approach for each dataset were presented as a part of the experimental results, but the results for the respective other approaches are presented here. In addition, it involves testing the unsupervised and normal setting on the English dataset without using network bias terms.

Hence, this appendix introduces a selection of additional experimental results. It is divided based on the two datasets.

A.1. Results using the English dataset

This section presents a selection of additional experimental results using the English dataset. It starts by presenting the results from test 1, without decreasing the number of normal samples. Not decreasing the number of normal samples were not conducted for the configurations from test 2. Furthermore, a comparison between the results for the unsupervised and normal setting with and without the use of network bias terms is described.

Test 1

The results obtained when keeping the original number of normal samples in the dataset are shown in Table A.1. Hence, this table presents the results when a smaller ratio of anomalous samples are added to the dataset. The table only includes the results from class 2: severe toxic, 4: threat and 6: identity hate, because these were the classes that did not contain enough samples to add 5% and 10% labelled anomalous samples. The actual ratio of anomalies in these experiments for class 2, 4 and 6 are 1.1%, 0.3% and 1.0%, respectively. Since the only difference is the amount of labelled normal data, the cases where $\gamma_a = 0.05$ and 0.10 are equal, and they are thus merged into one column; γ_a. $\gamma_l = 0.05$ or 0.10 means that all available anomalous samples are included, in addition to 5% or 10% normal samples.

Table A.1.: Results without decreasing normal samples in the English dataset. AC = anomaly class.

English dataset: Adding labelled training data						
	GloVe			**fastText**		
	γ_a	γ_l		γ_a	γ_l	
AC		0.05	0.10		0.05	0.10
2	88.0	88.8	88.7	87.8	88.2	86.4
4	78.0	81.0	80.9	71.9	74.0	72.7
6	88.0	88.4	88.3	88.9	88.3	89.3

For easier comparison, the results using both approaches with the English dataset is presented in Table A.2. This table shows the results with and without a reduced amount of normal data as AUC_D and AUC_ND, respectively. Furthermore, it presents the difference between the obtained results using both approaches, as well as the average difference for each anomaly class.

As can be seen from Table A.2, both the largest increase and decrease are achieved when adding labelled anomalies from class 4. When using GloVe, the system experiences a decreased performance by 6.5%. On the other hand, when using fastText, the performance increase by 11.7%. For anomaly class 2 the system obtains a reduced performance by an average of 0.2%, but for anomaly class 4 and 6, it experiences an increase of 2.6% and 0.7%, respectively. This results in an overall performance increase by 1.1% when decreasing the amount of normal data.

Table A.2.: Comparison between the two approaches (with and without the deduction of normal samples) on the English dataset. AC = anomaly class. AUC_D is the AUC scores (in %) when normal samples are deducted, while AUC_ND is the scores when normal samples are not deducted. Diff is the difference between AUC_D and AUC_ND and Diff_avg is the average difference.

Comparing approaches						
AC	**Embedding**	**Setup**	**AUC_D**	**AUC_ND**	**Diff**	**Diff_avg**
2	GloVe	$\gamma_a = 0.05$	87.8	88.0	-0.2	-0.9
		$\gamma_a = 0.10$	88.2	88.0	0.2	
		$\gamma_l = 0.05$	87.3	88.8	-1.5	
		$\gamma_l = 0.10$	86.6	88.7	-2.1	
	fastText	$\gamma_a = 0.05$	88.0	87.8	0.2	0.5
		$\gamma_a = 0.10$	88.2	87.8	0.4	
		$\gamma_l = 0.05$	87.5	88.2	-0.7	
		$\gamma_l = 0.10$	88.6	86.4	2.2	
4	GloVe	$\gamma_a = 0.05$	80.2	78.0	2.2	-6.5
		$\gamma_a = 0.10$	73.3	78.0	-4.7	
		$\gamma_l = 0.05$	78.7	81.0	-2.3	
		$\gamma_l = 0.10$	59.6	80.9	-21.3	
	fastText	$\gamma_a = 0.05$	87.7	71.9	15.8	11.7
		$\gamma_a = 0.10$	82.9	71.9	11.0	
		$\gamma_l = 0.05$	83.8	74.0	9.8	
		$\gamma_l = 0.10$	82.7	72.7	10.0	
6	GloVe	$\gamma_a = 0.05$	89.1	88.0	1.1	0.4
		$\gamma_a = 0.10$	88.4	88.0	0.4	
		$\gamma_l = 0.05$	88.9	88.4	0.5	
		$\gamma_l = 0.10$	87.8	88.3	-0.5	
	fastText	$\gamma_a = 0.05$	89.4	88.9	0.5	1.0
		$\gamma_a = 0.10$	90.3	88.9	0.4	
		$\gamma_l = 0.05$	89.7	88.3	1.4	
		$\gamma_l = 0.10$	90.9	89.3	1.6	

The effect of network bias terms

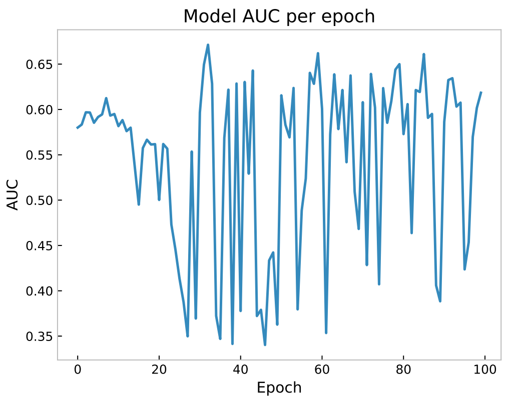
The method achieves poor performance in the two setups which involves not adding any labelled anomalies, i.e. in the unsupervised and normal setting. A potential cause is the use of network bias terms. When the network includes bias terms, it is possible to achieve a trivial, uninformative solution when the training set does not contain a sufficient amount of labelled anomalous data. Test 1 for unsupervised and normal setting is conducted with and without the inclusion of the network bias terms. The results are presented in Table A.3. The table includes the performance in each of the cases, as well as the difference in performance between the two approaches.

Table A.3.: Results with and without network bias terms for the unsupervised ($\gamma_l = 0.00$) and normal setting ($\gamma_n = 0.10$) using the English dataset.

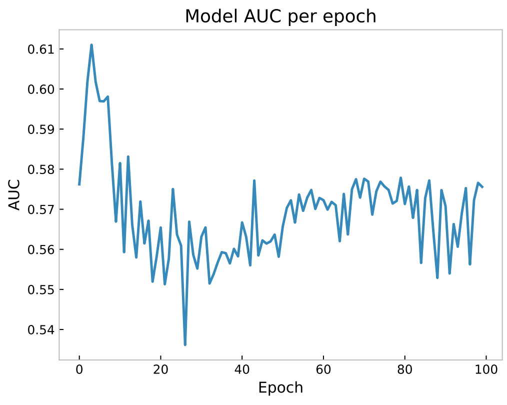
English dataset: The effect of network bias terms				
	GloVe		**fastText**	
Bias terms	$\gamma_l = 0.00$	$\gamma_n = 0.10$	$\gamma_l = 0.00$	$\gamma_n = 0.10$
With	60.9	62.6	53.5	50.0
Without	57.1	56.5	55.9	53.7
Difference	3.8	6.1	-2.4	-3.7

As can be observed, both scores decreased when the bias terms were removed when using GloVe, with 3.8% and 6.1%, respectively. On the other hand, both scores increased with fastText, with 2.4% and 3.7%.

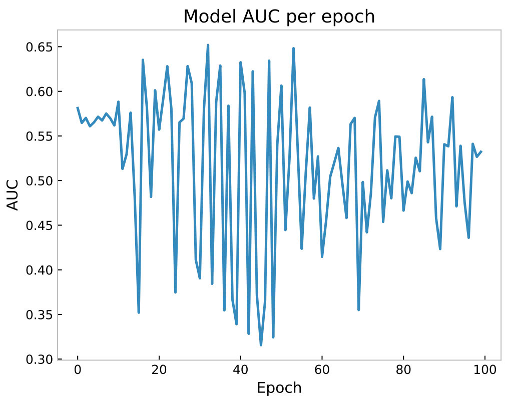
The validation AUC curves for both settings, can be found in Figure A.1 and A.2. The figures show the achieved validation scores for each epoch, with and without the inclusion of network bias terms.



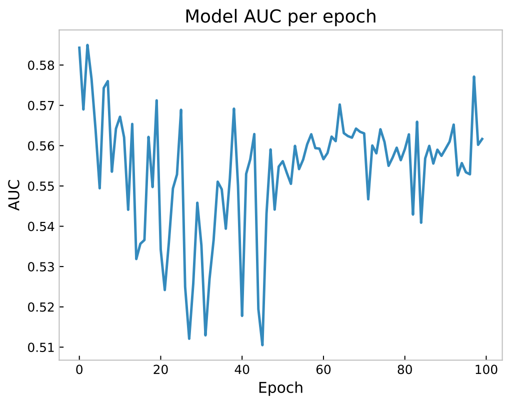
(a) Using GloVe with bias terms



(b) Using GloVe without bias terms

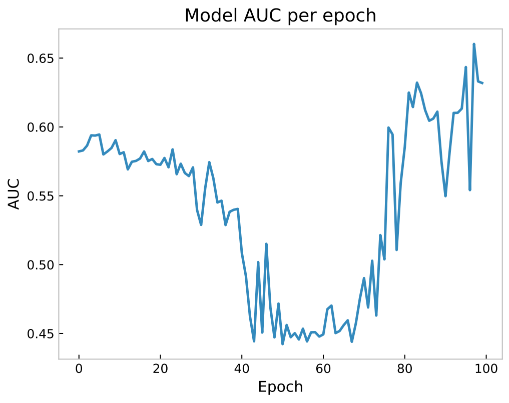


(c) Using fastText with bias terms

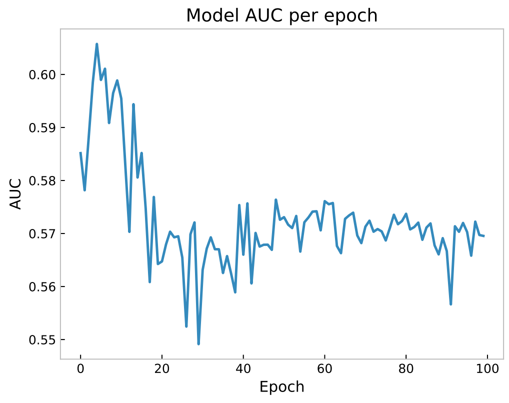


(d) Using fastText without bias terms

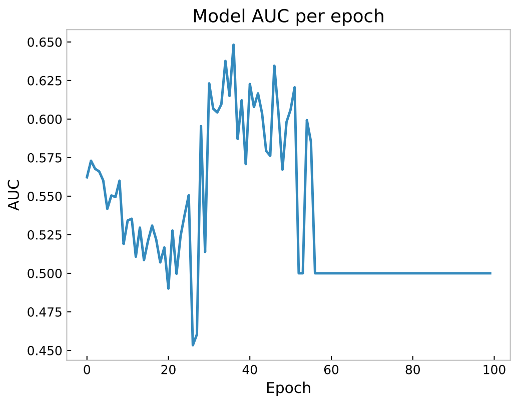
Figure A.1.: Validation AUC scores for the English dataset in the unsupervised setting



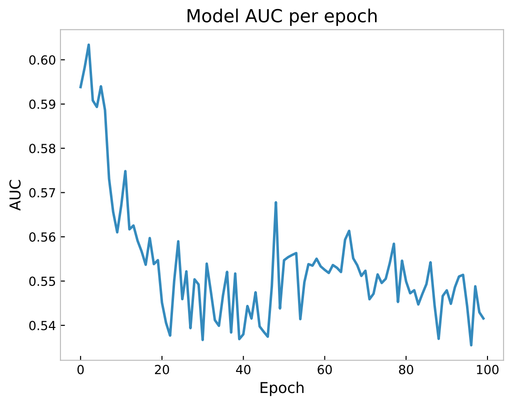
(a) Using GloVe with bias terms



(b) Using GloVe without bias terms



(c) Using fastText with bias terms



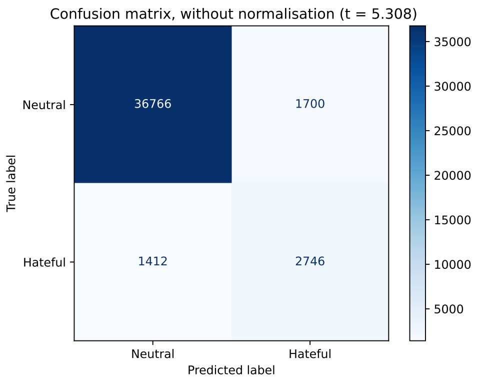
(d) Using fastText without bias terms

Figure A.2.: Validation AUC scores for the English dataset in the normal setting

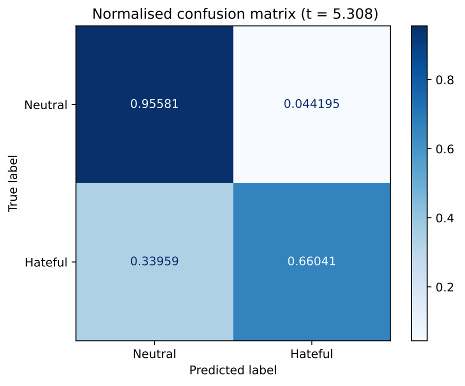
From the figures, it can be observed that in the unsupervised setting, the validation AUC score varies significantly more when bias terms are included. Here, the AUC varies from approximately 0.35 to 0.65, while it only varies between 0.5 to 0.6 when bias terms are removed. This is also the case for the normal setting, but the difference in variation is smaller than for the unsupervised setting. From both the table and figures above, we can see that the model performs poorly both with and without bias terms. However, when removing bias terms, the model's result does not longer appear to be solely based on the number of epochs.

Confusion matrices

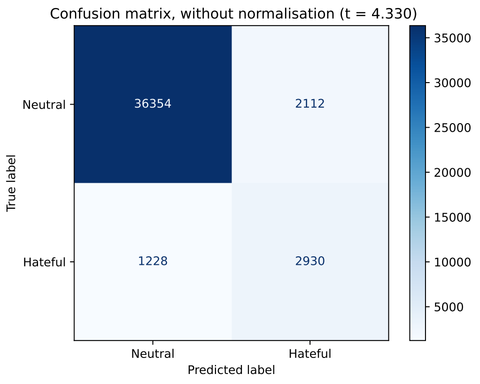
The confusion matrices for the thresholds corresponding to the frequency limits 20 and 30, that were not included in the evaluation in Section 7.1, are presented in Figure A.3.



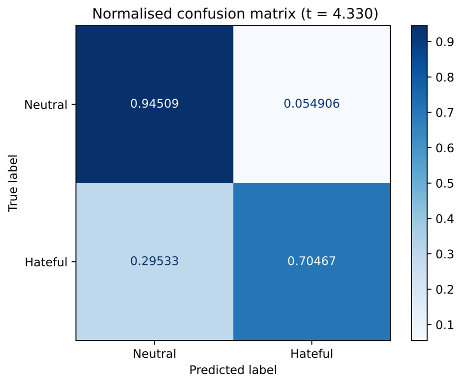
(a) Confusion matrix with t = 5.308



(b) Normalised confusion matrix with t = 5.308



(c) Confusion matrix with t = 4.330



(d) Normalised confusion matrix with t = 4.330

Figure A.3.: Additional confusion matrices for the English dataset. (a) and (b) are found by setting the frequency limit to 20, and hence, threshold t = 5.308. On the other hand, (c) and (d) are found by setting the frequency limit to 30, and hence threshold t = 4.330. (b) and (d) contains normalised predictions.

A.2. Results using the Norwegian dataset

The results obtained when the amount of normal samples in the dataset are decreased, are shown in Table A.4 and Table A.5. Table A.4 shows the results from test 1, while Table A.5 shows the results from test 2. Hence, these tables present the results when the desired ratio of anomalous samples are added to the dataset.

Table A.4.: Results from test 1 when decreasing normal samples in the Norwegian dataset. AC = anomaly class.

Norwegian dataset: Adding labelled training data				
	γ_a		γ_l	
AC	0.05	0.10	0.05	0.10
{4, 5}	70.8	72.0	71.3	67.9
{3, 4, 5}	73.0	76.5	71.3	73.6

Table A.5.: Results from test 2 when decreasing normal samples in the Norwegian dataset. AC = anomaly class.

Norwegian dataset: Polluted training data			
AC	$\gamma_p = 0.00$	$\gamma_p = 0.01$	$\gamma_p = 0.05$
{4, 5}	71.3	65.7	50.6
{3, 4, 5}	71.3	72.3	53.5

For easier comparison, the results using both approaches with the Norwegian dataset are presented in Table A.6. This table shows the results with and without a reduced amount of normal data as AUC_D and AUC_ND, respectively. Furthermore, it presents the difference between the obtained results using both approaches, as well as the average difference for each set of anomaly classes.

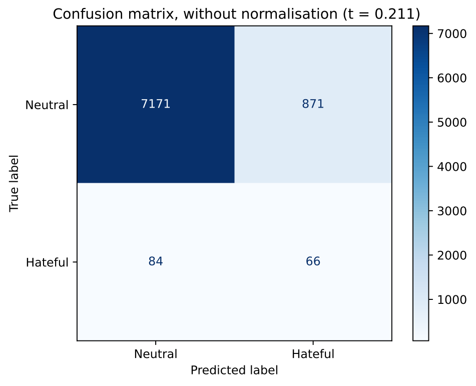
As can be seen from Table A.6, for all setups the system achieves an increased performance when the original amount of normal data is retained. The most significant differences are observed when 5% pollution is added to the training set, where the performance is 24.3% and 21.9% better when all samples are kept. Overall, the results are in this case, on average, 7.7% better.

Table A.6.: Comparison between the two approaches (with and without the deduction of normal samples) on the Norwegian dataset. AC = anomaly class. AUC_{ND} is the scores when normal samples are not deducted, while AUC_{D} is the AUC scores (in %) when normal samples are deducted. Diff is the difference between AUC_{ND} and AUC_{D} and Diff_{avg} is the average difference.

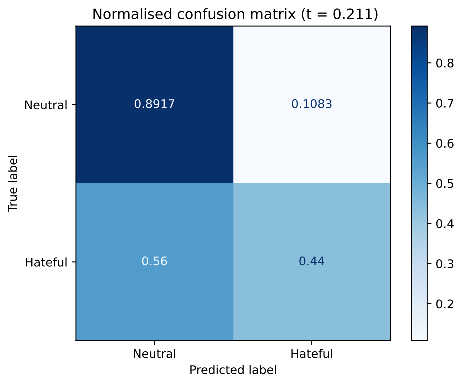
Comparing approaches						
Test	**AC**	**Setup**	$\textbf{AUC}_{\textbf{ND}}$	$\textbf{AUC}_{\textbf{D}}$	**Diff**	$\textbf{Diff}_{\textbf{avg}}$
1	{4, 5}	$\gamma_a = 0.05$	74.4	70.8	3.6	4.0
		$\gamma_a = 0.10$	74.4	72.0	2.4	
		$\gamma_l = 0.05$	73.7	71.3	2.4	
		$\gamma_l = 0.10$	75.3	67.9	7.4	
	{3, 4, 5}	$\gamma_a = 0.05$	76.8	73.0	3.8	2.8
		$\gamma_a = 0.10$	77.3	76.5	0.8	
		$\gamma_l = 0.05$	75.6	71.3	4.3	
		$\gamma_l = 0.10$	75.8	73.6	2.2	
2	{4, 5}	$\gamma_p = 0.00$	73.7	71.3	2.4	11.5
		$\gamma_p = 0.01$	73.6	65.7	7.9	
		$\gamma_p = 0.05$	74.9	50.6	24.3	
	{3, 4, 5}	$\gamma_p = 0.00$	75.6	71.3	4.3	9.5
		$\gamma_p = 0.01$	74.7	72.3	2.4	
		$\gamma_p = 0.05$	75.4	53.5	21.9	

Confusion matrices

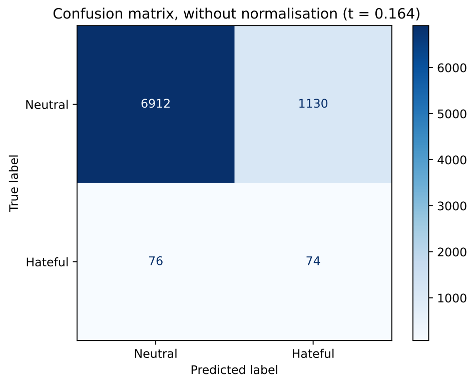
The confusion matrices for the thresholds corresponding to the frequency limits 30 and 40 are presented in Figure A.4.



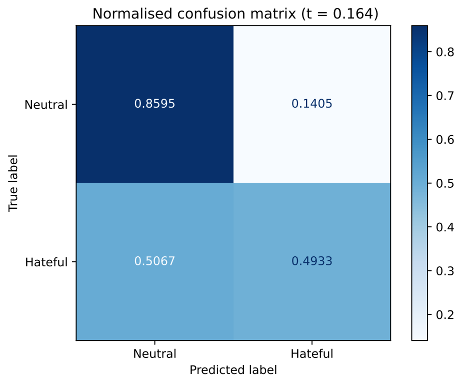
(a) Confusion matrix with t = 0.221



(b) Normalised confusion matrix with t = 0.221



(c) Confusion matrix with t = 0.164



(d) Normalised confusion matrix with t = 0.164

Figure A.4.: Additional confusion matrices for the Norwegian dataset. (a) and (b) are found by setting the frequency limit to 30, and hence, threshold t = 0.221. On the other hand, (c) and (d) are found by setting the frequency limit to 40, and hence threshold t = 0.164. (b) and (d) contains normalised predictions.

B. Annotation Guidelines

This appendix presents the guidelines provided to the outside annotators as a part of the user-based annotation of the Norwegian dataset. The guidelines are written in Norwegian, but a short version is included in English in Section 4.2.2.

Retningslinjer for klassifisering

Du skal nå lese gjennom en og en kommentar og avgjøre hvilken kategori den tilhører på en skala fra 1-5. Oppdater kolonnen "category" med tallet. Ikke nøl med å ta kontakt dersom du har noen spørsmål.

NB! Dersom noe skal fjernes, marker med x. Dette er typisk kommentarer på et annet språk enn norsk
NB2! Det er viktig med ytringsfrihet i Norge, og alt som er knyttet til ytringer rundt politikk osv har høyere terskel for å oppfattes som ikke-nøytrale i en debatt. Det er helt lov å være sterkt imot ulike politiske saker, men det blir regnet som ikke-nøytralt når det bikker over i definisjonene vi har nedenfor.

Tips! Se for deg at du er en varslingsrobot. Hadde du reagert og ville sagt ifra til brukeren at de burde tenke over hva de har skrevet en gang til før de publiserer denne kommentaren?
Tips! Bruk de vedlagte eksemplene og følg retningslinjene, men om du fortsatt er i tvil anbefales det å gå for den mildeste klassifiseringen, altså lavere tall.

Table B.1.: Annotation guidelines

Markering	Kategori	Definisjon
1	Nøytral	Alt annet
Eksempler på nøytrale kommentarer: 1. Artig at det å påpeke hva som skjer i våre gater har blitt en forbrytelse. Også en smule morsomt at 1.7 milliarder muslimer kalles en minoritet mot Norges 4 millioner nordmenn 2. blir forbanna over at listhaug sier at spagetti er en norsk verdi eller noe sånt 3. Tenker begge kjønn ghoster men at menn som regel er de som kommer med møkkete svar. Uansett føles det vondt å bli ghostet, man er jo tross alt håpefull når man matcher med noen. Jeg er hvertfall det. 4. Gud hjelpes for en gjeng Mange kommentarer kan ofte oppleves å ikke være nøytrale hvis man tenker på kontekst eller andre underliggende faktorer. Det er viktig i dette prosjektet å analysere og annotere hver enkelt kommentar separert, og heller ikke ilegge forfatteren meninger som ikke står eksplisitt. Med dette mener vi ikke at ironi og sarkasme alltid skal tolkes nøytralt, men at kommentarer ikke skal tolkes i verste mening hvis den hadde blitt rettet mot en bestemt type gruppe eller i en viss setting. Les den som den står med andre ord.		
2	Provoserende	En ytring blir definert som provoserende om den inneholder et aggressivt språk for å uttrykke en mening eller kan oppfattes upassende. Dette inkluderer bruk av banneord, hersketeknikk, sarkasme og ironi for å dekredibilitere motstanderen.
Eksempler på provoserende kommentarer: 1. Oi så flink du er til å forenkle fremstillinger. Er det derfor du ellers ser samfunnet svart/hvitt? Media skriver om noe = løgn Gjevjon/Steigan etc skriver noe = ufeilbarlige sannheter... Mester i nyanser og dyptenkning er du vel ikke 2. @USER Hvordan er det mulig å hate mennesker så intenst? Hva i all verden har disse menneskene gjort denne gjengen? FRP er jo et råttent og superegoistisk parti. Det er høyreekstreme som står for terroren i dette landet ikke muslimene. Håper at folk begynner å se det nå 3. Ehhh ... om du er religiøs er du hjernevasket. Islam er en eneste stor sekt, hva annet? Sekten holder 1,5+ milliarder mennesker nedtrykt i patriarkalske strukturer, underutvikling og uvitenhet, ekstrem fattigdom ... hva godt gjør d for menneskeheten å sitte å pugge koranvers? 4. Nok en gang viser det seg at islamkritikk ikke tåles, og er dødsens farlig. Og at feminister og venstresiden™ heller bøyer seg for muslimer, enn å forsvare ytringsfrihet og likestilling.,#JeSuisMila		

5. Greta Thunberg er en BLØFF ! En stor løgn skapt kunstig av sosialister som blir betalt for å lyve !
6. “NRK - Norges Islamistiske fyrtårn”

I utgangspunktet kan det se ut til at kommentar nummer 5 er nøytral ettersom den stort sett er saklig og ikke kommer med nedsettende eller provoserende utsagn. Likevel kan den ikke klassifiseres som et saklig innlegg i debatt fordi den har sterkt negativt ladede ord som beskriver FrP. Selv om det er stor takhøyde for å diskutere politikk og ha ulike synspunkt er det fortsatt ikke greit å dekredibilitere FrP ved å bruke så aggressivt språk.

3	Støtende (“offensive”)	En ytring blir definert som støtende dersom den inneholder sårende, nedsettende eller obskøne (=vulgær, slibrig, ‘kjønnsord’) kommentarer, enten rettet mot en gruppe mennesker eller mot et individ.

Eksempler på støtende kommentarer:

1. Ja for bergens-feita, høyre SOSIALISTEN Solberg, gjorde ikke noe for å få ham slengt ut; er hun mascokist?? Han erklærte jo på NRK at han skulle drepe henne. Noe galt er det med feita
2. Vil bare hoppe inn her og si at du er ond og ræva, men først og fremst veldig dum, Someone20004. Dårlig fungerende hjerne! Og det må vi respektere, alle er ikke født med en ordentlig hjerne
3. Ingen respekt for de selvgode, snillisme-feminist-myndighetene våre...... Og hvem ville vel det?????,Disse kjerringene støtter tvert imot denne kriminelle "berikelsen"!!!!!!
4. Poenget me heila tingen va jo ka så va d stusta i England din fitte kuk
5. Milan bare endra tissen min fra stiv til halvkram, litt som å pule en feit hore #fetehorer #æsj #fårdenikkeopp

Generelt oppfattes obskøne ord, som fitte, kuk, pikk, pule og lignende, som støtende. Det kan finnes unntak, som her hvor det brukes til å kommentere innhold i et innslag som har vært på Dagsrevyen: *For mye pikk og fitte! -Det blir for mye? -Ja... for å si det pent.* Det som hovedsakelig skiller støtende kommentarer (kategori 3) fra moderat hatefulle (kategori 4) er at de ikke retter seg mot en av de nevnte gruppene/karakteristikkene i definisjonen nedenfor. Forskjellen mellom kategori 2 og 3 vil være alvorlighetsgraden, altså hvor nedsettende en kommentar er f.eks.

4	Moderat hatefullt	Ytringer som er helt eller delvis motivert av hat eller negative holdninger mot grupper eller individ, basert på etnisitet, religion, seksualitet, kjønn, alder, politisk syn, sosial status eller funksjonsnedsettelser. Ytringene oppfordrer ikke til handling, men de er et angrep på integritet og er sterkt nedverdigende.

Eksempler på moderat hatefulle kommentarer:

1. Begriper ikke at noen kan slippe unna med et mordforsøk som dette var - eller terror.mange media er også redde for å skrive at d er innvandrere - er på en måte unødvendig da alle overfall utføres av disse jævlene... #islam #overfall #mordforsøk
2. En ting er utlendinger, noe annet er radikal islam. Er mange typer kulturer innvandret til Norge feks, ingen problem med dem, kun berikelse, noe komplett annet er islam med økende andel radikaliserte, de må uten unntak vekk fra Vesten.
3. Ja, de flytter hit og vi får verre liv = klankultur, misogyni, høye økte offentlige utgifter, press på skole og helsevesen, gjengproblematikk, æresrelatert vold, og aller verst - en retardert, avskyelig religion! Bruker vi din logikk bør vel alle pakistanere få komme inn og bli?
4. @USER Lan Marie Berg er ei fitte som har rævpult det norske folk på beskjed fra idligere leder for MDG. Han har nok knulla henne så hardt at hun bare handler uten å tenke. Samme med Raymond Myrengen i AP. Han er en svær kukk som har vært i hele partiet som tar i mot svarte penger de

Ser man på de tre første kommentarene forstår man at dette er sterkt nedverdigende ytringer mot en gruppe som innehar spesielle karakteristikker, i dette tilfellet deres religion eller etnisitet. Kommentar nummer 4 er et eksempel på en ytring som er et angrep på integriteten til et individ og er sterkt nedverdigende. Det er verdt å merke seg at ingen av disse eksemplene oppfordrer til handling (call to action). Eksempler på handling kan være “Send dem ut av landet”, “skyt dem”, “håper du hopper fra en bro”, “finn deg en tbane og dett foran”, “repatrieres (=sende de ut av landet)”

5	Hatefult	Ytringer som er helt eller delvis motivert av hat eller negative holdninger mot grupper eller individ, basert på etnisitet, religion, seksualitet, kjønn, alder, politisk syn, sosial status eller funksjonsnedsettelser og som oppfordrer til handling basert på dette.

Eksempler på hatefulle kommentarer:

1. Send henne hjem til Somalia med hele familien!
2. Når du trenes opp i bomber, kutter. strupen på barn og kvinner. Da er du et hjernevasket sharia dyr. Og du er etnisk pakistansk. Bør du bli sendt til pakistan med ett lite bidrag. De klarer av slikt. Send hennes influensere dit også!
3. Alle som jobber og sliter og blir mishandlet og fornedriget av disse kriminelle monstrene er de samme som betaler regninga for deres liv i Europa. Send ALLE kriminelle tilbake til hjemlandet og FORBY ISLAM!

Disse kommentarene oppfordrer til en handling basert på ulike karakteristikker av grupper eller enkeltindivider, og det er dette som skiller kategori 4 og 5. Det er verdt å merke seg at ikke all oppfordring til handling kan klassifiseres i denne kategorien. I denne kommentaren: “Start med at sende kriminelle utlendinger hjem, og stopp innvandring fra ikke vestlige land.” er det oppfordret til å sende kriminelle utlendinger hjem og til å stoppe innvandring. Dette er politiske synspunkt som må være helt lov å komme med i debatten, fordi ytringen angriper ingen spesielle grupper eller sier noe nedverdigende om hvorfor innvandringen bør stoppes.

C. Collecting Social Media Content

This appendix presents the search words, pages and names used to collect and preprocess the annotated Norwegian dataset. The first section contains the search words used to collect tweets from Twitter, while the second section contains the Facebook pages that all used posts were found. In addition, this section lists all the first and last names that were removed from the dataset.

C.1. Twitter search words

The following list of words were used to collect Twitter messages using the Twitter API.

pikk	joede	mdg	transe	feminist
lgbt	heterofil	utrydde	reis	fitte
tispe	homse	reis	hjem	hore
parasitt	send	korrupt	islam	homo
hatprat	rasist	hat	muslim	

C.2. Facebook

C.2.1. Sites and posts

The following Facebook pages were browsed for posts to crawl:

VG	Frp	Dagens Næringsliv
TV2	Aftenposten	Nei til mer bompenger
TV2 Nyhetene	Venstre	Norge fritt for Islam
Grønn Ungdom	Dagbladet	Vi støtter Sylvi Listhaug, innvandring- og integreringsminister
NRK Nyheter	Siv Jensen	
Hege Storhaug	Sylvi Listhaug	
Se og Hør	Lan Marie Berg	

C.2.2. Names

This section contains two lists of first and last names, commonly used in Norway. The lists are based on two lists of names provided by *Statistisk Sentralbyrå (SSB)*, with additional manually added names found in the dataset.

First names:

aaron	aleksander	amir	annika	ava
abdirahman	aleksandra	amira	anniken	aven
abdul	alette	amna	anthony	axel
abdullah	alex	amund	anton	aya
abdullahi	alexander	amy	antoni	ayla
abel	alexsander	ana	antonio	aylin
abigail	alexandra	anas	are	ayman
ada	alf	anastasia	ari	aagot
adam	alfred	anders	arian	aage
adele	ali	andré	ariana	aase
adina	alice	andrea	ariane	adolf
adine	alicia	andreas	ariel	adrian
adnan	alicja	andrine	arild	agnes
adrian	alida	ane	arin	aksel
adriana	alina	anea	arman	albert
agathe	aline	anette	arn	alexander
agnar	alisa	angela	arna	alf
agnes	alise	angelica	arne	alfhild
agnete	alisha	angelika	aron	alfred
agnethe	allan	angelina	arthur	alma
ahmad	alma	anine	arvid	amalie
ahmed	alva	anisa	arve	anders
aida	alvar	anita	arvin	andre
aiden	alvilde	anja	aryan	andrea
ailin	alvin	ann	asbjørn	andreas
ailo	amadeus	annette	asgeir	anette
aimee	amalia	anna	ask	anita
aina	amalie	annabel	askil	ann
aisha	amanda	annabell	aslak	anna
ajla	amandus	anne	astri	anne
aksel	amar	anne-kristine	astrid	anne-lise
alan	amelia	anneli	atle	anton
alba	amelie	anneline	audun	arild
albert	amin	annelise	august	arne
albertine	amina	anne-berit	aune	arthur
albin	amine	annie	aurora	arvid

asbjørn
aslaug
astrid
aud
august
balder
bartosz
bastian
beate
beatrice
bella
ben
bendik
benedicte
benedikte
benjamin
berit
bernhard
bertine
betina
bettina
bianca
bilal
birgitte
birk
bjarne
bjørn
bjørnar
bo
brage
brede
brian
bror
brynjar
beate
benedicte
benjamin
bent
bente
benthe
berit
bernhard
bernt
bertha
birger
bjarne
bjørg
bjørn
bodil
borghild
brit
brita
britt
bård
børge
børre
camilla
carina
carine
carl
carlos
carmen
caroline
casper
caspian
cassandra
cathrine
cathrina
catarina
catalina
catharina
catharine
cecilia
cecilie
celin
celina
celine
cicilie
charlie
charlotte
chloe
chris
christer
christian
christiane
christin
christina
christine
christoffer
christopher
cindy
clara
colin
conrad
cornelia
cornelius
camilla
carina
cecilie
charlotte
dag
damian
dan
dani
danial
daniel
daniela
daniella
darin
david
dawid
denis
dennis
diana
didrik
dina
dominic
dominik
dorthe
dorthea
dag
dagmar
dagny
daniel
ea
ebba
edel
edin
edith
edvard
edvart
edvin
edward
edwin
egil
eileen
eilif
einar
eir
eira
eirik
eiril
eirill
eirin
eivind
eyvind
eivor
ela
eldar
elea
elen
elena
eli
eliah
elias
elida
elin
elina
eline
elisa
elisabeth
elise
eliza
elizabeth
ella
elle
ellen
ellie
ellinor
elliot
elsa
else
elvira
ema
emanuel
embla
emelie
emely
emil
emilia
emilian
emilie
emilija
emilio
emily
emina
emine
emir
emma
emmeli
emmelin
emmeline
emmy
emre
emrik
endre
enya
eric
erica
erik
erika
erle
erlend
erling
eskil
eskild
espen
ester
esther
eva
evan
evelina
evelyn
even

	fredrik	gøran	halvor	inger
edith	freddy	haakon	hanna	ingmar
edvard	frederikke	hafsa	hanne	ingri
egil	fredrikke	hallvard	hans	ingrid
einar	freja	halvard	harald	ingunn
eirik	freya	hallvar	harry	ingvild
eivind	frida	halvor	hege	ingvill
eli	fride	hamza	heidi	iqra
elin	frøya	hana	helene	iren
elisabeth	frøydis	hanna	helga	irene
elise	finn	hannah	helge	irine
ellen	frank	hanne	henriette	iris
elsa	frida	hans	henrik	isa
else	frode	harald	henry	isaac
emil	gabriel	hasan	hilda	isabel
emilie	gabriela	hassan	hilde	isabell
emma	gabriella	hauk	hjalmar	isabella
erik	gabrielle	hedda	hjørdis	isabelle
erlend	gard	hedvig	håkon	isac
erling	gaute	hege	håvar	isak
erna	geir	heidi	håvard	iselin
espen	georg	heine	håkon	ismail
ester	gina	helen	håvard	ivan
eva	gitte	helena	ian	ivar
even	gjermund	helene	iben	iver
fabian	glenn	helga	iben	ivo
fadi	gunhild	helge	ibrahim	ida
falk	gunnhild	helin	ida	inga
fanny	gunnar	helle	idun	ingebjørg
fatima	guro	helmer	idunn	ingeborg
felicia	gustav	hennie	ilyas	inger
felix	geir	henning	iman	ingrid
ferdinand	georg	henny	imre	ingvald
filip	gerd	henriette	ina	ingvild
filippa	grethe	henrik	inan	irene
fillip	gro	henrikke	ine	ivar
finn	gudrun	henry	ines	jack
fiona	gunda	herman	inga	jacob
franciszek	gunhild	hermann	inge	jakob
frank	gunn	hermine	ingolf	jakub
frederik	gunnar	hilde	ingebjørg	james
fredrick	gunvor	hugo	ingeborg	jamie
friedrich	gustav	hussein	ingelin	jahn

```
jan
janet
janett
janie
janicke
janikke
janine
janne
jannicke
jaran
jarand
jarle
jasmin
jasmine
jasper
jeanett
jeanette
jennie
jennifer
jenny
jens
jeppe
jesper
jessica
jim
jimmy
jo
joachim
joakim
joar
joel
johan
johann
johanna
johanne
johannes
john
johnny
jon
jonah
jonas
jonatan
jonathan
jone
josef
josefine
joseph
josephine
joshua
jostein
julia
julian
juliane
julianne
julie
julius
june
juni
justin
jakob
jan
janne
jarle
jeanette
jenny
jens
joakim
johan
johanna
johanne
johannes
john
jon
jonas
jonny
jorun
jorunn
josefine
julie
jøran
jørgen
jørgen
jørn
kacper
kai
kaia
kaisa
kaja
kajsa
kamil
kamilla
karen
kari
karianne
kariann
karin
karina
karine
karl
karolina
karoline
karsten
kaspar
kasper
kaspian
katarina
katharina
kathrine
katinka
katja
katrine
kaya
kayla
ken
kenneth
kent
keth
ketil
kevin
khalid
kian
kim
kine
kira
kjartan
kjell
kjersti
kjetil
klara
klaudia
knut
konrad
kornelia
kornelius
krister
kristian
kristiane
kristin
kristina
kristine
kristoffer
krystian
kyrre
kaare
karen
kari
karin
karl
karoline
katrine
kenneth
kim
kine
kirsten
kjell
kjell-remi
kjersti
kjetil
klara
knut
konrad
kristian
kristin
kristina
kristine
kristoffer
kåre
laiba
laila
lana
lara
lars
lasse
laura
laurits
lauritz
lavrans
lea
leela
leah
leana
leander
leif
leila
lena
lene
leo
leon
leona
leonard
leonardo
leonora
lerke
levi
lia
liam
liana
lilja
lilje
lilli
lillian
lilly
lily
lina
linda
linde
line
linea
linn
linnea
linus
lisa
lise
liv
```

	lærke	marit	melina	muna
liva	maciej	marita	melinda	mustafa
live	madeleine	marius	melisa	magne
livia	madelen	markus	melissa	magnhild
loke	madelene	marlene	melvin	magnus
lone	mads	marta	mia	malin
lotta	magdalena	marte	michael	maren
lotte	magne	martha	michaela	margit
louis	magnus	marthe	michal	mari
louise	mahad	marthine	michelle	maria
lovise	mahamed	martin	mie	marianne
luca	mahdi	martina	mika	marie
lucas	mai	martine	mikael	marit
lucia	maj	martinius	mikail	marita
lucy	maj-linda	martinus	mikal	marius
ludvig	maia	martyna	mikkel	markus
ludvik	maiken	marwa	mikolaj	marte
luis	maja	mary	mila	martha
luka	maksymilian	maryam	milan	martin
lukas	malaika	mateo	milena	martine
luna	malak	mateusz	milian	mary
lycke	malena	mathea	milja	mathias
lydia	malene	matheo	milla	mathilde
lykke	mali	matheus	mille	mats
laila	malin	mathias	milo	may
lars	marcel	mathilda	mina	merete
laura	marco	mathilde	mio	merethe
laurits	marcus	matias	mira	mette
leif	maren	matilda	miriam	michael
lena	margit	matilde	moa	mina
lene	margrete	mats	mohamed	mona
lillian	margrethe	matteo	mohammad	monica
lilly	mari	matteus	mohammed	morten
lina	maria	mattias	molly	nadia
linda	mariam	mattis	mona	najma
line	marianne	maud	monica	nanna
linn	marie	max	monika	naomi
lisbeth	mariel	maxim	monique	natalia
lisbet	mariell	maximilian	mons	natalie
lise	marielle	maximillian	morgan	nataly
liv	marina	may	morten	natan
lovise	mario	maya	muhammad	nataniel
ludvig	marion	medina	muhammed	natasha

```
nathalie
nathan
nathaniel
nellie
nelly
neo
nichlas
nicholas
nicklas
nickolai
niclas
nicolai
nicolas
nicolay
nicole
nicoline
niklas
nikola
nikolai
nikolas
nikolay
nikoline
nila
nils
nina
njål
noa
noah
noel
nojus
noor
nora
norah
nova
nikolai
nils
nina
nora
oda
odd
oddbjørn
oddvin
oddrun
odin
ola
olaf
olai
olander
olav
olava
ole
olea
oline
oliver
olivia
olivier
oliwia
oliwier
olve
omar
omer
oscar
oskar
othelie
othilie
otilie
ottar
otto
oda
odd
oddvar
ola
olaf
olaug
olav
olava
ole
olga
oline
oskar
ove
patrick
patrik
patryk
paul
paula
paulina
pauline
peder
pelle
per
pernille
peter
petra
petter
philip
phillip
pia
piotr
preben
paul
pauline
peder
per
petra
petter
pål
pål
rachel
rafael
ragna
ragnar
ragnhild
rakel
ramona
randi
rania
rasmus
ravn
rayan
raymon
raymond
rebecca
rebecka
rebekka
regine
remi
renate
renata
renathe
renatha
richard
rikke
rikard
robert
robin
rolf
ronja
rose
ruben
runa
runar
rune
ruth
ryan
ragna
ragnar
ragnhild
ragnvald
randi
rasmus
rebecca
reidar
reidun
rigmor
rita
roar
robert
robin
roger
rolf
ronny
roy
rune
ruth
sabrina
saga
sahra
said
sakarias
salma
sam
samantha
samira
samson
samuel
sander
sandra
sanna
sanne
sara
sarah
savannah
scott
sean
sebastian
selin
selina
seline
selma
selmer
serina
serine
severin
shawn
sienna
sigmund
sigbjørn
signe
sigrid
sigrun
sigurd
sigve
siham
silas
silja
silje
simen
simon
simone
sina
sindre
sine
sinisa
```

	szymon	thea	torkil	ulrik
siren	sander	thelma	torhild	ulrikke
siri	sandra	theo	torild	una
siril	sara	theodor	tormod	unni
siv	sebastian	therese	torstein	valentina
sivert	signe	theresa	tove	vanessa
sjur	sigrid	thilde	trine	vanja
snorre	sigurd	thomas	tristan	varg
sofia	silje	thor	trond	varman
sofie	simen	thora	troy	vebjørn
sol	simon	thorbjørn	trude	vegar
solveig	sindre	thorvald	truls	vegard
sondre	siri	tia	trygve	vemund
sonja	sissel	tian	trym	vera
sophia	siv	tilda	tuva	veronica
sophie	sofie	tilde	tyra	veronika
stefan	solveig	tilla	terje	veslemøy
steffen	sondre	tim	thea	vetle
stein	stein	timian	theodor	vibeke
steinar	steinar	tina	therese	vibekke
stella	stian	tindra	thomas	victor
stian	stig	tine	thorvald	victoria
stig	stine	tinius	thorild	vida
stina	susanne	tiril	thorhild	vidar
stine	svein	tirill	tina	viktor
storm	sverre	tobias	tobias	viktoria
sturla	synnøve	tom	tom	vilde
ståle	sølve	tomas	tommy	vilhelm
sumaya	tage	tomine	tone	vilja
sunniva	taha	tommy	tonje	viljar
susann	tale	tone	tor	vilje
susanna	tallak	tonje	tora	vilma
susanne	tara	tony	torbjørn	vincent
svein	tarald	tony-andré	tore	vinjar
svein-johnny	tarik	tor	torill	viola
sveinung	tarjei	tora	torild	vivian
sven	tea	torben	tove	valborg
svenn	teo	torbjørn	trine	vegard
sverre	teodor	tord	trond	veronica
syed	terese	tore	trude	vibeke
synne	teresa	torgeir	trygve	victoria
synnøve	terje	torje	turid	vidar
syver	thale	torjus	ulla	vigdis

	wilma	yosef	zara	øystein
vilde	wilmer	younes	zoe	øysten
vår	wenche	yousef	zofia	øistein
vårin	willy	yusra	zuzanna	øivind
weronika	yahya	yusuf	ådne	øyvind
wibeke	yasin	yvonne	åse	åge
wiktor	yasmin	zahra	åshild	ågot
wiktoria	yassin	zainab	åsmund	
wilhelm	ylva	zakaria	åsne	
william	yngve	zander	ørjan	

Last names:

aa	aarsand	adolfsen	alnæs	anglevik
aabel	aarseth	afzal	alsaker	anthonsen
aaberg	aarskog	aga	alstad	anti
aaby	aarsland	aglen	alsvik	antonsen
aabø	aarstad	ahlsen	alver	anwar
aadland	aarum	ahmad	alvestad	apeland
aae	aarvik	ahmadi	alvheim	arctander
aagaard	aarø	ahmed	alvær	arneberg
aagesen	aas	aker	amble	arnesen
aakre	aasberg	akhtar	amdahl	arnestad
aakvik	aasbø	akram	amdal	arnevik
aaland	aase	aksdal	amdam	arntsen
aalberg	aasebø	akselsen	amin	arntzen
aalvik	aasen	aksnes	amiri	arnøy
aam	aaserud	albertsen	amlie	aronsen
aamodt	aaseth	albrigtsen	amundsen	arshad
aamot	aasgaard	aleksandersen	anda	arvesen
aandahl	aasheim	alexandersen	andersen	asbjørnsen
aandal	aasland	alfheim	anderson	asghar
aanensen	aaslund	alfredsen	anderssen	asheim
aanerud	aass	alfsen	andersson	ashraf
aanes	abbas	algrøy	andorsen	ask
aanestad	abdi	ali	andreassen	aske
aanonsen	abdullah	allum	andresen	askeland
aanstad	abdullahi	alm	andvik	askevold
aardal	abelsen	almaas	anfinsen	askildsen
aarflot	abrahamsen	alme	angell	askim
aarhus	adam	almli	angelsen	askvik
aarnes	adan	almås	angeltveit	aslaksen
aarrestad	aden	alnes	angvik	aslam

```
asp
aspaas
aspelund
aspen
aspenes
asphaug
astrup
auestad
augestad
aukland
aulie
aunan
aune
aurdal
aure
aurstad
ausland
austad
austbø
austnes
austrheim
axelsen
aziz
baardsen
bach
backe
bakk
bakka
bakkan
bakke
bakkehaug
bakkejord
bakkeli
bakkelund
bakken
bakker
bakkerud
bakketun
bakstad
ballestad
balstad
baltzersen
bang
barka
barmen
barstad
barth
bartnes
bashir
bastiansen
bauer
bauge
baumann
baustad
bay
bech
beck
becker
begum
bekkelund
bekken
bekkevold
bell
belsvik
bendiksen
bendixen
bengtsson
benjaminsen
benonisen
bentsen
bentzen
berdal
berentsen
berg
bergan
berge
bergem
bergene
berger
bergersen
bergerud
bergesen
berget
berggren
bergh
bergheim
bergland
bergli
berglund
bergman
bergmann
bergo
bergquist
bergseng
bergset
bergseth
bergstrøm
bergsvik
bergtun
bergum
berisha
berland
bernhardsen
berntsen
berntzen
berre
berstad
bertelsen
berthelsen
bertheussen
betten
bhatti
bibi
bilstad
birkedal
birkeland
birkeli
birkelund
birkenes
bjelland
bjerga
bjerk
bjerkan
bjerke
bjerkeli
bjerkeland
bjerkelund
bjerkestrand
bjerkli
bjerknes
bjerkvik
bjerkås
bjordal
bjorøy
bjune
bjønnes
bjønness
bjørdal
bjørgan
bjørge
bjørgen
bjørgo
bjørgum
bjørheim
bjørk
bjørke
bjørkedal
bjørkhaug
bjørkheim
bjørkli
bjørklund
bjørkmo
bjørkum
bjørkås
bjørlo
bjørn
bjørndal
bjørndalen
bjørnerud
bjørnes
bjørnestad
bjørnevik
bjørnsen
bjørnstad
bjørnå
bjørnø
bjørnøy
bjørseth
bjørsvik
bjørvik
blakstad
blindheim
blix
blom
blomberg
blomvik
blystad
bodin
boge
bogen
bolme
bolstad
boldvik
bones
bonsaksen
borch
borchgrevink
bore
borg
borgan
borge
borgen
borgersen
borlaug
borsheim
bostad
botnen
botten
boye
braaten
braathen
bragstad
brakstad
brandal
brandt
brandtzæg
brandvik
braseth
brastad
brataas
bratberg
bratland
bratli
bratlie
```

```
bratsberg
brattli
brattås
bratvold
braut
bredal
bredesen
breen
breiland
breistein
breivik
brekka
brekke
brekken
bremnes
bremseth
brenden
brenna
brenne
brevig
brevik
bringedal
brobakken
broch
brodahl
broen
brovold
brown
brox
bru
bruaset
brubakken
brudevoll
brudvik
bruheim
bruhaug
bruland
brun
brunhaug
brunborg
brunes
brunstad
brunvoll
bruseth
brustad
bruun
bruvik
bruvoll
bryhn
bryn
bryne
brynildsen
brynjulfsen
bråten
bråthen
brække
brækken
brænd
brænden
brøndbo
brønstad
bu
buan
bue
buene
buer
bugge
bugten
bui
buljo
bull
bunes
buraas
burås
burud
busch
buseth
busk
butt
buvarp
buvik
by
byberg
bye
byrkjeland
bårdsen
bækken
bærheim
bævre
bø
bødtker
bøe
bøen
bøhler
bøhn
børresen
børseth
børsheim
børstad
børve
bøthun
bøyum
cappelen
carlsen
carlson
carlsson
caspersen
celius
chan
chaudhry
chen
christensen
christiansen
christoffersen
christophersen
clausen
claussen
corneliussen
daae
dagestad
dagsland
dahir
dahl
dahlberg
dahle
dahlen
dahlgren
dahlstrøm
dalby
dale
dalen
dalene
daleng
dalhaug
dalheim
dalland
dalseth
dammen
dang
danielsen
davidsen
dehli
derås
devik
devold
didriksen
diesen
digernes
digre
dimmen
dirdal
ditlefsen
djupvik
djuve
djønne
do
doan
dokken
drabløs
drage
drageset
dragland
dramstad
drange
drevland
dreyer
drivenes
drønen
due
duong
dvergsdal
dyb
dybdahl
dybdal
dybvik
dybwad
dyngeland
dyrdal
dyrhaug
dyrkorn
dyrnes
dyrseth
dyrstad
dyrøy
dørum
ebbesen
eckhoff
edland
edvardsen
edvartsen
eeg
eek
ege
egeberg
egeland
egeli
egenes
eggan
egge
eggen
eggum
eid
eide
eidem
eidet
eidissen
eidsheim
eidsvik
eidsvåg
eie
eik
eike
eikeland
eikemo
```

```
eiken
eikenes
eikrem
eikås
eilertsen
einan
einarsen
einvik
eira
ek
ekeberg
ekeland
ekeli
ekelund
ekerhovd
ekker
eklund
eknes
ekre
ekrem
ekren
ekroll
ekstrøm
elde
elden
eldevik
eliassen
ellefsen
ellingsen
elmi
elnes
elstad
elton
eltvik
elvebakk
elvenes
elverum
elvestad
elvik
emanuelsen
emberland
emblem
emilsen
endal
endresen
enersen
eng
engan
engdal
enge
engebakken
engebretsen
engebråten
engedal
engeland
engelsen
engelstad
engelund
engen
engenes
enger
engeset
engeseth
engevik
engh
englund
engstrøm
engum
engvik
enoksen
ensrud
erdal
erga
erichsen
eriksen
eriksrud
eriksson
erikstad
erland
erlandsen
ernstsen
ersland
erstad
ertsås
ervik
eskedal
eskeland
espe
espedal
espeland
espelid
espenes
espeseth
estensen
evanger
evenrud
evensen
evenstad
evertsen
evje
evjen
evju
fadnes
fagerbakke
fagereng
fagerhaug
fagerheim
fagerland
fagerli
fagermo
fagernes
fagertun
fagervik
fagervold
fagerås
falch
falck
falk
fallet
fanebust
farah
fardal
farestveit
farstad
fauskanger
fauske
feragen
ferkingstad
ferstad
fevang
fidje
fidjeland
figenschau
figenschou
fiksdal
fimland
fimreite
finne
finnerud
finnesand
finnøy
finseth
finsrud
finstad
fischer
fiske
fiskerstrand
fiskum
fiskvik
fiskå
fjeld
fjeldberg
fjelde
fjeldheim
fjeldstad
fjell
fjellanger
fjelldal
fjellestad
fjellheim
fjellstad
fjellvang
fjermestad
fjær
fjæreide
fjærestad
fjærli
fjørtoft
flaa
flataker
flatebø
flaten
flatland
flatmo
flatås
flatøy
fleischer
flem
flesland
flo
flåten
flø
flønes
fløtre
fløysvik
foldnes
folgerø
folkedal
folkestad
folkvord
folland
follestad
fonn
fonnes
forberg
forbord
forland
formo
fornes
forsberg
forseth
forsland
forsmo
fosen
foss
fossan
fossdal
fosse
fossen
fosshaug
fossheim
fossland
fossli
fossmark
```

fossmo
fossnes
fossum
fotland
frafjord
framnes
frantzen
fredheim
fredriksen
fremstad
fretheim
friberg
friestad
frigstad
friis
frisvold
frivold
frogner
frostad
frydenberg
frydenlund
frøiland
frøland
frøseth
frøshaug
frøyen
frøyland
frøysa
frøystad
fuglerud
fuglestad
fure
furnes
furre
furset
furseth
furu
furuheim
furulund
furunes
furuseth
fyhn
fykse
fylkesnes
fylling
færøy
følstad
førde
føreland
førland
førre
førsund
gaarder
gabrielsen
gamst
gangstad
garberg
garcia
garnes
garshol
garvik
gashi
gaup
gausdal
gaustad
georgsen
gerhardsen
gilberg
gilde
gilje
gill
gillebo
gimse
giske
giæver
gjedrem
gjelstad
gjelsten
gjelsvik
gjendem
gjengedal
gjengstø
gjerde
gjerdingen
gjerdrum
gjerløw
gjermundsen
gjernes
gjerstad
gjersvik
gjersøe
gjertsen
gjeruldsen
gjervik
gjesdal
gjessing
gjestad
gjestvang
gjøen
gjønnes
gjøsund
glad
gleditsch
glesnes
gloppen
glosli
goa
gomes
gomez
godø
gonzalez
graff
gram
gramstad
gran
granberg
grande
granerud
granheim
granli
granlund
granly
granmo
grannes
granum
granås
grav
gravdahl
gravdal
gravem
grebstad
green
gregersen
greve
grimelund
grimelund-kjelsen
grimsmo
grimsrud
grimstad
grinde
grinden
grindhaug
grindheim
grini
grongstad
groth
grotle
grov
groven
grude
grue
grytten
grødem
grønbech
grønberg
grøndahl
grøndal
grøndalen
grøneng
grønhaug
grønli
grønlie
grønlien
grønlund
grønn
grønnestad
grønnesby
grønnevik
grønning
grønningen
grønningsæter
grønseth
grønstad
grønvik
grønvold
grønås
grøstad
grøtan
grøterud
grøtte
grøtterud
grøtting
gudbrandsen
guddal
gudmestad
gudmundsen
gulbrandsen
guldahl
guldberg
gullaksen
gulli
gulliksen
gundersen
gunnarsen
gunnerud
gunnerød
gunnes
gunnestad
gussiås
gustad
gustafsson
gustavsen
gustavson
guttorm
guttormsen
gården
gåsland
haagensen
haakonsen
haaland
haarberg
haarr
haarstad
haave

haavik
habberstad
habbestad
haddal
haddeland
hadland
hafstad
haga
hagberg
hage
hageland
hagelund
hagen
hagenes
hagerup
hagland
haglund
haile
halden
haldorsen
hall
hallan
halland
hallberg
halle
halleland
halleraker
hallingstad
hals
halse
halsen
halseth
halstensen
halvorsen
halvorsrud
hamborg
hamilton
hammer
hammeren
hammersland
hammerstad
hammervold
hammerø
hamnes
hamre
hana
handeland
hannestad
hannisdal
hansen
hanssen
hansson
hanstad
haraldseid
haraldsen
haraldstad
haram
hareide
harestad
harkestad
harnes
harstad
harsvik
hartmann
hartveit
hartvigsen
hartviksen
hasan
hashi
hasle
hassan
hassel
hasselberg
hasund
hatland
hatlem
hatlen
hatlestad
hatlevik
hatløy
hatteland
hauan
haug
haugan
haugane
hauge
haugen
hauger
haugerud
haugland
haugli
hauglid
haugnes
haugo
haugom
haugsdal
haugseth
haugstad
haugsvær
haugum
haugvaldstad
haukaas
haukedal
haukeland
haukenes
haukland
haukås
hausken
haveland
haver
havik
havn
havnen
hay
hebnes
heen
hegdal
hegg
heggdal
hegge
heggedal
heggelund
heggem
heggen
heggernes
heggestad
heggheim
heggland
heggset
heggøy
hegland
hegna
hegre
hegstad
heia
heiberg
heide
heien
heier
heim
heimdal
heimstad
heistad
heitmann
helberg
heldal
helgeland
helgerud
helgesen
helgestad
helgheim
hella
helland
helle
hellebust
helleland
hellem
hellenes
helleren
hellerud
hellesø
hellevik
helliesen
hellstrøm
hellum
helmersen
helset
helseth
helstad
heltne
helvik
hem
hemmingsen
hemnes
henden
henningsen
hennum
henriksen
hepsø
herdlevær
herfindal
herheim
herigstad
herland
hermann
hermansen
hernes
hernæs
herstad
hervik
hesjedal
heskestad
hessen
hestad
hestenes
hestetun
hestnes
hestvik
hetland
hevrøy
hexeberg
heyerdahl
hidle
hilde
hildre
hildrum
hille
hilleren
hillestad
hilstad
hindenes
hinna
hjelde
hjelle

hjelm
hjelmeland
hjelseth
hjelvik
hjorth
hjortland
hjulstad
hjørnevik
ho
hoang
hoddevik
hodne
hodneland
hoel
hoem
hoff
hoffmann
hofseth
hofstad
hoftun
hognestad
hogstad
hojem
hokstad
hol
holand
holberg
holden
hole
holen
holgersen
hollund
holm
holmberg
holme
holmedal
holmefjord
holmen
holmgren
holmsen
holmstrøm
holmvik
holmøy
holsen
holst
holstad
holt
holta
holtan
holte
holten
holter
holtet
holth
holthe
holum
holvik
homme
honningsvåg
hop
hope
hopen
hopland
hordvik
horgen
horn
horne
hornnes
horntvedt
horpestad
horten
horvei
hoseth
hotvedt
hov
hovd
hovda
hovdal
hovde
hovden
hove
hoven
hovind
hovland
hult
hunstad
hurlen
husa
husabø
husby
husdal
huse
huseby
husebø
husevåg
hussain
hussein
hustad
hustoft
hustveit
husum
husøy
huus
huynh
hval
hvattum
hveding
hveem
hvidsten
hylland
hynne
hystad
hågensen
håheim
håkonsen
håland
hånes
håpnes
hårstad
håvik
håvarsen
håvardsen
hægeland
hætta
høgberg
høgli
høglund
høgset
høgseth
høgås
høiberg
høiby
høie
høiland
høines
høiseth
høivik
hølland
høvik
hønsvik
høydahl
høydal
høye
høyem
høyer
høyland
høyvik
ibrahim
iden
idland
idsø
idsøe
igland
ihle
ihlen
ims
indahl
indergård
indrebø
ingebretsen
ingebrigtsen
ingvaldsen
innvær
iqbal
irgens
isachsen
isaksen
isdahl
isdal
isene
ismail
istad
iversen
jacobsen
jahnsen
jahr
jahren
jakobsen
jama
jansen
janssen
jansson
jensen
jensrud
jenssen
jentoft
jeppesen
jespersen
jessen
jevnaker
jevne
joa
joakimsen
johannesen
johannessen
johansen
johanson
johanssen
johansson
johnsen
johnson
johnsrud
johnstad
jonassen
jones
jonsson
jordal
jordalen
jordan
jordet
jordheim
josdal
josefsen
juell

juliussen
justad
juul
juvik
jåsund
jæger
jønsson
jørgensen
jørstad
jøssang
kaasa
kaland
kaldestad
kaldal
kalland
kalleberg
kallestad
kallevik
kalstad
kalvik
kanestrøm
kapstad
karim
karimi
karlsen
karlson
karlsrud
karlsson
karlstad
karlstrøm
karoliussen
karstensen
kasin
kaspersen
kaupang
kaur
kausar
kavli
kaya
khalid
khalil
khan
kielland
kihle
kiil
kildahl
kildal
kile
kilen
killi
kind
kinn
kirkeby
kirkerud
kirkhus
kirknes
kittelsen
kittilsen
kjeldsberg
kjelsen
kjelsnes
kjeldsen
kjellevold
kjelsberg
kjelsrud
kjelstad
kjerstad
kjesbu
kjos
kjær
kjærstad
kjærvik
kjølberg
kjølstad
kjørstad
kjørsvik
kjøsnes
klakegg
klausen
klaussen
klavenes
kleiv
kleiveland
kleiven
klemetsen
klepp
kleppa
kleppe
klette
klev
kleveland
kleven
klingenberg
klokk
kloster
klungland
klungtveit
klyve
klæbo
klæboe
kløvstad
knapstad
knarvik
knoph
knotten
knudsen
knutsen
koch
kofoed
kolberg
kolbjørnsen
kolle
kolltveit
kolnes
kolseth
kolsrud
kolstad
kolstø
kolås
kommedal
kongshaug
kongsvik
konradsen
koppang
kopperud
koren
korneliussen
korslund
korsmo
korsnes
korsvik
korsvold
kowalski
kraft
krane
krasniqi
kringstad
kristensen
kristiansen
kristoffersen
krog
krogh
krogsrud
krogstad
krogsæter
krohn
kroken
kronstad
krosby
kruse
kryger
kråkenes
kulseth
kumar
kvaal
kvaale
kvale
kvalheim
kvalnes
kvalsund
kvalsvik
kvalvik
kvalvåg
kvalø
kvaløy
kvam
kvame
kvamme
kvammen
kvamsdal
kvandal
kvande
kvarme
kvello
kvernberg
kverneland
kvernmo
kvia
kvien
kvilhaug
kvinge
kvinnesland
kvistad
kviteid
kvitseid
kvitland
kvåle
kværner
kydland
kyllingstad
kyvik
kårstad
kåsa
laastad
lade
lagesen
laland
lam
lamo
landa
landaas
lande
landmark
landro
landrø
landsem
landsverk
langaas
langbråten
langdalen
lange
langedal
langeland
langemyr

```
langen
langerud
langfeldt
langhelle
langholm
langli
langlo
langmo
langnes
langset
langseth
langvik
langåker
langås
langø
langørgen
langøy
larsen
larssen
larsson
lassen
laugen
lauritsen
lauritzen
laursen
lausund
lauten
lauvrak
lauvås
lavik
le
lea
lee
lehmann
lehn
lehne
leidland
leikanger
leiknes
leikvoll
lein
leine
leira
leirdal
leirvik
leirvåg
leistad
leite
leithe
leivestad
leknes
lekve
lekven
lende
lenes
leonhardsen
leren
lervik
lervåg
lerøy
levang
li
lia
liabø
lian
liberg
lid
lidal
lie
lied
lien
lieng
lier
liknes
liland
lilleberg
lilleby
lillebø
lilleeng
lillegård
lillehagen
lillejord
lilleland
lillemo
lilleng
lillestøl
lilletvedt
lillevik
lilleås
lima
lind
lindahl
lindal
lindanger
lindberg
lindblad
linde
lindgaard
lindgren
lindhjem
lindholm
lindland
lindquist
lindqvist
lindseth
lindstad
lindstrøm
lindtveit
lindvik
lindås
linge
linnerud
liseth
liu
lium
liverød
ljosland
lockert
lode
loe
loen
lofthus
log
lohne
lokøy
lomeland
lona
lone
longva
lorentsen
lorentzen
lossius
lothe
ludvigsen
ludviksen
lund
lundal
lundberg
lundby
lunde
lundeby
lundekvam
lundemo
lunden
lunder
lundestad
lundgren
lundgård
lundh
lundquist
lura
lutro
lydersen
lygre
lyng
lyngmo
lyngstad
lyngvær
lynum
lysaker
lyse
lysgård
lysgaard
lyshaug
lysne
lyssand
lystad
lysø
låstad
lægreid
lærum
løberg
løchen
lødemel
løge
løken
løkke
løkkeberg
løkken
løkås
løland
lønning
løset
løseth
løtveit
løvaas
løvberg
løvdal
løver
løvik
løvland
løvli
løvlie
løvlien
løvold
løvoll
løvseth
løvstad
løvås
løyning
madland
madsen
magerøy
magnus
magnussen
magnusson
mahamed
mahamud
mahmood
malde
malik
malm
malmedal
malmin
```

malmo
malvik
mandal
mandelid
mannes
marcussen
marken
markhus
markussen
marstein
martens
marthinsen
marthinussen
martin
martinsen
martinussen
marvik
marøy
mathiassen
mathiesen
mathisen
matre
maudal
maurstad
mauseth
medhus
meek
mehl
mehlum
mehus
meier
meisingset
meland
melberg
melby
melbye
melgård
melheim
melhus
meling
mellem
mellemstrand
mellingen
mellum
melsom
melum
melvold
mentzoni
merkesdal
meyer
michaelsen
michalsen
michelsen
midtbø
midtgård
midthun
midtskogen
midttun
midtun
mikaelsen
mikalsen
mikkelborg
mikkelsen
milde
miljeteig
miller
minde
minge
mirza
misje
misund
mittet
mjelde
mjånes
mjøen
mjølhus
mjønes
mjøs
mo
moa
moan
moberg
moe
moen
mogen
mogstad
mohamed
mohammad
mohammadi
mohammed
mohamud
mohn
moholt
moi
moksnes
moland
molde
molden
moldestad
moldskred
molland
molnes
molstad
molteberg
moltu
molund
molvik
molvær
mong
monge
mongstad
monsen
monstad
mork
morken
morland
mortensen
morstad
moseng
moss
mossige
mostad
mostue
moum
muggerud
mundal
munkvold
munthe
muren
muri
mustafa
mydland
myhr
myhra
myhre
myhren
myhrer
myhrvold
myking
mykland
myklebost
myklebust
myller
myran
myrdal
myre
myren
myrene
myreng
myrekrok
myrhaug
myrland
myrseth
myrstad
myrvang
myrvold
myrvoll
mæhle
mæhlum
mæhre
mæland
mæle
mælen
møgster
møller
mørch
mørk
mørkved
mørland
naalsund
naas
nakken
narum
narvestad
natland
natvig
natvik
natås
naustdal
nedrebø
nedregård
negård
nekstad
nerdal
nereng
nergaard
nergård
nerheim
nerhus
nerland
nerli
nervik
nes
nesbø
nese
neset
nesheim
nesje
nesland
nesmoen
ness
nessa
nesse
nesset
nesvik
neteland
netland
neumann
neverdal
nevland
ngo
nguyen
nicolaisen
nicolaysen

nielsen
nikolaisen
nilsen
nilssen
nilsson
nissen
nistad
njærheim
nodeland
nodland
nome
nomeland
noor
norberg
nord
nordahl
nordal
nordang
nordanger
nordberg
nordby
nordbø
nordeide
nordeng
nordengen
norderhaug
nordgaard
nordgård
nordhagen
nordhaug
nordheim
nordhus
nordland
nordli
nordlie
nordlien
nordlund
nordmark
nordmo
nordnes
nordrum
nordseth
nordskag
nordskog
nordstrand
nordstrøm
nordsveen
nordtveit
nordtømme
nordvik
nordås
noreng
norheim
norland
norli
norman
normann
nornes
norum
nowak
nur
nybakk
nybakken
nyberg
nyborg
nybråten
nybø
nydal
nygaard
nygren
nygård
nyhagen
nyheim
nyhus
nyhuus
nyland
nylund
nymark
nymo
nymoen
nyquist
nyrud
nyseth
nystad
nysted
nystrøm
nystuen
nysveen
nysæter
nysæther
nyvold
nyvoll
nærland
nærø
næs
næss
nævdal
nøkleby
nørstebø
nøttveit
obrestad
odden
odland
oen
ofstad
oftedal
ohnstad
oksnes
okstad
olafsen
olaisen
oland
olaussen
olavesen
olden
olesen
oliversen
olsen
olsnes
olsrud
olsson
olstad
olsvik
olufsen
oma
omar
omdahl
omdal
omholt
omland
ommedal
ommundsen
onarheim
onstad
opdahl
opdal
opedal
opgård
ophaug
opheim
ophus
opland
oppedal
oppegaard
oppegård
opsahl
opsal
opseth
opstad
orre
orvik
os
ose
osen
oshaug
oskarsen
osland
osman
osmundsen
osnes
oterhals
otnes
ottem
otterlei
ottersen
otterstad
ottesen
ottosen
overvik
overå
ovesen
owren
paasche
palm
parveen
paulsen
paulsrud
paus
pedersen
persen
persson
petersen
pettersen
petterson
pettersson
pham
phan
pihl
pilskog
pladsen
plassen
polden
pollestad
poulsen
prestegård
presthus
prytz
qvale
raa
raaen
raastad
rabben
rafoss
rahimi
rahman
raja
raknes
rakvåg
ramberg
ramfjord
ramsdal
ramsland
ramstad
ramsvik

ramsøy
rana
randen
ranheim
ranum
rasch
rashid
rasmussen
ravn
ravndal
ravnås
raza
ree
reed
refsdal
refseth
refsland
refsnes
refvik
rehman
reiersen
reigstad
reime
rein
reinertsen
reinholdtsen
reistad
reitan
reite
reiten
rekdal
rekkedal
rekstad
reksten
remen
remme
remmen
remøy
reppe
reppen
repstad
resell
revheim
reyes
rian
ribe
richardsen
richter
riis
riise
riiser
rikardsen
rikstad
rimstad
rindal
rinde
rinden
ring
ringdal
ringen
ringheim
ringnes
ringstad
ringvold
risa
risan
risberg
risdal
rise
rishaug
risnes
rist
risvik
risøy
roald
robberstad
roberg
robertsen
robstad
rodal
roen
rognan
rogne
rognerud
rognes
rognli
rognlien
rognmo
rognstad
rogstad
rohde
rokne
rokstad
roksvåg
roland
rolfsen
rolfsnes
rolland
rolstad
romsdal
romstad
romundstad
rong
rongved
roos
rosenberg
rosenlund
rosenvinge
rosland
rosnes
ross
rossebø
rosseland
rossland
rostad
rosvold
roth
rovik
rud
rudshaug
rudi
rudolfsen
rue
rugland
rui
runde
rustad
rustand
rusten
ruud
ryan
rydland
rydningen
rye
ryen
ryeng
rygg
rygh
rykkje
ryland
rynning
rystad
rånes
ræstad
røberg
rød
rødal
rødland
rødseth
rødsjø
røe
røed
røen
røgeberg
røhne
røine
røise
røisland
røkenes
røkke
rømo
røneid
rønne
rønneberg
rønnestad
rønning
rønningen
røren
rørstad
rørtveit
rørvik
røsand
røsnes
røssland
røst
røstad
røste
røsten
røsvik
røv
røvik
røyland
røyrvik
røyset
røysland
saastad
sachse
saeed
safi
saga
sagli
sagbakken
sagen
sagmo
sagstad
sagstuen
said
sakariassen
sakshaug
salamonsen
salberg
salbu
saleh
salih
salomonsen
salte
saltnes
salvesen
samdal
samnøy
samuelsen
sand
sandaker
sandal

sandanger
sandbakk
sandbakken
sandberg
sandbæk
sande
sanden
sander
sandersen
sandhåland
sandland
sandli
sandmo
sandnes
sandsmark
sandstad
sandtorv
sandum
sandve
sandven
sandvik
sandvold
sandvoll
sandvær
sandø
sandøy
sangolt
sanne
sannes
sara
saue
saur
schanche
schanke
schau
schei
scheie
schie
schjelderup
schjerven
schjetne
schjølberg
schmidt
schou
schrøder
schultz
schøyen
seeberg
seglem
seierstad
seim
sekkingstad
sekse
seland
seldal
sele
seljeseth
selle
sellevold
sellæg
selmer
selnes
selstad
selvik
selvåg
sem
semb
seppola
serigstad
sevaldsen
severinsen
shah
shala
sharif
sharma
sheikh
sigurdsen
sigvartsen
sikveland
silden
silseth
silva
simensen
simonsen
singh
singstad
sinnes
sirevåg
sirnes
sivertsen
sjaastad
sjo
sjursen
sjåstad
sjåvik
sjøberg
sjøen
sjøgren
sjøli
sjølie
sjøstrøm
sjøthun
sjøvold
sjøvoll
skaar
skaare
skadberg
skage
skagen
skagestad
skailand
skansen
skar
skarbø
skare
skaret
skarstad
skarstein
skarsvåg
skarshaug
skau
skauen
skaug
skauge
skaugen
skavhaug
skei
skeide
skeie
skevik
skille
skinnes
skinstad
skistad
skjeggestad
skjelbred
skjelstad
skjelvik
skjerdal
skjerping
skjerve
skjerven
skjervold
skjetne
skjemstad
skjevik
skjold
skjolden
skjong
skjæret
skjærstad
skjærvik
skjæveland
skjølberg
skjønberg
skjønhaug
skjørestad
skog
skoge
skogen
skogheim
skogland
skogli
skoglund
skogly
skogmo
skogseth
skogsrud
skogstad
skogvold
skogvoll
skomedal
skorpen
skorstad
skotnes
skotte
skovly
skram
skramstad
skrede
skretteberg
skretting
skråmestø
skuland
skullerud
skulstad
skurdal
skålevik
skår
skårdal
skåre
skårland
skøien
sleire
sletner
sletta
slette
slettebø
sletten
sletthaug
slettum
sletvold
slinning
slyngstad
slåen
slåtten
smeby
smedsrud
smedstad
smeland
smestad
smistad
smith
smørdal

sneve
sogn
sola
solbakk
solbakken
solberg
soldal
solem
soleng
solevåg
solhaug
solheim
solli
sollid
sollie
sollien
solstad
solstrand
solsvik
soltvedt
soltveit
solum
solvang
solvik
solvoll
solås
soma
somby
sommer
sommerseth
sommerstad
sommervold
sortland
sparby
sperre
spilde
sporsheim
stabell
stadheim
staff
stamnes
standal
stang
stange
stangeland
starheim
stav
stava
stave
stavem
staven
stavenes
stavik
stavland
stavnes
stavrum
steen
steffensen
stein
steindal
steine
steinnes
steinsland
steinsvik
steiro
stenberg
stendal
stene
stenersen
stenerud
stenhaug
stensby
stensen
stenseth
stensland
stensrud
stenstad
stensvold
stenvik
stephansen
stiansen
stien
stigen
stjern
stokka
stokkan
stokke
stokkeland
stokken
stokkenes
stokland
stokstad
storbakk
stordahl
stordal
storebø
storeide
storesund
storhaug
storheim
storli
storlien
storm
stormark
stormo
stormoen
stornes
storsve
storsveen
storvik
storås
storøy
strand
strandberg
strande
stranden
strandheim
straume
stray
strøm
strømberg
strømme
strømmen
strømnes
strømsnes
strømstad
strømsvåg
stubberud
stuen
sture
styve
stålesen
støa
støen
stølan
støle
stølen
størdal
størkersen
størksen
størseth
støylen
suhr
sund
sundal
sundberg
sundby
sunde
sundet
sundgot
sundland
sundli
sundnes
sundseth
sundstrøm
sundt
svanberg
svanes
svanevik
svardal
svarstad
sveberg
svee
sveen
svela
svellingen
svendsen
svendsrud
svenkerud
svenning
svenningsen
svensen
svensli
svensson
sverdrup
sveum
sviggum
sviland
svindland
svingen
sværen
sylta
sylte
synnes
synnevåg
syrstad
systad
syversen
syvertsen
sæbø
sæland
sæle
sælen
sæter
sæterbø
sæterdal
sætermo
sæternes
sæther
sæthre
sætra
sætran
sætre
sætren
sævareid
sæverud
sævik
søberg
søbstad
sødal
søderholm
søfteland
søgaard
søgård

søiland
sølvberg
sømme
søndenå
sønderland
søndergaard
sønsteby
sønstebø
sønsterud
sørby
sørbø
sørdal
søreide
søreng
sørensen
sørgård
sørhaug
sørheim
sørhus
sørland
sørli
sørlie
sørmo
sørnes
sørum
sørvik
søvik
søyland
tafjord
tahir
takle
taksdal
takvam
talberg
tallaksen
tandberg
tande
tandstad
tang
tangen
tangstad
taraldsen
taranger
taule
teien
teig
teige
teigen
teigland
telle
tellefsen
tengesdal
tenold
terjesen
tesdal
tessem
tetlie
tharaldsen
theodorsen
thoen
thomas
thomassen
thommesen
thompson
thomsen
thon
thorbjørnsen
thoresen
thorgersen
thorkildsen
thorsen
thorshaug
thorsrud
thorstensen
thorsø
thorud
thorvaldsen
thrane
throndsen
thu
thue
thuen
thune
thunes
thygesen
thøgersen
tidemann
tiller
titlestad
tjelle
tjelta
tjemsland
tjessem
tjore
tjøstheim
tjøsvoll
tjøtta
tobiassen
todal
todnem
toft
tofte
tokerud
tokle
tollefsen
tomren
tomter
tonheim
tonning
tonstad
torbergsen
torbjørnsen
toresen
torgersen
torheim
torjussen
torkelsen
torkildsen
torland
tornes
torp
torset
torstensen
torsvik
torvik
torvund
totland
tran
trana
trengereid
trondsen
trones
tronrud
tronstad
trulsen
truong
trydal
træet
trøan
trøen
trønnes
tuft
tufte
tungland
tunheim
turøy
tvedt
tveit
tveita
tveitan
tveite
tveiten
tveito
tverberg
tverå
tvete
tveten
tveter
tyldum
tysnes
tysse
tøllefsen
tømmervik
tømmerås
tømte
tøndel
tønder
tønnesen
tønnessen
tønsberg
tørstad
tøsse
ueland
ugelstad
ugland
ulland
ulleberg
ullestad
ulriksen
ulset
ulseth
ulstein
ulven
ulvestad
ulvik
ulvin
ulvund
underhaug
undheim
undrum
unhjem
unneland
urdal
urke
ursin
urstad
ustad
utgård
uthaug
utheim
uthus
utne
utnes
utsi
utvik
vaa
vaage
vabø
vadseth
vagle
vaksdal
valberg
valde
valderhaug

valen
valla
valland
valle
vallestad
valseth
valstad
valvik
valø
vang
vangen
vangsnes
vanvik
varhaug
vartdal
vassbotn
vassdal
vatland
vatle
vatn
vatne
vea
vedal
vedvik
vedå
vedøy
vegsund
veiby
veivåg
veland
velde
velle
venås
vestad
vestby
vestbø
vestli
vestly
vestnes
vestre
vestrheim
vestvik
vestøl
vetrhus
veum
vevang
vevle
viddal
vie
vigdal
vigre
vik
vika
vikan
vikane
vikanes
vike
viken
vikene
vikse
vikøren
vilhelmsen
villanger
vindenes
vinje
visnes
vist
viste
vistnes
vo
vogt
vold
volden
voldsund
voll
vollan
volle
vollen
vonheim
vorland
vu
våga
våge
vågen
vågenes
vårdal
værnes
waage
waaler
wagner
wahl
wahlstrøm
walberg
walde
walderhaug
walle
wallin
walseth
walstad
wang
wangberg
wangen
wanvik
warholm
warsame
wathne
weberg
weiseth
welde
welle
wennberg
wennevold
wergeland
werner
wessel
westad
westberg
westby
westbye
westerheim
westerlund
westgaard
westgård
westlie
westrum
wetteland
weum
wibe
wiborg
wickstrøm
wie
wien
wiggen
wiig
wiik
wik
wiken
wiking
wiker
wiklund
wikstrøm
wilberg
wilhelmsen
willassen
wille
williams
willumsen
wilson
winge
winger
winje
winsnes
winther
wisløff
wist
wisth
with
wold
wolden
woldseth
wolff
woll
wollan
wong
worren
wright
wroldsen
wulff
wærnes
wøien
yagoub
yildirim
yildiz
yilmaz
yndestad
yri
ytreland
ytterland
ytterstad
yttervik
yusuf
zachariassen
zahl
zakariassen
zhang
zimmermann
ådland
ågotnes
åkre
åmodt
ånensen
ånestad
årdal
årnes
årseth
årvik
ås
åsebø
åsen
åsheim
åsland
ødegaard
ødegård
ødegården
øen
øglænd
øie
øien
øiestad
øiseth
økland
øksendal
øksnes
ølberg

```
ølmheim
ørbeck
øren
ørmen
østberg
østby
østbye
østbø
østebø
østensen
østenstad
østerbø
østerhus
østerud
østerås
østgaard
østgård
østhus
østli
østlie
østlund
østmo
østrem
østreng
østvang
østvik
østvold
øverby
øverbø
øvereng
øvergaard
øvergård
øverland
øverli
øverås
øvrebø
øvrelid
øvretveit
øvrum
øvstebø
øvstedal
øvsthus
øwre
øyan
øyberg
øye
øyen
øygard
øygarden
øynes
```