Master's thesis

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

Marie Andreassen Svanes & Tora Seim Gunstad

Detecting and Grading Hateful Messages in the Norwegian Language

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

NTNU
Norwegian University of Science and Technology

Marie Andreassen Svanes & Tora Seim Gunstad

Detecting and Grading Hateful Messages in the Norwegian Language

Master's thesis in Computer Science, Spring 2020

Data and Artificial Intelligence Group
Department of Computer Science
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Supervised by Herindrasana Ramampiaro

Abstract

Today, with the widespread use of social media and discussion forums, it has become easy to express one's opinion. Also, it has become increasingly difficult to maintain safe environments online and prevent cyberbullying due to the enormous amounts of user-generated data being published. At the same time, research on the detection of hateful and offensive utterances has grown substantially over the past years. Hate speech can be seen as a deliberate attack directed towards a target group based on their characteristics. A manual approach to filter and moderate such utterances is not efficient enough, and thus the field of automatic hate speech detection is becoming increasingly more important.

There are several challenges within the research field of hate speech detection, such as the lack of a universal definition of hate speech, no common dataset and that binary classification, which has been the most frequently used method recently, does not fit the real-world scenario. Therefore, the work conducted in this thesis aims at investigating methods on how to identify offensive utterances online using multiclass classification and grade them based on how offensive they are, in the hopes of achieving a more factual and neutral tone in online debates and forums.

In order to solve this task, a thorough literature review related to hate speech detection, existing data collections, and various classification methods have been conducted to attain valuable insights. Based on the knowledge obtained, a large dataset consisting of Norwegian comments from various sources was collected and annotated. This dataset is a significant contribution to the field of hate speech detection in Norwegian, seeing as an annotated dataset of this size did not previously exist.

Both classic models and deep learning models with word embeddings have been used to experiment with the dataset in an all-in-one approach and a two-step approach. The experimental results indicate that the two-step approach is advantageous when the goal is to detect as many non-neutral comments as possible. However, all the models struggled with distinguishing between categories and with the imbalanced dataset. The best result was achieved using a combination of classic and deep learning models. Based on this, there is a potential for future research to detect and grade offensive utterances with a deep learning model in the first step, combined with a classic model in the second step. Furthermore, there is a need for a balanced dataset that is extended with more comments in the most severe categories.

Sammendrag

I dagens samfunn har det blitt stadig enklere å uttrykke sin mening gjennom utstrakt bruk av sosiale media og diskusjonsfora. Det har også blitt vanskeligere å forebygge nettmobbing og opprettholde trygghet på Internett på grunn av de store mengdene med brukergenerert data som blir publisert. Samtidig har forskning innen deteksjon av hatefulle ytringer økt betraktelig de siste årene. Hatefulle ytringer kan bli sett på som et bevisst angrep rettet mot en målgruppe basert på dens karakteristikker. Å manuelt filtrere og moderer slike ytringer vil ikke være effektivt nok, og dermed har automatisk deteksjon av hatefulle ytringer stadig blitt et viktigere forskningsområde.

Det finnes flere utfordringer innen deteksjon av hatefulle ytringer, slik som mangelen på en universell definisjon av en hatefull ytring, et manglende felles datasett og at binær klassifisering, som tidligere har vært den mest brukte metoden, ikke representerer den virkelige verden på en god måte. Arbeidet i denne masteroppgaven har derfor som mål å undersøke metoder for å identifisere ytringer ved hjelp av multiklasse-klassifisering, og gradere dem basert på hvor støtende de er, i håp om å oppnå en mer saklig og nøytral tone i debatter og fora på nett.

For å løse denne oppgaven har vi derfor utført et grundig litteratursøk relatert til deteksjon av hatefulle ytringer, eksisterende datasamlinger og ulike klassifiseringsmetoder for å oppnå verdifull innsikt. Basert på denne kunnskapen ble et stort datasett bestående av norske kommentarer fra ulike kilder samlet inn og annotert. Dette datasettet er et betydelig bidrag til forskningsområdet, ettersom et datasett av denne størrelsen ikke tidligere har eksistert.

Både klassiske modeller og dype læringsmodeller med *word embeddings* har blitt brukt for å eksperimentere med datasettet i en alt-i-ett-metode og en to-stegsmetode. De eksperimentelle resultatene indikerer at to-stegsmetoden er fordelaktig når målet er å detektere så mange ikke-nøytrale kommentarer som mulig. Alle modellene slet likevel med å skille mellom de ulike kategoriene i det ubalanserte datasettet. Det beste resultatet ble oppnådd ved å kombinere klassiske og dype læringsmodeller. Basert på dette mener vi at det er et potensiale for fremtidig forskning å se på bruken av en dyp læringsmodell i første steg kombinert med en klassisk modell i andre steg. Videre er det også et behov for et mer balansert datasett som er utvidet med flere kommentarer i de groveste kategoriene.

Preface

This Master's thesis was written as a part of the master degree program in Computer Science at the Department of Computer Science (IDI), at the Norwegian University of Science and Technology (NTNU). The project preceding this thesis (Andreassen Svanes et al., 2019) was conducted during fall 2019 in collaboration with Maria Hilmo Jensen. In agreement with our supervisor, we have included some of the work that was conducted during fall 2019 because we all see the value of considering this to be a part of this thesis. The focus of this thesis has been detection and classification of the different degrees of the offensiveness of user-generated content in the Norwegian language using multiclass classification. We would like to thank Heri Ramampiaro for supervision and providing helpful feedback through guidance and discussions throughout the semester. Heidi Wyller, senior advisor at *Diskrimineringshjelpen og Meglingsbanken*, has also been an important resource for us in the process of defining the different categories of hate speech. We would also like to thank Dr. Basant Agarwal at Indian Institute of Information Technology Kota (IIIT Kota) for sharing his knowledge of deep learning models with us. Lastly, we would also like to express our gratitude towards fellow students and family who have volunteered to help us with annotation of the dataset.

Marie Andreassen Svanes & Tora Seim Gunstad

Trondheim, June 16, 2020

Contents

List of Figures

List of Tables

1. Introduction

Social media, discussion forums, and news media are something most of us use in our daily lives today. These platforms allow us to easily communicate, spread news, and express opinions, resulting in enormous amounts of user-generated data spread across the Internet. Unfortunately, not all of this user-generated data is considered friendly. Norsk Telegrambyrå found that almost 10% of comments published on the Facebook pages of *NRK* and *TV 2* were hateful, according to a report published by *Likestillings- og diskrimineringsombudet* in 2017[1]. With the growing amount of user-generated data, it is increasingly challenging to distinguish hateful and offensive language, especially without a formal definition in place. This makes it harder to moderate and detect hateful comments, even though identifying and moderating hateful comments is of high priority for newspapers, forums, and other social media platforms. Due to these mentioned difficulties, there are also challenges with regards to creating and annotating new datasets to develop better machine learning models.

This chapter discusses the background and motivation behind our study of hate speech detection and also identifies potential challenges connected to this field of research. The research questions we have focused on, our contributions and the research method we have used will also be presented.

1.1. Background and motivation

Freedom of speech is a principle that allows one to articulate an opinion without fear of retaliation, censorship, or legal sanction. It is recognised as a human right under Article 19 of the Universal Declaration of Human Rights[2] where it is defined as follows: "*Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers*".

In Norway, freedom of speech is considered one of the pillars of society. However, racist or otherwise hateful opinions are still illegal (Elden et al., 2018). Despite this, it seems like the threshold for writing hateful utterances online is lower, which might be due to anonymity and lack of moderation in online forums. A report from *Medietilsynet* in Norway (Medietilsynet, 2020) shows that 43% of Norwegian teenagers have seen hateful

[1] https://www.ldo.no/nyheiter-og-fag/brosjyrar-og-publikasjonar/rapporter/fb-rapport/
[2] https://www.un.org/en/universal-declaration-human-rights/

utterances that target vulnerable groups online in the past year. Furthermore, research conducted by *Megafon* on behalf of the Danish Institute for human rights (Zuleta and Burkal, 2017) found that more than half of Danish Facebook users refrain from expressing their opinions and participating in the online debate because of the negative tone. In addition, the research showed that most of the users expected Facebook and other news media to take more responsibility for the debates, with more intervention and supervision. To maintain safe environments online and to prevent cyberbullying, it is important to detect and remove hate speech. However, it is also crucial to not induce censorship of users since freedom of speech is a fundamental human right.

The process of manual moderation is becoming increasingly time-consuming for human annotators due to information overload. Automation of online moderation can save resources and contribute to a more neutral tone in the debate, allowing for more individuals to participate and express their opinion freely. According to Van Royen et al. (2014), expert moderators favour automatic monitoring, but with some conditions. Conditions include effective follow-up strategies and protecting both the commentators' privacy and their self-reliance.

Today, hate speech detection is an important field of study as a result of the massive increase of user-generated data and hateful comments online. Previous studies have focused mainly on dataset construction, text classification and automatic identification of cyberbullying events (Van Hee et al., 2015), as well as machine classification and statistical modelling for hate speech detection (Burnap and Williams, 2015). Simple surface features have been widely used, and models such as support vector machine, Bayesian logistic regression and similar classic machine learning methods have been used on self-made Twitter datasets. Some studies have used finance and news data from Yahoo to develop a hate speech corpus and train a machine learning model (Nobata et al., 2016). In more recent studies, many state-of-the-art approaches have used word embeddings, often in combination with deep learning models (Badjatiya et al., 2017; Zhong et al., 2016). Furthermore, transfer learning models such as BERT (Devlin et al., 2018) have also been frequently used to improve various methods further. A commonality for most of these studies is that they collect and label their own data, due to the lack of a common dataset.

A challenge within the field of hate speech detection is that most of the studies have thus far been conducted in English. Results have shown that the methods tested usually do not perform equally well when tested on a different dataset than the one they were trained on. This is a challenge for hate speech detection in other languages, such as Norwegian. Furthermore, most studies have applied a standard binary classification approach to the problem of hate speech detection. Traditionally, hate speech detection has focused on classifying whether an utterance is neutral or not, and what category of hate speech an utterance belongs to, i.e. racism or sexism. However, deciding whether an utterance is hateful or not can be more complicated than a basic yes/no question. Thus, multistep and multiclass approaches have been proposed since they are better equipped to distinguish between different levels of utterances.

Sentiment analysis and opinion mining are common tools in text classification to help decide "what other people think". The rating-inference problem is about determining the overall sentiment implied by the user and map such sentiment onto some fine-grained rating scale. An example of this is movie reviews, where the focus is on how a movie is rated by a consumer and how these ratings can be predicted. Instead of a binary classification task determining whether a review is either good or bad, rating-inference is about determining an author's evaluation to a multipoint scale, for instance from one to five "stars". An interesting aspect of this type of multiclass classification is that there are several different degrees of similarity between class labels, where an item classified as 4 on the scale is intuitively closer to an item classified as 3 than an item classified as 1. In the case of hate speech detection, it is natural to discuss whether or not to use tools such as sentiment analysis and opinion mining. Our approach is somewhat the opposite of the rating-inference problem with movie reviews because in our case, the system is rating the user instead of the user rating a movie. A challenge with sentiment analysis in hate speech detection is that a hateful comment can have positive sentiment, and a neutral comment can have a negative sentiment. This implies that to map the sentiment of each comment to a rating scale will not necessarily provide an accurate classification and thus possibly not filter out the hateful comments for moderation.

The motivation behind this thesis is to achieve satisfying results experimenting with a Norwegian dataset by developing a definition of various degrees of hate speech and use multiclass classification to detect hate speech. The focus will be on creating a method that will help moderators in their work by providing guidance to the users. The vision is that if a user is notified before posting a comment that it might be perceived as provocative, offensive or hateful, the user will rethink, reformulate and thus contribute to a healthier debate. To find the correct balance between freedom of speech and detection of hate speech in online debates appears to be a highly relevant and complicated question today. We are aware of the challenges with automated hate speech detection, such as misuse by directing our implementation at voices that do not express hatred or to suppress constructive criticism or opinions that dissent with a system, for instance, those criticizing a dictatorial regime. These concerns were also addressed by Saleem et al. (2017) when researching how to tackle hate speech in online social spaces. To use our findings and results in such a way mentioned above would be antithetical to our intention, which is to contribute to a healthier debate online by using a more neutral tone and facilitating more users to use their freedom of speech and participate in discussions.

1.2. Goals and research questions

The goal of this research project is to investigate methods on how to identify and grade offensive utterances online using multiclass classification, in the hopes of achieving a more factual and neutral tone in online debates and forums. Based on this, the overall research question can be formulated as follows:

Research question *How to identify offensive utterances and grade them based on how offensive they are?*

Both a theoretical and practical approach will be used to answer this. First, an extensive literature review will be conducted, building upon the literature review done in the project preceding this thesis. The practical approach will be based on the findings from this literature review and will include the creation of a Norwegian dataset and experiments on both standard text classifiers using multiclass classification and experiments with deep learning models. This objective is further divided into the three sub-questions below.

Research question 1 *Which existing methods and models have been effective when detecting offensive utterances this far?*

An extensive literature review was conducted during the project preceding this thesis (Andreassen Svanes et al., 2019) to obtain an overview of the state of the art within the field of hate speech detection. The findings from the previous review are amended with a discussion of several papers that have been studied during this research project. The findings will be analysed and used to decide which methods and models are most effective when detecting offensive utterances.

Research question 2 *How can multiclass classification be used to determine the different degrees of the offensiveness of user-generated content systematically?*

First, a review of state of the art within multiclass classification approaches will be conducted. The findings from this review will create the foundation for the development of our methods with multiclass classification. Next, several standard text classifiers will be implemented to serve as a baseline for later comparison. This provides useful data for measuring the effect of a two-step multiclass classification approach compared to a more traditional classification approach. Finally, several deep learning models will be implemented to see if this will improve the results further.

Research question 3 *How can we build a good dataset, in a specific language, for experimental evaluations?*

The construction of a Norwegian dataset will be based on the findings from the conducted literature review. Here, we will analyse the sources that have been used, how to select good search terms and known weaknesses in existing data collections. Multiple sources will be crawled to gather comments to create a more generalised and balanced dataset. Furthermore, text preprocessing and data cleaning adapted to the content of online comments will be performed. Annotation guidelines will be based on our findings from research on the annotation of hate speech. This includes experiences from annotation processes in previous studies and also consulting reports and professionals regarding the legal perspective of hate speech.

1.3. Contributions

The work conducted in this master's thesis will contribute with a deeper insight into the evolving field of hate speech detection. This includes a thorough, in-depth literature review with various perspectives on hate speech detection and a detailed overview of existing data collections and classification methods. Hopefully, an increased amount of research in this area will improve the methods that are currently adopted by online communities for monitoring and moderating user's comments. Furthermore, there is, to the best of our knowledge, neither available research done on Norwegian text nor a labelled dataset in Norwegian. Thus, it is our goal that this research can be used as a basis by other researchers. More specifically, the work conducted in this thesis has the following main contributions:

C1 A literature review on hate speech detection, including a detailed overview of existing data collections and classification methods.

C2 Building a Norwegian dataset from various sources, completely annotated.

C3 A definition of hate speech and offensive utterances using a 5-point scale.

C4 The implementation of several standard models and deep learning models using two different approaches to detect various degrees of the offensiveness of user-generated comments.

1.4. Research method

We have used several methodologies to accomplish the overall goal of this research project and to answer the research questions. First, a literature review on hate speech detection, creation and annotation of datasets and multiclass classification has been conducted, partly during the project that preceded this thesis (Andreassen Svanes et al., 2019). The findings went through a qualitative analysis to provide some of the answers to the three sub-questions. Based on this, a Norwegian dataset was created and annotated for experimental evaluations. Furthermore, an experimental research strategy has been followed throughout the work conducted in this thesis, where the experiments were designed to provide answers to Research question 2. The results from these experiments were qualitatively analysed by evaluating the distribution of correctly and incorrectly predicted comments and factors that impacted the classifiers. The work in this thesis will help gain experience and insight into the use of multiclass classification in hate speech detection through an exploratory design rather than to prove or disprove a hypothesis.

1.5. Thesis structure

Chapter 2 introduces relevant theoretical concepts and methods that are used in the field of hate speech detection, either in this thesis or in related work.

Chapter 3 provides an overview of state of the art within the field of hate speech detection with regards to dataset creation, annotation procedure and multiclass classification, and provides an overview of existing data collections and classification methods.

Chapter 4 provides a definition and examples of the varying degrees of hate speech.

Chapter 5 describes the architecture and implementation of the developed models.

Chapter 6 presents the creation of the dataset, annotation procedure and experimental setup, including the results from the different approaches and models used.

Chapter 7 evaluates and compares the experimental results presented in the previous chapter.

Chapter 8 discusses interesting aspects of our research, both in light of the state of the art analysis and with regards to the research questions presented.

Chapter 9 concludes the thesis by summarizing the research contributions and also includes suggestions for future work.

In addition, four appendices with additional information is included.

2. Background theory

This chapter provides the necessary background information to understand the concepts discussed in the related work in Chapter 3, as well as the architecture, the experimental approach and evaluation and discussion of the results presented in Chapter 5, Chapter 6 and Chapter 7. It is intended to provide an overview of the basic theory on which the methods presented in this thesis is based on.

First, in Section 2.1, machine learning and its traditional approaches are explained, before deep learning is described in Section 2.2. The rest of this chapter describes various relevant topics such as natural language processing, text analysis, text classification, evaluation methodologies and the tools and libraries used in this project.

The project preceding this thesis (Andreassen Svanes et al., 2019) was conducted in collaboration with fellow student Maria Hilmo Jensen. The main goal of the project was to research and give an overview of relevant background theory for this master's thesis. Most of the background theory was written during fall 2019, and an identification of the relevant background material was carried out during spring 2020. This is now amended with further elaboration of some topics and the inclusion of new ones as seen fit by the project description of this master's thesis.

2.1. Machine learning

The following sections will present the different types of learning algorithms used in machine learning and standard machine learning models. The work with the background theory in subsection 2.1.1 - 2.1.5 has been carried out in the project preceding this thesis (Andreassen Svanes et al., 2019).

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). 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. However, machine learning algorithms can solve many of these difficult 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 "has shown to be a decisive component in the field of text classification because of their simplicity and versatility", which is why we have focused on machine learning models for our research.

2.1.1. Types of learning algorithms

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, Semi-supervised Learning* and *Reinforcement Learning*, which are presented in the consecutive sections.

Supervised learning

Supervised machine learning algorithms are used for predicting future events based on previously attained knowledge. This is achieved by building a mathematical model based on the analysis of a provided training set. The training set contains both the inputs and the known desired output, i.e., it is a labelled dataset. After sufficient training, new input data can be provided to the algorithm and based on the key characteristics the model will predict the most likely output (Rebala et al., 2019; Russell and Norvig, 2010).

Unsupervised learning

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 model can identify clusters or groups of similar data items, and the algorithms are mainly used in pattern detection, text clustering and descriptive modeling[1].

Semi-supervised learning

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. Clustering techniques are used to identify groups within the given dataset, and the few existing labelled data points within each group are used to provide labels to the other data points in the same cluster/group. 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).

Reinforcement learning

Regarding reinforcement learning algorithms, the system learns from a series of reinforcements, either in the form of rewards or punishments (Russell and Norvig, 2010). Based on how the system performed, it must decide for itself which event in the course of events was most responsible for the performance. In this way, the system can learn a method that interacts with its external environment by producing actions that give an interpretable result. Reinforcement learning is particularly useful in situations that involve changing environments or huge state space, e.g., multiplayer games or driving (Rebala et al., 2019). Trial and error search and the use of delayed rewards are the main characteristics of reinforcement learning. This gives the system the ability to decide the ideal behaviour for its internal state and a given context[1].

Learning algorithms in text classification

In the field of text classification, various learning algorithms can be used. However, supervised algorithms are typically used to solve problems such as classification and regression problems, and are therefore frequently used to classify text. In supervised text classification, a batch of text documents containing social media comments, news articles or other text will be annotated. This dataset is then used to train a model which will be able to classify new, unannotated text. Popular supervised learning algorithms are Bayesian networks, support vector machines and various deep learning models.

2.1.2. Logistic regression

There are two kinds of regression analysis techniques: linear regression and logistic regression (LR). The main difference between these is that linear regression categorises as a regression problem, while logistic regression categorises as a classification problem. Linear regression allows one to predict the future value of a continuous variable and assumes that the variables can be expressed as a linear function (Rebala et al., 2019).

LR is a supervised machine learning model used for classification that originated from the field of statistics. Generally, regression analysis is a statistical technique used for creating prediction models based on labelled datasets. The model was first proposed by Cox (1958) as a way of using regression analysis to determine discrete, categorical variables. Originally, 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 multinomial values.

The goal of LR is to find a model that is best suited for a given set of independent variables. Typically, LR is easy to implement and it uses a prediction function to do the classification. This function is called a logistic function, or a sigmoid function, and is shown in Equation 2.1.

[1] https://expertsystem.com/machine-learning-definition/

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

The logistic function maps any real-valued number into a number in the range $[0, 1]$. This number equals the probability of the input belonging to a class, where values close to 1 indicates one class while values close to 0 indicate the other class. When LR is used in machine learning, the value f is an estimated probability, and the logistic function is hence parameterised. This equation is shown in Equation 2.2.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.2}$$

The input vector x has a coefficient, θ_i, for each x_i, and the vector θ is what the learning algorithm aims at discovering. T means that the transpose of the vector is used.

2.1.3. Support vector machines

Support vector machines (SVMs) were first introduced by Cortes and Vapnik (1995) and are a popular approach used in solving supervised learning problems. They are particularly prominent when one does not have any prior knowledge about a domain (Russell and Norvig, 2010). SVMs often produce significant accuracy with less computation power, and they can be used for both regression and classification tasks. They try to solve problems by finding a hyperplane in an N-dimensional space, where N is the number of features, that distinctly classifies all the data points. When separating data points from two classes, there may be several possible hyperplanes, but it is desired to find the plane that has the maximum distance between data points of both classes[2].

2.1.4. Decision trees

A decision tree is a decision support tool that represents a function for making a decision based on input data. The input is a vector of attributes, and the output is a single value, where both can either be discrete or continuous values. The decision tree has a tree-like structure where each internal node in the tree performs a test of the value of one of the input attributes. Furthermore, the branches from the node represent the outcome of the test and are labelled with the possible values of the attribute. Each leaf node represents a target variable, which is the value to be returned. If these target variables are class labels, the tree is a classification tree, while if the target values are continuous variables/numerical, it is called a regression tree (Russell and Norvig, 2010). The full paths from the root node to the leaves serve as the classification rules.

[2]https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

In decision tree learning, a decision tree is used as a predictive model to be able to go from an item's observations to a decision/conclusion about the item. This is an approach used in statistics, machine learning and data mining. Russell and Norvig (2010) states that the decision tree learning algorithm uses a greedy divide-and-conquer approach, where the attribute that makes the most difference to the classification is always tested first. As a result, the main problem can be divided into smaller sub-problems that can be solved recursively.

Random forest

Random forest is an ensemble learning method and a meta estimator that fits several decision tree classifiers on various sub-samples of the dataset. Each decision tree in the random forest make a prediction, and the class with the most predictions become the model's prediction. 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[3]. Since the trees have a low correlation between them, they will protect each other from the individual errors, thus improving the performance of the random forest as a whole. The method is used for classification and regression by using averaging to improve predictive accuracy. Random forest is also known to correct decision trees' disposition to overfit.

2.1.5. Gradient boosting

Gradient boosting is a technique that produces a prediction model in the form of an ensemble of weak prediction models. It can be used in machine learning, typically for decision trees[4]. Boosting is a method of converting weak learners into strong learners. The intuition behind boosting is to strengthen a model with weak predictions several times, thus improving it by refocusing on the examples that the previous prediction misclassified. Thus, each new tree is a fit on a modified version of the original dataset trained in a gradually and sequentially manner.

There are three elements in gradient boosting, namely a loss function to be optimised, a weak learner to make predictions and an additive model to add weak learners to minimise the loss function. Decision trees are used as the weak learner in gradient boosting, meaning that Gradient Boosting Decision Trees are decision trees that use the gradient boosting technique.

[3] https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[4] https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14

2.1.6. Learning to rank

Learning to rank (LTR), also known as machine-learned ranking, is a class of machine learning techniques applied to solve ranking problems (Li, 2011). A ranking problem is defined as a derivation of ordering over a list of examples that maximises the utility of the entire list[5]. The items can be sorted according to their degree of relevance, preference or importance. This differs from traditional prediction problems using classification or regression, where the aim is to find a class or a single numerical score for each item rather than focus on the relative ordering among all the items. To train the model, a list of items and a score for each of those items are used as training data. In the case of a search engine ranking, which is the most common application of LTR[6], this equals a list of results for a query and a relevance rating for each of those results.

Several LTR algorithms have been developed using mainly three different approaches (T.-Y. Liu, 2011): pointwise, pairwise and listwise. A common practice in the algorithms developed by Microsoft Research[6] is to transform the ranking problem into a pairwise prediction problem. That means that the algorithm compares a pair of items at a time, find the optimal ordering for that pair of items and then use it to find the final ranking for all items. The pointwise approach looks at a single item at a time, and train the model to predict how relevant it is for the current query. The final ranking is achieved by sorting the list of document scores. The listwise approach looks at the entire list of items and tries to find the optimal ordering for it. This is known as a more complex approach compared to the two others.

2.2. Deep learning

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

To extract features, deep learning builds representational hierarchies containing multiple abstraction levels. Goodfellow et al. (2016) describes 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 simpler concepts, and it is typically working on less complex 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.

[5] https://towardsdatascience.com/introducing-tf-ranking-f94433c33ff
[6] https://medium.com/@nikhilbd/intuitive-explanation-of-learning-to-rank-and-ranknet-lambdarank-and-lambdamart-fe1e17fac418

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. The black box problem is another challenge with deep learning models. It is based on the failure to produce a description of the results from the model, and the trouble with describing the most important features that led to the results. Thus, the model can be seen as a black box where parameters are given as input and the desired output is received, but one knows nothing of the process in general. There are several variations to deep learning models, and this section will briefly describe some of the models used in natural language processing which are relevant to our experiments. The work with the background theory in Section 2.2 has been carried out in the project preceding this thesis (Andreassen Svanes et al., 2019).

2.2.1. Artificial neural networks

Artificial neural networks (ANNs), also called multilayer 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:

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right) \tag{2.3}$$

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. Neural networks can be used to create supervised machine learning models for classification problems and are very useful for solving complex problems where LR does not produce accurate results (Rebala et al., 2019).

Feed-forward

A feed-forward network has its name because information only travels forward in the network. Hence, it 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. The hidden layers perform nonlinear transformations of the inputs by applying

weights and directing them through an activation function. 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 multilayer 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.

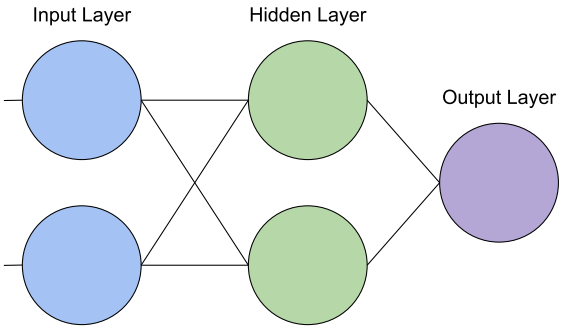

Figure 2.1.: Architecture of a feed-forward network with one input layer, one hidden layer and one output layer.

2.2.2. 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.

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 enable 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. J.-H. Wang et al. (2018) conducted experimental results which showed that given enough training data, the methods could learn the word usage in the context of social media. These findings can be useful for further experiments with text classification.

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, it is a technique for measuring the degree of overlap between two different figures. 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.

As opposed to regular multilayer networks, the first layers involved in convolution in a convolutional network are not fully connected. This means that all the nodes in one layer are not connected to all the 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 for text classification by splitting sentences into words and use as input. Next, the words are split into features and then fed into a convolutional layer. A representative number is given as output from pooling the results of the convolution and sent to a fully connected neural network. The classification

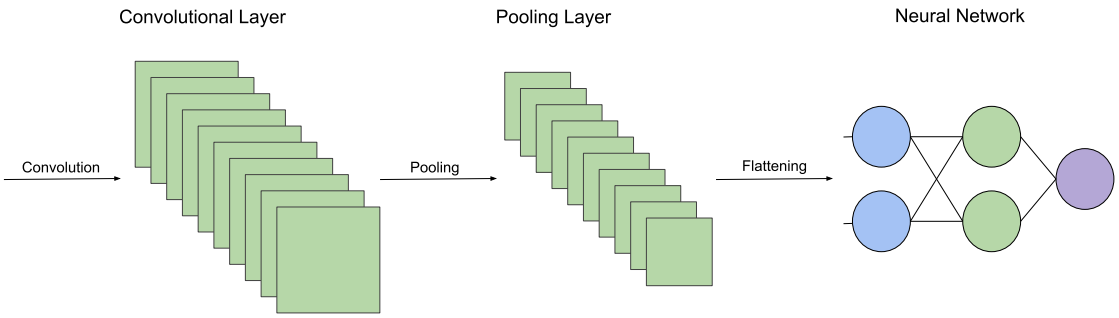

Figure 2.2.: A simplified architecture of a convolutional neural network. The network contains one convolution layer, one pooling layer and a fully connected neural network.

decision is then 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, whereas RNN is good for modelling orderly sequence learning problems.

2.3. Multistep classification

Multistep classification is a technique that can be used for a more efficient classification model and to further distinguish between categories. Assume a large set of features is to be applied to a dataset consisting of document-entity pairs. In general, it is inefficient and not feasible on a scale to compute these features for each pair. A solution to this problem is multistep classification. The classification task can be split into several smaller classification tasks to increase efficiency. When dealing with non-binary classification, a common approach is first to transform the task into a binary classification. An example of this can be first to classify whether a document is relevant or not. All the non-relevant documents can be discarded. Next, all documents that were classified as relevant can again be classified into new and more specific categories.

2.4. 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.

The work with the background theory in subsection 2.4.1 has been carried out in the project preceding this thesis (Andreassen Svanes et al., 2019). Subsection 2.4.2 is new and subsection 2.4.3-2.4.4 has been revised during spring 2020. In the following sections,

we will present topics such as various features used in NLP to achieve an understanding of the human language, and also how text classification can be used in the field.

2.4.1. Features

In text classification, 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 default for text classification is to use terms as features, but 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 number of characters, while the time an e-mail arrived can be an example of extrinsic information. Features related to text classification will be further elaborated in Section 2.4.3, while the rest of this section will focus on methods used for text representation and 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 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 BoW 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:

$$IDF_i = \log \frac{N}{n_i} \tag{2.4}$$

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

$$w_{i,j} = \begin{cases} (1 + \log\, f_{i,j}) \times \log\, \frac{N}{n_i} & \text{if } f_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

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, by looking n words into the past. 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 2.3.

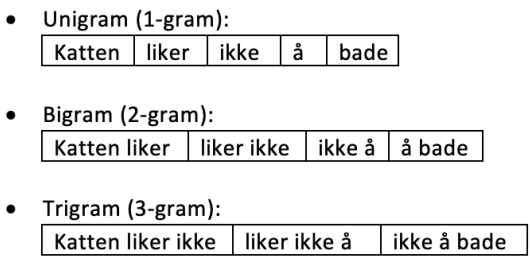

Figure 2.3.: Word n-gram representation.

2.4.2. Word embedding

Word embedding is a set of feature learning and language modelling techniques used to improve the ability of networks to learn from text data. A word embedding is a vector representation of words or phrases from a vocabulary. It is one of the most popular representations of document vocabulary because word embeddings can capture the context of a word in a document, its relation with other words, and semantic similarities. A word embedding model uses a text corpus as input and outputs word vectors. The vocabulary will be constructed from the training text data. Next, the model will learn vector representations of words resulting in a word vector file which can, for instance, be used as features in machine learning applications[7]. To capture context, distributed representations are generated, meaning that word embeddings introduce some dependence of one word on other words, contrary to a one hot encoded representation where all words are independent. To illustrate, consider the following sentences: "Ha en fin dag" and "Ha en god dag". The idea is that words with the same meaning, such as "fin" and "god" in these examples, 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. There are several commonly known word embedding methods in use today, such as Word2vec, GloVe and fastText.

Word2vec[8] is a word embedding model provided by Google (Mikolov et al., 2013). The Language Technology Group at the University of Oslo[9] has contributed to a variety of word representation models for different languages, including Norwegian. They provide Norwegian models for Word2vec, GloVe and fastText trained on a common crawl of different Norwegian corpora. The Norwegian Word2vec model resulted in a vocabulary size of 4 480 046 words. The mapping of words to vector size, known as dimension size, is 100. The window size equals the span used to determine the context of the words and is set to 5 in this case. This model provides an efficient implementation of two different learning algorithms, namely continuous bag-of-words, CBOW, and continuous skip-gram architectures. These algorithms are used to learn the representation of a word to make a

[7] https://towardsdatascience.com/what-the-heck-is-word-embedding-b30f67f01c81
[8] https://code.google.com/archive/p/word2vec/
[9] http://vectors.nlpl.eu/repository/

prediction of other words in the sentence. CBOW tries to predict based on a context window, whereas skip-gram predicts the word based on the surrounding words in the sentence instead of context.

GloVe, or Global Vectors for words representation, is a word embedding model provided by Stanford (Pennington et al., 2014). Their English word embeddings consist of various models from 25, 50, 100, 200 to 300 dimensions based on 2, 6, 42, 840 billion tokens[10]. The Norwegian model is trained on the same amount of words as for the Word2vec model, with a dimension size of 100 and window size of 15. This is an unsupervised learning algorithm where training is performed on aggregated global word-word co-occurrence statistics from a corpus. The idea behind the model is that ratios of these word-word co-occurrence probabilities have the potential for encoding some type of meaning.

FastText[11] is a library for efficient learning of word representations and sentence classification developed by Facebook (Bojanowski et al., 2017). They provide a Norwegian model with a vocabulary size of 2 million words, dimension size of 300 and a window size of 5. The model is making use of character-level representations using a hash table for both word and character n-grams. It represents each word as a bag of character n-grams in addition to representing the word itself. The fastText model from the Language Technology Group at the University of Oslo consists of 4 428 648 words and have a dimension size of 100, a window size of 5, and the model used is skip-gram.

2.4.3. Text classification

Text classification is the task of organising information by separating documents into a set of predefined classes or categories. The need for automatic text classification has increased in line with the growth of information found online. Text classification can be used to build personalised filters that learn about the preferences for each user or to deliver better search results. More formally, Baeza-Yates and Ribeiro-Neto (2011) defines it as follows:

Text classification: *if d_i is a document of the entire set of documents D and $\{c_1, c_1, ..., c_n\}$ is the set of all categories C, then text classification assigns one category c_j to a document d_i.*

Ranking classification and hard categorisation are two distinct types of text classification. The former occurs when a document is assigned to more than one category, known as a multilabel task, and the latter occurs when a document is assigned to a single category, known as a single-label task. Even though the above definition from Ikonomakis et al. (2005) states that F is a binary classification function, it is preferred within the field of information retrieval to compute a degree of membership for a document in a class instead. A structured view of all information can be achieved by associating one or

[10] https://nlp.stanford.edu/projects/glove/
[11] https://fasttext.cc/docs/en/crawl-vectors.html

more class labels with a document, which helps to understand and interpret data better (Baeza-Yates and Ribeiro-Neto, 2011).

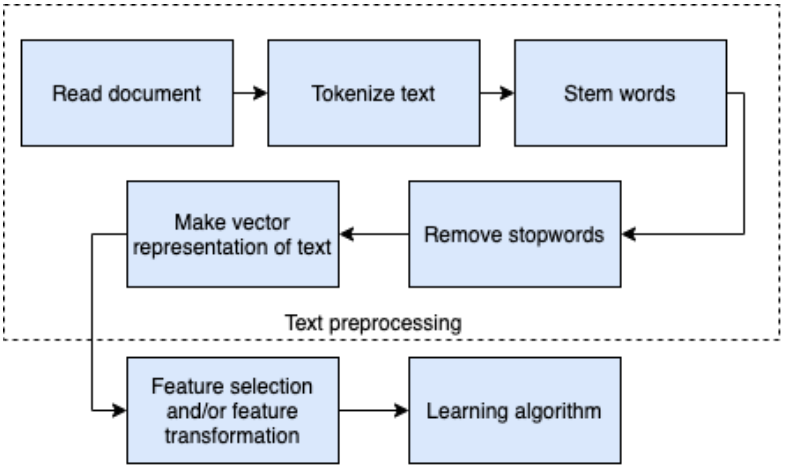

Figure 2.4.: The process of text classification (Ikonomakis et al., 2005).

The process of text classification is illustrated in Figure 2.4 and starts when a document is read. Next, the text is tokenized, 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. The tokenization 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 (Büttcher et al., 2016).

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).

Lemmatization 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 lemmatization is the word *saw*. While stemming might return only the character *s*, lemmatization 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, therefore these words are removed from the query at query time. 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. Examples of stopwords in the Norwegian language are "alle", "det" and "og" (Baeza-Yates and Ribeiro-Neto, 2011).

The next step in the text classification process is to make vector representations of the text. This is useful when the ordering of terms is relevant or when terms are repeated (Baeza-Yates and Ribeiro-Neto, 2011). Each term is represented by a vector filled with zeros. The index corresponding to the word in the text is represented by a one. These vectors can be used further as classification features. Word embeddings, as described in Section 2.4.1, is a method of representing text as vectors and can be used for feature selection.

The goal of feature selection for text classification 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).

The whole process of text classification is heavily influenced by machine learning, where learning algorithms such as clustering, decision trees and k-nearest neighbour are some of the best-known examples (Baeza-Yates and Ribeiro-Neto, 2011). It can be discussed whether or not all the steps in text preprocessing, as illustrated in Figure 2.4, is desirable when working with deep learning models For instance, when using models such as LSTM or other models which capture semantic meaning, it is not always advantageous to remove stopwords or apply word-stemming. This is because one might lose potentially valuable information and also because it depends on what kind of stopwords that are removed and how good the model can perform on noisy data.

2.4.4. Sentiment analysis

As one of the most common tools used for text classification, sentiment analysis determines whether the underlying sentiment towards a topic is positive, neutral or negative. It is often used to aid businesses to monitor brand and product reputation, conduct market research or gauge public opinions, also known as opinion mining. In addition, for a deeper understanding of real user opinions, intent analysis and Contextual Semantic Search can be performed[12].

[12]https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

In natural language processing and machine learning, sentiment analysis assigns a weighted sentiment score to either an entity, topic, theme or category found within the query being analysed. The process of sentiment analysis starts with splitting each document into smaller parts, such as sentences and tokens. Next, each component is analysed and sentiment-bearing ones, often adjective-noun combinations, are identified and assigned a sentiment score between -1 and +1[13].

2.5. Evaluation methodologies

When dealing with a machine learning problem, it is important to evaluate and test the performance of the implemented model. 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. Another factor to consider is that the model might give decent results when evaluated with one metric, e.g. accuracy score, but give poor results when evaluated with another. Hence, the performance of the machine learning model must be evaluated using several techniques and metrics.

Furthermore, it is also important to evaluate the annotation procedure. It is common practice to compare the annotations of a single instance done by multiple people when labelling a large dataset. This is done to validate and improve annotation guidelines, identify difficulties with the dataset and to defend not using several annotators on the rest of the dataset. A way of comparing annotator performance is to measure inter-annotator agreement. This score indicates to what extent the annotation process is consistent on the same source data (Artstein, 2017). This section describes the process of calculating inter-annotator agreement, techniques for evaluation of a machine learning model and also presents various metrics that are often used when evaluating classification systems.

2.5.1. Inter-annotator agreement

Fleiss' kappa is a statistical measure for calculating inter-annotator agreement. This method is used when there is a fixed number of raters who assign categorical ratings to a number of items. It is defined by Artstein (2017) as follows:

Fleiss' Kappa Let c be the number of annotators, and let n_{ik} be the number of annotators who annotated item i with label k. For each item i and label k there are $\binom{n_{ik}}{2}$ pairs of annotators who agree that the item should be labelled with k; summing over all the labels, there are $\sum_k \binom{n_{ik}}{2}$ pairs of annotators who agree on the label for item i, and agreement on item i is the number of agreeing pairs divided by the total number of pairs of annotators $\binom{c}{2}$. Overall observed agreement is the mean agreement per item, that is the sum of observed agreement for each item i divided by the total number of items $\mathbf{i}$.

[13] https://www.lexalytics.com/technology/sentiment-analysis

$$\text{Fleiss } \kappa \qquad A_0 = \frac{1}{\mathbf{i}c(c-1)} \sum_i \sum_k n_{ik}(n_{ik}-1) \tag{2.6}$$

Kappa values range from -1 to +1. There is no universally accepted interpretation of Fleiss' kappa, seeing as the number of categories and items annotated will affect the magnitude of the value. However, when $\kappa = 1$, a perfect agreement exists, and when $\kappa = 0$, the inter-annotator agreement is the same as would be expected by chance[14].

2.5.2. Techniques

The work with the background theory in Section 2.5.2 has been carried out in the project preceding this thesis (Andreassen Svanes et al., 2019).

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 consists 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).

2.5.3. Metrics

Evaluating the quality of the results from processed queries is known as retrieval evaluation. A quantitative metric is associated with the results produced when a user specifies a set of queries to an information retrieval system (Baeza-Yates and Ribeiro-Neto, 2011). Evaluation methodologies are used to make informed decisions 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).

[14] https://support.minitab.com/en-us/minitab/18/help-and-how-to/quality-and-process-improvement/measurement-system-analysis/how-to/attribute-agreement-analysis/attribute-agreement-analysis/interpret-the-results/all-statistics-and-graphs/kappa-statistics/

Accuracy

Accuracy is a simple evaluation of a classifier's performance, shown in Equation 2.7.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.7}$$

However, due to a possible imbalance in the dataset, a high accuracy does not equal a good model. Three types of measures can be used to compensate for this imbalance, namely precision, recall and F_1-score.

Precision and recall

Precision and recall are two of the most widely used evaluation metrics. The metrics are applied to an unordered set of documents retrieved with a given retrieval algorithm which processed a user query. Let *Rel* be the set of relevant documents, and *Res* the set of retrieved documents. Recall is the fraction of relevant documents that have been retrieved, shown in Equation 2.8.

$$Recall = R = \frac{|Res \cap Rel|}{|Rel|}. \tag{2.8}$$

Recall is used to evaluate the effectiveness of tasks where the user wants to find all relevant documents, for instance, legal search or literature reviews. This evaluation metric measures how meticulously the search results meet the user's information need. A potential flaw with recall is the fact that search engines can simply retrieve all documents to achieve a score of 1.0. However, this will result in a long list of non-relevant documents (Baeza-Yates and Ribeiro-Neto, 2011).

Precision is based on the assumption that a user wishes to find a reasonable number of relevant documents, for instance, in a Web search, and is defined as the fraction of retrieved documents that are relevant, shown in Equation 2.9.

$$Precision = P = \frac{|Res \cap Rel|}{|Res|}. \tag{2.9}$$

F-measure

F-measure, also known as the F_1-score, is an accuracy measure that is the harmonic mean between precision (P) and recall (R) metrics. It is defined as follows:

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}} = \frac{2 * R * P}{R + P}. \tag{2.10}$$

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.

There are three different ways of calculating the average F_1-score: micro, macro, and weighted averaging. Micro averaging sums the contributions from each class to compute the overall average, and is used when the goal for a classifier is to maximise hits and minimise misses. Macro averaging computes the average for each class unweighted and is often used in cases where the minority class is valued the most. Weighted average calculates the metrics for each class and uses the support of each class to calculated their weighted average.

ROC curve

A receiver operating characteristic (ROC) curve is a probability curve used as a performance measure for classifiers. The degree of separability between classes is measured in the area under the curve (AUC). The higher the AUC score, the better the model is at predicting the correct label (Büttcher et al., 2016). A model with a good measure of separability will have an AUC score closer to 1. Likewise, a poor model will have an AUC score closer to zero. An AUC score around 0.5 means the model cannot separate between classes, which equal random guessing.

2.6. Tools and libraries

Many tools and open-source libraries are developed in order to provide easily accessible and reusable components/modules. There exist many such tools and libraries for working with textual data, and they make the job more convenient rather than to implement the same module over and over again. Python was chosen to be the main programming language, primarily because of its extensive support for machine learning libraries. `Pandas`[15], The Natural Language Toolkit (`NLTK`[16]), Scikit-learn (`sklearn`[17]), `Keras`[18] and `TensorFlow`[19] are examples of such python libraries used in our work. Other tools and libraries that we used include The Twitter Developer API[20] and `LightGBM`[21].

[15] https://github.com/pandas-dev/pandas
[16] https://www.nltk.org/
[17] https://scikit-learn.org/stable/
[18] https://keras.io/
[19] https://www.tensorflow.org/
[20] https://developer.twitter.com
[21] https://lightgbm.readthedocs.io/en/latest/

3. State of the art

In this chapter, we discuss the current state of the art within the field of hate speech detection. This includes the existing research on hate speech detection, hate speech datasets, an overview of the current classification methods for both English and non-English languages, and lastly a look into multiclass classification. Hate speech detection is a popular field making it hard to cover everything. Hence, while we do not claim this investigation to be a comprehensive or representative review of all existing research for all languages, we aim at giving a good overview of previous work.

Parts of this work were started during the project proceeding this thesis (Andreassen Svanes et al., 2019) and have been reviewed during Spring 2020. Reformulations and new material, such as Section 3.4, are added in the following chapter where it seems fit.

3.1. Hate speech detection

Hate speech is a field which continuously increases in popularity, 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 big 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.

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. Schmidt and Wiegand (2017) underline that many different aspects might influence what is considered a hate speech message and Davidson et al. (2017) point out that different offensive terms often have cultural implications; a word can have different meaning to certain groups. Due to these difficulties, Ross et al. (2017) and Waseem (2016) investigate the difficulty of annotating data by analysing the reliability of the annotators and the difference between the annotation quality when the annotators have a definition or not or whether they are experts. Waseem (2016) found that amateur annotators are more likely than expert annotators to label an utterance as hate speech due to amateurs being more prone to subjectivity. If only cases of full agreement among amateur annotators are considered,

the results produced are comparable to expert annotators. This can reduce the amount of work required by an expert annotator since they would only need to consider samples where the amateur annotators disagree. Ross et al. (2017) concluded that the reliability did not improve when they were given a definition. However, both papers agreed that annotators should be given a more detailed definition of hate speech before annotating. This highlights the problem of subjective interpretation of hate speech.

Another problem for hate speech detection is the evolving of 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 increases. 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 field of hate speech. The first point of discussion was whether hate speech classification should differentiate only between hate and non-hate, i.e. binary classes, or multiple classes. As binary classification is considered simpler method-wise, this has previously been the most commonly used approach (Malmasi and Zampieri, 2017). However, as noted by Dinakar et al. (2012), models that are trained on hateful data often rely on the frequency of offensive or profane words to distinguish between the classes. Therefore, it is important to discriminate between hate speech and profane words. As a result, classifying multiple categories has recently become more popular through either multilabel classification (Malmasi and Zampieri, 2017; Sharma et al., 2018) or multistep classification (Park and Fung, 2017). This will be further elaborated in Section 3.4.

3.2. Existing data collections

Despite the proliferation of hate speech as a research field, one commonly accepted corpus does not yet exist. Because of this, authors usually have to collect and label their own data. The datasets created are often constructed specifically for the domain. Since the datasets have been constructed for different purposes, they may display different sub-types of hate speech and have special 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 are using 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 utterances is much lower than the number of neutral utterances. Thus, to get a sufficient number of hateful utterances, a larger number of comments has to been annotated. Furthermore, most of the datasets have not been made publicly available. One reason can be that due to the offensive language and profane words in 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 racist, sexist 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 contains a considerable proportion of neutral tweets, which makes it unbalanced. The authors state in their paper that this is intentional for a better real-world representation. Currently, a lot of the tweets are not available anymore 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 annotation through employing CrowdFlower[1] 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 certain lexical methods are effective to identify the potentially 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 it as non-hate speech.

Another publicly available dataset, published on Kaggle[2], contains around 150k Tweets where 16k are toxic. As Sharma et al. (2018) point out, this dataset only classifies whether an instance is toxic or not. Therefore they 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 that expanded the currently available datasets which consists of Waseem and Hovy (2016) and Davidson et al. (2017). More recently, A.-M. Founta et al. (2018) published a dataset containing 100k English tweets with cross-validated labels. This is more useful 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 seven state-of-the-art hate speech models using datasets from Waseem and Hovy (2016), Wulczyn et al. (2017) and Z. Zhang

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

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) emphasised; the lack of a definition of hate speech results in subjective differences when annotating datasets, which leads to the dataset being highly domain-specific. 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. (2019b) 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. Classification methods

Hate speech detection can be divided into classic methods and deep learning methods. As stated by Schmidt and Wiegand (2017), classic methods were mainly used previously, but more recently, neural networks and deep learning methods tend to outperform these methods. This section will present the current state of the art within the field of classification methods, including feature extraction, classic methods and deep learning. In addition, the current state of the art within hate speech detection for non-English languages will be covered.

3.3.1. Feature extraction

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. 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), and 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 widely used 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) looked into which features that are the most prominent when detecting hate speech and found that character n-grams contribute the most to the result. Mehdad and Tetreault (2016) investigated the difference by using word n-grams over character n-grams and 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. Gitari et al. (2015) built a list of hate verbs and more recently *hatebase.org*[3] has been widely used. Davidson et al. (2017) showed that for detecting hate speech, one could not rely entirely 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) tags which are employed in the feature set of 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 simple surface features, linguistic features and syntactic 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 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 frequently affiliated; Often negative sentiment belongs to a hateful utterance, and several approaches such as 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), which assigns 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 "hund" and "katt" may end up having a more similar vector than "hund" and "båt". There exist several popular word embedding models including Word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2019). Word embeddings have 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

[3]https://hatebase.org/

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 LR classifier, addressing the issues mentioned above 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 "lyst" would have different representation, whether it is used in the sentence "Det er lyst ute" or "Jeg har ikke lyst". 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, which will be further investigated in Section 3.3.3.

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 *multimodal 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 much 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.

3.3.2. Classic methods

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

LR and SVM have been the most popular choices among the classic methods within hate speech, as stated by Schmidt and Wiegand (2017). Waseem and Hovy (2016) used

LR in combination with extra-linguistic features and character n-grams to detect hate speech. Gaydhani et al. (2018) investigated an approach which combined n-gram and TF-IDF with NB, LR and SVM. LR outperformed the previous state-of-the-art methods by achieving an F_1-score of 0.96. 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. 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, which implies that classifying hate speech is a hard task. Lee et al. (2018) used the dataset from A.-M. Founta et al. (2018) to conduct the first comparative study using the most frequently studied classic machine learning and deep learning methods. For the classic methods they used 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 with a pre-trained GloVe word embedding model. The models performed more or less equal, with an F_1-score ranging from 0.73 to 0.81, 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, competing against deep learning methods. Nobata et al. (2016) and Sharma et al. (2018) also used a supervised model combining various surface and linguistic features.

As previously mentioned, supervised machine learning methods are often used as a baseline for comparison 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, classic 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 that SVM outperformed the other methods. In fact, it beat all other approaches for Task 6.2 of the PolEval 2019 Workshop (Ogrodniczuk and Kobyliński, 2019): multiclass classification of hate speech. The reason probably being that it was a small and unbalanced dataset which can make more complicated models overfit. Classic methods are often chosen and preferred due to simplicity and when there is a smaller amount of data.

3.3.3. Deep learning

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.

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 which will be further elaborated in Section 3.4. 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 the best. 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 an extensive 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 the 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 the current 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 model 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 competitive 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. A. M. 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 models.

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 prove to be more powerful than solely base on either. In hate speech, CNN is useful to extract word or character combinations (word embeddings), and RNN to learn word or character dependencies (order information). They hypothesise that a combined structure can be more effective

since 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, which also incorporated various user features. Their solution achieved a higher classification quality than the current state-of-the-art algorithms. One of these current state-of-the-art algorithms included Badjatiya et al. (2017). 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) suggested a fuzzy multistep method to classify, and it was compared to SVM, CNN and LSTM beating the current state of the art. Meyer and Gambäck (2019) tested 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 0.79.

Schmidt and Wiegand (2017) stated that there do 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.

Since 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. One of these was to identify and categorise offensive language in social media, with three sub-tasks, and Zampieri et al. (2019b) presented the results and the main findings. 104 teams submitted a result for the first sub-task where the goal was to discriminate between offensive and non-offensive posts. As can be seen in Figure 3.1, the most popular models involved deep learning approaches, and within these, there was a variation of models used, with 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.

Both Zampieri et al. (2019b), Badjatiya et al. (2017), Z. Zhang et al. (2018) and Fagni et al. (2019) showed that deep learning methods outperformed the more classical machine learning methods. Considering this, using deep learning for hate speech detection in combination with word embeddings and/or pre-trained language models seems like the most promising direction to go.

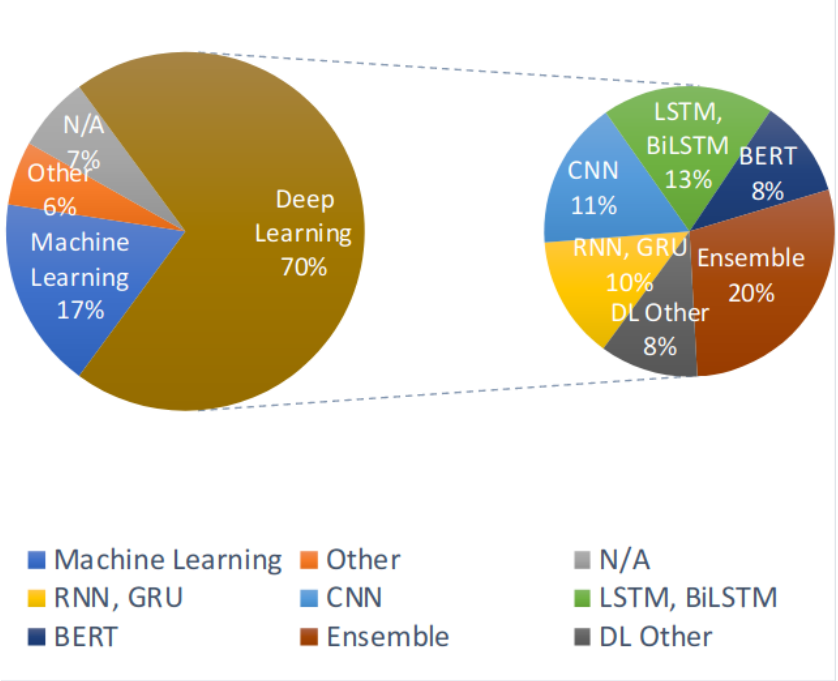

Figure 3.1.: Overview of models used in SemEval 2019 sub-task A (Wiegand et al., 2018).

3.3.4. Hate speech detection for non-English languages

Since our goal is to detect hate speech in Norwegian, it would be relevant to look into what approaches have been tested when detecting hate speech 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. This approach achieved an F_1-score of 0.55, 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[4] 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.

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

Sigurbergsson and Derczynski (2019) were the first to detect hate speech in Danish and are to the best of our knowledge the only ones who have done it for Scandinavian languages. The authors 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 consisted of classifying each post as either offensive or not. Sub-task B was to do an automatic categorisation 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 LR 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 achieved an F_1-score of 0.70 by using LR.

Several national semantic evaluation workshops for hate speech have lately taken place, which have focused on their respective languages. Wiegand et al. (2018), Bosco et al. (2018) and Ogrodniczuk and Kobyliński (2019) consist of various research papers in German, Italian and Polish. In general, many state-of-the-art approaches were explored as well as 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 was 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 0.65 (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.

3.4. Multiclass classification

There have been conducted several studies on the analysis of hate speech annotation and multiclass classification of hate speech with a varying number of annotation categories over the past years. Most of these studies suggest that a simple binary classification method is not complex enough to analyse hate speech. This means that to define a statement as either hateful or neutral is not nuanced enough with regards to the fact that there are several degrees of intensity of the utterance. Since our goal is to develop an effective method for hate speech classification using multiclass classification, it would be relevant to research how these studies developed their annotation guidelines and compare this with the results obtained, including studies on multistep classification as well.

Balog et al. (2013) address the cumulative citation recommendation (CCR) task for knowledge base acceleration (KBA) where the goal is to identify central documents, by looking into multistep classification approaches. The dataset used was gathered from three sources and annotated based on relevance (multiclass) and if it contained mentions of target entities (binary). Relevance was judged on a 4-point scale: *Garbage*, *Neutral*,

Relevant, and *Central*. The same initial step is shared by both the 2-step and the 3-step approach, which is to identify entity mentions in the documents. The 2-step approach classifies whether a document is central or not. The 3-step approach splits this task in two by first separating relevant and central documents from the garbage and neutral ones, and then distinguish between relevant and central documents. This is done by mapping a document score to a (0, 1000] range, where (0, 500] are the documents marked as garbage or neutral, (500, 750] is the relevant documents and (750, 1000] is the central documents based on the classifier's confidence values. Experimental evaluation was done using two decision tree classifiers, J48 and Random forest, two relevance levels, and two ways of determining confidence cutoffs. They report that both approaches performed very similar, making the 2-step approach the preferred choice given its relative simplicity.

As discussed in Section 3.3.3, Park and Fung (2017) also explore a two-step approach to see if dividing the problem space into two steps makes automatic detection of abusive language more effective. They compared a one-step approach of multiclass classification on sexist and racist language with a two-step approach which performed binary classification on abusive language and then classified into specific types. Their results indicated that the two-step approach can perform as well as the one-step approach, and is a way of boosting the performance of simpler models and combine different types of classifiers. For future work, they suggest training a two-step classifier on separate datasets. First, train the model on a large dataset with abusive language and then a smaller specific-labelled dataset, aiming to build a more robust and detailed abusive language detector.

H. Liu et al. (2019) conducted further research on multistep learning. They pointed out that the assumption that different classes are mutually exclusive does not always hold in real applications, and proposed a three-level framework for cyberhate detection. First, they did a polarity classification, determining whether something is hate or not hate, followed by a multitask classification for identification of different types of hate speech. The third level detected topics and context of hate speech. This framework is aimed at different abstraction levels of hate speech detection tasks to reduce the complexity of a single task on a single level. The study further focused on the multilabel classification of the different types of hate speech.

One of the earliest studies on multiclass classification was done by Kwok and Y. Wang (2013), who applied a supervised machine learning approach to detect anti-black hate speech in tweets. A balanced training set of 24 582 tweets was annotated as either racist or not. If a tweet was annotated as racist, a label for why it was found racist was assigned. These labels included *contains offensive words*, *reference to painful historical context*, *stereotypes* and *threatening*. Of the tweets that were racist, they found that 86% was due to the use of offensive words. This resulted in the use of unigram features when constructing the vocabulary. Next, they applied 10-fold cross-validation and achieved an average accuracy of 76%. In the end, they concluded that a simple BoW model is insufficient. Future work should include bigrams, sentiment analysis and further exploration of who the target of the hateful tweets is, to develop a better understanding

of hate and to improve the annotation guidelines.

Poletto et al. (2017) analysed an Italian Twitter corpus to understand hate speech annotation better. They designed a multiclass model with five categories which they applied to a dataset of 1828 annotated tweets. The five categories were *Hate Speech* (yes or no), *Aggressiveness* (no, weak or strong), *Offensiveness* (no, weak or strong), *Irony* (yes or no) and *Stereotypes* (yes or no). In order to classify a tweet as hate speech, two factors had to co-occur: the tweet was addressed to a minority group identified as a target group, and the utterance incited actions towards this target. They found that 16% of all tweets were annotated as hate speech. Also, they found that some labels co-occur more frequently with hate speech, and that was those indicating the presence of either aggressiveness, stereotypes or offensiveness, or all three combined.

The first hate speech classifier for Italian texts was introduced by Vigna et al. (2017), who proposed a variety of categories to distinguish what kind of hate a comment contained. A dataset consisting of 17 567 comments collected from Facebook was annotated with one of the following classes: *No hate, Weak hate* and *Strong hate*. Next, the hate messages were divided into distinct categories: *Religion, Physical and/or mental handicap, Socio-economical status, Politics, Race, Sex and gender issues* and *Other*. Two different classifiers were used, namely SVM and LSTM, and both sentiment polarity and word embedding lexicons were applied. They followed a 10-fold cross-validation process on two subsets of the original dataset. A three-class dataset composed of comments in the classes *No hate*, *Weak Hate* and *Strong hate* that had been annotated by at least three annotators, and a two-class dataset divided into *No hate* and *Hate*. The results showed that SVM performed best on the three-class dataset, but in general, both SVM and LSTM performed better on the two-class dataset. In conclusion, the classifiers performed better on the binary classification task.

At EVALITA[5] 2018, a periodic evaluation campaign of NLP and speech tools for the Italian language, nine teams participated and Bosco et al. (2018) describe the methods and results achieved. Two datasets from two different online social platforms, Twitter and Facebook, were used for different sub-tasks and thus annotated differently. Facebook comments were assigned to one of the following categories of hate: *No hate*, *Weak hate*, or *Strong hate*. Furthermore, the hateful (*Strong hate*) messages were again divided into distinct categories similar to those presented by Vigna et al. (2017). Tweets were annotated as either hate speech or not based on its intensity degree. Here, 0 indicated No hate, and 1 through 4 indicated some degree of hate speech. The winning team tested three different classification models: one based on linear SVM, one based on a 1-layer BiLSTM and one based on a 2-layer BiLSTM. Additional resources such as two word-embedding lexicons and polarity and subjectivity lexicons were also applied. This resulted in an improvement of the state of the art of hate speech detection in the Italian language.

Several studies have focused on developing a balanced corpus of hate speech combined

[5] evalita.it

with proper annotation guidelines. However, without a universal definition of hate speech, it is challenging to make these guidelines, and the reliability of annotations are often questioned. Golbeck et al. (2017) provided a large hand-labelled corpus of online harassment data, with approximately 15% of 35 000 tweets being examples of harassment. They gathered tweets by focusing on hashtags, derogatory terms and word structures that would result in a dense set of offensive tweets in the final dataset. Tweets annotated as harassing were further divided into sub-categories: *The Very Worst, Threats, Hate Speech, Directed Harassment, Potentially Offensive*, and *Non-Harassing*. However, they found that these sub-categories had a lot of overlap amongst them, so the final corpus does not include these labels. Nonetheless, the labels were useful to specify which types of content to look for. In the end, the tweets were annotated as either Harassing or Non-Harassing. The latter was a combination of the neutral tweets and the tweets labelled as Potentially Offensive. This was done because the annotators struggled with annotating things such as jokes in poor taste as non-harassing, even though it did not meet the criteria of the other labels. Again, this is an example of the difficulty of annotating potentially hateful text and the difficulty of separating different labels from one another.

A study conducted by Ross et al. (2017) researched the reliability of hate speech annotations, and found that the reliability was overall low. This lead to the conclusion that hate speech perhaps should be considered a multiclass classification problem rather than a binary classification problem. The research was done by collecting a German corpus from 10 hashtags that could be used in an offensive or insulting way. The annotation scheme consisted of two surveys with the same three steps. One group of annotators were given a definition of hate speech before annotating, and the other was not. First, annotators were asked to answer if they considered the tweet hate speech with either yes or no. Second, annotators were asked to answer either yes or no to if the tweet should be banned from Twitter. Lastly, they were asked to rate on a 6-point scale how offensive the tweet was. Label 1 indicated not offensive at all, and label 6 indicated very offensive. The results showed that on average tweets were rated as slightly more offensive by annotators with a given definition of hate speech than by those without a definition. To conclude, they stated that since hate speech is such a vague concept, it requires better definitions and guidelines to let annotators better distinguish hate speech of varying degree from other content. They suggest looking at hate speech as a regression problem, predicting the degree of the hatefulness of a message instead of as a binary classification task.

Sanguinetti et al. (2018) developed an annotation scheme specifically designed to account for the numerous factors that can contribute to the definition of a hate speech notion. This annotation scheme was then used to annotate about 6000 tweets for hate speech against immigrants. Five categories were defined, namely *Hate Speech* (yes or no), *Aggressiveness* (no, weak, strong), *Offensiveness* (no, weak, strong), *Irony* (yes or no) and *Stereotype* (yes or no), similar to Poletto et al. (2017). In addition, a sixth category was added after reviewing the annotation scheme. *Intensity*, which accounted for the degree of incitement that was present in the tweet, ranged from 1 to 4 for tweets annotated as Hate Speech, and 0 for the other ones. They found that stereotype, aggressiveness and

offensiveness were the categories that frequently co-occurred with tweets annotated as hate speech. Another finding was that most hateful tweets had an intensity of 1 or 2, meaning that the incitement was conveyed subtly. Few hateful tweets explicitly incited to violent or discriminatory actions, resulting in few tweets being annotated with intensity 3 or 4. This research contributed with a more complex annotation scheme, leading to more detailed and systematic analyses of possible linguistic patterns associated with the different categories.

Another interesting variable in the annotation of hate speech is how the individual annotator affects the results. Salminen et al. (2018) look into how the interpretation of hate speech varies by country and by the individual. They retrieved comments from a major online news and media company on Facebook and YouTube, and crowd workers were asked to annotate the comments using a 4-point scale: *Very hateful* (4), *Moderately hateful* (3), *Slightly hateful* (2), or *Not hateful at all* (1). The approach used was to fit a generalised linear model with a binomial distribution to predict the hatefulness rating using the *Country* variable. Results show that the Country variable is highly significant, with great distinction in proportions of answers rated as 3 or 4. However, the results also showed a significant amount of variance within countries, implying that the interpretation of hate is highly dependent on the individual annotator's definition of hate speech.

Most of the reviewed studies have either developed a classification model or gathered a dataset and annotated with non-binary labels. As previously mentioned, Sharma et al. (2018) have combined these tasks and created an ontological classification of harmful speech based on the degree of hateful intent. *Class I* is the most severe, with examples of violence, extremism and propaganda. *Class II* contains examples of threats and fear, intimidation and accuse. *Class III* is the mildest, with examples of bullying, trolling, irony and sarcasm. They performed experiments with three different classifiers using 10-fold cross-validation, where Random forest with a BoW approach performed best. To conclude, the authors propose considering hate speech detection as a regression problem rather than a classification problem, by giving scores to each tweet based on the degree of hateful intent rather than assigning it to a defined class.

3.5. Overview

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 3.1 summarises the approaches discussed in this chapter and give an overview of some of the most important methods within the field of hate speech detection today.

[6]https://github.com/clips/pattern

Table 3.1.: Overview of the approaches presented in Chapter 3.

Work	Description	Resources used	Classification	Dataset
Burnap and Williams (2015)	Machine 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 machine learning based method to detet hate speech	Several different NLP features, including lexicon, n-grams, Word2vec and syntactic	Skip-bigram and distributed memory model	Own Yahoo comments dataset, WWW2015 dataset Djuric et al. (2015)
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
Van Hee et al. (2015)	Dataset construction, text classification and automatic identification of cyberbullying events	Pattern[6], surface and sentiment features, POS tagging	SVM	Own Dutch Twitter dataset

Work	Description	Resources used	Classification	Dataset
Davidson et al. (2017)	Automated hate speech detection through using classic machine learning 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 through classic and deep learning approaches	Text features, BoW, Word2vec	LR, NB, SVM and CNN, GRU and Ensemble	Own Twitter 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)
Gambäck and Sikdar (2017)	Deep learning hate speech classification system	Character n-grams, one-hot encoding and Word2vec	CNN	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
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

Work	Description	Resources used	Classification	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 initialised 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)
A. M. Founta et al. (2019)	Unified deep learning architecture for abuse detection	TF-IDF, BoW and GloVe	NB as baseline, RNN, BiRNN, GRU and LSTM	Chatzakou et al. (2017), Waseem and Hovy (2016), Davidson et al. (2017), Rajadesingan et al. (2015)

Work	Description	Resources used	Classification	Dataset
Biesek (2019)	Automatic cyberbullying in Polish tweets	TF-IDF and fastText	SVM, BiGRU, Flair framework[7] with Contextual String Embeddings (Akbik et al., 2018)	Ogrodniczuk and Kobyliński (2019)
H. Liu et al. (2019)	Fuzzy multistep 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 categorise offensive language in social media	GloVe	classic methods baseline, CNN, BiLSTM, BiGRU	OLID dataset Zampieri et al. (2019b)
Sharma et al. (2018)	Degree based classification of harmful speech in social media	TF-IDF and BoW	SVM, NB and RF	Own Twitter dataset
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
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)
Balog et al. (2013)	Multistep classification approaches	Document, entity, document-entity and temporal features	Decision trees J48 and Random Forest	KBA Stream Corpus 2012[8]

Work	Description	Resources used	Classification	Dataset
Kwok and Y. Wang (2013)	Anti-black hate speech detection	Unigram features and BoW	Naïve Bayes	Own Twitter dataset
Poletto et al. (2017)	Analysis of annotator agreement on multiclass hate speech annotation	-	-	Own Italian Twitter dataset
Bosco et al. (2018)	Human-labelled corpus for online harassment research	-	-	Own Twitter dataset
Golbeck et al. (2017)	Human-labelled corpus for online harassment research	-	-	Own Twitter dataset
Ross et al. (2017)	Measuring reliability of hate speech annotations in German	-	-	Own German Twitter dataset
Sanguinetti et al. (2018)	Summary of EVALITA 2018 Hate speech detection task	Word embeddings, polarity and subjectivity lexicons	Linear SVM, 1-layer BiLSTM, 2-layer BiLSTM	Own Twitter and Facebook dataset
Salminen et al. (2018)	Variations in interpretation of online hate	-	-	Own Twitter dataset

4. Definition of hate speech and offensive language

One of the major challenges linked to the field of hate speech detection is how to accurately define, categorise and annotate different types of hate speech. To the best of our knowledge, and as can be inferred from the discussion in Section 3.1, there is no universal definition of hate speech today. Thus, we have contributed to the research field with our definition of hate speech and offensive language based on research from previous studies and also conferred with an expert on Norwegian law. This chapter will present our definition and the process behind it, followed by a description of the resulting five categories we decided on. Lastly, we will discuss some challenges related to the grading of hateful and offensive utterances.

4.1. Definition of hate speech and offensive language

Although there is no universal definition for hate speech, the most accepted definition 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; Djuric et al., 2015; Gitari 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), examples on hate speech are

(1) *Go fucking kill yourself and die already useless ugly pile of shit scumbag.*

(2) *The Jew Faggot Behind The Financial Collapse.*

(3) *Hope one of those bitches falls over and breaks her leg.*

As can be inferred, (1) and (3) can be considered hateful even though they are not directed towards a specific group. Nevertheless, it is quite hard to define what is hateful or not. As the authors point out, what is considered a hateful utterance 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, and identity of the author, target recipient, as well as context consisting of co-occurring media objects (images, videos, audio). We will not focus on the identity of authors and co-occurring media objects, but we emphasise the importance of these areas.

Another factor to take into consideration is that hate speech may have substantial 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 somewhat normal when quoting rap lyrics, but in another context, they should be perceived differently. Besides, whether an utterance is offensive or not is often subjective. As Xiang et al. (2012) state, "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) who underpin that even though the annotators have common annotation guidelines, the agreement score amongst the annotators is often very low.

Hate speech can be considered a rather broad umbrella term for several types of insulting user-created content. Based on the already existing articles on this topic, hate speech is the most frequently used expression. It is important 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. As pointed out earlier, (1) and (3) are not labelled as hate speech per definition, but they can still be hateful. Offensive language is, on the other hand, defined as a term that is applied to hurtful, derogatory or obscene comments made by one person to another person. In our case, offensive comments can also be directed towards groups. The difference from hate speech is the severity of the comment.

As indicated by the reflections done in Andreassen Svanes et al. (2019), the problem of defining hate speech is quite complex. Thus, using a binary classification will not be sufficient. This is because a binary classifier will not be able to distinguish between relatively similar utterances correctly. Furthermore, results from several papers (Sanguinetti et al., 2018; Sharma et al., 2018) on the field of hate speech detection indicate that a simple binary classification method does not cover the complexity of analysing hate speech. Also, the category with offensive utterances will include everything that is not classified as either hateful or neutral, indicating that the category itself will consist of utterances with a relatively large variation in severity.

Next, we researched *Straffeloven* §185 and §186 regarding hateful utterances and discrimination; the law covers hate crime against a person based on their ethnicity, religion, homosexual orientation or disability. A report from the Oslo police department further elaborates how they act based on this legislation (Hansen, 2015). However, the Danish Institute for human rights, found that approximately 15% of online comments can be classified as hateful and that the comments mostly target groups or individuals based on their political view (Zuleta and Burkal, 2017), which is not included in the characteristics

mentioned by *Straffeloven*. Other characteristics such as ethnicity, religion, social status and gender were also prevalent, and the report showed that news articles regarding these targeted groups spawned almost the double amount of hateful comments compared to news articles about other topics. Similar research has been done in Norway, where a report from *Likestillings- og diskrimineringsombudet* also emphasised different targeted groups than those mentioned by *Straffeloven*. Based on this, we have decided to implement a grading system for utterances consisting of five categories ranging from neutral to hateful. The five categories, Hateful, Moderately hateful, Offensive, Provocative and Neutral, are defined in the following sections.

4.1.1. Hateful

The most hateful utterances were split into two categories, moderately hateful and hateful, based on whether they incite violent or discriminatory actions or not. This was based on similar definitions formulated by Sanguinetti et al. (2018) and Sharma et al. (2018), where an explicit incitement to either violent or discriminatory actions was considered to be of the most severe degree. Whether or not the authors only justifies the actions and wish for them to happen or declare themselves ready to carry them out did not affect the severity, and all utterances that incite actions in any way is included in this category.

Definition 4.1.1. Hateful utterances are utterances which are partly or wholly motivated by hate or negative attitude towards groups or individuals based on ethnicity, religion, sexuality, gender, age, political views, social status or disabilities and which encourage violent actions based on this.

Examples of hateful utterances found in the dataset:

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!*
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!*

As can be inferred from the examples above, this category contains utterances which are severely degrading and disparages a group or individual's dignity. The utterances all contain a clear call to action, which in these cases is a message directed at either immigrants or Muslims telling them to "go home" or to be sent out of the country. The context makes it evident that the author carries hatred towards the target group and wants to inflict pain upon them. The motivation behind the statement is the key factor when classifying these kinds of utterances. It is important to notice that a simple mention of religion or immigrants is insufficient reason for classifying an utterance as non-neutral.

4.1.2. Moderately hateful

In general, moderately hateful and hateful utterances are characterised by their encouragement to violation of an individual or group's integrity and how it is a severe disparagement of dignity. These are factors that have been emphasised when legal cases are ruled and *Straffeloven* §185 has been violated[1]. The main difference between moderately hateful and hateful utterances is that moderately hateful utterances do not incite actions.

Definition 4.1.2. Moderately hateful utterances are utterances which are partly or fully motivated by hate or negative attitude towards groups or individuals based on ethnicity, religion, sexuality, gender, age, political views, social status 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 utterances found in the dataset:

1. *Begriper ikke at none 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 @USER*
2. *@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*
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?*

It is evident that (1) and (3) are dehumanising the targeted group and also blaming them for alleged problems in the society. Comment (2) targets an individual's dignity and is severely degrading. These utterances raise aversion and hatred towards groups or individuals based on their characteristics, but they do not incite further action.

4.1.3. Offensive

Offensive utterances ascribe negative features to a target, either a group or an individual, and differs from the two previously described categories by the severity of the comment. These utterances are considered less severe because they do not direct insults at individuals or targeted groups based on the characteristics described in Definition 4.1.2. Offensive utterances are more general in whom they target, and the degree of derogatory intent of the utterance is less severe. Thus offensive utterances are defined as follows:

[1]Decision in Eidsivating lagmannsrett: https://lovdata.no/pro/avgjorelse/le-1995-2083

Definition 4.1.3. An utterance is defined as offensive if it contains hurtful, derogatory or obscene comments, either directed towards an individual or a group.

Examples of offensive utterances found in the dataset:

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, @USER. Dårlig fungerende hjerne! Og det må vi respektere, alle er ikke født med en ordentlig hjerne*
3. *Milan bare endra tissen min fra stiv til halvkram, litt som å pule en feit hore #fetehorer #æsj #fårdenikkeopp*

Looking at the examples above, (1) and (2) are annotated as offensive because of hurtful and derogatory descriptions of individuals, while (3) contain obscene words and is annotated as offensive for that reason.

4.1.4. Provocative

Provocative language such as profane and filthy words are in general not enough to annotate an utterance as provocative. If an utterance is formulated with a neutral tone and language and contains one profanity, it may still be annotated as neutral. However, the number of profanities in a comment can sometimes be enough to switch from a neutral statement to a provocative if the use of profane and filthy words is not in the context of making a coherent remark.

Definition 4.1.4. A provocative utterance contains aggressive language to express an opinion or can be perceived as inappropriate. This includes the use of profane words, patronising language or the use of irony and sarcasm to lower the credibility of an opponent.

Examples of provocative utterances found in the dataset:

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. *Greta Thunberg er en BLØFF ! En stor løgn skapt kunstig av sosialister som blir betalt for å lyve !*
3. *@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*

Here, (1) is an example of an utterance with patronising language and irony with regards to the targeted individual's skills. The use of aggressive language to express an opinion is exemplified in (2). (3) can be perceived as inappropriate with the use of unnecessary aggressiveness and very negative words to describe a political party.

4.1.5. Neutral

Utterances that do not meet the criteria of any of the other categories will be annotated as neutral. These utterances are characterised by a neutral language, without profanities, aggression or other non-factual debate techniques.

Definition 4.1.5. An utterance which contains a neutral language and is a factual contribution to the debate.

Examples of neutral utterances found in the dataset:

1. *blir forbanna over at listhaug sier at spagetti er en norsk verdi eller noe sånt*
2. *@USER Du har vel allerede fått så hatten passer i kommentarene her, men ønsker likevel å påpeke at du ikke presenterer noe evidensbasert argument. Som andre har nevnt - ruspolitikk bør flyttes fra justis- til helsesektoren, noe som har fungert svært godt i flere land; jamfør Portugal.*
3. *Frykter at Finnmark blir uten meieri*

As we can see from (1), it is possible to use profane and loaded words such as "forbanna" and still publish a neutral utterance. Furthermore, (2) is an example of how it is possible to debate and disagree and still neutrally convey the message. (3) is clearly a neutral utterance, with no sign of provocative, offensive or hateful tendencies.

4.2. Challenges with grading utterances

From many of the studies reviewed in Chapter 3, it is clear that it is a challenge to distinguish between various degrees of hate. It is difficult to classify utterances in distinct categories when it appears to be smooth transitions and a sometimes unclear distinction between some of the categories. Another challenge is related to the subjectivity of those involved in the detection of hate speech, both with the process of defining hate speech and with annotating the collected data. As discussed in Section 3.4, Salminen et al. (2018) found that interpretation of hate was highly dependent on the individual's opinion and feelings. Thus, it is especially challenging to classify comments that are close to or in-between categories, because the subjectivity of the individual annotator will affect the decision. For example, some may classify a comment that has a provocative tone as category 2, while others find the comment neutral and classify it as category 1. Essentially, this implies that it can be very challenging to follow a universal definition of hate speech

and its various degrees because the individual's opinion will always affect the perception of an utterance.

Another challenge worth discussing is by whom such a definition should be made. Researchers with expertise in the field of computer science have often either created their own definition or based theirs upon the work of similar research. Should there be a universal definition that future research can be based upon? Moreover, who should be responsible for this definition? It is essential to ensure that no kind of bias affects the outcome. There have been conducted much research within the field of sociology on this topic, but this is beyond the scope of this thesis.

Based on these reflections, it is clear to us that our definition might have flaws and that the annotation work can be subject to subjectivity of the annotators. This will be further discussed in Chapter 8.

5. Architecture

This chapter will explain the architectures for the models that have been implemented. Two different approaches were used, namely an all-in-one approach and a two-step approach. The architecture of both approaches will be presented in the following sections, including descriptions of how each model was implemented for the correlated approach. Finally, the architecture for the deep learning models will be presented.

5.1. Standard models

A standard model is simple to set up and has a reasonable chance of providing decent results, and is also used to put a more complex model into context. Thus, we chose to implement the four supervised learning algorithms described in Section 2.1 and Section 2.2 as our standard models and use them as baselines for comparison later. This included learning to rank (LTR), logistic regression (LR) and linear support vector machine (LinearSVM) with different transformation methods, and a simple LSTM model.

5.1.1. All-in-one approach

The idea of the all-in-one approach is to follow the standard procedure related to classification problems. First, the dataset is split into a test set and a train set. Next, feature extraction is performed on the train set before these are being added as features to the machine learning algorithm. The algorithm is then tested using the test set, which outputs an array of predictions. The architecture for the all-in-one approach is illustrated in Figure 5.1.

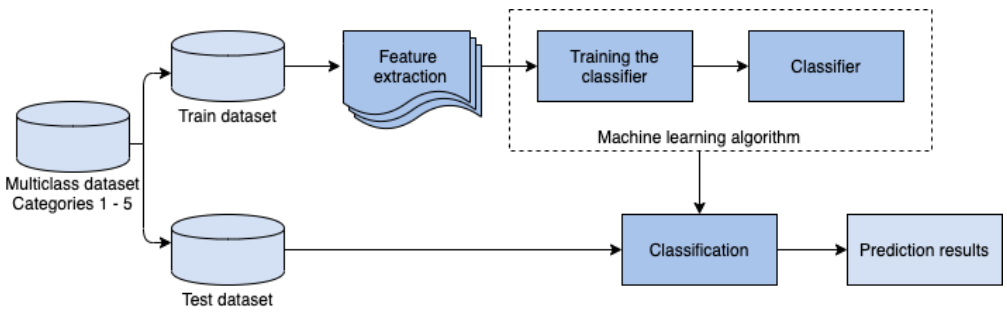

Figure 5.1.: Architecture of the all-in-one approach.

In the all-in-one approach with the Norwegian dataset, all models were trained to classify comments on a scale ranging from 1 to 5, as described in Chapter 4. LTR constructs a gradient boosting model and was implemented with Random forest as the boosting type, a multiclass objective, and multiclass logloss as a metric. LR was implemented with the *liblinear* solver with L2 regularization. LinearSVM was implemented with balanced class weights and a reduced penalty parameter due to being trained on an unbalanced dataset. The LSTM model was implemented by first creating an embedded layer that uses vectors with a length of 100 to represent each word. Next, a layer called SpatialDropout1D was added to perform variational dropout. The LSTM layer was then added with 100 memory units before the output layer was defined to create one output value for each class that the model was trained on, in this case, five. Because this is a multiclass classification problem, softmax was used as the activation function, and sparse categorical cross-entropy was used as loss function due to the categories being mutually exclusive for each comment. The LSTM model was then trained in five epochs.

For experiments with the English dataset, the models were trained on a binary dataset to classify comments as either neutral or offensive. LR and LinearSVM were implemented the same as for multiclass classification. LTR was again implemented with Random forest as the boosting type, but with a binary objective and binary logloss as a metric. The LSTM model was also implemented mostly the same, except for changing the number of output values in the Dense layer to two instead of five. In addition, due to this being binary classification, sigmoid was used as the activation function, and binary cross-entropy was used as the loss function.

5.1.2. Two-step approach

The first step in the two-step approach is similar to the English dataset's binary approach, with the only exception being which dataset is used as input. To create the Norwegian binary dataset, we merged all non-neutral comments, i.e., categories 2 to 5, to a non-neutral category. Thus, we ended up with prediction results in categories Neutral and non-Neutral. The architecture for step one in the two-step approach is illustrated in Figure 5.2.

In step two, the original multiclass dataset is altered to only contain categories 2 to 5. This dataset is split into a test set and a training set, but the test set is discarded. Instead, the output from step one, the comments predicted to be non-Neutral, is used as the test set. The architecture for step two is illustrated in Figure 5.3.

Thus, for the second step in the two-step approach, all models were trained only to classify non-neutral comments. All the models, except LSTM and LR, were implemented the same as described for experiments with the Norwegian dataset with the all-in-one approach in Section 5.1.1. In this approach, LR was implemented with balanced class-weights. For LSTM, a simple moderation of output values from five to four classes was made to accommodate that the model had been trained on only four classes in the two-step approach.

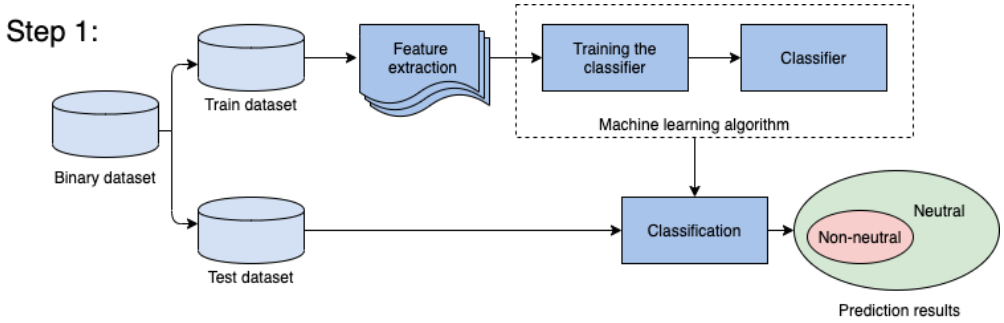

Figure 5.2.: Architecture of step one in the two-step approach.

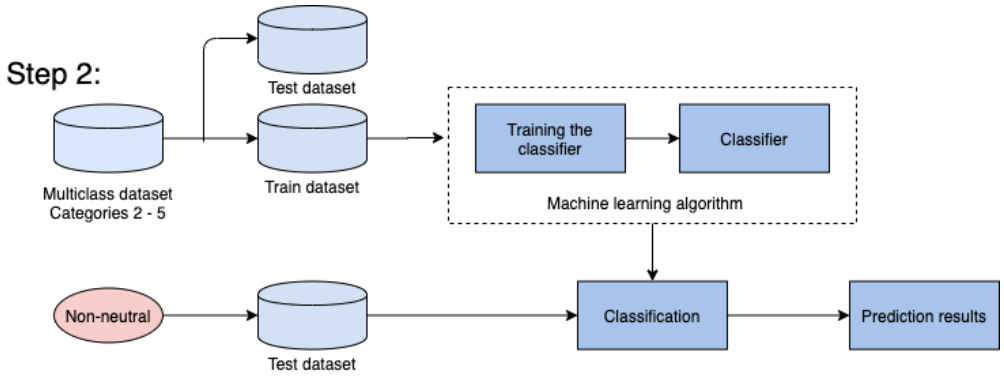

Figure 5.3.: Architecture of step two in the two-step approach.

5.2. Deep learning architecture

Next, we wanted to examine models that were more complex than those examined in our research project (Andreassen Svanes et al., 2019). Thus, we investigated the different deep learning models described in Section 2.2, in combination with different types of word embeddings. LSTM, CNN, and a bidirectional LSTM (BiLSTM) were tested, including a combination of CNN with BiLSTM and LSTM. The models are presented in this section, and the comparison between the different models' performance is presented in Section 6.7.

For the deep learning models, only the two-step approach is considered. Thus, the general implementation and architecture are similar to the standard models' two-step approach, which is explained in Section 5.1.2. We will explain the differences in detail in the following paragraphs.

The first layer of the deep learning models is a word embedding layer, and the same embedding layer was used for all of the tested models. In order to use word embeddings in our models, we had to create an embedding matrix which would be inputted as the weights to the embedding layer. The matrix can be thought of as a sequence of embedded

words. Each word was mapped onto either a 100 or 300-dimensional vector, depending on the word embedding used. The input shape, i.e., the number of words in each comment, was set to 135. Trainable was set to *True* in order for the weights to be updated during training.

Other than the embedding layer, the LSTM model was implemented the same as described in Section 5.1.2. The CNN model, as explained in Section 2.2.2, consisted of a hidden CNN layer with a filter size, i.e., number of neurons, for the convoluted feature map of 100, and a feature map of 4. Rectified linear unit function (ReLu) was used as the activation function, and max-pooling was set to a pool size of 4.

The BiLSTM model was based on the work done by Z. Zhang et al. (2018). It consisted of a bidirectional layer on top of an LSTM layer, which means that the network will do both forward and backward learning. Also, a dropout layer was added to prevent overfitting. However, the results deteriorated, and we chose not to proceed with this layer in further experiments. The number of filters for the BiLSTM model was set to 100. Furthermore, the BiLSTM and CNN models were tested with a double layer. Here, the second layer had half the filter size. There were not found any significant differences in the results, and a decision was made to not include the two-layered BiLSTM and CNN in further results and discussion.

We based the combined CNN and LSTM model on Løfqvist and Odden (2018), and the architecture is shown in Figure 5.4. The CNN layer had a filter size of 100 and a feature map of 4 and used ReLu as the activation function. On the generated feature map, a max-pooling layer with size 4 was used. The LSTM layer had a filter size of 200, and the activation function selected was the hyperbolic tangent function, Tanh. CNN and BiLSTM were also tested, and the same model was used. However, a bidirectional layer was added on top of the LSTM layer, and the number of filters was reduced to 100, due to the bidirectional layer doubling the filter size, totalling to 200.

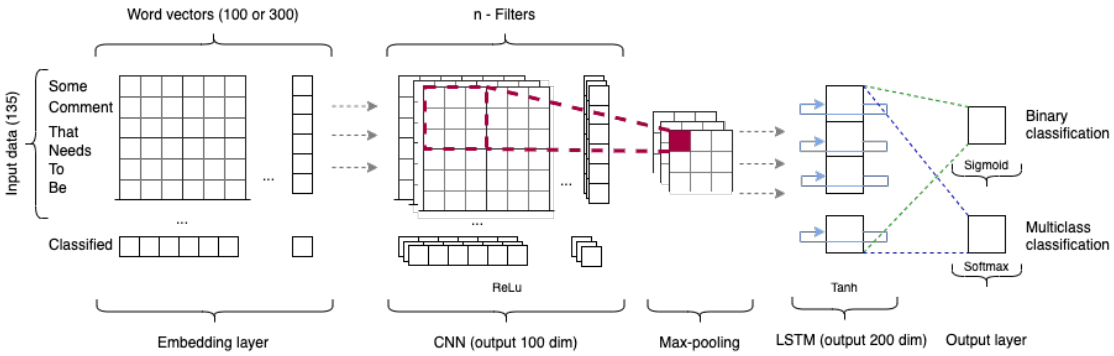

Figure 5.4.: Architecture of the CNN-LSTM model.

For all the models, the output layer consisted of two neurons to classify the data as either neutral or non-neutral in step one, and four neurons to do the same in step two. In the case of binary classification, sigmoid activation was used with binary cross-entropy, and in the case of multiclass classification, softmax activation with categorical cross-entropy was used. An overview of the models' configuration can be found in Table 5.1.

Table 5.1.: Overview of the architecture of the deep learning models.

Model	**Desc.**	**Filter size**	**Dropout**	**Feature map**	**Pooling size**	**Activation function**
LSTM	One layer	100	0.2			Tanh
CNN	One layer	100		4	4	ReLu
BiLSTM	One layer	100	0.2			Tanh
CNN-LSTM	CNN	100		4	4	ReLu
	LSTM	200				Tanh
CNN-BiLSTM	CNN	100		4	4	ReLu
	BiLSTM	100				Tanh

6. Experiments and results

This chapter will give a thorough presentation of the approach for identifying hateful utterances in the Norwegian language. First, since no common hate speech dataset exists for the Norwegian language, an explanation of how we have created such a dataset will be given. This includes the entire process from the data extraction and preprocessing to the annotation procedure. Next, the feature extraction steps that have been done will be described, and the evaluation methodology used will also be presented. In the following sections, the results from our experiments will be presented, starting with the all-in-one approach on the Norwegian dataset in Section 6.4, followed by the two-step approach in Section 6.5. In Section 6.6, results from experiments with an English dataset will be presented for comparison of the standard models' performances. The results from the ablation study with word embeddings and experiments with the deep learning models are presented in Section 6.7. Finally, results from experiments with a combination of models is presented in Section 6.8.

6.1. Dataset creation

Creating a labelled dataset can be a time-consuming and demanding task. Therefore, a significant part of this research project has been to create a benchmark dataset for the Norwegian language. When dealing with classification tasks, a thorough annotation procedure is essential for achieving adequate results. Precise labelling, proper representation of the data, and the amount of it play a significant part in the annotation procedure. We will explain how the data was extracted and preprocessed in the following sub-sections, as well as the annotation procedure and what guidelines we decided to follow when annotating.

6.1.1. Collection of data

It is important to gather data which is representative of a real-world scenario. Thus, we decided to gather data from various sources from social media, which includes *Resett*[1], Facebook, and Twitter. We did two rounds of collecting and processing data. One during Fall 2019 for Andreassen Svanes et al. (2019), which consisted of comments from Resett, and one round during the Spring semester of 2020 consisting of publicly available Facebook and Twitter data.

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

To collect data from Resett, we modified a web crawler[2] found on GitHub that collects comments from Resett's comment section, which use the comment platform of `Disqus`[3]. The crawler uses `Scrapy`[4], an open-source framework for extracting data from websites, and `Splash`[5], a headless browser designed for web scraping. We stored the comments in a `MongoDB`[6] database. We collected the 1000 most recent articles from the website, which mainly consisted of discussions of controversial topics such as immigration, the environment, and politics. The results were 21 087 comments gathered over two weeks in October 2019.

To collect tweets from Twitter, we applied for access to Developer accounts on Twitter and used the Twitter Search API with Full Archive and 30-Days search as options. Full Archive search allows 50 requests a month collecting 100 tweets per request totalling to 5k a month, while 30-Days Search allows 250 requests a month collecting 25k tweets in total. The search query requires a search word. Thus we decided on a word list of common Norwegian words and words that had been used frequently in recent debates in Norway, which we thought would represent Norwegian tweets. The word list can be found in Appendix C. We also added a language filter in the query, but it did not eliminate all non-Norwegian tweets since many words are common for different languages. In addition, a considerable amount of the tweets collected were retweets, which would derange our models in subsequent steps. They were therefore removed and not added to the final dataset.

We did several rounds of collecting data with different word lists. For the first three iterations, we first checked approximately how many results each query returned over four days. We removed words that did not accumulate a sufficient amount of results. As pointed out in Section 4.1, hateful utterances seldom appear in social media, and Twitter was no exception. We agreed after some iterations to try a search for other words than in the original word list, which also included removing terms considered to be neutral such as "fotball, håndball, langrenn, ski" from this list. After adding and removing different words for some iterations, we decided to add words that are generally seen as inflamed, such as "homse, abort, lesbisk, jøde" to the word list. This decision was because we could see after a while from the collected tweets that generally, there were few hateful utterances in the dataset. To investigate this assumption, we decided to annotate a small part of the tweets to check the percentage of hateful comments. After annotating 1000 tweets, we saw that we needed to collect more inflamed words since about 6.5% of them were hateful. Thus, we decided to only search for words considered to be inflamed, such as "rasist, trans, islam, fitte" to hopefully increase the percentage of hateful utterances. In total, we collected 31 667 tweets.

Facebook data was gathered manually from public Facebook pages such as Norwegian

[2]https://github.com/louisguitton/disqus-crawler
[3]https://disqus.com/
[4]https://scrapy.org/
[5]https://scrapinghub.com/splash
[6]https://www.mongodb.com/

newspapers and public persons. A list of the Facebook posts used can be found in Appendix D. Our wish was to collect data from perceived controversial and heavily debated posts at that point in time. Immigration, the environment, and politics where the most heavily debated posts we could find, as was also the trend among inflamed topics on Resett and Twitter. In the end, the results were 14 418 comments collected from 65 different posts.

Even though the social media data we collected are publicly available, to preserve the privacy of the users, we have ensured their identity was protected whenever necessary. This will be further elaborated in the next section.

6.1.2. Processing of collected data

Generally, user-generated text tends to contain abbreviations, emoticons, links, and slang, which are often considered noisy data. When using machine learning methods, this data can degrade a possible model. Thus, it is essential to process the data properly before proceeding with feature extraction and modelling. For classic methods, which were mainly used in the project thesis, and for the standard models in this thesis, noisy data can adversely affect the overall performance. Thus, abbreviations, emoticons, links, and slang were removed from the dataset. However, when working with deep learning models, some of these data, such as abbreviations and slang, can be useful and are therefore retained. This section explains how the collected data was processed.

With simple calculations in Python, we found that the average length of a comment from Resett, Twitter, and Facebook was respectively 234, 178, and 123 characters. Often with classic methods, it is argued that comments that are too short would not contain a sufficient amount of information to be useful when extracting features. Besides, it would be hard for a human annotator to classify a short comment due to the lack of context and information. Therefore, for the Resett dataset, a decision was made to remove comments consisting of ten characters or less. However, this step was not considered for the Twitter dataset since deep learning models do not require as many specifications on text length. Moreover, we also figured out through examining data from Twitter that it is possible to make a hateful utterance in ten characters or less. Nonetheless, we decided to remove comments on Facebook with ten characters or less after examining the data. On both Facebook and Resett, it is common to reply to comments with "Enig!" and "Godt sagt!". On Twitter, this same reaction to agreeing with an utterance would be to retweet or click like. Generally, on Facebook, many comments consist solely of emoticons or a user tagging another user. This explains the significant difference in the average comment lengths for the three datasets. Thus, this type of response would not contribute noteworthy to the models and was therefore removed.

Keeping too much data can result in a too detailed vocabulary for classic methods, and it can also be hard to decide the underlying meaning of a large text for an annotator. For both Resett and Facebook, some of the comments were several pages long. Based on this, all comments longer than 250 characters for Resett and all comments over 500 characters

for Facebook were eliminated. Since the average length of a Facebook comment was significantly shorter than for Resett and Twitter, we decided to not filter out as many of the longer Facebook comments. It was not necessary to take such measures for the Twitter dataset since Twitter has a character limit of 280 characters.

Resett

Unprocessed, we had 21 087 comments from Resett. We decided to remove URLs and emoticons from the comments since they contributed to noise in the dataset. We removed possible duplicates to ensure that the crawler had not made any mistakes, which resulted in removing 1 496 comments. Besides, we discarded comments with less than ten and more than 250 characters, thus removing respectively 288 and 5 666 comments. In the end, the dataset consisted of 13 637 comments.

Twitter

We started preprocessing with 31 668 tweets. In the first step, we checked whether the tweets were Norwegian by comparing them to a list of stopwords, which resulted in removing 1 106 tweets. We could see when annotating that it still did not catch all non-Norwegian tweets as many words appear in both Norwegian and other languages. Since we want to keep as much information as possible from the tweets, we decide only to remove URLs and "via mentions", a type of retweeting. After dropping duplicates, we are left with 27 613 tweets, thus removing 2 949 instances. From the sample that we annotated, we noticed that the words we considered neutral and had removed earlier from the word list were, in fact, mainly neutral. Thus, we decided to remove: 'fotball', 'håndball', 'langrenn', 'ski', 'forsker', 'utdanning', 'damelaget', 'lykkelig', 'streik', 'åpenhet' and add these to a new list considered to be neutral. This step removed 2 450 tweets, resulting in 25 128 tweets.

To anonymise users mentioned in the tweets, we replaced all usernames with @USER. Afterwards, we removed "gass" and "vind" which totalled to 945 tweets, which left us with 24 183 tweets. We removed "#' at the beginning of sentences, such as "#Oslo", as well as some retweets that were not on the regular format, because we observed that they were mostly neutral statements from bots. From these steps, we remove 624 tweets, ending up with 23 559 in total.

We removed duplicates one more time after replacing usernames, resulting in 44 more tweets removed. We removed case sensitive words of the neutral words and also added some other words we had noticed were common and mostly neutral. This step resulted in removing 209 tweets. In addition, we observed that some tweets containing (+), (pluss), (abo) and (DN+) were all neutral as they were directly from newspaper accounts. When these were removed as well, we ended up with 23 140 tweets. Note, however, that tweets we see as neutral such as tweets containing (pluss) will be kept and used for later processing where all of them will be annotated as neutral.

The collection and preprocessing of data was done in iterations, meaning that the list consisting of only inflamed words collected later in the process was preprocessed separately. 3 412 tweets were gathered and processed. We manually went through parts of the dataset and moved neutral tweets to the neutral tweet list. Finally, we were left with 1 384 tweets which we added to the original dataset of 23 140 tweets. We removed duplicates one last time and got a dataset of 24 510 tweets.

Facebook

For Facebook, we gathered a total of 15 518 comments. After removing comments bigger than 500, we were left with 15 034 comments. Facebook does not highlight a person's name in a comment in the same way as Twitter does. Thus, we needed to remove all names from the comments except public persons such as the prime minister and other famous persons. We found a list of common Norwegian first names and surnames on Github[7] and added names as we manually went through all the comments. Also, as explained earlier, we decided to remove comments under ten characters. This step resulted in removing 1 909 comments, achieving a total of 13 125 comments. We also ran the same function as for the Twitter dataset removing non-Norwegian instances and URLs, getting 12 157 comments. Through dropping duplicates and also removing comments that just consisted of names we finally achieved a dataset of 11 400 comments ready to be annotated.

We decided to use a subset of 6000 of the collected comments from Resett from Fall 2019. All comments annotated as non-neutral in the project preceding this thesis were kept, as well as a sample of 2000 neutral comments. This decision was due to the imbalance of topics since most of the posts were written against Muslims. Thus, the complete dataset for annotation consisted of 41 910 utterances. We present the amount of data removed in each step of preprocessing in Table 6.1.

6.1.3. Annotation procedure and guidelines

In our analysis of state of the art, we looked into annotator agreement and annotation reliability and the importance of this when creating and annotating a dataset. In the following section, we will explain the annotation process more thoroughly.

Based on the definitions in Section 4.1, we decided to annotate the data by implementing a grading system consisting of five categories ranging from neutral to hateful. The categories are Hateful, Moderately hateful, Offensive, Provocative and Neutral. They are marked using a scale from 1 to 5 based on increased severity, where Neutral is marked with 1, and Hateful is marked with 5.

Considering that we will collect and annotate even more data than in the project preceding this thesis, we had to agree on how the annotation process should be done.

[7] https://github.com/0301/ordliste

Table 6.1.: The amount of data from each source at each step.

Preprocessing step	**Resett**	**Twitter**	**Facebook**
Unprocessed	21 087	31 668	15 518
Drop duplicates	19 591		
Remove comments ≥ 250 and ≥ 500	13 925		15 034
Remove comments ≤ 10 characters	13 637		13 125
Remove non-Norwegian		30 562	12 157
Drop duplicates after removing URLs and "via mentions"		27 613	11 400
Remove common words and add inflamed dataset		24 510	
Total	**13 637**	**24 510**	**11 400**

As in Andreassen Svanes et al. (2019), we decided to base our annotation approach on De Gibert et al. (2018). Here, the authors created and discussed the annotation guidelines together to ensure that all annotators, both internal and external, would have the same understanding of hate speech. Since we were three people working together on the dataset, we decided to annotate 2 500 comments first later to calculate the inter-annotator agreement rate and the Fleiss's Kappa score. Then we used majority vote to determine what label an annotation should have. Throughout the annotation process, we discussed the annotations and modified the annotation guidelines accordingly. The resulting annotation guidelines can be found in Appendix B, and the definitions of the classes can be found in Section 4.1.

Table 6.2.: Annotation agreement.

	Neutral	**Prov.**	**Offensive**	**Mod. Hateful**	**Hateful**	**Total**
Full agreement	91%	3%	9%	13%	28%	89.5%
Majority vote	97%	26%	35%	9%	39%	99.1%

As can be seen in Table 6.2, there is generally a high agreement between the annotators whether an utterance is neutral or not, but the agreement amongst the other classes are not as coherent. There is a significantly higher agreement between the annotators when considering majority vote compared to full agreement. The increase is especially notable in category 2 and 3, which can imply that the annotators particularly struggled to distinguish between the two classes. The Fleiss's kappa score was 0.3869, which generally

means fair agreement.

We decided to use external annotators as part of our evaluation. Nobata et al. (2016) stated that there are large differences in agreement amongst annotations done by crowdsourcing compared to annotations done by experts. Regardless of this, due to a need to reduce our workload and our agreement scores not being satisfactory, we decided to employ external annotators. Thus, assuming that the authors' subjectivity would be reflected in the external annotators. The decision was based on the fact that we were collecting more data than what we were able to annotate on our own, and for credibility purposes; to minimise bias in the evaluation as there may be some disadvantages to annotating a complete dataset by ourselves. Thus, we decided to employ 20 computer science students to annotate 500 comments, each gaining a 250,- NOK gift certificate. In addition, we asked friends and family to annotate voluntarily in order to lessen our workload. 62% of the data were labelled by us, and the remaining 38% by external annotators. The challenges with the annotation procedure will be discussed in more detail in Section 8.3.

Waseem (2016) found that amateur annotators are more likely than expert annotators to label an utterance as hate speech. However, they also found that if there is full agreement between amateur annotators, the annotation agreement is comparable to expert annotators. Thus, expert annotators would only have to annotate the cases where the amateur annotators disagreed. Optimally we would like to have three annotators for each comment, which is considered the normal amount, but we did not have sufficient funds for this to be feasible. Thus, after the external annotators were done annotating, we decided to check all the annotations from category 2 - 5 to verify that the annotations were done according to the annotation guidelines. Nonetheless, when the authors were in doubt, the external annotation was retained. Despite Ross et al. (2017) pointing out that reliability did not improve when given a definition, we considered it crucial for our annotators to have such a guide. We especially thought this would be useful since we had chosen to implement multiclass classification where the categories where mutually exclusive and in need of strict definitions.

After annotating all of the utterances, some more preprocessing was done. When we annotated the 2 500 first tweets, we did not replace mentions with @USER as it was still early in the collection process. Thus, there were still some duplicates in the complete Twitter dataset. We removed 68 duplicated tweets which left us with a total of 24 444. Since both manual and automatic collection of utterances happened over a span of two months, we were particularly aware of this type of utterances during this period. Therefore, we added an extra 51 utterances considered to be hateful to the final dataset. We split the neutral dataset of the previously filtered comments in two; one consisting of the most generic tweets, such as RTs, bots, hashtags and newspapers, and the other of the filtered tweets from using the wordlists. These lists consist of respectively 4 854 and 1 001 tweets. There is no guarantee that all of the tweets in these lists are neutral, and we decided not to use these list in further implementations considering the number of neutral utterances already in the dataset. Finally, we ended up with a dataset consisting

of 41 891 utterances.

The distribution of the final dataset are shown in Table 6.3. As expected, most of the utterances in the dataset were neutral. Despite trying to decrease the number of neutral comments by removing neutral statements and actively search for inflamed words, 82% was still considered neutral. This can both be explained by lack of context and that, in general, few hateful utterances can be found in social media. However, this is consistent with the findings of both the Danish and Norwegian reports (Zuleta and Burkal (2017); Veledar (2018)) stating that respectively 15% and 9% of online comments can be classified as hateful.

Table 6.3.: Distribution of the preprocessed annotated dataset.

Category	**Number of comments**
Neutral	34 085
Provocative	4 737
Offensive	1 563
Moderately hateful	510
Hateful	250
Removed (X)	746
Total	41 891

We also desired our dataset to be as representative as possible, meaning that we did not want to solely collect tweets containing profane words or only specific topics considered to be inflamed.

As anticipated, the distribution of the four categories of hateful utterances was not even. This can be seen in Figure 6.1. Veledar (2018) showed in their paper the degree of abusiveness through the values: slightly, medium and very abusive. They found a predominance of 64% slightly abusive comments followed by 17% with a medium degree and 7% with a very abusive character. From our pie chart, we can see that there is compliance between our results and the paper's results.

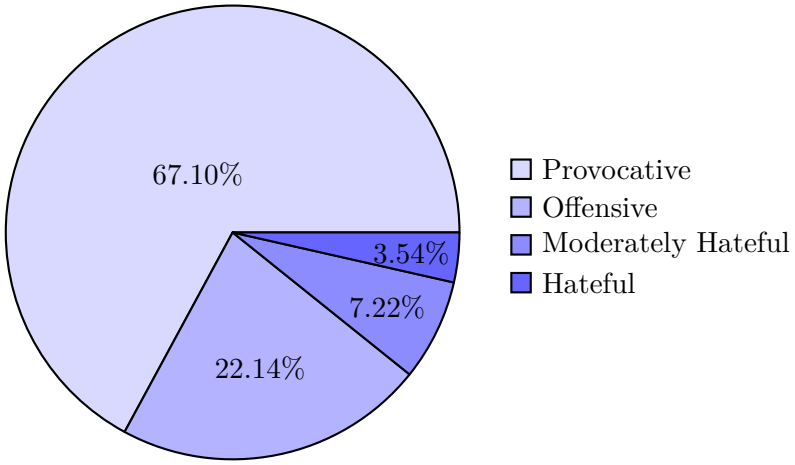

Figure 6.1.: Distribution of non-neutral utterances.

6.2. Feature extraction

The general process of implementation, from a raw unlabelled dataset to prediction results, is illustrated in Figure 6.2.

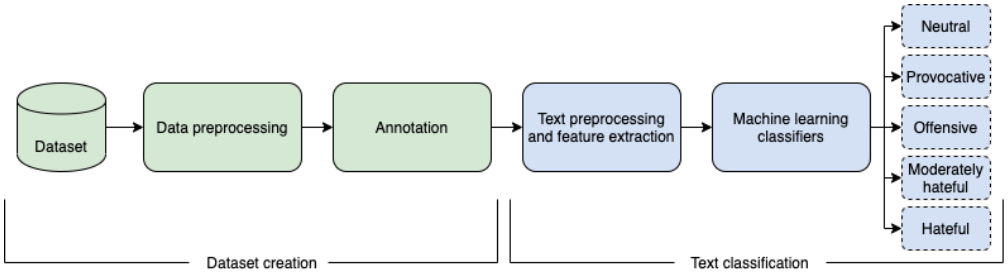

Figure 6.2.: The general process of implementation.

It was decided to implement and test some simple standard machine learning models on the labelled dataset to see if these methods could achieve satisfactory results when used for text classification. The process of dataset creation was presented in Section 6.1.2, with explanations of the data preprocessing step and the annotation step. The following section will describe the performed feature extraction steps for the Norwegian dataset.

An analysis of relevant textual features was desired to investigate if extracted features could be applied to the dataset and contribute to better accuracy when detecting and grading offensive language using the standard models. We used the methods presented in Andreassen Svanes et al. (2019) with the following results.

Statistical features

Relevant statistical features belonging to each comment were extracted from the corpus. Some features did not have a significant difference between the different categories, meaning that a comment in category 1 had the same features as a comment in category 5. Thus, we chose to apply only features with a significant difference between the categories, namely total length, character count, punctuation count, number of exclamation marks, number of symbols and number of question marks.

N-gram analyses

Word n-gram analyses were performed, counting occurrences of words either alone, in pairs or triples. The analyses showed that many of the most used n-grams were similar amongst the five categories, but some occurred more frequently in the non-neutral categories. Thus, we chose to apply the following n-grams as features:

islam
muslim
muslimer
fitte
landet
norsk
Norge
ut land
send ut
send hjem

Part-of-Speech tagging

Next, the POS tagger called RippleTagger was applied to the dataset. RippleTagger has support for the Norwegian language, and the count of each lexical category in the text was added as features to the respective comment. Tags with no significant difference between the categories were removed, and the rest was applied as features.

The vectors now contain the following features:
'total_length', 'char_count', 'punc_count', 'num_exclamation_marks', 'num_symbols', 'num_question_marks', 'nb_islam', 'nb_muslim', 'nb_muslimer', 'nb_fitte', 'nb_landet', 'nb_norsk', 'nb_norge', 'nb_utlandet', 'nb_sendut', 'nb_sendhjem', 'adj_count', 'adp_-count', 'adv_count', 'aux_count', 'det_count', 'intj_count', 'noun_count', 'num_count', 'pron_count', 'propn_count', 'verb_count'.

Text representation

Finally, the feature vectors were transformed into input vectors for the standard machine learning models. Four transformation methods were used as input vectors for LR and LinearSVM, namely count vectors, word-level TF IDF vectors, n-gram level TF-IDF vectors and character-level TF-IDF vectors. Count vectors is a matrix notation of the dataset, where every row represents a comment from the corpus, and every column represents a specific term. In combination, they represent the frequency count of a particular term in a particular comment. The last three are variations of the TF-IDF score, used to represent the relative importance of a term.

For LTR, the feature vectors were transformed into an input vector using TfidfVectorizer from the Scikit-learn library, which converts a collection of raw documents to a matrix of TF-IDF features. Tokenizer from Keras is a text tokenization utility class which allows to vectorize a text corpus. It was used to produce an input vector for the deep learning models, by first creating the vocabulary index based on word frequency and then transforming each comment to a sequence of integers.

6.3. Evaluation methodology

The models were trained and tested on different datasets, depending on which approach that was used. However, all datasets were split into 70% training data and 30% test data, with 10% of the training data being used for validation. Furthermore, k-fold cross-validation was used with $k = 3$ to identify important parameters and to avoid unwanted bias from the train-test split. On average, the Norwegian dataset contained 82.7% neutral comments in the training set and 83.1% neutral comments in the test set.

To evaluate the models' overall performances, accuracy, precision, recall and F_1-score were computed for each of the models. Because we value the minority classes the most, the macro-averaged F_1-score was used as a metric to determine which transformation of each model that performed the best. In addition, ROC curves were created for the binary classification done on the English dataset, and AUC scores were calculated. Built-in functions from scikit-learn were used to evaluate each model's performance, resulting in a confusion matrix. This is a table used to describe a model's performance on a set of test data for which the true values are known, also known as an error matrix. Each row of the matrix represents instances in a predicted class while each column represents the instances in an actual class, making it possible to determine the number of correct predictions and see where the model makes a mistake.

6.4. Results: All-in-one approach with standard models

In this section, results from experiments on the four standard models tested in the all-in-one approach will be presented. The architecture and the implementation of each model were described in Chapter 5.

6.4.1. Learning to rank

We started with LTR, to investigate the effect of looking at our research question as a ranking problem rather than a prediction problem. Table 6.5 displays evaluation metrics for the LTR model, which achieved an accuracy of 81.6% and a macro-averaged F_1-score of 0.29. The confusion matrix is presented in Table 6.4.

Table 6.4.: Confusion matrix for LTR.

True / Predicted	**1**	**2**	**3**	**4**	**5**
1	9913	46	38	109	125
2	1242	47	24	65	33
3	342	10	40	32	24
4	93	5	13	35	6
5	40	1	3	9	21

Table 6.5.: Evaluation metrics for LTR.

Category	**Precision**	**Recall**	**F_1-score**	**Support**
1	0.85	0.97	0.91	10 231
2	0.43	0.03	0.06	1411
3	0.34	0.09	0.14	448
4	0.14	0.23	0.17	152
5	0.10	0.28	0.15	74
Macro avg	0.37	0.32	0.29	12 316
Weighted avg	0.77	0.82	0.77	12 316

6.4.2. Logistic regression

LR was used with four different simple surface features, which achieved very similar accuracy scores varying from 82.7% to 83.2%, and macro-averaged F_1-scores ranging from 0.19 to 0.26. Count vectors achieved the highest score, and thus, we have presented the confusion matrix and evaluation metrics for this model in Table 6.6 and Table 6.7.

Table 6.6.: Confusion matrix for LR with count vectors.

True / Predicted	1	2	3	4	5
1	9972	206	42	11	0
2	1207	177	20	5	2
3	354	55	34	5	0
4	104	31	13	5	0
5	57	11	4	1	1

Table 6.7.: Evaluation metrics for LR with count vectors.

Category	Precision	Recall	F_1-score	Support
1	0.85	0.97	0.91	10 231
2	0.37	0.13	0.19	1411
3	0.30	0.08	0.12	448
4	0.15	0.03	0.04	152
5	0.33	0.01	0.03	74
Macro avg	0.40	0.24	0.26	12 316
Weighted avg	0.77	0.83	0.78	12 316

6.4.3. LinearSVM

LinearSVM was tested with three different simple surface features. The models achieved similar accuracy scores, varying from 78.2% to 79.6%, and macro-averaged F_1-scores ranging from 0.26 to 0.31, where word-level TF-IDF vectors achieved the highest score in both metrics. Thus, we have presented the confusion matrix and evaluation metrics for this model in Table 6.8 and Table 6.9.

Table 6.8.: Confusion matrix for LinearSVM with word-level TF-IDF vectors.

True / Predicted	1	2	3	4	5
1	9449	389	220	115	58
2	994	237	119	47	14
3	243	77	88	29	11
4	68	28	40	16	0
5	35	9	11	6	13

Table 6.9.: Evaluation metrics for LinearSVM with word-level TF-IDF vectors.

Category	Precision	Recall	F_1-score	Support
1	0.88	0.92	0.90	10 231
2	0.32	0.17	0.22	1411
3	0.18	0.20	0.19	448
4	0.08	0.11	0.09	152
5	0.14	0.18	0.15	74
Macro avg	0.32	0.31	0.31	12 316
Weighted avg	0.77	0.80	0.78	12 316

6.4.4. LSTM

A simple LSTM model was implemented and tested on the dataset. The confusion matrix and evaluation metrics for each category is shown in Table 6.10 and Table 6.11, where LSTM achieved an accuracy of 78.7% and a macro-averaged F_1-score of 0.28.

Table 6.10.: Confusion matrix for LSTM.

True / Predicted	1	2	3	4	5
1	9257	867	96	11	0
2	971	377	51	12	0
3	229	156	54	9	0
4	75	43	28	6	0
5	29	25	14	4	2

Table 6.11.: Evaluation metrics for LSTM.

Category	Precision	Recall	F_1-score	Support
1	0.88	0.90	0.89	10 231
2	0.26	0.27	0.26	1411
3	0.22	0.12	0.16	448
4	0.14	0.04	0.06	152
5	1.00	0.03	0.05	74
Macro avg	0.50	0.27	0.28	12 316
Weighted avg	0.77	0.79	0.78	12 316

6.4.5. Summary

To summarise, the accuracy and macro-averaged F_1-score of each model are presented in Table 6.12. Even though LR achieved the highest accuracy, it is debatable whether or not this is the best model, seeing as it achieved the lowest F_1-score. In addition, it appears from the relatively low F_1-scores that all models are heavily influenced by their ability to correctly classify most of the neutral comments, meaning that we cannot assume that these models are equally good at detecting non-neutral comments.

Table 6.12.: Accuracy score for each model using the all-in-one approach.

Model	**F_1-score**	**Accuracy**
Learning to rank	0.29	81.6%
Logistic regression with count vectors	0.26	83.3%
LinearSVM with word-level TF-IDF vectors	0.31	79.6%
LSTM	0.28	78.7%

6.5. Results: Two-step approach with standard models

Here, results from experiments on the four standard models tested with the two-step approach will be presented. In the first step, all models were tested on 12 316 comments where 10 231 were annotated as neutral and 2085 as non-neutral.

For the second step, all models were trained using a training set made from the original dataset containing comments in category 2-5 only, in total 3967 comments. Then, the models were tested on the output from step one. The distribution of comments in each category in the training set is 3321 in category 2, 1115 in category 3, 356 in category 4 and 175 in category 5.

6.5.1. Learning to rank

First step: Binary classification

The first step resulted in 454 comments being predicted as non-neutral and 11 862 as neutral. The non-neutral comments were further preprocessed where the original labels were paired with the comment, and the comments in category 1 (false negatives predicted in step one) were removed. This left us with a dataset of 266 comments to use as test data for the model in step two.

The confusion matrix for the binary classification with LTR is presented in Table 6.13. This resulted in an accuracy of 83.7%, a macro-averaged F_1-score of 0.56 and an AUC score of 0.55.

Table 6.13.: Step one: Confusion matrix for LTR.

True / Predicted	0	1
0	10 043	188
1	1819	266

Second step: Multiclass classification

Next, the model was tested on the output from step one; the 266 comments predicted to be non-neutral. This resulted in the following confusion matrix presented in Table 6.14 and the evaluation metrics presented in Table 6.15. Overall, LTR achieved an accuracy of 51.5% and a macro-averaged F_1-score of 0.32.

Table 6.14.: Step two: Confusion matrix for LTR.

True / Predicted	2	3	4	5
2	111	14	4	3
3	55	23	3	2
4	28	10	1	1
5	8	0	1	2

Table 6.15.: Step two: Evaluation metrics for LTR.

Category	Precision	Recall	F_1-score	Support
2	0.55	0.84	0.66	132
3	0.49	0.28	0.35	183
4	0.11	0.03	0.04	40
5	0.25	0.18	0.21	11
Macro avg	0.35	0.33	0.32	266
Weighted avg	0.45	0.52	0.46	266

6.5.2. Logistic regression

First step: Binary classification

Initially, LR with count vectors performed marginally better than the other transformations with an accuracy of 79.5% and a macro-averaged F_1-score of 0.66, compared to values ranging from 72.94% to 76.5% and from 0.59 to 0.65 for the others. Thus, it was decided to use the output from this transformation for step two. This resulted in 2507 comments being predicted non-neutral and 9809 as neutral, as shown in the

confusion matrix presented in Table 6.16. It achieved an AUC score of 0.68. With further preprocessing, we were left with a dataset of 1035 comments to use as a test set in step two.

Table 6.16.: Step one: Confusion matrix for LR with count vectors.

True / Predicted	**0**	**1**
0	8759	1472
1	1050	1035

Second step: Multiclass classification

In the second step, LR was tested on the output from step one. Word-level TF-IDF vectors performed the best with an accuracy of 58.7% and a macro-averaged F_1-score of 0.42. The confusion matrix for logistic regression with word-level TF-IDF vectors is presented in Table 6.17 and the evaluation metrics are presented in Table 6.18.

Table 6.17.: Step two: Confusion matrix for LR with word-level TF-IDF vectors.

True / Predicted	**2**	**3**	**4**	**5**
2	507	60	44	11
3	169	59	30	6
4	49	22	25	3
5	22	6	5	17

Table 6.18.: Step two: Evaluation metrics for LR with word-level TF-IDF vectors.

Category	**Precision**	**Recall**	**F_1-score**	**Support**
2	0.68	0.82	0.74	622
3	0.40	0.22	0.29	264
4	0.24	0.25	0.25	99
5	0.46	0.34	0.39	50
Macro avg	0.44	0.41	0.42	1035
Weighted avg	0.56	0.59	0.56	1035

6.5.3. LinearSVM

First step: Binary classification

In the first step, LinearSVM with count vectors performed better than the two other models tested, with an accuracy of 79.6% and a macro-averaged F_1-score of 0.66. Thus, it was decided to use the output from LinearSVM with count vectors for step two. This model predicted 2384 comments as non-neutral and 9932 as neutral, as shown in the confusion matrix in Table 6.19. It achieved an AUC-score of 0.67. After further preprocessing and removal of false negatives, we were left with 979 comments to use as a test set in the second step.

Table 6.19.: Step one: Confusion matrix for LinearSVM with count vectors.

True / Predicted	0	1
0	8826	1405
1	1106	979

Second step: Multiclass classification

Similar to LR, LinearSVM performed best with word-level TF-IDF vectors. It achieved an accuracy of 56.8% and a macro-averaged F_1-score of 0.41 when tested on the 979 comments from step one. The confusion matrix for LinearSVM with word-level TF-IDF vectors is presented in Table 6.20 and the evaluation metrics are presented in Table 6.21.

Table 6.20.: Step two: Confusion matrix for LinearSVM with word-level TF-IDF vectors.

True / Predicted	2	3	4	5
2	451	74	43	19
3	149	62	30	8
4	44	21	27	3
5	20	5	6	17

Table 6.21.: Step two: Evaluation metrics for LinearSVM with word-level TF-IDF vectors.

Category	Precision	Recall	F_1-score	Support
2	0.68	0.77	0.72	587
3	0.38	0.25	0.30	249
4	0.25	0.28	0.27	95
5	0.36	0.35	0.36	48
Macro avg	0.42	0.41	0.41	979
Weighted avg	0.55	0.57	0.55	979

6.5.4. LSTM

First step: Binary classification

LSTM achieved an accuracy of 82.6%, a macro-averaged F_1-score of 0.65, and an AUC-score of 0.64. As we can see from the confusion matrix in Table 6.22, 1500 comments were predicted non-neutral and 10 816 were predicted neutral. After preprocessing and removing false negatives, we were left with a dataset consisting of 724 comments to use in step two.

Table 6.22.: Step one: Confusion matrix for LSTM.

True / Predicted	0	1
0	9455	776
1	1361	724

Second step: Multiclass classification

The model was tested on the output from step one, which resulted in an accuracy of 49.4% and a macro-averaged F_1-score of 0.25. The confusion matrix and the evaluation metrics for LSTM is presented in Table 6.23 and Table 6.24.

Table 6.23.: Step two: Confusion matrix for LSTM.

True / Predicted	2	3	4	5
2	309	73	9	0
3	167	45	5	0
4	48	21	4	0
5	33	8	2	0

Table 6.24.: Step two: Evaluation metrics for LSTM.

Category	Precision	Recall	F_1-score	Support
2	0.55	0.79	0.65	391
3	0.31	0.21	0.25	217
4	0.20	0.05	0.09	73
5	0.00	0.00	0.00	43
Macro avg	0.27	0.26	0.25	724
Weighted avg	0.41	0.49	0.43	724

6.5.5. Summary

To summarise, the accuracy and macro-averaged F_1-score of each model in the second step is presented in Table 6.25. From this, we can see that LinearSVM and LR with word-level TF-IDF vectors and class weights stand out, with higher accuracies and F_1-scores.

Table 6.25.: Accuracy score for step two for each model using the two-step approach.

Model	F_1-score	Accuracy
Learning to rank	0.32	51.5%
Logistic regression on word level TF-IDF vectors	0.42	58.7%
LinearSVM on word level TF-IDF vectors	0.41	56.8%
LSTM	0.25	49.4%

Compared to the results from the all-in-one approach, it is evident that there is more distance in the performance of the model using the two-step approach. Here, two models perform significantly better, unlike for the all-in-one approach where the models performed more similarly.

6.6. Experiments with English dataset

As discussed in Chapter 3, most state-of-the-art models are trained on English datasets and, on average, perform relatively well when tested. Because of this, we chose to evaluate our models on an existing English dataset. By comparing the results from an English dataset with the results from our Norwegian dataset, we would be able to determine significant factors that affect the performance of the models when trained on the Norwegian language. The main goal behind this decision was to get an indication of whether or not the models themselves were satisfactory. If the Norwegian and the English dataset performed equally bad, it could indicate a problem with the models, and this was something we wanted to investigate.

Zampieri et al. (2019a) compiled the Offensive Language Identification Dataset (OLID), consisting of 13 240 tweets using a fine-grained three-layer annotation scheme. Each comment contains up to three labels corresponding to one of the following: Offensive language identification (sub-task A), Automatic categorisation of offence types (sub-task B), and Offence target identification (sub-task C). We chose to only use the labels from sub-task A, resulting in a dataset consisting of 13 240 comments labelled as either Offensive or Not Offensive. As described by Zampieri et al. (2019a), the dataset was already preprocessed and ready to be used.

The same statistical textual analysis and feature extraction as described in Section 6.2 was conducted. The results differed from the Norwegian dataset, and here the five most significant features were total length, character count, punctuation count, number of exclamation marks and number of symbols. Analyses of both POS tags and n-grams were also conducted, resulting in the top five unigrams and top two bigrams being added as features as well as the POS tags.

Finally, the dataset was split into 70% training data and 30% test data and k-fold cross-validation with $k = 3$ was implemented before the data was used as input vectors to the standard models described in Section 5.1.1. Some of the parameters were adjusted to fit the English dataset, including the adjustment from multiclass to binary classification.

6.6.1. Results on English dataset

In total, the same nine standard models that were used in the experiments with the Norwegian dataset were tested on the English dataset. The ROC curves for all the models are shown in Figure 6.3.

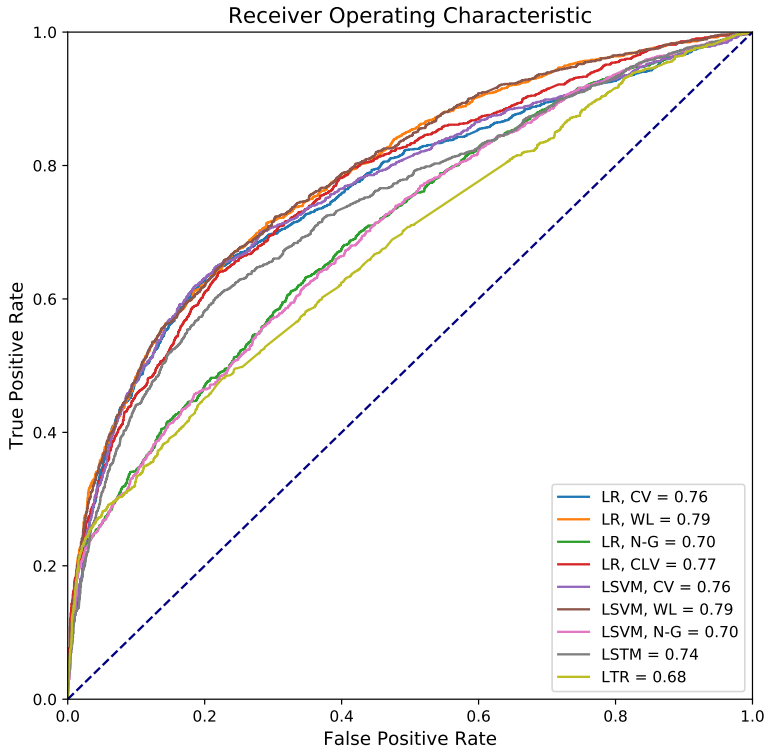

Figure 6.3.: ROC curves for models tested on the English dataset.

From this, we found that both LR and LinearSVM achieved decent AUC scores, with LSTM and LTR somewhat lower. Table 6.26 shows the macro-averaged F_1-score, accuracy and AUC score for each of the four standard models.

Table 6.26.: Evaluation metrics for the standard models tested on English dataset.

Model	**F_1-score**	**Accuracy**	**AUC score**
Learning to rank	0.58	0.72	0.68
Logistic regression with count vectors	0.70	0.75	0.77
LinearSVM on word-level TF-IDF vectors	0.72	0.74	0.79
LSTM	0.69	0.73	0.74

The evaluation metrics for each model is presented in Table 6.27. Based on the results shown in both tables, it appears to be LinearSVM with word-level TF-IDF vectors that performs best on the English dataset. This model found 845 true negatives and 2112 true positives. Also, we see that 61.4% of the offensive comments and 81.4% of the neutral comments were correctly classified. This can be seen in Table 6.28.

Table 6.27.: Results for the standard models tested on English dataset.

Model	Category	Precision	Recall	F_1-score
Learning to rank	N	0.70	0.98	0.82
	O	0.87	0.21	0.34
Logistic regression with count vectors	N	0.77	0.89	0.82
	O	0.70	0.49	0.58
LinearSVM with word-level TF-IDF vectors	N	0.80	0.81	0.81
	O	0.64	0.61	0.62
LSTM	N	0.77	0.84	0.81
	O	0.64	0.53	0.58

Table 6.28.: Confusion matrix for LinearSVM with word-level TF-IDF vectors.

True / Predicted	N	O
N	2112	484
O	531	845

6.7. Deep learning models

A description of the simplified ablation study done on experimenting with different word embeddings will be presented in this section. Next, the results of these experiments are presented. A simplified ablation study was also conducted on the different deep learning models that were described in Section 5.2. The results from these experiments are presented in Section 6.7.3.

6.7.1. Word embeddings

According to the current state of the art, we decided to explore three neural language text representation methods, namely Word2vec, GloVe and fastText. They are previously explained in Section 2.4.2. In addition, we wanted to investigate transfer learning models such as BERT. We used both pre-trained and own trained word embeddings in our research. To determine which representation model should be used, we tested the methods as input for different deep learning methods. The results from this will be evaluated in Section 7.3. In this section, we will present how the different word representation models were obtained.

The pre-trained Norwegian models of Word2vec, GloVe and fastText were all found on a

common site for NLP models trained on a variety of languages[8]. They were trained on three different corpora from common crawl from different Norwegian web page domains. In addition, a common crawl version from the official site of fastText[9] was used. An overview of the different parameters used for the pre-trained embeddings can be found in Table 6.29.

Table 6.29.: Overview of pre-trained word embeddings.

Word embedding	Vocab. size	Model	Dimension size	Window size
Word2vec	4 480 046	CBOW	100	5
GloVe	4 480 046	Global Vectors	100	15
fastText	4 428 648	SG	100	5
fastText	2 000 000	CBOW	300	5

We had a dataset of user-generated texts from social media containing different dialects and misspellings. Presumably, the dataset would not fit well to the pre-trained embeddings due to them being trained on factual texts with formal language. However, by own training our models, we could run a risk with our models becoming too adapted to the dataset. 85% of the vocabulary from our dataset is covered by the pre-trained model from the NLP directory, and the fastText common crawl covered 78.8%. This is a decent coverage for our vocabulary. Nonetheless, we wanted to check whether there was a significant difference in results when using own trained word embeddings. Thus, we wanted to test both pre-trained and own trained embeddings both to keep the vocabulary as general as possible, but also to check whether an own trained model would improve the results. Word2vec and fastText embeddings were trained on our own as there were already existing libraries for this purpose such as gensim[10] and a fastText Python library[11] which would simplify our process. An explanation of our own trained word embeddings follows below.

For both Word2vec and fastText, we trained on dimension size 100 and 300, as these are the most common and we already had pre-trained embeddings with these dimensions. A window size of 5 and 15 were used since we wanted to see whether an increase in window size would have an impact on the results. In addition, both continuous bag-of-words (CBOW) and skip-gram (SG) were tested to check whether they would influence the result. The default threshold for the minimum count of words in the vocabulary, *min count*, that is used when training these models is 5. However, this would return the total number of words found by the model only to be 18%. Thus, a decision was made to use *min count* of 1 and 2. The number of words for min count = 1 is 65 533 and 27 499 for min count = 2. The difference in vocabulary size was 49.9% for min count = 2 and 99.7% for min count = 1.

[8] http://vectors.nlpl.eu/repository/
[9] https://fastText.cc/docs/en/crawl-vectors.html
[10] https://radimrehurek.com/gensim/
[11] https://pypi.org/project/fasttext/

The gensim library was used for training our own word embeddings on our dataset with Word2vec and tested with the parameters mentioned above on the standard two-step LSTM model. In addition, we used the pre-trained Word2vec model to expand our own trained vocabulary. This means that the vocabulary size would not be increased, but the model would include the word embedding of the pre-trained model for a specific word. If in our dataset the representation of "send" is more close to "ut" than "pakke", expanding the own trained model with a pre-trained model could make the word "pakke" more similar to "send". This is because these words are supposedly more frequently used together in common Norwegian texts.

The fastText library provides two methods: *train_unsupervised* and *train_supervised*, in which we will only use train_unsupervised. Train_unsupervised works the same way as the gensim method, where the processed dataset is inputted, and the same parameters as above are configured and tested on the standard two-step LSTM model. In addition, a *pretrainedVectors* parameter can be added to train on both pre-trained vectors and our own dataset. With this approach, only the fastText common crawl dataset on vector format is compatible, which means that the embedding size has to be set to 300. The original vocabulary size of pre-trained embedding was 2 000 000, as seen in Table 6.29. After training, we ended up with a vocabulary of 2 001 962 for min count = 2 and 2 015 880 for min count = 1.

An overview of the different parameters for the own trained embeddings can be found in Table 6.30. As already mentioned, all the different word embeddings were tested on the standard LSTM model. The results are presented in Section 6.7.2.

Table 6.30.: Overview of own trained word embedding parameters

Word emb.	Voc. size	Model	Dim. size	Win. size	Word count
Word2vec	65 533	CBOW, SG	100, 300	5, 15	1
Word2vec	27 449	CBOW, SG	100, 300	5, 15	2
fastText	65 533	CBOW, SG	100, 300	5, 15	1
fastText	27 449	CBOW, SG	100, 300	5, 15	2
fastText extended	2 001 962	CBOW	300	5, 15	1
fastText extended	2 015 880	CBOW	300	5, 15	2

The pre-trained BERT model was collected from a nordic BERT GitHub[12] repository that was recently published containing both Danish, Norwegian and Swedish models. The downloadable Norwegian weights are trained on 4.5 GB data from common crawl. The repository contains weights and a vocab-file similar to the original BERT models[13]. However, we did not manage to properly implement the Norwegian models in the same way as for the English example tested. We realised there were created "issues" regarding

[12]https://github.com/botxo/nordic_bert
[13]https://github.com/google-research/bert

this on GitHub, and thus, we decided not to move forward using BERT as there would be a need for more analysis and studies to find out what did not work.

6.7.2. Results: Word embeddings

Here, the results from the experiments done with different word embeddings will be presented. The architecture and implementation of the two-step LSTM approach are described in Section 5.1.2 and Section 5.2, with the only difference in architecture being the inclusion of word embeddings in the embedding layer and adding class weights in order to balance the dataset.

First step: Binary classification

Table 6.31 shows the results for the first step of the two-step approach. The table is split with a double line between the own trained models and the pre-trained models, which are grouped with the models without word embeddings. The different versions of each model are enumerated accordingly. As we can see, all models perform more or less equally for all metrics. However, some interesting observations can be made. Overall, the pre-trained fastText model (1) is the one that performs the best with a macro-averaged F_1-score of 0.69, a non-neutral F_1-score of 0.50 and an AUC score of 0.72. Nevertheless, the other models are not inferior, with macro-averaged F_1-scores ranging from 0.63 to 0.69 and AUC-scores from 0.64 to 0.71. The confusion matrix for the pre-trained fastText model (1) can be found in Table 6.32.

Table 6.31.: Step one: Result for word embeddings on standard LSTM.

Word emb.	**Params.** (Model, Dim.size, Win.size, Word count)	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Macro F_1-score**	**AUC**
None w/o cw	–	0 1	0.87 0.48	0.93 0.32	0.90 0.39	0.64	0.63
None w cw	–	0 1	0.89 0.37	0.83 0.49	0.86 0.42	0.64	0.66
fastText w/o cw	SG, 100, 5, 5	0 1	0.88 0.55	0.93 0.39	0.91 0.46	0.68	0.66
fastText pre-trained (1)	SG, 100, 5, 5	0 1	0.91 0.43	0.84 0.60	0.87 0.50	0.69	0.72
fastText pre-trained (2)	CBOW, 300, 5, 5	0 1	0.90 0.38	0.82 0.54	0.86 0.44	0.65	0.68
Word2vec pre-trained	CBOW, 100, 5, 5	0 1	0.91 0.43	0.84 0.58	0.87 0.49	0.68	0.71
GloVe pre-trained	GV, 100, 15, 5	0 1	0.91 0.40	0.81 0.61	0.86 0.48	0.67	0.71
Word2vec own trained (1)	CBOW, 100, 5, 1	0 1	0.89 0.39	0.84 0.49	0.86 0.43	0.65	0.67
Word2vec own trained (2)	CBOW, 100, 5, 2	0 1	0.90 0.37	0.81 0.55	0.85 0.44	0.65	0.68
Word2vec own trained (3)	CBOW, 300, 5, 2	0 1	0.89 0.42	0.87 0.46	0.88 0.44	0.66	0.67
Word2vec own trained (4)	CBOW, 300, 15, 2	0 1	0.90 0.38	0.81 0.56	0.86 0.45	0.65	0.69
Word2vec own trained (5)	SG, 100, 5, 1	0 1	0.90 0.40	0.83 0.56	0.86 0.47	0.66	0.69
Word2vec own trained (6)	SG, 100, 5, 2	0 1	0.90 0.41	0.84 0.53	0.87 0.46	0.66	0.69
Word2vec own trained (7)	SG, 300, 5, 2	0 1	0.91 0.39	0.82 0.58	0.86 0.47	0.68	0.69
Word2vec own trained (8)	SG, 300, 15, 2	0 1	0.90 0.44	0.87 0.51	0.88 0.48	0.68	0.69
Word2vec improved (1)	100, 5, 1	0 1	0.88 0.38	0.86 0.41	0.87 0.40	0.63	0.64
Word2vec improved (2)	100, 5, 2	0 1	0.90 0.40	0.84 0.52	0.87 0.46	0.66	0.68
Word2vec improved (3)	300, 5, 2	0 1	0.89 0.43	0.88 0.45	0.88 0.44	0.66	0.67
Word2vec improved (4)	300, 15, 2	0 1	0.90 0.40	0.83 0.57	0.86 0.47	0.67	0.70

Word emb.	Params. (Model, Dim.size, Win.size, Word count)	Cat.	Prec.	Rec.	F_1-score	Macro F_1-score	AUC
fastText own trained (1)	CBOW, 100, 5, 1	0 1	0.89 0.45	0.88 0.48	0.89 0.46	0.67	0.68
fastText own trained (2)	CBOW,100, 5, 2	0 1	0.90 0.38	0.81 0.58	0.85 0.46	0.66	0.70
fastText own trained (3)	CBOW, 300, 5, 2	0 1	0.89 0.42	0.86 0.50	0.88 0.46	0.67	0.68
fastText own trained (4)	CBOW, 300, 15, 2	0 1	0.90 0.44	0.87 0.50	0.88 0.47	0.67	0.69
fastText own trained (5)	SG, 100, 5, 1	0 1	0.90 0.42	0.84 0.56	0.87 0.48	0.68	0.70
fastText own trained (6)	SG, 100, 5, 2	0 1	0.90 0.42	0.84 0.56	0.87 0.48	0.68	0.70
fastText own trained (7)	SG, 300, 5, 2	0 1	0.90 0.40	0.83 0.56	0.87 0.47	0.67	0.69
fastText own trained (8)	SG, 300, 15, 2	0 1	0.89 0.45	0.88 0.49	0.89 0.47	0.68	0.68
fastText extended (1)	300, 5, 2	0 1	0.90 0.42	0.85 0.55	0.87 0.47	0.67	0.70
fastText extended (2)	300, 5, 1	0 1	0.90 0.43	0.85 0.56	0.87 0.48	0.68	0.70
fastText extended (3)	300, 15, 2	0 1	0.91 0.39	0.81 0.62	0.86 0.48	0.67	0.71

Table 6.32.: Step one: Confusion matrix for pre-trained fastText LSTM.

True / Predicted	0	1
0	8599	1650
1	845	1248

Second step: Multiclass classification

All the results from the second step of the two-step approach can be found in Appendix A. The word embedding models have an average macro-averaged F_1-score of 0.25, with the scores varying from 0.20 to 0.27. A decision was made to only present three different models due to there barely being any differences in the metrics, and no model notably differ from the others. We choose to present one pre-trained model and one each of the own trained Word2vec and fastText models. They are shown in Table 6.33. The pre-trained fastText model (1) is presented because it was the best-performing word embedding from the first step. The own-trained Word2vec model (8) and the own-trained

fastText extended model (2) are also presented due to having the best performance from each word embedding class, respectively. The confusion matrix for the pre-trained fastText model (1) is presented in Table 6.34.

Table 6.33.: Step two: Evaluation metrics on word embeddings LSTM.

Word emb.	**Params.** (Model, Dim.size, Win.size, Word count)	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Macro F_1-score**
fastText pre-trained (1)	SG, 100, 5, 5	2	0.60	0.66	0.62	0.23
		3	0.26	0.13	0.18	
		4	0.09	0.12	0.10	
		5	0.02	0.04	0.03	
Word2vec own trained (8)	CBOW, 300, 15, 2	2	0.64	0.62	0.63	0.26
		3	0.35	0.20	0.26	
		4	0.09	0.23	0.12	
		5	0.03	0.02	0.03	
fastText extended (2)	300, 5, 2	2	0.60	0.62	0.61	0.27
		3	0.31	0.24	0.27	
		4	0.10	0.13	0.12	
		5	0.07	0.11	0.09	

Table 6.34.: Step two: Confusion matrix for the pre-trained fastText model (1).

True / Predicted	**2**	**3**	**4**	**5**
2	481	114	81	58
3	221	48	60	28
4	75	15	14	8
5	31	6	6	2

6.7.3. Results: Deep learning models

After investigating the impact word embeddings had on the standard LSTM model, we wanted to try different deep learning models to check how they would affect the final results. The same models presented in Section 5.2, i.e. LSTM, CNN, BiLSTM and a combination between these, are tested. For simpleness, all the models are tested on a pre-trained fastText embedding since this was the best-performing word embedding from the ablation study presented in the previous section.

First step: Binary classification

As can be seen in Table 6.35, there is only a moderate difference in the metrics between the tested models, with the macro-averaged F_1-scores ranging from 0.67 to 0.68 and

AUC scores ranging from 0.68 to 0.69. Overall, looking at the non-neutral F_1-score, the macro-averaged F_1-score and the AUC score, CNN in combination with LSTM and BiLSTM performs best. The best deep learning model here still does not outperform the standard LSTM model with word embeddings. The confusion matrix for CNN with LSTM is shown in Table 6.36.

Table 6.35.: Step one: Evaluation metrics for the deep learning models.

Model	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Macro F_1-score**	**AUC**
CNN	0	0.89	0.89	0.89	0.68	0.68
	1	0.46	0.48	0.47		
BiLSTM	0	0.90	0.85	0.87	0.67	0.69
	1	0.42	0.53	0.47		
CNN + LSTM	0	0.90	0.89	0.89	0.68	0.69
	1	0.47	0.49	0.48		
CNN + BiLSTM	0	0.90	0.87	0.89	0.68	0.69
	1	0.45	0.51	0.48		

Table 6.36.: Step one: Confusion matrix for CNN + LSTM.

True / Predicted	**0**	**1**
0	9071	1178
1	1064	1029

Second step: Multiclass classification

The results from the second step are presented in Table 6.37. As can be seen, there are some differences in the total F_1-score. The average F_1-score is 0.26, with the scores ranging from 0.25 to 0.27. The confusion matrix for CNN with LSTM is presented in Table 6.38.

Table 6.37.: Step two: Evaluation metrics for the deep learning models.

Model	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Macro F_1-score**
CNN	2	0.58	0.56	0.57	0.25
	3	0.30	0.32	0.31	
	4	0.09	0.10	0.10	
	5	0.03	0.02	0.03	
BiLSTM	2	0.61	0.60	0.61	0.27
	3	0.29	0.25	0.27	
	4	0.08	0.14	0.11	
	5	0.08	0.07	0.08	
CNN-LSTM	2	0.57	0.62	0.59	0.27
	3	0.34	0.36	0.35	
	4	0.08	0.04	0.05	
	5	0.11	0.04	0.06	
CNN-BiLSTM	2	0.58	0.56	0.57	0.26
	3	0.32	0.42	0.36	
	4	0.06	0.02	0.03	
	5	0.07	0.05	0.06	

Table 6.38.: Step two: Confusion matrix for CNN-LSTM.

True / Predicted	**2**	**3**	**4**	**5**
2	359	182	31	9
3	155	112	16	4
4	61	28	4	3
5	33	10	0	2

6.8. Combining classic and deep learning models

Lastly, we conducted experiments where we combined the deep learning and classic models from our previous experiments. In this section, we will present the best results from the second step where the three different classic models, namely LTR, LR and LinearSVM, were tested. In the first step, the standard LSTM model was used.

LTR achieved low F_1-scores for categories 3-5, indicating that the model is mostly able to predict category 2 and the macro-averaged F_1-score was 0.24. LinearSVM achieved relatively high macro-averaged F_1-scores with the best model achieving 0.48, but on average, LR with character-level vectors achieved the highest macro-averaged F_1-score with 0.51. Thus, we have chosen to present the confusion matrix and evaluation metrics for this model in Table 6.39 and Table 6.40.

Table 6.39.: Step two: Confusion matrix for LR with character-level vectors.

True / Predicted	**2**	**3**	**4**	**5**
2	600	59	46	9
3	191	116	29	4
4	44	21	48	3
5	16	10	1	17

Table 6.40.: Step two: Evaluation metrics for LR with character-level vectors.

Category	**Precision**	**Recall**	**F_1-score**	**Support**
2	0.71	0.84	0.77	714
3	0.56	0.34	0.42	340
4	0.39	0.41	0.40	116
5	0.52	0.39	0.44	44
Macro avg	0.54	0.50	0.51	1214
Weighted avg	0.63	0.64	0.62	1214

7. Evaluation

This chapter provides an interpretation and evaluation of the experimental results presented in Chapter 6. An explanation of how the results may have been impacted will also be provided. Lastly, we will present a comparison of the different models that have been tested and look at the results in a bigger picture.

7.1. Standard models

In this section, we will present the evaluation of the experimental results with standard models. First, an evaluation of the results using the all-in-one approach is presented, followed by an evaluation of the results using the two-step approach.

7.1.1. All-in-one approach

Looking at the summary presented in Table 6.12, we can see that the models achieved relatively high accuracy scores, ranging from 78.7% to 83.3%. However, as we briefly mentioned in Section 6.4.5, based on their low macro-averaged F_1-scores, it appears that all models are heavily influenced by their ability mostly to classify the neutral comments correctly. This implies that the models might not be equally good at classifying the non-neutral comments.

LTR achieved the second-highest accuracy and macro-averaged F_1-score of 81.6% and 0.29, respectively. When looking at the evaluation metrics presented in Table 6.5, it is clear that the F_1-score is highest for category 1, and significantly lower for the other categories, with decreasing values for category 2-5. The amount of correctly classified comments in category 1 is a major contributor to the accuracy score. However, we can also see from the confusion matrix in Table 6.4 that LTR is the model that classifies most comments in category 5 correctly, with a total of 21 comments. Regardless of this, the model performs poorly for categories 2-5, ranging from 3% to 28% in the number of comments classified correctly.

We can see from Table 6.6 that LR only predicted a single comment in category 5, and rather predicted 77% of them as neutral comments, and still achieved an accuracy score of 83.3%. The evaluation metrics in Table 6.7 and the macro-averaged F_1-score of 0.26 further emphasises this model's poor performance for the non-neutral categories. We can see from the confusion matrix in Table 6.6 that the amount of correctly classified

comments in category 1 is almost the sole contributor to the high accuracy score. It may seem like the model simply categories most of the comments as neutral. Thus, the accuracy of 83.3% is not an accurate description of this model's ability to predict non-neutral comments. Furthermore, we can see that LR performed slightly better in categories 2 and 3 than categories 4 and 5. However, with F_1-scores ranging from 0.03 to 0.04, we can conclude that the high accuracy score is simply due to the amount of correctly classified neutral comments, as we can see from the F_1-score of 0.91 for category 1. With the main research question in mind, where the goal is to identify offensive utterances and grade them, it is not very helpful that this model predicts 94.9% of the comments to be neutral and is good at predicting neutral comments only.

LinearSVM displays more similar behaviour to LTR than to LR with predictions in all categories, as seen from Table 6.8. Also, it achieves somewhat higher F_1-scores for categories 2-5 than what LR performed, as seen from Table 6.9. It is clear that this model also performs best in category 1, and achieves relatively good F_1-scores for category 2 and 3 compared to the previous models. However, it appears that this model struggles with category 4, predicting most of these comments to be in lower categories. Compared to LR, LinearSVM achieved a lower accuracy score at 79.6%, even though this model was able to classify 13 comments in category 5 correctly, and in total predicted a greater amount of comments correctly in categories 2-5. It also achieved the highest macro-averaged F_1-score at 0.31, which might be due to the use of balanced class weights.

Again, it seems to be a trend where LTR and LinearSVM achieves a lower accuracy score than LR because they spread their predictions a bit more evenly throughout the five categories. They achieve better F_1-scores for the non-neutral categories, but this also means that the model wrongly predicts more comments in each category, which in total lowers the accuracy. From the confusion matrices, we see that LTR predicts 94.4%, LR predicts 94.9% and LinearSVM predicts 87.6% of the comments to be neutral. Remembering that 83.1% of the test set was annotated as neutral, it is easy to see that the high accuracy score of each model is related to the number of neutral comments it is able to predict correctly. With the goal of detecting non-neutral comments in mind, we can argue that it is a significantly better result to achieve a higher macro-averaged F_1-score than simply the highest accuracy score.

Lastly, we take a closer look at LSTM. This model achieved the lowest accuracy score of 78.7% and the second-lowest macro-averaged F_1-score of 0.28. However, looking at Table 6.11, it is not evident that this model performs significantly worse than, for instance, LR. LSTM achieves the lowest F_1-score for category 1, but the highest F_1-score for category 2 and second-highest for category 3. It also performs slightly better than LR in category 4 and 5. However, from Table 6.10, we can see that this model also predicts most of the comments in categories 1 - 3 and only two comments in category 5. This implies that the model struggles with the two most severe categories and that it might not be the best model to use when the goal is to detect all of the non-neutral comments.

A commonality for all of the models is that they perform significantly better for category

1 than the rest of the categories. As discussed, some of the models perform better than others on categories 2-5, but it seems to be a trend where the performance decreases for each category. One reason as to why LinearSVM performs, in general, better for the non-neutral categories than the others is that it is implemented with balanced class weights. Without these class weights, the models will most likely classify comments as neutral due to the unbalanced dataset with a significantly larger proportion of neutral comments.

Another factor worth noticing is how the unbalanced dataset affects the amount of correctly classified comments for each category. Seeing as the training set contained 82.7% neutral comments, it is natural to assume that the decreasing number of correctly classified comments can be due to a smaller and not sufficient amount of training data in these categories.

7.1.2. Two-step approach

In the first step, the models performed roughly the same. Accuracy scores ranged from 79.5% to 84.4%, micro-averaged F_1-scores from 0.56 to 0.67, and AUC scores from 0.55 to 0.68. However, there was a significant difference in the number of true negatives found, from 266 to 1035, which can affect the results in step two.

In the second step, LTR used a test set of only 266 comments. From Table 6.15, we can see that the model performed best in category 2 and that category 4 stands out with a significantly worse F_1-score of only 0.04. This indicates that the model struggles with this category, and as we can see from Table 6.14, it is classifying most of these comments as category 2 or 3. In total, 75.9% of the comments in the test set were classified as category 2, and very few comments were classified as either category 4 or 5.

With 1035 comments, LR was evaluated with the most extensive test set. LR achieved the highest accuracy of 58.7% and also the highest macro-averaged F_1-score of 0.42. From Table 6.18, we see that the model is best at predicting comments in category 2, but also achieves relatively high F_1-scores for categories 3-5. Looking at Table 6.17, we see that this model predicts 72.2% of the comments to belong to category 2, but that it also predicts a more considerable amount of comments in each of the other categories compared to LTR.

LinearSVM was also evaluated with a large test set, consisting of 979 comments. On average, it achieved F_1-scores similar to LR, and performed evenly across categories 3-5 as can be seen from Table 6.21. It is also worth noticing from Table 6.20 that this model predicts comments more evenly in all categories, as LR did, instead of mainly predicting comments in category 2, as LTR did. This is reflected in the model's performance, where LinearSVM performed second-best in both accuracy and macro-averaged F_1-score.

LSTM, on the other hand, achieved the lowest accuracy and macro-averaged F_1-score of the models in step two with scores of 49.4% and 0.25. From the confusion matrix in Table 6.23, we see that this model predicts zero comments in category 5, almost no

comments in category 4, and few in category 3. Furthermore, we can see from Table 6.24 that the model has the lowest F_1-scores in all categories, with 0.00 in category 5. The model seems to classify a large majority of the comments as category 2, resulting in the low accuracy and F_1-scores. Also, this model achieved lower macro-averaged F_1-score in step two than all the models did in the all-in-one approach.

Overall, all models seem to classify most comments in category 2. This can be compared to the results from the all-in-one approach, where the models tended to classify most comments in category 1. Again, the results in the two-step approach might be due to the fact that the training set contained most comments in category 2, the category closest to neutral. Looking at the F_1-scores from all models further supports this. Another important factor is the number of comments in each of the test sets. LTR had the fewest comments, and this might also be a contributing factor as to why this model classifies almost no comments in higher categories. However, LSTM showed similar tendencies and performed worse than LTR, but still had a test set that was almost three times the size. This again indicates that the size of the test set is not the only contributing factor behind these results. In addition, LTR and LSTM did not have balanced class weights, which LR and LinearSVM were implemented with. With a macro-averaged F_1-score of 0.41 and 0.42, LinearSVM and LR seem to perform better than the others with regards to accuracy, F_1-scores and the distribution of predictions. This indicates that the use of balanced class weights will have a positive impact on the results.

7.2. Experiments with the English dataset

Experiments were conducted on an English dataset to evaluate the models and compare the results from an English dataset with our Norwegian dataset, as mentioned in Section 6.6. Seeing as the experiments on the English dataset is binary classification, it is most natural to compare with the results from the first step in the two-step approach. The summary presented in Table 6.26 aligns well with the results for the Norwegian dataset, which ranges from 79.6% to 84.4% in accuracy scores. The macro-averaged F_1-scores also aligns well, with the results from the Norwegian dataset being slightly lower, ranging from 0.56 to 0.66. For both datasets, it is clear that LTR performs significantly worse than the other models when using the macro-averaged F_1-score as an evaluation metric, and that the other three models perform very similarly. It is also worth noticing that the models tested on the English dataset achieved almost identical accuracy scores and relatively close AUC scores in addition to the percentage of neutral and non-neutral comments classified correctly, indicating that they performed approximately the same.

To summarise, the similar results for the English dataset compared to the Norwegian dataset implies that the models are working as they should and that the results for the Norwegian dataset can be trusted.

7.3. Deep learning

In this section, we will present the evaluation of the experimental results with deep learning models. First, an evaluation of the results using word embeddings with a standard LSTM model is presented, followed by an evaluation of the results testing various deep learning models.

7.3.1. Word embeddings

Several interesting observations can be made from Table 6.31, which shows the results from the first step with the word embedding models. One of these is that the own trained Word2vec models with CBOW representation generally have a worse AUC and F_1-score for the non-neutral category than the SG representation. The same observation can be made for the fastText own trained models. In general, the fastText own trained models have a better AUC and F_1-score than the Word2vec own trained models. One explanation to this can be that fastText matches unknown words through using n-grams, which could be an advantage in a corpus with potentially many spelling mistakes and use of dialect. Thus, one could assume that this would give better results since GloVe and Word2vec do not have this same option for generating word embeddings for "OOV" (out of vocabulary) words.

The pre-trained fastText model (2) performs the worst of all the pre-trained models, with a macro-average F_1-score of 0.65 and a non-neutral F_1-score of 0.44. This may be explained by this model just having half the vocabulary size of the other pre-trained models. Despite this, it still covers 78.8% of the vocabulary compared to the other pre-trained models which covered 85%. Thus, a decision was made to extend this vocabulary with our dataset, resulting in three extended own trained fastText models. Interestingly, the results seem to improve with the additional vocabulary. The non-neutral F_1-score of the extended models had an average of 0.48, meaning that the score increased by 0.04 from the pre-trained model.

Both models without word embedding and without class weights are tested to investigate whether including them have an impact on the final result or not. Unsurprisingly, the neutral F_1-score for the models without class weights are significantly higher than the models with class weights. This is due to the imbalanced dataset with an overweight of neutral in the first step and overweight of category 2 in the second step. Looking at the differences in recall for the models without class weights compared to the ones with, we can observe a notable decrease in recall. Without class weights, the models, None w/o cw and fastText w/o cw, achieve a recall of 0.32 and 0.39 respectively. The same model without word embedding, but with count weight, has a recall of 0.49, while the pre-trained fastText model (1) achieves 0.60. The general average for the other models is 0.53. Thus, it is evident that using class weights is reasonable, considering the goal of detecting offensive utterances.

In addition, there is a considerable difference in the non-neutral F_1-score between the

models using word embeddings and not. The models without word embeddings had an F_1-score of 0.39 and 0.42 and the general average of F_1-score for the models with word embeddings was 0.46. The model without word embedding but with class weights had higher recall than the fastText without class weights. Despite this, the average for the models using a combination of class weights and word embedding is notably higher. Thus, using word embeddings seems to improve the standard LSTM model further.

From the confusion matrix in Table 6.32 of the first step with the pre-trained fastText model (1), we can see that 1248 comments were classified as true non-neutral. Thus, there are a greater amount of predicted cases of non-neutral in the first step than for the standard models. This is a general trend for all the word embedding models, and this is supported by the significantly higher recall for the models with, compared to the models without word embeddings.

Table 6.33 is a general representation of the results from the second step presented in Appendix A. No model stands out positively, and there are no clear differences between the models in terms of which type of model and representation is being used, dimension size, window size or word count. There is also no correspondence between the performance of the models in the first and second step.

As with the standard models, the models are able to correctly classify utterances in category 2 quite well, while misclassifying most of the severe categories such as categories 4 and 5. This is reflected in how all the metrics for these categories are generally low. Thus, these models' F_1-score is mainly increased due to the two first categories being high.

The confusion matrix in Table 6.34 shows that the model tends to have a lower threshold to predict most comments in the two first categories, meaning that the model confuses between the classes. This corresponds with the evaluation metrics in Table 6.33, and the metrics from the standard models. In addition, this also supports the findings from the standard models that the results may be due to the dataset being skewed.

7.3.2. Deep learning

The results from the first step with the deep learning models show that there is not much differing from the word embedding models. The average macro-averaged F_1-score is 0.68, which is a bit higher than the average for word embeddings with 0.67. However, the deep learning models are only tested on the pre-trained fastText model (1), which had the best metrics overall from all the word embedding models tested on the standard LSTM model. Thus, there is a possibility that some word embeddings might be better suited for the tested deep learning models, but testing all the different word embedding models with different deep learning models were considered out of the scope for this thesis. The deep learning models have the same AUC score as the word embeddings with an average of 0.69. However, the spread between the word embedding models is larger, with several models having AUC scores of 0.71 and 0.72. In addition, the non-neutral F_1-score has an

average of 0.48, which is higher than the word embedding models with an average of 0.44. Here, again, the spread is much bigger between the different word embedding models.

An interesting observation which can be made about the deep learning models is that they on average have a higher F_1-score for category 2 compared to the word embedding models. Also, the same trend with the skewness of the predicted classes happens here as well; categories 2 and 3 are mainly the reason for high precision and recall values. However, as pointed out earlier, this is natural due to the distribution of instances for each class in the dataset. This can also be seen from the confusion matrix in Table 6.38. The model mostly predicts categories 2 and 3, and it is struggling with predicting categories 4 and 5. This is also consistent with the previous findings.

Overall, the deep learning models perform satisfactorily in the first step, since they achieve comparable results to the standard LSTM model. LSTM models are frequently used in text classification because they are known to learn the long-distance sequence order information among words, and this might be a reason as to why the LSTM model is giving good results. The CNN model also gets acceptable results and has more frequently been proven to work well on text classification, since it learns features and focuses on semantically close data. As proposed by the recent state of the art, a combination of the two types of networks might turn out beneficial. The combined model is the best performing of the tested deep learning models, thus supporting the recent state of the art.

Optimally, more hyperparameter optimisation could have been done throughout the testing of the deep learning models. However, as our goal was not to propose one final model as our solution, doing extensive testing was seen as out of the scope. The deep learning layers were still tested with different parameters such as dropout, activation functions and dimension sizes. Thus, the proposed models are the best-performing ones within this parameter optimisation.

7.4. Combining classic and deep learning models

The results of step two of the two-step deep learning approach were not satisfactory. However, for the standard models, step two of the two-step approach seemed promising. Thus, a decision was made to combine classic and deep learning models.

The classic methods from the standard models were tested in combination with the pre-trained fastText LSTM model (1). LTR was performing poor despite using class weights, and most of the instances were classified as category 2. LR and LinearSVM were performing approximately the same, but LR with character-level vectors had a slightly better performance with a macro-averaged F_1-score of 0.51. As can be seen from the evaluation metrics in Table 6.40, all categories' F_1-score are relatively good. The different predicted categories are evenly distributed which the confusion matrix in Table 6.39 confirms. This can be seen by the diagonal line in the matrix and the high amount

of predicted instances for each class. We can see that the model is not just trying to predict the category which has the most instances like the other models tend to do. Thus, this indicates that a combination of classic and deep learning model can be more effective when detecting hateful and offensive utterances.

7.4.1. Classification errors

In order to get a better understanding of what type of comments the models wrongfully classified, we decided to analyse the predictions further. It was also desirable to investigate the predicted comments more closely to gain insight into issues related to the dataset and annotation procedure.

From the analysis of all the models' performance, we decided to investigate the predictions made by the combination model of the pre-trained fastText model (1) in the first step and LR with character-level vectors in the second step, because this model achieved the best metrics. Table 7.1 first presents three comments that were annotated Neutral but predicted non-Neutral, followed by three comments that were annotated non-Neutral but predicted Neutral. Based on these comments, it is hard to draw any conclusions. Generally, the comments shown are highly dependant on context and the subjectivity of the annotator. The misclassified non-Neutral comments are hard to interpret and do not have a clear hateful message such as a call to action or common hateful terms. As an example, in the fourth comment, the word "kulørt" is used, which is a synonym for "coloured", but most times this word is used in a non-hateful way. This may be difficult for the model to interpret since the model is most likely not trained on similar hateful examples using "kulørt".

The two other misclassified non-neutral comments are originally annotated to category 2, the mildest degree of hatefulness. This might indicate that it is hard to differentiate between the mildest categories and neutral utterances. This issue is further emphasised if we look at the third comment, which contains a profane word such as "faen". However, as we stated in the annotation guidelines, this does not necessarily mean that an utterance is hateful. In this case, depending on the context and subjectivity of the annotator, the comment could possibly have been annotated as non-neutral.

Table 7.1.: Wrongful predictions made by pre-trained fastText model (1) in the first step. Neutral comments are labelled with 0, and non-Neutral are labelled with 1.

Category		Comment
True	**Predicted**	
0	1	Norsk statsborger fra Somalia hevet stønad fra NAV på vegne av sin døde mor
0	1	Er det noen i eller nær Oslo som vil ha en roomie? Er litt lei av sørlandet, så driver å kikker rundt etter noe nær Oslo. PM meg om noe kunne passe.
0	1	Navn Navn husmorsmafiaen er noe du faen ikke kødder med!
1	0	Navn Navn Navn glad i kulørte gutter tror jeg
1	0	udokumentert? kanskje du skal våkne snart?
1	0	Litt artig at de betegner seg selv som skeive, i motsetning til de rette. En slags ufrivillig selvinnsikt.

As pointed out earlier, most of the models are struggling to predict categories 4 and 5 correctly, and tend to classify these comments in lower categories. Thus, we looked at the span between the comments predicted as categories 2 and 5 to examine whether there were any significant findings behind these classification errors. Looking at Table 7.2 for the comments predicted as category 2 but annotated as 5, it is clear that the model struggles to understand the most severe comments and to find the call to action in the utterances when it is not written explicitly. An example of this can be seen from the second comment. In addition, comments such as the first comment may be too long for the model to be able to understand it properly. The third sample again demonstrates the difficulties of annotating, and it should probably have been annotated as neutral since we do not have enough context to decide the real intention of the utterance. However, from the samples, it is clear that there is a difference between the comments annotated as category 2 but predicted to be category 5, and those annotated as category 5 but predicted to be category 2. Thus, it should be possible to train models to be able to distinguish between them accurately. The misclassified comments in category 2 all contain some form of a call to action. Despite this, the fact that a comment contains a call to action does not necessarily mean that it automatically is considered a category 5 utterance. Even though category 5 mainly consist of the same types of comments that incite action, such as "burde skytes", there may not be enough instances of these types of comments in total, considering there are only 250 category 5 comments in the dataset.

Table 7.2.: Wrongful predictions made by LR in the second step. Provocative comments are labelled with 2, and Hateful are labelled with 5.

Category		Comment
True	**Predicted**	
5	2	Navn Navn Du er en idiot. Min mening er at alle som er som du burde skytes. Jeg har lov til å ha min mening. Jeg har også ytringsfrihet i Norge. Og tro du meg. Jeg ytrer meg. På lik linje med deg. Bare at jeg snakker sant. Du lager hysteri blant de yngre som vil føre til selvmord og i værste fall og en ungdom som konstant er redd. Jeg er overbevist om at du har en eller annen diagnose.
5	2	Jeg hadde til og med slengt de i en grill, steikt deilig grise koteletter. Koranen kan mette mange mager.
5	2	Det er Navn Navn! Fang han så fort som mulig. Dette er ikke greit, Navn
2	5	rydd opp i møkka
2	5	TV2 - nå må dere faen meg slutte med dette tullet og få fjernet den dommedagsreklamen dere sender som skremmer vettet av våre barn og unge!!!!!!!!!!
2	5	@USER @USER @USER Start med at sende kriminelle utlendinger hjem, og stopp innvandring fra ikke vestlige land.

There is a much higher percentage of comments in categories 2 and 3 in the dataset, which implies that there should be sufficient comments to train the model. However, whether an utterance is offensive or provocative can be hard to distinguish. Looking at Table 7.3, the samples collected confirm this. We can also see that these categories seem to suffer from overlapping annotations and the subjectivity of the annotator. The problem with the subjectivity of the annotators is displayed through the third comment, which in our opinion should be classified as Neutral. Again, it is essential to highlight the importance of context when annotating these comments. As an example, the fourth comment is an utterance that is highly dependent on the context, whether it is neutral or should be classified as something else. The overlapping annotations might imply that category 2 is too broad, meaning that there are too many varying comments in this category. A model will often not have been trained on all the possible utterances in such a broad category, which might result in the problems with distinguishing these comments.

Table 7.3.: Wrongful predictions made by LR in the second step. Label 2 indicates Provocative, and label 3 indicates Offensive.

Category		Comment
True	**Predicted**	
2	3	Alt er i sin skjønneste orden her i Norge, så hold kjeft!
2	3	For noen vanvittig store tapere dere begge er som sitter og snakker online om 2 mennesker som ikke engang aner dere eksisterer, hahaha. Jævla fjortis-jenter
2	3	Sutrelandet har det vel bedre enn de fleste.
3	2	@USER For en komplett usympatisk og ekkel mann.
3	2	Minner meg mer og mer om Stalin—
3	2	Navn Navn Gå å se deg i speilet og skjemmest, ditt naive sauhau.

In general, the difficulties in distinguishing between the categories might imply that our definition of the five categories can be improved. Also, these classification errors support the assumption that the annotation of comments is dependent on the subjectivity of the annotators and the context in which the comment is written.

7.5. Comparison

An apparent observation, when comparing all the models, is that all of them can predict neutral comments reasonably well compared to their ability to predict non-neutral comments. In general, all models seem to be affected by the imbalanced dataset. Class weights have been used to reduce the effect of this imbalance hopefully.

Looking at the results for the standard models with the all-in-one approach compared to the two-step approach, it is clear that the F_1-score for the non-neutral categories are almost without exception better for the two-step approach. Another observation is that the all-in-one approach, in general, predicts too few comments in the non-neutral categories. This implies that the use of a two-step approach is advantageous when the goal is to detect offensive and hateful utterances of varying degree. However, it is worth mentioning that the differences might be due to factors such as the random seed or the train-test split as well.

When dealing with imbalanced classes, the AUC score is often the preferred metric, which is why we also have evaluated our results using this. Looking at results from the two-step approach, the AUC score for the first step of the standard models ranges from 0.55 to

0.67, with an average of 0.61. This is significantly lower than for the deep learning models, which had an average of 0.67. The AUC score is reflected in the confusion matrices for the different models. As can be seen from the confusion matrix of the pre-trained fastText model (1) in Table 6.32, it returns more true non-neutral comments than all the standard models do, which further becomes valuable in step two of the two-step approach. A high number of true positives also has a positive impact on our goal of detecting as many offensive utterances as possible.

In the first step, class weights make the distribution of categories more balanced, as can be seen in the difference between the confusion matrix for the standard LSTM model compared to the confusion matrices for pre-trained fastText (1) and CNN-LSTM. LR and LinearSVM from the standard models are also using class weights, which might explain the good AUC score achieved in both steps for this model. This effect is not as prominent when dealing with multiclass classification for the deep learning models, but LR and LinearSVM still achieve good results. However, when looking at the confusion matrix for LSTM with class weights in Table 6.38, we see that using class weights still have a positive impact.

Furthermore, almost all models predict more comments in categories 2 and 3 when using multiclass classification, regardless of whether it is a standard model, a model with word embeddings, or a deep learning model being used. An explanation for this might be that there are more instances of these comments in the dataset.

Although the initial goal of experimenting with deep learning models was to improve the performance from the standard models, there only appears to be positive differences in the results for the first step of the two-step approach. The deep learning models struggle with multiclass classification in step two, especially with predicting comments in categories 4 and 5. The average macro-averaged F_1-score for the deep learning and word embedding models are 0.26 and 0.25, respectively. This is inferior to the classic models in the standard two-step approach. LR and LinearSVM achieve the best results of all models for multiclass classification. Combining classic and deep learning models proved to be effective, where a combination of pre-trained fastText (1) model and LR is evaluated to be the best model since it in total predicted the most non-neutral instances correctly. Thus, this indicates that a combination of classic and deep learning model can be more effective when detecting hateful and offensive utterances.

8. Discussion

In this chapter, we will discuss interesting aspects of our research, in light of the state of the art analysis conducted in Chapter 3 and with regards to possible restrictions of the work itself. Also, our work is discussed with regards to the research questions presented.

8.1. Dataset

A drawback to our dataset is the imbalance between the categories. Especially the proportion of comments in the higher categories seem to be insufficient to properly train our models, despite our efforts to search for inflamed words actively and to exclude presumably neutral topics. As an alternative solution, it would be preferable to exclude a portion of the neutral comments we have gathered to balance the dataset. However, this would result in a smaller dataset which is a disadvantage when working with machine learning. Also, the amount of non-neutral comments in our dataset aligns with the results from both the Danish and Norwegian reports (Zuleta and Burkal (2017); Veledar (2018)) as well as the findings from Veledar (2018), as mentioned in Section 6.1.3. This indicates that our dataset complies with the current state-of-the-art with regards to hate speech datasets and also gives us the impression that our dataset reflects a natural representation of comments in each category.

Considering that we collected parts of the dataset from a known xenophobic website, our dataset contains a high proportion of comments discussing immigration and Muslims. However, this corresponds with the findings from Veledar (2018). When asked the question "Which groups are most exposed to hateful comments on Facebook?", 41% answered people with another ethnicity than Norwegian, and 31% answered politically active persons. This supports our choice of choosing inflamed topics such as politics and immigration as search terms. Nonetheless, this will still lead to an imbalanced dataset with regards to distributions of topics which are considered hateful. This imbalance might result in our models being overfitted to such topics and performing worse for new topics that are not included in the dataset. When evaluating classification errors in Section 7.4.1, we found the same tendency to misclassify comments related to unknown topics.

A challenge when collecting data through search terms was to discover more general bullying. To the best of our knowledge, hate speech detection on topics such as gender, appearance and obesity is less prevalent in the research field. However, these topics are considered typical in cases of bullying. Thus, a future improvement would be to conduct

a more targeted collection of data to balance the dataset by including more topics like these.

The sources from which a dataset is made is also an interesting factor to discuss. Our dataset consists of comments from Twitter, Facebook and Resett. However, all three sources were crawled over time and from various posts and search terms to increase the diversity of the dataset and avoid imbalance with regards to a specific topic or trend. Our goal is to increase the quality of the public debate, and thus all of our comments are gathered from public pages and not from groups on Facebook. Some of these groups are known for their debates on inflamed topics such as immigration and gender equality, and participants often have strong political opinions. An example of this is the group *Stopp islamiseringen av Norge* (SIAN), which is frequently debated in media. We have not had access to these groups, and we have also chosen not to try to gain access because we do not consider them as part of the general public debate. However, we suspect that the proportion of comments in category 3-5 would increase significantly if a dataset was made from these sources.

8.2. Definition of hate speech

As previously discussed in Section 4.2, there are several challenges related to grading utterances and the definition of hate speech. When working with our definition of hate speech, we focused on the fact that the appearance of a specific word was not enough to classify a comment as non-neutral. Examples of this include sentences such as "Muslimer har ikke gjort noe galt" and sentences that include obscene words used in a neutral setting. From our n-gram analyses that were presented in Section 6.2, we found ten n-grams that were added as features to help train the standard models, including words such as "muslimer". However, we believe that it is not right to link these words automatically to the non-neutral categories when defining comments. This is partly based on the fact that most of the content published online is neutral. Thus such words will be part of neutral comments as well, and it would be very generalising to assume that all comments regarding Muslims are hateful. A consequence of this choice might be that more comments were defined as neutral when the annotators followed our annotation guidelines. The lack of a pattern regarding these words might have confused our models which are trained with some of them as features in the non-neutral comments.

The difference between tone and toxicity is also a factor we considered when working with our definition of hate speech, after studying an analysis of Google's Perspective API[1]. A comment might be written with a neutral tone and not contain any offensive terms or curse words, but still be hateful. Tone is closer aligned to sentiment, where the emotions of the author shine through, whereas toxicity can be more unconscious bias. Thus, it was important for us to define hate speech in such a way that a comment with a negative tone but no aggressive language or derogatory opinions directed towards a target could be classified as neutral, and that a comment which expressed its opinion in a neutral tone but ascribed negative features to a target would not be classified as neutral. Nonetheless, if a text does not contain any offensive terms or curse words, we are most likely to misclassify hate speech.

8.3. Annotation procedure

How the annotation procedure is done is a factor to consider when evaluating our research. There is a trade-off between a more time-efficient process and the lack of domain experts when crowd-sourcing the task of annotating our dataset. As discussed in Section 3.4, studies indicate that interpretation of hate is highly dependent on the individual's perception, resulting in low inter-annotator agreement. The subjectivity of each annotator will affect the dataset whether or not crowd-sourcing is used because the bias of the authors/researchers will also become evident if we annotate the dataset ourselves. However, there are large differences in agreement amongst annotations done by crowd-sourcing compared to annotations done by domain experts, as pointed out in Section 6.1.3.

One argument for crowd-sourcing is that it would be more time-efficient. Unfortunately, we did not have sufficient funds to ensure that at least three annotators annotated each comment. This could also have contributed to avoiding unwanted bias. However, as stated by Waseem (2016), the annotation agreement of amateur annotators is comparable to expert annotators if there is full agreement between the amateur annotators. Based on this, we provided the crowd-sourced annotators with detailed guidelines, in the hopes of eliminating some of the individual subjectivity. The initial test where the authors annotated 2500 comments resulted in only fair agreement, with a Fleiss's kappa score of 0.3869. This might be due to the annotators not being familiar enough with the annotation guidelines, the subjectivity of each annotator and perhaps sometimes annotating the comments too quickly and not fully grasping the content they read. It is natural to assume that these factors might also have affected the crowd-sourced annotators.

Our experience with external annotators varied. When analysing the annotations, we found that some of the annotators classified more comments than average in the non-neutral categories, often generating clusters of 2s and 3s. Some took their time and

[1] https://medium.com/@carolinesinders/toxicity-and-tone-are-not-the-same-thing-analyzing-the-new-google-api-on-toxicity-perspectiveapi-14abe4e728b3

split the task into smaller pieces which might have contributed to the annotators being more focused and not following a particular pattern when annotating. Others finished the task very quickly, which might result in their work being more prone to possible misclassifications. Although the gender distribution of the annotators was quite balanced, the group of annotators is not a representative sample of society. Most of them study computer science, are in their twenties and have the same ethnicity. This is worth noticing as it might have affected how our annotation guidelines have been interpreted, and we emphasise the issue of subjectivity.

8.4. Classification approaches

As can be inferred from Section 6.4, results from the standard models with the all-in-one approach were not satisfactory, and most of the comments were classified as neutral. Thus, in accordance with the recent state of the art, we decided to try out a two-step approach. As pointed out in Section 3.4, H. Liu et al. (2019) looked into multistep learning because using only binary classification would not be optimal for real-world scenarios. Vigna et al. (2017) achieved better results with a binary classification compared to a three-class classification. Thus, we decided to implement a two-step approach using a binary classification in the first step and multiclass classification in the second step. If we compare the number of predicted non-neutral comments in the all-in-one approach to the two-step approach, it is clear that the total amount increases significantly in the first step of the two-step approach. For the standard models, the number of instances found increased by 168% on average. Our underlying goal is to find as many hateful utterances as possible. Thus, these results indicate that a two-step approach would be more suitable to find non-neutral comments than the all-in-one approach. Hopefully, a two-step approach will give a more nuanced classification because it first classifies the comments which are clearly neutral, instead of the more complex problem of predicting a comment into one of five categories.

It is evident from both our state of the art analysis and further experimentation with the standard models, that the classic models were not an unfavourable choice. However, due to our interest in exploring deep learning to classify hate speech, it was desirable to research this more. The results showed that the deep learning models indeed found more instances of non-neutral comments in the first step of the two-step approach, but struggled with multiclass classification. The standard models had a better distribution of instances predicted in the second step overall, and thus a combination of both deep learning and classic models might seem favourable, as is also suggested by Park and Fung (2017).

8.5. Censorship and freedom of speech

There are several challenges and ethical dilemmas related to the development of automatic hate speech detection. As we briefly mentioned in Chapter 1, it is extremely important not to induce censorship of users as freedom of speech is a basic human right. Possible negative outcomes with this technological advance include misuse by directing the system at voices that do not express hatred and the possibility to suppress constructive criticism or opinions that dissent with a system. Our stance is that the process of removing and censoring comments ultimately should be done by a human. In other words, if our model classifies a comment as non-neutral, it should be inspected by a human moderator who makes the decision on whether to remove, moderate or allow the comment.

However, the challenge of censorship is always present even when human moderators control online debates. The freedom of speech is made to ensure that utterances that do not comply with the majority's opinions are also allowed a place in the public debate. Thus, it is crucial that the development of automatic hate speech detection does not cause only utterances that are formulated in a certain way or with certain opinions to be part of the online debate.

Our goal is more efficient filtering of utterances that harass others to maintain safe environments online that encourage everyone to participate in the online communities. To receive hateful comments can have a major impact on an individual's life and willingness to continue to participate in the debate. After our conversations with Heidi Wyller at *Diskrimineringshjelpen og Meglingsbanken* we became aware of the issues with utterances that are not forbidden by law but still is perceived as hateful by the majority. Thus, we hope to eliminate especially these comments and to do so it will be particularly relevant to identify and correctly classify comments in category 4 and 5 seeing as these are the most severe.

8.6. Anomaly detection

Anomaly detection could have been chosen as an alternative approach to answer our research questions. This is a method in machine learning and data mining to identify rare items, also known as outliers, that differ significantly from the majority of the data. In the case of hate speech detection, hate speech can be considered an anomalous variant of ordinary speech. A dataset will be labelled as "normal" and "abnormal" to train a classifier to detect the anomalies. As previously discussed, a dataset on the topic of hate speech is often imbalanced, seeing as most of the comments online is considered to be normal. This fits well with the task of supervised anomaly detection because of the inherently unbalanced nature of this method.

In our case, anomaly detection could have been interesting to apply in the second step of the two-step approach. After the first step, we are left with small amounts of data in each category, which can cause problems for standard classification methods. This case

is more similar to anomaly detection because our goal is to detect those utterances that deviated from normal.

However, it is worth noting that this approach is also subject for the same vulnerabilities that we have discussed in the previous sections regarding the collection of the dataset, definition of hate speech, the annotation procedure and censorship of utterances. Furthermore, anomaly detection will only find the most severe comments as they differ the most from the ordinary ones. This might result in offensive and provocative comments being unnoticed by the moderators.

8.7. Transferability to other languages

An interesting aspect of our research, and hate speech detection in general, is whether or not the work is transferable to other languages than the original one. Based on our results when experimenting with an English dataset, it is clear that both approaches and the various models can be applied to a dataset in another language. However, to use the deep learning models, word embeddings must be downloaded or own trained for the desired language. Another possible approach is to use BERT multilingual[2], but there are some known issues with smaller languages, such as Norwegian, not having sufficient coverage in the model.

Research in other languages will have to collect their own dataset or use an existing one since our dataset is in Norwegian. Our approach on how to collect a dataset is transferable by crawling sources with the use of search terms in the desired language. However, as previously discussed in Section 8.3, it is important to consider which search terms that are used based on relevant topics in the country or region where the corresponding language is used, and not just translate our Norwegian search terms.

The annotation guidelines we have produced might not be fully transferable to another language. A lot of context and information can be lost in translation, which implies that the dataset can be annotated incorrectly if the annotation guidelines are translated from another language. This is also why we decided to make a definition for hate speech using Norwegian examples, to ensure in the best way possible that the dataset is annotated correctly.

[2]https://github.com/google-research/bert/blob/master/multilingual.md

8.8. Revisiting the research questions

The goal of this thesis has been to investigate the ability to identify offensive utterances and grade them based on how offensive they are. Our focus has been on utterances in the Norwegian language. Three sub-questions were formulated in order to answer our main research question, and we will discuss these in this section.

Research question 1 *Which existing methods and models have been effective when detecting offensive utterances this far?*

As previously mentioned, hate speech detection is an important field of study today as a result of the massive increase in user-generated data and hateful comments online. In Chapter 3, we present the current state of the art within existing research on hate speech detection, existing data collections, various classification methods and multiclass classification. Table 3.1 presents an overview of this.

Our state of the art analysis found that several of the more recent papers and workshops are combining multiple feature extraction methods, and that word embeddings have increased in popularity. Word2vec, GloVe and fastText have for instance been used by Z. Zhang et al. (2018) and Pavlopoulos et al. (2017), and also at international workshops such as SemEval 2019 (Zampieri et al., 2019b). Furthermore, transfer learning models such as BERT has increased in popularity as well. Based on this, we chose to implement our deep learning models with both pre-trained and own trained word embeddings. We also investigated the use of BERT, as mentioned in Section 6.7.1, but found it too time-consuming to precede with within the scope of this thesis.

In Section 3.3, we found that supervised machine learning approaches have been state-of-the-art previously, but deep learning models are now preferred. However, these supervised machine learning models are often used as a baseline for comparison, and as Biesek (2019) points out, often perform well on unbalanced datasets. Based on this, we decided to implement the four standard models to use as a baseline for comparison and to explore an ensemble of deep learning models. The architecture and implementation of both the standard models and the deep learning models are described in Chapter 5.

Another important finding from Park and Fung (2017), which we presented in Section 3.4, was that a two-step approach can perform as well as a one-step approach, and might even be advantageous if trained on separate datasets. Based on this, we decided to experiment with both a one-step approach and a two-step approach to detect more non-neutral comments, as previously discussed in Section 8.4.

Research question 2 *How can multiclass classification be used to determine the different degrees of the offensiveness of user-generated content systematically?*

We provided our definition of hate speech and offensive language, consisting of five categories ranging from neutral to hateful, in Chapter 4. This was essential to be able to perform multiclass classification on our dataset. These definitions were based on

our review of relevant literature, as described in Section 3.4, as well as consulting with Heidi Wyller regarding *Straffeloven* §185 and §186, and several reports from the Oslo Police Department.

Discussion on how to define hate speech has been conducted in Section 8.2. From our research, it is evident that there are several challenges with multiclass classification and how to determine different degrees of hate speech. We found that the number of classes used in multiclass classification varied. Often, three categories of hate were used, and then hateful messages were divided into distinct categories, as done by Bosco et al. (2018) and Vigna et al. (2017). There is no clear indication as to what the ideal number of categories for multiclass classification is. However, looking at the evaluations made in Chapter 7, it appears to be a trend where the models struggle to predict the most severe non-neutral comments, such as comments in categories 4 and 5. Also, the models seem to struggle with distinguishing between categories 2 and 3 as well. One reason for this might be the lack of data in each of these categories, resulting in the models not being adequately trained to identify these types of utterances.

An alternative approach to try and improve the results can be to merge categories 4 and 5 into a category of hateful utterances and merge categories 2 and 3 into a category of offensive utterances. With these alterations, it would presumably be more comments in each of the two non-neutral categories, which in turn will result in the model being better trained to classify these types of comments than before. This is supported by the findings of Golbeck et al. (2017), who started with six sub-categories, but then found that they overlapped quite a bit. From this, we can deduce that it is a common problem within the field of hate speech, and might be a reason as to why some of our models seem to struggle to distinguish categories 4 and 5 in particular.

Also, the total number of comments predicted as non-neutral was lower when using multiclass classification with the all-in-one approach compared to the two-step approach, as previously mentioned in Section 7.5. This corresponds to the research done by H. Liu et al. (2019), who implemented a three-level framework, first doing binary classification, and then identifying different types of hate speech. We tested a method with a deep learning model in the first step and a classic model in the second step of the two-step approach. Based on the evaluations made in Section 7.4, this seems like a promising method when using multiclass classification to determine the different degrees of the offensiveness of user-generated content.

Research question 3 *How can we build a good dataset, in a specific language, for experimental evaluations?*

In Section 3.2, we present the current state of the art within existing data collections and highlights the lack of a commonly accepted corpus. In order to build a good dataset, we reviewed relevant literature on existing data collections and how these had been constructed. This was then used as an inspiration to build our own dataset. Thus, our description of how to create a dataset in Section 6.1 is highly relevant for future

researchers when they are tasked with building their own dataset in another language.

The process of data collection, data cleaning and annotation procedure were often done in various ways, as described in the state of the art analysis. Thus, we had to analyse and find what was most suitable for our goal of developing a dataset in Norwegian. We wanted the dataset to consist of relevant topics to use it in hate speech detection, and thus it was essential to use good search terms and keep up with the current trends in media. As discussed in further detail in Section 8.1 - Section 8.3, it is important to gather data from different sources on various topics, have a clear definition of the labels that are used and focus on the annotation procedure to build a good dataset.

9. Conclusion and future work

9.1. Conclusion

Over the years, detection of hate speech and offensive utterances has become an important field of study because of the challenge with enormous amounts of user-generated data being published online. However, the existing methods are still not performing as good as desired with regards to filtering non-neutral comments. There are several challenges associated with the detection and grading of offensive utterances, such as the lack of a universal definition of hate speech and the issue of censorship. Furthermore, recent studies highlight the issue of using binary classification to detect offensive utterances and suggests trying a multiclass approach instead. The work conducted in this thesis is aiming to investigate methods on how to identify offensive utterances online using multiclass classification and to collect and annotate a Norwegian dataset.

A comprehensive review of literature related to hate speech detection, existing data collections, and various classification methods including multiclass classification has given a detailed overview of which existing methods and models that have been effective when detecting offensive utterances this far. Also, Table 6.12 can be used as a starting point for future research within the field.

We have conducted experiments with both standard models and deep learning models, using an all-in-one approach and a two-step approach. The experimental results showed that using a two-step approach might be advantageous in the case of multiclass classification, which is further confirmed by several state-of-the-art studies. We achieved the best results with the two-step approach using a combination of classic and deep learning models. Overall, the results indicated that an imbalanced dataset with an insufficient number of comments in the more severe categories is a challenge, and might result in the models mostly being able to predict the neutral comments correctly.

Furthermore, we have contributed with a completely annotated Norwegian dataset with more than 41k comments, crawled from various sources over time. To the best of our knowledge, there exists no such dataset today, and thus this is a significant contribution to the field of hate speech detection in Norwegian. We have also presented a definition of hate speech and offensive utterances using a 5-point scale, which can contribute to better annotation of datasets in future research and has been vital for the work conducted in this thesis.

To conclude, the contributions from this work has been an extensive literature review

on the field of hate speech detection, a Norwegian dataset, and experiments done to investigate how multiclass classification can be used to detect hate speech and offensive utterances. To the extent of our knowledge, this has not been done before in Norwegian and can, therefore, be considered valuable contributions to the research field.

9.2. Future work

Throughout working with this thesis, we became especially aware of some improvements that could be applied to our work. Thus, this section presents some suggestions for future research within the field of hate speech detection and improvements for the work done in this project.

Annotation and creation of dataset

We have earlier pointed out the challenges regarding annotation of datasets in Section 8.3. Our dataset consists of 41k comments from three different social media resources, with an imbalanced distribution between the categories. Ideally, one should employ external annotators to the whole dataset to prevent possible bias which might be present in our case because the authors largely annotated the whole dataset. However, using external annotators does not necessarily mean that the annotations are done correctly. In addition, more data could be collected in order to balance the dataset further, since the final dataset only consisted of 7052 non-neutral comments and many of the comments consisted of the same type of hate speech on topics such as immigration and politics.

Spectrum

Furthermore, we annotated the dataset using distinct categories, and it was classified accordingly. A suggestion for future work may be to implement a model that classifies a comment on a spectrum, in terms of its position on a scale between two extreme or opposite points. Thus, not using distinct classes but instead using a smooth transition between the predicted classes. This might be a solution to the problem with distinguishing between classes in multiclass classification, and might also improve performance since a comment classified as category 4 would naturally be seen as closer to a comment in category 5 than category 1.

Context of comments

Next, we suggest that the context of the comment could show to be important to improve the detection of hateful utterances. In general, comments on social media are rather short, and it can be difficult to decide the context. In this thesis, we have distinguished between replies to other comments and comments posted as a reaction to an original post. Thus it can be hard to know precisely what is meant by an utterance when annotating a

comment. For example, the comment "De omringer oss!" is not necessarily hateful, but in the context of the comment "Jeg har en mørkhudet nabo" it is obviously hateful.

When extracting comments from Facebook and Resett, the comments are often related to a specific news article. Thus, extracting this information can be valuable in the future. As an example, the comment "Slike vil vi ikke ha her i landet!" is most likely hateful, but in the context of an article about imported rotten bananas, it is probably neutral.

In addition, different features, such as meta and multimodal information, could prove valuable. Demographic information such as user characteristics and user history have earlier shown to improve the detection rate of hate speech (Pitsilis et al., 2018). Pictures, videos and links are frequently found in social media such as Twitter and Facebook. Context from multimodal information could also provide useful to prevent hate speech in the future as the debate evolves from pure text comments to hateful pictures or videos.

Transfer learning models

In this thesis, an attempt was made to implement Norwegian transfer learning models such as BERT. The experiments were not successful mainly due to the Norwegian models being recently published, and thus it has scarce documentation and were therefore not thoroughly investigated. However, we still believe that these methods have great potential, which recent state of the art within hate speech support. Thus, in the future, we encourage the Norwegian research community to investigate transfer learning models further.

Improved performance

Lastly, some of the standard models performed comparably to the deep learning models, indicating that the effect of using more complex architecture can only be minor. It seems that some of the problems with performance might stem from the annotation of the dataset, which can differ a lot between datasets used in various studies. Further work which looks into linguistic features related to different types of hate speech and the various degrees is needed. We, therefore, suggest that future work focus more on this by creating a good dataset, as this might improve the performance more than just fine-tuning the models.

Bibliography

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

Andreassen Svanes, M., Seim Gunstad, T., & Hilmo Jensen, M. (2019). *Detecting offensive utterances in the Norwegian language.* Department of Computer Science, NTNU - Norwegian University of Science and Technology.

Artstein, R. (2017). Inter-annotator Agreement. In N. Ide & J. Pustejovsky (Eds.), *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.

Balog, K., Takhirov, N., Ramampiaro, H., & Nørvåg, K. (2013). *Multi-step Classification Approaches to Cumulative Citation Recommendation.*

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.*

Berthold, M. R. (2003). Mixed fuzzy rule formation. *International Journal of Approximate Reasoning*, *32*(2-3), 67–84.

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.

Blei, D. M., Ng, A. Y., & Edu, J. (2003). *Latent Dirichlet Allocation Michael I. Jordan.*

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

Bosco, C., Dell'orletta, F., Poletto, F., Sanguinetti, M., & Tesconi, M. (2018). Overview of the EVALITA 2018 Hate Speech Detection Task. In *Ceur workshop proceedings (vol. 2263).*

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. *Computational Linguistics*, *18*, 467–479.

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.

Chatzakou, D., Kourtellis, N., Blackburn, J., De Cristofaro, E., Stringhini, G., & Vakali, A. (2017). Mean Birds: Detecting Aggression and Bullying on Twitter. In *Websci 2017 - proceedings of the 2017 acm web science conference* (pp. 13–22).

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Association for computational linguistics (acl)* (pp. 1–8).

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. In *Proceedings of the 11th international conference on web and social media, icwsm 2017* (pp. 512–515).

De Gibert, O., Perez, N., García-Pablos, A., & Cuadros, M. (2018). Hate Speech Dataset from a White Supremacy Forum. In *Association for computational linguistics (acl)* (pp. 11–20).

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*, *abs/1810.0*.

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*, 29–30.

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

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. In *Ceur workshop proceedings (vol. 2315)*.

Fortuna, P. (2018). A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys*, *51*.

Fortuna, P., Soler-Company, J., & Nunes, S. (2019). Stop PropagHate at SemEval-2019 Tasks 5 and 6: Are abusive language classification results reproducible? In *Association for computational linguistics (acl)* (pp. 745–752).

Founta, A. M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A., & Leontiadis, I. (2019). A Unified Deep Learning Architecture for Abuse Detection. In *Proceedings of the 10th acm conference on web science - websci 19* (pp. 105–114).

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. In *12th international aaai conference on web and social media, icwsm 2018* (pp. 491–500).

Gambäck, B., & Sikdar, U. K. (2017). Using Convolutional Neural Networks to Classify Hate-Speech. In *Association for computational linguistics (acl)* (pp. 85–90).

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*.

Gitari, N. D., 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.

Golbeck, J., Ashktorab, Z., Banjo, R. O., Berlinger, A., Bhagwan, S., Buntain, C., Cheakalos, P., Geller, A. A., Gergory, Q., Kumar Gnanasekaran, R., Gunasekaran, R., Hoffman, K. M., Hottle, J., Jienjitlert, V., Khare, S., Lau, R., Martindale, M. J., Naik, S., Nixon, H. L., Ramachandran, P., Rogers, K. M., Rogers, L., Sarin, S., Shahane, G., Thanki, J., Vengataraman, P., Wan, Z., & Wu, D. M. (2017). A Large Human-Labeled Corpus for Online Harassment Research. In *Websci 2017 - proceedings of the 2017 acm web science conference* (pp. 229–233).

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

Gröndahl, T., Juuti, M., Conti, M., & Asokan, N. (2018). All You Need is "Love": Evading Hate Speech Detection. In *Proceedings of the acm conference on computer and communications security* (pp. 2–12).

Hansen, I. (2015). *HATKRIM Rettslige og praktiske spørsmål*. Oslo.

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. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (pp. 49–66).

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. In *Acl 2018 - 56th annual meeting of the association for computational linguistics, proceedings of the conference (long papers)* (pp. 328–339).

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*.

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

Kumar, R., Ojha, A. K., Malmasi, S., & Zampieri, M. (2018). Benchmarking Aggression Identification in Social Media. *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, (1), 1–11.

Kwok, I., & Wang, Y. (2013). Locate the Hate: Detecting Tweets against Blacks. *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, *360*(17), 3266–3270.

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

Li, H. (2011). A Short Introduction to Learning to Rank - Microsoft Research. *IEICE Transactions on Information and Systems*, *E94-D*(10).

Liu, H., Burnap, P., Alorainy, W., & Williams, M. L. (2019). Fuzzy Multi-task Learning for Hate Speech Type Identification. In *The web conference 2019 - proceedings of the world wide web conference, www 2019* (pp. 3006–3012).

Liu, T.-Y. (2011). *Learning to Rank for Information Retrieval*.

Løfqvist, K., & Odden, C. (2018). *Extending Decision Support for the Norwegian Labour Inspection Authority Through Open and Unstructured Data Sources Methods for detecting relevant information based on external data*.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *Plos One*, *14*(8).

Malmasi, S., & Zampieri, M. (2017). Detecting Hate Speech in Social Media. In *International conference recent advances in natural language processing, ranlp* (pp. 467–472).

Medietilsynet. (2020). *Om sosiale medier og skadelig innhold på nett*.

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

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. In *Advances in neural information processing systems*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2019). Advances in Pre-Training Distributed Word Representations. In *Lrec 2018 - 11th international conference on language resources and evaluation* (pp. 52–55).

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*, 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*.

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. In *Empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Peters, M. E., Neumann, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Naacl hlt 2018 - 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies - proceedings of the conference* (pp. 2227–2237).

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

Poletto, F., Acmos, M. S., Sanguinetti, M., Patti, V., & Bosco, C. (2017). Hate Speech Annotation: Analysis of an Italian Twitter Corpus. In *Ceur workshop proceedings*.

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*, 97–106.

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*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

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

Saleem, H. M., Dillon, K. P., Benesch, S., & Ruths, D. (2017). *A Web of Hate: Tackling Hateful Speech in Online Social Spaces*.

Salminen, J., Veronesi, F., Almerekhi, H., Jung, S., & Jansen, B. J. (2018). Online Hate Interpretation Varies by Country, But More by Individual: A Statistical Analysis Using Crowdsourced Ratings. In *2018 fifth international conference on social networks analysis, management and security (snams)* (pp. 88–94).

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

Sanguinetti, M., Poletto, F., Bosco, C., Patti, V., & Stranisci, M. (2018). An Italian Twitter Corpus of Hate Speech against Immigrants. In *Lrec 2018 - 11th international conference on language resources and evaluation* (pp. 2798–2805).

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

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. In *Proceedings of the 10th international conference on web and social media, icwsm 2016* (pp. 687–690).

Unsvåg, E. F. (2018). *Investigating the Effects of User Features in Hate Speech Detection on Twitter*.

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. In *International conference on human and social analytics (huso 2015)* (pp. 13–18).

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*.

Veledar, A. (2018). *HATEFULLE YTRINGER I OFFENTLIG DEBATT PÅ NETT*.

Vigna, F. D., Cimino, A., Dell'orletta, F., Petrocchi, M., & Tesconi, M. (2017). Hate me, hate me not: Hate speech detection on Facebook. In *Ceur workshop proceedings* (pp. 86–95).

Wang, J.-H., Liu, T.-W., Luo, X., & Wang, L. (2018). An LSTM Approach to Short Text Sentiment Classification with Word Embeddings. *2018 Conference on Computational Linguistics and Speech Processing ROCLING*, 214–223.

Warner, W., & Hirschberg, J. (2012). Detecting Hate Speech on the World Wide Web. *Proceeding LSM '12 Proceedings of the Second Workshop on Language in Social Media*, 19–26.

Waseem, Z. (2016). *Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter*.

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., Meyer, M., Amann, A., Anikina, T., Azoidou, A., Borisenkov, A., Kolmorgen, K., Kröger, I., Schäfer, C., De Smedt, T., Jaki, S., Bachfischer, M., Akujuobi, U., Zhang, X., Risch, J., Krebs, E., Löser, A., Riese, A., Krestel, R., Padilla Montani, J., Schüller, P., Scheffler, T., Haegert, E., Pornavalai, S., Lee Sasse, M., Stammbach, D., Zahraei, A., Stadnikova, P., Klakow, D., Bai, X., Merenda, F., Zaghi, C., Caselli, T., & Nissim, 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. In *26th international world wide web conference, www 2017 (2017)* (pp. 1391–1399).

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. In *Acm international conference proceeding series* (pp. 1980–1984).

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019a). *Predicting the Type and Target of Offensive Posts in Social Media*. Association for Computational Linguistics.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., & Kumar, R. (2019b). *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, Z., Robinson, D., & Tepper, J. (2018). Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics) (2018) 10843 lncs* (pp. 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. In *Ijcai international joint conference on artificial intelligence* (pp. 3952–3958).

Zuleta, L., & Burkal, R. (2017). *HADEFULDE YTRINGER I DEN OFFENTLIGE ONLINE DEBAT*.

A. Additional experimental results

Table A.1.: Result on word embeddings in second step of two-step LSTM.

Word emb.	**Params.** (Model, Dim.size, Win.size, Word count)	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Total F_1-score**	**Macro F_1-score**
None w/o cw	–	0	0.53	0.72	0.61	0.42	0.22
		1	0.31	0.25	0.28		
		2	0.00	0.00	0.00		
		3	0.00	0.00	0.00		
None w cw	–	0	0.58	0.53	0.55	0.41	0.23
		1	0.26	0.28	0.27		
		2	0.09	0.12	0.10		
		3	0.00	0.00	0.00		
fastText w/o cw	SG, 100, 5, 5	0	0.55	0.89	0.68	0.41	0.21
		1	0.30	0.10	0.15		
		2	0.00	0.00	0.00		
		3	0.00	0.00	0.00		
fastText pre-trained	SG, 100, 5, 5	0	0.60	0.66	0.62	0.43	0.23
		1	0.26	0.13	0.18		
		2	0.09	0.12	0.10		
		3	0.02	0.04	0.03		
fastText pre-trained	CBOW, 300, 5, 5	0	0.57	0.53	0.55	0.41	0.23
		1	0.29	0.30	0.29		
		2	0.06	0.08	0.07		
		3	0.00	0.00	0.00		
Word2vec pre-trained	CBOW, 100, 5, 5	0	0.62	0.54	0.58	0.43	0.27
		1	0.29	0.26	0.27		
		2	0.08	0.17	0.11		
		3	0.12	0.09	0.10		
GloVe pre-trained	GV, 100, 15, 5	0	0.62	0.54	0.58	0.43	0.25
		1	0.28	0.23	0.25		
		2	0.11	0.20	0.14		
		3	0.04	0.08	0.05		
Word2vec own trained	CBOW, 100, 5, 1	0	0.60	0.49	0.54	0.42	0.26
		1	0.34	0.33	0.33		
		2	0.10	0.22	0.14		
		3	0.04	0.05	0.04		
Word2vec own trained	CBOW, 100, 5, 2	0	0.62	0.58	0.60	0.43	0.24
		1	0.28	0.23	0.25		
		2	0.07	0.12	0.08		
		3	0.04	0.05	0.04		
Word2vec own trained	CBOW, 300, 5, 2	0	0.57	0.77	0.65	0.43	0.23
		1	0.31	0.11	0.17		
		2	0.10	0.08	0.09		
		3	0.02	0.03	0.02		

Word emb.	**Params.** (Model, Dim.size, Win.size, Word count)	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Total F_1-score**	**AUC**
Word2vec own trained	CBOW, 300, 15, 2	0	0.64	0.62	0.63	0.46	0.26
		1	0.35	0.20	0.26		
		2	0.09	0.23	0.12		
		3	0.03	0.02	0.03		
Word2vec own trained	SG, 100, 5, 1	0	0.60	0.65	0.63	0.44	0.24
		1	0.28	0.20	0.24		
		2	0.04	0.04	0.04		
		3	0.06	0.09	0.07		
Word2vec own trained	SG, 100, 5, 2	0	0.57	0.59	0.58	0.41	0.22
		1	0.27	0.22	0.25		
		2	0.04	0.04	0.04		
		3	0.02	0.02	0.02		
Word2vec own trained	SG, 300, 5, 2	0	0.59	0.66	0.62	0.43	0.25
		1	0.27	0.16	0.20		
		2	0.09	0.08	0.09		
		3	0.06	0.13	0.08		
Word2vec own trained	SG, 300, 15, 2	0	0.58	0.55	0.57	0.42	0.24
		1	0.28	0.30	0.29		
		2	0.05	0.04	0.05		
		3	0.03	0.05	0.04		
Word2vec improved	100, 5, 1	0	0.57	0.56	0.56	0.35	0.20
		1	0.17	0.02	0.04		
		2	0.12	0.15	0.13		
		3	0.03	0.22	0.06		
Word2vec improved	100, 5, 2	0	0.64	0.48	0.55	0.42	0.26
		1	0.29	0.26	0.28		
		2	0.11	0.26	0.15		
		3	0.04	0.10	0.05		
Word2vec improved	300, 5, 2	0	0.60	0.52	0.56	0.43	0.26
		1	0.33	0.35	0.34		
		2	0.10	0.15	0.12		
		3	0.02	0.03	0.02		
Word2vec improved	300, 15, 2	0	0.61	0.64	0.63	0.46	0.25
		1	0.31	0.26	0.28		
		2	0.07	0.09	0.08		
		3	0.00	0.00	0.00		
Fasttext own trained	CBOW, 100, 5, 1	0	0.58	0.72	0.64	0.45	0.25
		1	0.37	0.21	0.27		
		2	0.09	0.06	0.07		
		3	0.02	0.03	0.02		
Fasttext own trained	CBOW,100, 5, 2	0	0.63	0.40	0.49	0.40	0.23
		1	0.29	0.43	0.34		
		2	0.07	0.15	0.10		
		3	0.00	0.00	0.00		

Word emb.	**Params.** (Model, Dim.size, Win.size, Word count)	**Cat.**	**Prec.**	**Rec.**	**F_1-score**	**Total F_1-score**	**Macro F_1-score**
Fasttext own trained	CBOW, 300, 5, 2	0	0.59	0.69	0.64	0.44	0.25
		1	0.29	0.13	0.18		
		2	0.08	0.13	0.10		
		3	0.10	0.10	0.10		
Fasttext own trained	CBOW, 300, 15, 2	0	0.58	0.67	0.62	0.44	0.24
		1	0.28	0.23	0.25		
		2	0.12	0.09	0.10		
		3	0.00	0.00	0.00		
Fasttext own trained	SG, 100, 5, 1	0	0.59	0.61	0.60	0.42	0.24
		1	0.24	0.17	0.20		
		2	0.09	0.13	0.11		
		3	0.05	0.07	0.06		
Fasttext own trained	SG, 100, 5, 2	0	0.62	0.69	0.66	0.46	0.25
		1	0.34	0.18	0.24		
		2	0.10	0.13	0.11		
		3	0.01	0.02	0.01		
Fasttext own trained	SG, 300, 5, 2	0	0.59	0.52	0.55	0.42	0.25
		1	0.30	0.33	0.31		
		2	0.06	0.05	0.06		
		3	0.05	0.14	0.08		
Fasttext own trained	SG, 300, 15, 2	0	0.59	0.46	0.52	0.37	0.22
		1	0.27	0.16	0.20		
		2	0.10	0.32	0.15		
		3	0.00	0.00	0.00		
Fasttext extended	300, 5, 1	0	0.58	0.54	0.56	0.42	0.25
		1	0.32	0.25	0.28		
		2	0.08	0.11	0.09		
		3	0.04	0.13	0.06		
Fasttext extended	300, 5, 2	0	0.60	0.62	0.61	0.45	0.27
		1	0.31	0.24	0.27		
		2	0.10	0.13	0.12		
		3	0.07	0.11	0.09		
Fasttext extended	300, 15, 2	0	0.60	0.59	0.60	0.44	0.24
		1	0.28	0.25	0.26		
		2	0.07	0.07	0.07		
		3	0.03	0.07	0.04		

B. Annotation guidelines

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.

Markering	**Kategori**	**Definisjon**
1	Nøytral	En ytring som har et nøytralt språk og er et saklig bidrag i debatten, uten preg av provoserende, støtende eller hatefulle tendenser.
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. *Greta Thunberg er en BLØFF ! En stor løgn skapt kunstig av sosialister som blir betalt for å lyve !* 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. *@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* 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 ﷺ 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	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 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. Search words

klima
muslim
olje
islam
Thunberg
kjoett
groenn
FrP
bompenger
politiker
Listhaug
skatt
tjukke
innvandring
abort
operasjon
Sophie Elise
blogger
Oslo
dame
ungdom
USA
MDG
AP
rike
parti
dyr
rasist
narkotika
dop
kriminelle
vold
Solberg
lykkelig
Finnmark
misbruk
katastrofe
miljø
utdanning
samfunnet
unge
Trump
NRK
forsker
vind
gass
ulv
rovdyr
streik
NAV
inkludering
aapenhet
IS
minoritet
overgrep
kamp
bistand
barn
norden
menneske
fordom
helse
ytringsfrihet
kvinneandel
gutt
jente
kvinne
mann
menn
utlending
land
radikal
ekstrem
trussel
ideologi
sosial
liberal
hoeyre
AP
kultur
diskriminer
kjerring
hore
stakkars
heks
hjerne
homo
kakerlakk
lesbisk
kjoenn
trans
abort
homse
fitte
feminist
kjerring
billig
pikk
lgbt
tispe
parasitt
hatprat
joede
heterofil
homse
send
utrydde
korrupt
hat
transe
reis
hjem

D. Facebook posts

Facebook pages used

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

Facebook post links

https://www.facebook.com/watch/?v=156751315775152
https://www.facebook.com/tv2nyhetene/posts/10158416836354750
https://www.facebook.com/tv2nyhetene/posts/10158415521994750
https://www.facebook.com/tv2nyhetene/posts/10158415341789750
https://www.facebook.com/tv2nyhetene/posts/10158413582469750
https://www.facebook.com/tv2nyhetene/posts/10158411764709750
https://www.facebook.com/tv2nyhetene/posts/10158408465749750
https://www.facebook.com/tv2nyhetene/posts/10158438979454750
https://www.facebook.com/tv2nyhetene/posts/10158400977669750
https://www.facebook.com/gronnungdom/posts/3183913484959412
https://www.facebook.com/gronnungdom/posts/3039651739385588
https://www.facebook.com/nrknyheter/posts/10157385101366715
https://www.facebook.com/nrknyheter/posts/10157384435571715
https://www.facebook.com/nrknyheter/posts/10157384324951715
https://www.facebook.com/nrknyheter/posts/10157384003561715
https://www.facebook.com/nrknyheter/posts/10157385465546715
https://www.facebook.com/nrknyheter/posts/10157381525091715
https://www.facebook.com/aftenposten/posts/10153362653470516
https://www.facebook.com/aftenposten/posts/10157843766115516
https://www.facebook.com/aftenposten/posts/10157839807565516
https://www.facebook.com/aftenposten/posts/10157830175890516
https://www.facebook.com/aftenposten/posts/10157829312735516
https://www.facebook.com/aftenposten/posts/10157826724625516
https://www.facebook.com/aftenposten/posts/10157823132105516
https://www.facebook.com/aftenposten/posts/10157818551255516
https://www.facebook.com/aftenposten/posts/10157815189600516
https://www.facebook.com/aftenposten/posts/10157814725850516
https://www.facebook.com/aftenposten/posts/10157670277410516
https://www.facebook.com/aftenposten/posts/10157846195085516

https://www.facebook.com/hege.storhaug.5/posts/2861377663882986
https://www.facebook.com/hege.storhaug.5/posts/2874023965951689
https://www.facebook.com/seoghor/posts/10163070482805607
https://www.facebook.com/lanmarieberg/posts/1428948783936871
https://www.facebook.com/lanmarieberg/posts/1453757501455999
https://www.facebook.com/lanmarieberg/posts/1443443719154044
https://www.facebook.com/watch/?v=781335449004214
https://www.facebook.com/venstre.no/posts/10158507546940676
https://www.facebook.com/fremskrittspartiet/posts/10159335377894046
https://www.facebook.com/fremskrittspartiet/posts/10159308116229046
https://www.facebook.com/neitilbomringer/posts/2608058495910429
https://www.facebook.com/neitilbomringer/posts/2602642333118712
https://www.facebook.com/listhaugfrp/posts/1338172996354256
https://www.facebook.com/listhaugfrp/posts/1339771882861034
https://www.facebook.com/permalink.php?story_fbid=1261389840569100id=706880609353362
https://www.facebook.com/Siv.Jensen.FrP/posts/10156857931016127
https://www.facebook.com/Siv.Jensen.FrP/posts/10156867277396127
https://www.facebook.com/norge.fritt.for.islam/posts/2655525601211811
https://www.facebook.com/vgnett/posts/10158302967531995
https://www.facebook.com/vgnett/posts/10158302802226995
https://www.facebook.com/vgnett/posts/10158300323591995
https://www.facebook.com/vgnett/posts/10158299806276995
https://www.facebook.com/vgnett/posts/10158294625211995
https://www.facebook.com/vgnett/posts/10158291473851995
https://www.facebook.com/vgnett/posts/10158290167021995
https://www.facebook.com/vgnett/posts/10158286094421995
https://www.facebook.com/vgnett/posts/10158278427896995
https://www.facebook.com/dagbladet/posts/10158099355253118
https://www.facebook.com/dagbladet/posts/10158098250373118
https://www.facebook.com/dagbladet/posts/10158097963773118
https://www.facebook.com/dagbladet/posts/10158092870023118
https://www.facebook.com/DNavis/posts/10162848903435287
https://www.facebook.com/DNavis/posts/10162845066895287

Marie A. Svanes & Tora S. Gunstad

NTNU
Norwegian University of
Science and Technology