Vilde Roland Arntzen

Haters Gonna Hate

Detecting Norwegian Abusive Language in Social Media with Transformer-based Models

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

Vilde Roland Arntzen

Haters Gonna Hate

Detecting Norwegian Abusive Language in Social Media with Transformer-based Models

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

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

Abstract

Social media and digital platforms have contributed to more people using their freedom of expression than ever. As a consequence, online debates are becoming increasingly polarized, which has an excluding effect on many. Removing abusive content has become crucial for social media platforms and is often assisted using automatic detection algorithms, a field that has multiplied in interest over the past years.

The recent introduction of Transformer-based approaches pre-trained on general language understanding has revolutionized models' ability to understand interdependencies in language. Models based upon the Transformer mechanism have shown promising results for nearly all language tasks so far, including abusive language detection. However, limited research has been conducted in this field for the Norwegian language due to the lack of available datasets and technologies. The recent introduction of Transformer-based models pre-trained on Norwegian corpora makes it possible to explore the successful approach also for Norwegian abusive language detection.

This thesis is focused on detecting Norwegian hateful and offensive social media language using Transformer-based models. Following a literature review, three experiments explore and compare five deep learning architectures based on the Bidirectional Encoder Representations from Transformers (BERT) model. All models are optimized through an extensive hyperparameter search, and several dataset manipulation techniques are tested.

The best model, NB-BERT, significantly outperformed the only existing solutions for offensive and hateful language detection, emphasizing the power of Transformer-based models also for the Norwegian language. However, the results indicate that this model also confused hate speech with offensive language and offensive with neutral language. The ambiguity between the hateful, offensive, and neutral categories originates from the dataset, which generally had a low inter-annotator agreement, a recurring problem of abusive language datasets. The findings further reveal a bias towards classifying Islam-related content as hate speech, which also propagated from the dataset used. Furthermore, an analysis of the model predictions reveals that the model struggled to understand non-standard languages such as typos, grammar mistakes, slang, and dialects, which is different from the corpus on which the model is pre-trained. The results motivate further research on developing better Norwegian abusive language datasets and using models transferring general language understanding to Norwegian abusive language detection.

Sammendrag

Fremveksten av sosiale medier og digitale plattformer har bidratt til at flere bruker ytringsfriheten sin enn tidligere. Dette har medført en økning i polariserte debatter på nett, som har en ekskluderende effekt på mange. Plattformene har derfor sett et økt behov for å moderere og fjerne uønsket innhold, en jobb som ofte gjøres ved bruk av automatiske detekteringsalgoritmer. Forskningen på slike algoritmer har økt betraktelig de siste årene, og flere gode metoder har blitt utviklet.

Introduksjonen av Transformer-baserte metoder har revolusjonert hvordan modeller forstår sammenhenger i språk, og har vist lovende resultater for nesten alle språkoppgaver. Modellene brukes til å overføre kunnskap fra generell språkforståelse til spesifikke oppgaver som å detektere uønsket innhold på nett. Forskningen innenfor dette feltet på norsk har derimot vært svært begrenset på grunn av mangelen på datasett og teknologier inntil nylig. Introduksjonen av flere Transformer-baserte modeller trent på norsk språkforståelse gjør det mulig å utforske effekten av slike modeller også for norsk.

Denne masteroppgaven fokuserer på detektering av krenkende og hatefulle ytringer i norske sosiale medier ved bruk av Transformer-baserte modeller. En litteraturstudie er gjennomført for å forstå forskningsfeltet og bruken av Transformer-baserte metoder, etterfulgt av tre eksperimenter. Eksperimentene tester og sammenligner fem arkitekturer basert på modellen Bidirectional Encoder Representations from Transformer (BERT). Alle modellene optimaliseres gjennom et systematisk søk og flere teknikker for datasett-manipulering er testet for å forsøke å håndtere utfordringer med datasettet.

Resultatene viser at den beste modellen, NB-BERT, signifikant utkonkurrerte de eksisterende metodene for deteksjon av krenkende og hatefulle ytringer, som understreker verdien av Transformer-baserte metoder også for norsk. På tross av det indikerer resultatene at modellen blander krenkende og hatefulle ytringer, og nøytrale og krenkende ytringer. Tvetydignheten for hva som skiller hatefult, krenkende og nøytralt språk stammer fra en lav enighet blant annoterere for hva som skiller de ulike klassene fra hverandre. Funnene indikerer også at modellen overgeneraliserte ved at den ofte detekterte islamrelaterte ytringer som hatefulle, et problem som også har forplantet seg fra datasettet. Videre viser funnene at modellen slet med å forstå ustandardisert språk som skrivefeil, slang og dialekter, som er ulikt fra det den er forhåndstrent på. Funnene motiverer videre arbeid for å utvikle bedre datasett for detektering av hatefulle og krenkende ytringer og bruk av Transformer-baserte modeller for detektering av slikt innhold i norsk språk.

Preface

This Master's thesis is the final submission of a Master's Degree in Computer Science at the Norwegian University of Science and Technology (NTNU), Trondheim. The work is conducted at the Department of Computer Science and supervised by Professor of Language Technology, Björn Gambäck. The thesis mainly concerns abusive language detection for the Norwegian language using the Transformer-based BERT model.

A special thanks go to my supervisor for providing relevant and interesting work and for his generous support throughout the work. Furthermore, I want to thank Per Egil Kummervold at the AI lab of the National Library of Norway for interesting discussions, support, and feedback throughout the project, and for assisting with any questions regarding the NoTraM models and making the NoTraM$_{large}$ model publicly available for use. A thanks go to Einar Næss Jensen for technical support on the IDUN high-performance computer cluster. I also want to thank Mathias Wahl and Sondre Skjåstad for the collaboration on annotations, and Joakim Warholm, Frikk Hald Andersen, Eirik Dahlen, and Karoline Bonnerud for knowledge sharing and engaging discussions on NLP related topics throughout the semester. A thanks also go to Marie Andreassen Svanes for sharing the Norwegian abusive language dataset and for answering any related questions. Finally, I want to grant a special thanks to my sister, Stine Bonnen, and Kristoffer Solberg Rakstad for assisting me with feedback on the thesis and encouraging me during this thesis's writing.

Vilde Roland Arntzen
Trondheim, 23rd June 2021

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network.
API	Application Programming Interface.
BERT	Bidirectional Encoder Representations from Transformers.
CNN	Convolutional Neural Network.
GPU	Graphical Processing Unit.
HPC	High Performance Computing.
mBERT	Multilingual BERT.
NB-BERT	Nasjonalbiblioteket BERT.
NLP	Natural Language Processing.
NLTK	Natural Language Toolkit.
NorBERT	Norwegian BERT.
RNN	Recurrent Neural Network.
Sklearn	Scikit-learn.
SVM	Support Vector Machine.

1 Introduction

The emergence of social media and digital platforms has revolutionized people's ability to connect across social, political, and geographic divides. Previously, access to mass media platforms was mitigated and had to be negotiated, whereas today, anyone can share content and instantly reach millions of people. Social media allow people to use their freedom of expression to participate in public discourse, which bears great opportunities for democratic participation. However, unmonitored online debates are becoming increasingly polarized and contain more abusive content than ever. Abusive content such as hate speech creates fear and may jeopardize people's ability to express themselves in public debates and, in the worst case, lead to violence towards targeted groups (Likestillings- og diskrimineringsombudet, 2018; Eggebø and Stubberud, 2016). This is one of the reasons why social media platforms have increased efforts to monitor user-generated content.

The monitoring of vast amounts of user-generated content requires extensive work and resources, which over the past years have been reduced using automatic detection algorithms. For example, during the period July to December in 2019, Twitter, an American microblogging and social networking platform, took action in 2.9 million cases of content violating their guidelines and more than 970,000 cases that contained hateful conduct (Twitter inc., 2020). Additionally, the COVID-19 pandemic has catalyzed changes in the technology sector, and the importance of automatic detection algorithms has correspondingly increased. Several companies have, for instance, reduced in-office staffing resulting in increased dependence on technology for hate speech detection (Google LLC, 2020). The sheer amount of data that needs to be monitored is so large that exhaustive and manual detection of abusive content is not feasible. For that reason, automatic detection algorithms are crucial in managing abusive content on digital platforms.

Nevertheless, automatic detection of abusive content has proven to be a challenging task. Several social media platforms state that they still rely on humans to review content in certain areas due to the challenge of interpreting nuances in natural human language (Rosen, 2020; Google LLC, 2020). However, the recent introduction of transfer learning and Transformer-based approaches for Natural Language Processing (NLP) has given promising results for abusive language detection in several languages (Zampieri et al., 2020). By considering the context, the more recent models provide better separability of abusive language types, which is one of the main challenges in the field. This thesis will focus on detecting offensive and hateful social media language for Norwegian using Transformer-based models pre-trained on general Norwegian language understanding.

1.1 Background and Motivation

In Norway, freedom of expression is seen as a matter of course. Despite this, there has been an ongoing debate over the past years discussing the trade-off between hate speech and the limits of freedom of expression. Although there is no standard national or international definition of what statements are considered hate, there is a general agreement that hate speech is a societal problem (Likestillings- og diskrimineringsombudet, 2018). Hate speech creates fear in affected groups and has a detrimental effect on the individual. It also weakens democracy as people tend to withdraw from public debates due to hatred contributions. Likestillings- og diskrimineringsombudet (2018) found that more than half in their study of 1,006 Norwegian users of the social media platform Facebook choose to refrain from debating due to the negative and harsh tone. The work against hate speech in general and in social media platforms is, based on this, an essential contribution to equality and democratic participation in society.

Pressure to combat the harmful effects of hate speech has emerged from regulations and shareholders worldwide. Several countries, including Norway, have created laws prohibiting expressions spreading hate towards individuals or groups (Kulturdepartementet, 2019). Digital platforms have, therefore, increased responsibility to monitor their content. Traditionally, the platforms relied on users to detect and report content, which was later reviewed by system moderators. However, manual moderation does not scale proportionally to the rapid increase in user-generated content online. Additionally, disturbing material can cause mental distress in people reviewing it (Roberts, 2019). Automatic detection algorithms are both scalable and can remove harmful content before it reaches the public.

Nevertheless, detecting hateful content has shown to be a nontrivial task for people and algorithms. There is no formal definition of hate speech due to the lack of an objective view of what separates it from other offensive languages (Nobata et al., 2016; Davidson et al., 2017). Pamungkas et al. (2019) stated that the degree of abusiveness of swear words is contextual, and not all expressions containing offensive terms are intended to be abusive. Likewise, there are expressions intended to be hurtful, without any explicit, offensive terms (Caselli et al., 2020). Despite the lack of a standard definition, most hate speech definitions typically concern insults targeting group characteristics such as religion, ethnicity, sexual orientation, gender, or other identity factors (Davidson et al., 2017; Founta et al., 2018; Zampieri et al., 2019b). In contrast, offensive language is more general and often includes profanities and curse words that do not necessarily have a harmful intention. Several studies, including this thesis, categorize hate speech as a subgroup of offensive language, whereas abusive language embraces all language types with an underlying harmful intention (Nobata et al., 2016; Swamy et al., 2019; Zampieri et al., 2019b).

The introduction of Transformer-based models in NLP has revolutionized how automatic detection algorithms learn languages. The transformer builds upon the *attention* mechanism, which enables it to weigh words in a sentence differently, and this way, it represents context to a greater extent than before. Devlin et al. (2019) developed the

Transformer-based model BERT which stands for Bidirectional Encoder Representations from Transformers. The model is pre-trained on general language understanding and transfers this knowledge to particular downstream tasks. BERT has proven successful for several NLP tasks, including abusive language detection in several languages (Zampieri et al., 2019a).

However, until recently, Norwegian has been considered a low resource language, meaning that it is a language with a small number of technologies and datasets relative to its international importance (Cieri et al., 2016). The state-of-the-art models available for languages such as English have therefore not been available for Norwegian until recently. Kummervold et al. (2021) and Kutuzov et al. (2021) newly developed two BERT models trained on general language understanding for the Norwegian language. This thesis investigates how the Norwegian BERT models can detect offensive and hateful language for Norwegian social media data. Additionally, it tests several dataset manipulation techniques to cope with challenges related to the Norwegian abusive language dataset.

1.2 Goals and Research Questions

Goal *Investigate how to accurately detect and distinguish neutral, offensive, and hateful Norwegian language in social media.*

This research aims to improve automatic abusive language detection in social media by separating hateful, offensive, and neutral language for Norwegian. The reason for detecting both offensive and hateful language is to separate the two to preserve freedom of expression in the models at the same time as moderating unintended content. The following research questions define steps toward achieving this goal.

Research question 1 *How well does the Bidirectional Encoder Representations from Transformers (BERT) – pre-trained on a large Norwegian corpus – detect offensive and hateful Norwegian language in social media?*

BERT and other similar Transformer-based approaches have achieved state-of-the-art results in many NLP tasks, including abusive language detection in several languages. The recent introduction of Norwegian BERT models enables testing the model for Norwegian abusive language detection, which will be done in the study of this thesis.

The Norwegian BERT models will be trained to detect offensive and hateful social media language in two separate tasks, to improve the current state-of-the-art results for Norwegian abusive language detection. The models include the multilingual BERT model (mBERT), and the two monolingual Norwegian BERT models: the Nasjonalbiblioteket BERT (NB-BERT) and the Norwegian BERT (NorBERT) model. The mBERT model is trained on more than a hundred different languages (including Norwegian), whereas the others are trained on large Norwegian corpora solely to understand the Norwegian language. The experiments will investigate how well the models distinguish between neutral and offensive Norwegian language and between offensive and hateful Norwegian language. Additionally, the results will be compared to more traditional non-Transformer-based approaches evaluated on the same dataset.

Research question 2 *How do the different BERT models pre-trained on the Norwegian language compare when applied to detecting offensive and hateful social media language?*

The purpose of the second research question is to decide what model to proceed with for the remaining experiments to further improve the detection of offensive and hateful social media language.

Research question 3 *What are the effects of applying different data manipulation techniques to the Norwegian dataset when detecting offensive and hateful social media language?*

In the specialization project (Arntzen, 2020) which served as a preface to this thesis, a lack of Norwegian abusive language datasets was discovered. The dataset by Svanes and Gunstad (2020) and Jensen (2020) is currently the only Norwegian abusive language dataset available. This dataset is highly unbalanced with few abusive instances, which has proven to negatively affect models' ability to detect abusive language (Svanes and Gunstad, 2020; Jensen, 2020).

The third research question aims to test two data manipulation techniques to possibly cope with these challenges. The first technique is undersampling which investigates the effect of removing instances from the majority class to balance the uneven class distribution. The second technique tests the effect of expanding the Norwegian dataset with instances from a Danish abusive language dataset, found to be the most similar dataset regarding language and class labels. The raw Danish data will be tested, additionally to several translated versions, to investigate what format increases the ability of the BERT model to detect abusive language. Based on this, the third research question will also investigate how the BERT model can generalize from the Danish abusive language dataset to the Norwegian.

1.3 Research Method

The methodology used to address the goal of this thesis combines an exploratory and experimental approach. As part of a preliminary project (Arntzen, 2020), secondary research was conducted to gain insight into abusive language detection and address gaps in research to be filled for further progression in the field. This research was based on a structured literature search protocol described in detail in Section 4.1. The search was extended with literature to address this thesis's research questions.

The experiments were carried out with a similar approach to the literature from the secondary research, with the addition of testing recently introduced models that had not yet been evaluated on the task of detecting Norwegian abusive language. The models were assessed on an existing dataset to produce evaluation baselines and verify the new manipulated datasets' integrity resulting from undersampling and aggregating Danish and Norwegian data. The implementation and setup is further described in Section 6.3.

1.4 Contributions

1. A thorough literature review of previous work within abusive language detection and Transformer-based approaches supporting the Norwegian language.
2. A qualitative and quantitative analysis of the Norwegian abusive language dataset by Svanes and Gunstad (2020) and Jensen (2020). A new label scheme is additionally proposed to reduce the ambiguity found in the dataset.
3. An overview of how to implement the Norwegian Transformer-based BERT models to the downstream task of detecting offensive and hateful Norwegian language, with corresponding open-source code.
4. An overview of how to optimize the Transformer-based models with precautions to variance in the fine-tuning process, with corresponding open-source code.
5. A list of optimized hyperparameters for each Norwegian BERT model fine-tuned on the Norwegian abusive language dataset.
6. Experiments with the systems mentioned above on the Norwegian abusive language dataset.

Corresponding code for the above systems can be found in the following GitHub repository: `https://github.com/vildera/abusive_language_detection`.

1.5 Thesis Structure

This thesis contains a total of nine chapters. The following list presents the chapters and their primary purpose.

1. **Introduction:** Gives the reader an introduction to the field of abusive language detection and the problems this thesis aims to address.
2. **Background Theory:** Describes preliminary knowledge and theory needed to understand the methods and results presented in the remaining chapters.
3. **Datasets:** Presents common challenges with abusive language datasets and two relevant Norwegian and Danish datasets used for the task of detecting hate speech and offensive language in the experiments of this thesis.
4. **Related Work:** Presents a literature review on abusive language detection. The approach used for selecting literature is described, followed by related literature in abusive language detection and Transformer-based approaches.

5. **Data Analysis:** Investigates quantitative and qualitative aspects of the datasets used. Based on these analyses, a new representation of the dataset is proposed, which is used in the experiments of the next chapters. A brief content analysis is first presented investigating word occurrences in the classes of the Norwegian and Danish datasets. This is followed by an annotator agreement analysis of a subset of the Norwegian dataset presenting inter-annotator agreement calculations and qualitative observations from the process.

6. **Architecture and Experimental Setup:** Describes a detailed overview of the system architectures of the BERT models and the experimental setup used to perform the experiments of Chapter 7. This includes every step from data and model setup and implementation to the hardware setup used to run the experiments.

7. **Experiments and Results:** Presents the experimental plan designed to achieve the goal and research questions of this thesis, and the results from running these experiments.

8. **Evaluation and Discussion:** Contains an evaluation of the experimental setup and results with regards to the goal and research questions, followed by a discussion of the research questions and goal.

9. **Conclusion and Future Work:** Summarizes the project and the essential findings and contributions, and proposes ideas to motivate future work in the field of Norwegian abusive language detection.

2 Background Theory

This chapter covers the background knowledge required to understand the field of Transformer-based abusive language detection applied to the Norwegian language. First, some fundamentals of the Norwegian language and textual pre-processing will be described, followed by machine learning concepts for text classification, deep learning, and Transformers. This chapter covers the concepts needed to understand the study and can be used to reference concepts and terms that are not assumed to be known by the reader. However, the background theory assumes that the reader is familiar with basic machine learning, linear algebra, and statistics. Section 2.2 is adopted from Section 2.1.1 and certain parts of Section 2.4 is adopted from Section 2.2.3 of the specialization project (Arntzen, 2020).

2.1 Fundamentals of the Norwegian Language

Understanding the fundamentals of the Norwegian language and its peculiarities is necessary to recognize what features an abusive language detection algorithm must interpret to detect abusive content and what challenges may arise during this process related to the Norwegian language.

The Norwegian language is a Germanic language mainly spoken in Norway and is closely related to the two other Scandinavian languages, Danish and Swedish. Additionally to the 26 letters of the English alphabet, Norwegian incorporates the three letters *æ*, *ø* and *å* not supported by standard ASCII encoding. In order to represent Norwegian text including these letters, `UTF-8` encoding is often used, also in the models of this thesis's experiments (Kutuzov et al., 2021; Kummervold et al., 2021).

Norwegian has two formal written standards called *Bokmål* and *Nynorsk*. Bokmål is the main variety used by close to 90% of the Norwegian population, while roughly 10% of the Norwegian population use Nynorsk (Vikør, 2015). The two standards are mutually intelligible and so similar that they may be regarded as written dialects. However, lexically there are some differences between the two, as illustrated in Figure 2.1 which shows an example sentence written in Nynorsk, Bokmål, and English. Although the Bokmål and Nynorsk sentences have many similarities, only two out of nine terms are identical across the sentences. The two varieties differ to some degree in vocabulary and morphosyntactic rules, the internal structure of terms, and syntax. Despite the differences, it has been proven that training a model on both varieties combined yields better results than achieved for models trained on each one of them in isolation (Velldal et al., 2017).

In addition to having two formal written languages, Norwegian also incorporates local

Ein	får	ikkje	noko	tilfredsstillande	fleirbrukshus	med	dei	pengane
Man	får	ikke	noe	tilfredsstillende	flerbrukshus	med	de	pengene
One	*gets*	*not*	*any*	*satisfactory*	*multiuse-house*	*with*	*those*	*money*
PRON	VERB	ADV	DET	ADJ	NOUN	ADP	DET	NOUN

Figure 2.1: Example sentence in Nynorsk (top row) and Bokmål (second row) with corresponding English glossary. The example is reprinted from Velldal et al. (2017) with permission from Erik Velldal.

oral dialects. In informal settings such as online debates or social media - as covered by this thesis - variations of written dialects are present because people write closer to how they speak. As with other types of social media language, such as slang, dialects cause further variations and noise in the dataset. The problem of handling dialects is generally more challenging than handling the standard language varieties. Nynorsk and Bokmål have their own written standards reflected in newspapers, books, and other corpora available for training. As opposed to these varieties, dialects have no standard and are more diverse in their forms.

As mentioned earlier, Norwegian is very similar to Danish and Swedish, and the three languages are all mutually intelligible. Norwegian and Danish are most similar in terms of vocabulary due to Norway being in union with Denmark between 1537 and 1814 (Weidling and Njåstad, 2020). Danish was, in fact, the only written language in Norway from roughly 1500 to 1850, even after the union was dissolved and a new union was entered with Sweden. Danish laid the foundation for Norwegian Bokmål, whereas Norwegian Nynorsk is somewhat closer to many oral dialects. The main difference between Norwegian and Danish today lies in the spelling of and the pronunciation of terms. In comparison, the Swedish vocabulary contains more terms that differ from both the Norwegian and Danish vocabularies.

Both Norwegian, Danish, and Swedish are agglutinative languages partly composed of compound terms, meaning sequences of terms acting as a single semantic construct. A compound's meaning is composed of the meanings of the individual constituents and how they are semantically related. The occurrence of out-of-vocabulary terms that are not found in the dictionary, can be observed and still be perfectly valid. This creates an almost infinite number of term combinations, which may pose challenges for machine learning models, if not broken down into separate pieces present in models' learned vocabulary. An example of a compound term for Norwegian is *Bussjåførstreik* which is composed of three separate terms translated to *bus driver strike* in English.

2.2 Textual Pre-processing

When working with textual data, some pre-processing is often performed before the analysis to convert the text into a predictable and analyzable form. Pre-processing is often performed before extracting features in the data to remove noise or otherwise emphasize discriminative features.

Three common steps for text pre-processing are *tokenization*, *normalization*, and *noise removal*. Tokenization divides the text into smaller chunks called tokens and, in many cases, removes special characters and punctuations. A token consists of a sequence of characters grouped as a useful semantic unit often referred to as terms. Further processing is often done after tokenization. Normalization refers to the task of converting text into common representations, so that text of similar content can be easily analyzed. Stemming and lemmatization are two common normalization techniques. Stemming is the simpler approach that eliminates affixes from a term resulting in the term stem or root, such as the example below.

$$\text{running} \longrightarrow \text{run}$$

For stemming, the term stem does not need to correspond to the dictionary-based morphological root but is rather rule-based. Lemmatization, on the other hand, is more complex and refers to the process of grouping inflected forms of terms together so they can be analyzed as a single item, called the term's *lemma* or dictionary form. Lemmatization is exemplified below.

$$\text{better} \longrightarrow \text{good}$$

Both stemming and lemmatization aim to reduce terms' inflectional and derivationally related forms to standard base forms. However, the two differ in the approach taken. Stemming typically removes the ending characters of a term and hopes to achieve a representative subterm, whereas lemmatization considers the vocabulary and morphological meaning of terms. Using pre-processing techniques such as stemming and lemmatization must be carefully considered, as some of the original information or meaning in the text can be lost in the process. An example of this is performing stemming on the terms *business* and *busy* which may end up with the same stem, *bus*, although the terms have very different meanings.

Other common normalization steps include converting characters to the same case (often lower case), removing numbers or converting them into their textual representation, and removing stopwords. Stopwords contribute little to the overall meaning of a text, given that they are the most common terms used overall in the document corpus. Stopwords often include terms such as *the* and *and* but may also depend on the domain used. For instance, *not*, which in some contexts is considered a stopword, may be necessary for particular domains, such as in sentiment analysis where the phrase *not good* will change the meaning if the term *not* is removed. Although some stopwords contribute to little meaning in itself, it is worth mentioning that it contributes to binding the text together and understanding the context of the language.

Finally, noise removal of the data refers to removing characters, digits, and pieces of text that interfere with the analysis. As opposed to tokenization and normalization, noise removal is highly domain-dependent. For social media platforms, noise removal often include removing whitespaces and converting URLs to a common term. For other web pages, it may also include the removal of noise such as HTML tags. Other characteristics with social media data to consider when pre-processing include users repeating characters to express feelings. A common way to handle such terms is to convert the term into its common form or to reduce the number of repeated characters to two (Brody and Diakopoulos, 2011). An example of this is given below.

$$\text{niiiiiiiiiiice} \longrightarrow \text{nice}$$

As social media text often holds a different format than many other text documents, it is essential to handle pre-processing with care to avoid losing important information from the text. For instance, hashtags and user tags have another meaning in social media posts and might be helpful for feature extraction. In some cases, noise such as misspellings of terms can also say something about the user writing it and may also be considered a basis for feature extraction. Therefore, the steps taken to pre-process the text should always be considered concerning the problem's domain.

2.3 Machine Learning for Text Classification

The task of detecting abusive language is often solved using machine learning. Machine learning is a field within artificial intelligence concerning the construction of computer algorithms that automatically improve with experience (Mitchell, 1997). The goal of machine learning algorithms is to generalize patterns in the data to predict future, unseen data samples. For text classification, the instance to be predicted is usually referred to as a document representing several types of texts, such as books, social media comments, or shorter sentences. Machine learning algorithms are often divided into supervised and unsupervised learning algorithms, although other approaches do exist. The primary focus of this thesis will be on supervised algorithms, as most research on abusive language detection is performed using supervised classification approaches. The following sections describe some central topics in machine learning for text classification, ranging from data handling and feature representations to the Support Vector Machine and evaluation measures. Section 2.3.6 is adopted with some modifications from Section 2.1.1 of the specialization project (Arntzen, 2020).

2.3.1 Supervised Classification

Supervised machine learning algorithms learn a function that maps input data to an output label based on pre-labeled training examples. For classification tasks, this label is categorical so that samples belong to a category, not continuous quantities. For instance, if the task is to predict whether a document contains hate speech or not, the supervised classification algorithm requires that each document in the dataset is labeled as hate or not hate speech. The model is trained to learn hate speech detection based on this data but needs a different set to be evaluated. Therefore, the available data is usually divided into three sets: a training set, a development set, and a test set. The development set is often also called the validation set.

The training set is usually the largest and contains the samples that are used to train the model. The training set size can vary depending on the specific task but is typically

70-90% of the available labeled data. The development set is mainly used to optimize model hyperparameters, which define the parameters whose value is used to control the learning process when developing the model, hence development. This set is used to evaluate the model during training. Finally, the test dataset is used to evaluate the final model and is typically represented by 10-30% of the available samples. The test dataset is completely unseen until final model evaluation to simulate how the model performs on new, unseen data. The three datasets should roughly have a similar probability distribution, as it makes it more likely for the model to fit both the training and test data well. A better fitting to the training dataset than the test dataset usually points to overfitting, meaning that the model is too customized to the training data and does not generalize well to the test data.

2.3.2 Inter-annotator Agreement: Fleiss' Kappa

When creating a dataset for classification, it is common to use several annotators to decide what labels each instance should have. A way to assess the quality of the annotations is to calculate the inter-annotation agreement between annotators. A high inter-annotator agreement means that there is consistency in what separates the classes across the dataset. Fleiss' kappa, κ, is a statistical measure to assess the inter-annotator agreement between a fixed number of annotators when assigning categorical labels to a number of instances. Artstein (2017) defines Fleiss' kappa as follows.

Let c be the number of annotators, and n_{ik} the number of annotators annotating instance i with class label k. For each instance i with label k there are $\binom{n_{ik}}{2}$ pairs of annotators who agree on label k. Summing over all labels, there are $\sum_k \binom{n_{ik}}{2}$ pairs who agree on the label, for instance, i, and the agreement on instance i is the number of agreeing pairs divided by the total number of annotator pairs. The overall observed agreement is then defined as the mean agreement per instance, that is, the sum of the observed agreement for each instance i divided by the number of items i.

$$\kappa = \frac{1}{ic(c-1)} \sum_i \sum_k n_{ik}(n_{ik} - 1)$$

The values of κ range between -1 and 1, where 1 means perfect agreement and $\kappa \leq 0$ means no agreement between the annotators other than what is expected by chance. Despite the lack of formal guidelines on how to interpret κ, Landis and Koch (1977) suggested the classifications displayed in Table 2.1.

2.3.3 Stratified Sampling

Machine learning algorithms often require data to be divided into subsets used for training, development, and testing, of which the *Stratified sampling* technique is often used. Stratified sampling is a sampling technique where the sampled instances are selected in the same proportion as they appear in the population (Särndal et al., 2003). This is achieved by dividing the population into strata, subpopulations, based on a common characteristic. For classification, this characteristic is often the class distribution, and

Value of κ	Strength of Agreement
$\kappa < 0.20$	Poor
$0.21 - 0.40$	Fair
$0.41 - 0.60$	Moderate
$0.61 - 0.68$	Good
$0.81 - 1.00$	Very good

Table 2.1: Suggested classifications of Fleiss' kappa by Landis and Koch (1977).

the stratified sampling aims at splitting the dataset such that each split is similar to this distribution. The instances are sampled using random sampling from each of the subgroups.

Consider the classification task of classifying emails into spam and non-spam emails, whereas only 20% of the emails are spam and the remaining 80% are non-spam. If we want to use stratified sampling to sample a subset of the data with equal distributions, the dataset would be divided into two subgroups of spam emails and non-spam emails. The resulting training sample is created by sampling 80% of the instances from the non-spam and 20% from the spam emails, using random sampling. Figure 2.3.3 illustrates the example, whereas the dark gray instances correspond to spam emails and the white instances correspond to non-spam emails. One can observe from the figure that the resulting sample also has a distribution of 20% spam emails and 80% non-spam emails.

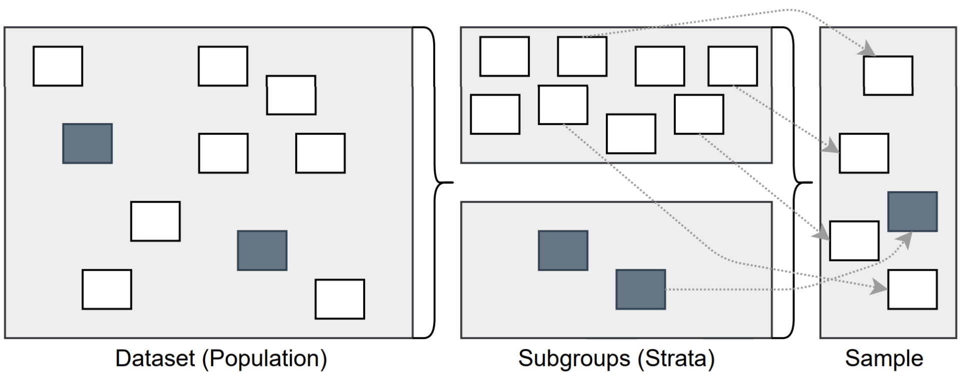


Figure 2.2: Illustration of stratified sampling using strata to sample a subset with equal class distribution to the full dataset. The dark gray and white instances illustrate instances belonging to two different classes, and the arrows illustrate one combination of random samples.

2.3.4 Features and Feature Space

For a classification model to learn separable patterns in the data, individual measurable properties or characteristics, also called *features*, are often extracted. Features can represent different aspects of a task, but informative and discriminative features are preferred to help the classifier to separate between instances of different classes. Consider a method that classifies price ranges for housing. Features that can be used to separate houses in different price ranges can, for instance, include house size in square meters and if there is a pool or not.

Most algorithms require a numerical representation of inputs as these facilitate statistical analysis. Therefore, features are usually represented numerically as n-dimensional vectors called *feature vectors* as shown below.

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Each position in the vector will usually represent a different feature, with n being the total number of features. Consider the example with housing price ranges, a house of 200 square meters and without a pool could be represented as the following feature vector.

$$x = \begin{bmatrix} 200 \\ 0 \end{bmatrix}$$

For textual documents, features can represent properties such as term occurrences or term presence. The collection of features represented by the n-dimensional feature vector spans an n-dimensional vector space called the *feature space.* This vector space represents the range of possible values for a chosen set of features in the selected data.

2.3.5 Word Embeddings

Word embeddings are forms of feature representations representing terms as dense numerical vectors mapped from a multi-dimensional space to a lower dimension. The values (or features) of the vector (*embedding*) capture information about the term's syntax and semantic meaning. This builds upon the assumption that terms of similar meanings tend to occur in a similar context. For instance, the features of a word embedding vector may be the underlying features of a term, i.e., terms occurring in similar contexts. Consider the two vector representations as follows.

$$\text{Father} = [\text{parent} : 1, \text{man} : 1, \text{woman} : 0]$$

$$\text{Mother} = [\text{parent} : 1, \text{man} : 0, \text{woman} : 1]$$

Mathematical operators can be used on the word embeddings to illustrate one application of such representations.

$$\text{Father} - \text{Man} + \text{Woman} = \text{Mother}$$

Traditionally, terms were represented as numbers often based on ordering and term frequency. A limitation of such approaches, along with the inability to capture context, is the increasing representation size as the vocabulary increases. This results in a sparse representation with vast amounts of zeros requiring more memory and computational resources, a problem commonly known as *the curse of dimensionality.* Word embeddings were initially founded by Bengio et al. (2003) who trained a neural language model on the distributed representation of terms to overcome this problem. However, the concept was first brought to the fore by Mikolov et al. (2013) who created the *word2vec* model, a toolkit enabling training and use of pre-trained embeddings.

Pennington et al. (2014) later introduced the competitive method *GloVe.* Word2vec and GloVe are fundamentally similar in that they both allow dense and expressive representations of terms based on contextual similarity. However, there are some differences. Word2vec embeddings are based on training a shallow feed-forward neural network capturing local term-to-term co-occurrence statistics. That is, the semantics learned for a given term are only affected by the surrounding terms. In contrast, GloVe embeddings are learned based on matrix factorization techniques that also capture global term-to-term co-occurrence statistics leveraging the entire corpus. The latter can be computationally expensive when trained on a large corpus but only needs to be trained once.

Word embeddings have played a critical role in the realization of *transfer learning*, the concept of transferring knowledge from one task used to solve a different task. For instance, word embeddings can be trained on general language understanding corpora and further used to represent documents in downstream tasks such as abusive language detection.

2.3.6 Support Vector Machine

A Support Vector Machine (SVM) is a supervised learning model that can be used for classification tasks. Given a set of training instances with corresponding labels, an SVM aims to build a line, hyperplane, or set of hyperplanes that separate the data into distinct classes. The idea is to find the "extreme" values in both classes and create a hyperplane that best explains the class boundary. The training instances are represented with coordinates in an n-dimensional space, in which n represents the number of features. The dimension of the hyperplane is given by $n - 1$ to separate the features, as visualized in Figure 2.3. The values closest to the hyperplane are called support vectors, and their distance to the hyperplane is called the margin. The model aims to maximize the average distance from the support vectors to the hyperplane to separate the classes. A *kernel* function is used to maximize the distance from the support vectors to the hyperplane. The kernel function is a mathematical function that calculates the similarity of two vectors. For linear SVMs, the kernel function corresponds to the vectors' inner product, which returns a scalar representing the similarity between the two vectors. Kernel functions can also hold different shapes, for instance, non-linear or polynomial, which maps the feature space into spaces of higher dimensions.

Suppose the data cannot be linearly separated with a hyperplane as in Figure 2.3. In that case, the data points are mapped into a higher dimension using a kernel function that transforms lower-dimensional spaces into spaces of higher dimensions. The idea is that the data will continue to be mapped into higher dimensions until it is linearly separable. SVMs are effective in high dimensional spaces in a memory-efficient way and have proven successful in many classification tasks.

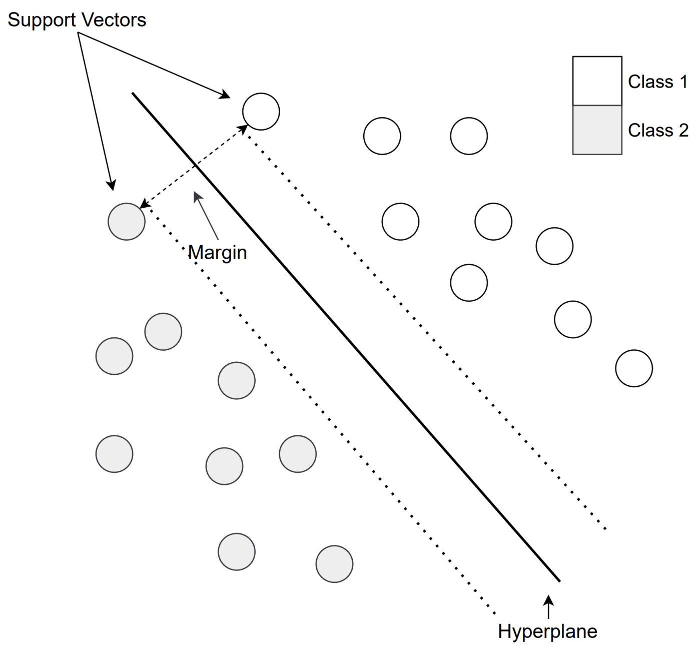


Figure 2.3: Visualization of an SMV hyperplane separatig datasets from two classes

2.3.7 Model Evaluation

Evaluation measures are used to evaluate how well machine learning models perform on a particular task and are an essential step toward selecting the best model and reporting its results. Several evaluation measures can be used to evaluate and assess the performance of classification models. The evaluation measures used to evaluate classification performance in this thesis includes *accuracy*, *precision*, *recall*, and F_1 *score* which will be described subsequently in this section. The descriptions of the measures are adopted from Section 2.3 of the specialization project (Arntzen, 2020) with some adjustments. At the end of this section, the *k-fold cross-validation* technique used in combination with these measures to evaluate models will be described.

The formulas are based on the following terms, with reference to Table 2.2. *True positives* (*tp*) is the number of correctly classified positive instances, *true negatives* (*tn*) is the number of correctly classified negative instances, *false positives* (*fp*) is the number of incorrectly classified positive instances, and *false negatives* (*fn*) is the number of incorrectly classified negative instances.

		Predicted Label	
		Negative	Positive
True Label	Negative	tn	fp
	Positive	fn	tp

Table 2.2: A confusion matrix representing the possible predicted values for a binary classification problem.

When calculating precision, recall and, f1 score, there are two commonly used calculation methods, namely micro- and macro-averaging. Micro-averaging is calculated by aggregating all class contributions and then averaging, while macro-averaging computes the metric independently for every class and then averages. As opposed to micro-average, macro-average penalizes poor performance in the minority class for unbalanced datasets. This is because the performance of the minority class will receive the same weighting as the majority class. Therefore, macro-average is suitable when the minority class of an unbalanced dataset is of high interest, such as for abusive language detection for datasets following a natural unbalanced distribution.

Accuracy

Accuracy is a simple and intuitive metric that quantifies correctly classified instances among the total instances. The resulting value lies between 0 and 1, whereas 1 means that all instances are correctly classified. The formula is given as follows.

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn}$$

Although accuracy is a simple and effective metric to measure overall model performance, it can be misleading for imbalanced datasets. With an imbalanced dataset, a model classifying all instances into the majority class can achieve high accuracy. For instance, if 95% of the dataset belongs to the majority class, classifying all instances into this class will achieve an almost perfect accuracy score of 0.95. However, if the goal is to predict the minority class, the model is far from reaching the goal. Therefore, accuracy is often used in combination with other measures to assess model performance.

Precision and Recall

Precision and recall are two highly correlated measures used in machine learning to evaluate classification models. The two metrics are defined as follows.

Precision quantifies the proportion of correct positive classifications or the proportion of relevant instances among the total retrieved instances. Precision is a good metric to measure a model's ability to label instances correctly. The formula is given as:

$$Precision = \frac{tp}{tp + fp}$$

On the other hand, recall quantifies the proportion of correct positive predictions made out of all positive predictions that could have been made. That is the fraction of relevant predicted instances among all relevant instances. Recall is good at measuring the ability of models to find relevant samples. The formula is given as:

$$Recall = \frac{tp}{tp + fn}$$

Precision and recall are usually compared at a fixed level or combined into a single measure, as demonstrated in the next section.

F_1 Score

The F1 score is the harmonic mean of the precision and recall values and conveys the trade-off between them. The value of the F1 score ranges between 0 and 1, whereas 1 is the best possible value and corresponds to both precision and recall being equal to 1. F1 score tends to be closer to the lower of precision and recall, and its formula is given as:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

To demonstrate the usefulness of combining the precision and recall metrics, consider a search engine that aims to collect relevant documents. In this case, the positives will correspond to relevant documents, whereas the negatives will correspond to irrelevant documents. Recall can then be maximized by collecting all documents available. However, a large portion of the collected documents may be irrelevant. In order to handle this problem, precision can be used to measure the proportion of retrieved documents that are relevant. Using macro F1 score in combination with accuracy will enable measuring the model's performance across all classes also for cases with unbalanced datasets.

Stratified K-Fold Cross-validation

K-fold cross-validation is a model validation technique that assesses how the results of a model generalize to independent datasets. As described in Section 2.3.1 a model is often provided a training and a test dataset, whereas only the test dataset is used to evaluate model performance. K-fold cross-validation aims to test how the model predicts unseen new data to reduce problems such as overfitting or selection bias. A model suffers from overfitting to a dataset when its production is too close or the same as a particular dataset (usually the training set). Therefore, it fails to predict new unseen instances reliably. On the other hand, selection bias is the bias introduced by the selection of instances so that the dataset obtained is not representative of the data intended to predict.

The cross-validation technique is based on splitting the dataset into k groups, where each instance in the dataset is assigned to only one group. It is common to use the stratified sampling technique previously described in Section 2.3.7 to ensure equal distribution for each group. This ensures that the cross-validation result is a closer approximation of the generalized value.

For each unique fold, as displayed in Figure 2.4, one group is held out and used for validation, whereas the k-1 remaining groups are used for training. The model is fitted to the training data and evaluated on the validation data for each fold. In the end, the average scores are reported, and the model has summarized been evaluated on the complete dataset.

A common way to use cross-validation is to combine the training and development dataset for k-fold cross-validation during model selection and selecting the best model from the averaged result. Finally, the model's results are reported on the test set, which was never seen during cross-validation. The reason for not including the test set during cross-validation is that it would result in a biased score. The test dataset is held out for final evaluation such that the best model from cross-validation can be evaluated on an unseen dataset. Cross-validation score, therefore, refers to the score averaged over each fold. The technique is displayed in Figure 2.4 with k equal to five.

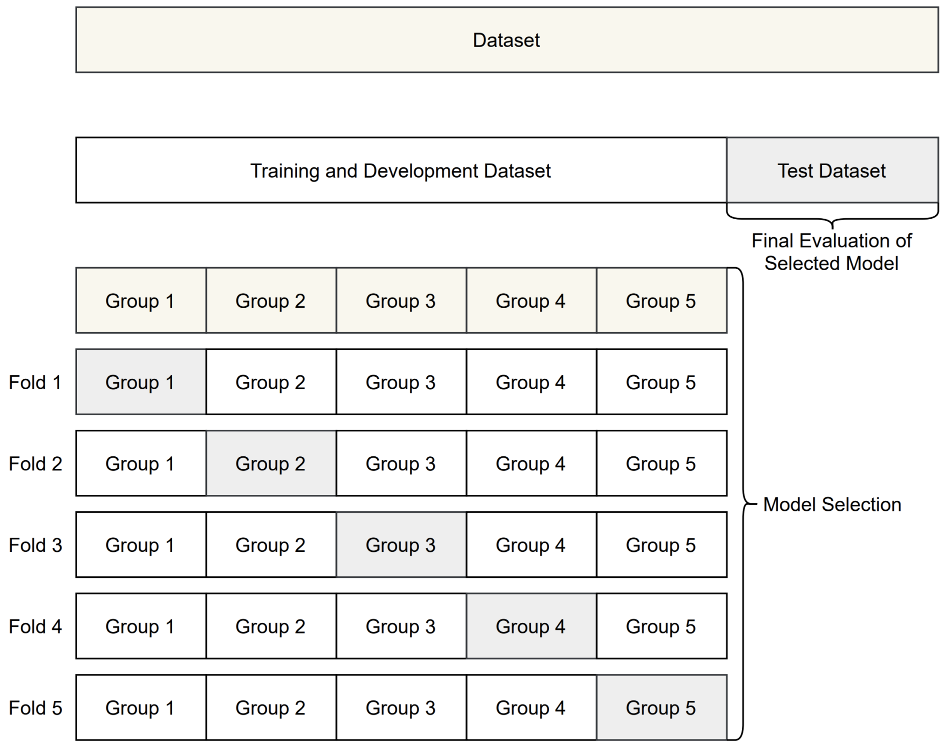


Figure 2.4: Illustrating five-fold cross-validation where the training dataset is divided into five groups and the model is evaluated on one group and trained on the other four groups for each fold.

When using the k-fold cross-validation technique, a crucial step is to select a good k-value to represent the model's skills to predict instances correctly. There is a trade-off between bias and variance when choosing this value. A larger k-value increases variance, whereas the bias of the technique decreases when k is smaller (Kuhn and Johnson, 2013).

The value of k is chosen such that each of the k groups is statistically representative of the dataset used. Although there is no standard rule when selecting k, a rule of thumb is that k's value often ranges between 5 and 10 depending on the dataset used (James et al., 2013).

2.4 Deep Learning

Deep learning is a subcategory of machine learning based on Artificial Neural Networks, which uses multiple layers to extract higher-level features from raw input data, such as text. The input data propagates through a weighted network of neurons, called perceptrons or nodes, and automatically learns new data representations. Deep learning is an essential part of the Transformer-based models used in this thesis, both for the attention mechanism they build upon and the classification layer used to classify the input documents. This section covers the basic theory behind deep learning and the attention mechanism, followed by the advanced Transformer-based architectures used to predict abusive language in this thesis. Sections 2.4.1, 2.4.6, and 2.4.7 are built upon Section 2.2.3 of the specialization project (Arntzen, 2020).

2.4.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) vary in shape and structure depending on the task to be solved. Common for all of them is that they seek to approximate the unknown function represented by the input data. ANNs are inspired by how information is processed in biological systems and the concept of communication neurons at an abstract level. A neural network is therefore composed of neurons and the connections between them. Increasing the number of neurons or connections increases the network parameters and provides a more flexible system that can learn more complex patterns, with the trade-off of requiring more computational effort.

The simplest form of an ANN is a Feed-forward Neural Network (FNN). As the name says, the network's information only moves in one direction, forward, from the input layer through the hidden layers (if any) to the output layer. A commonly used feed-forward neural network is the *Multilayer Perceptron* which will be described in the next section, followed by a section describing the training process of such networks. Subsequently, the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) models will be described.

Multilayer Perceptrons

A *Multilayer Perceptron* consists of an input and output layer with one or more hidden layers in between (Haykin, 2009). The input and output layers work as network gates to the outside world. In contrast, the hidden layers perform some non-linear functional transformation to the input data, extracting higher-level features. The hidden layers consist of hidden neurons, whereas the *hidden* term refers to the fact that the neurons are not accessible from the external world. Figure 2.5 illustrates a multilayer perceptron with one hidden layer. The network illustrated is *fully connected* which means that for each layer, every neuron in the network is connected to every neuron in the subsequent layer.

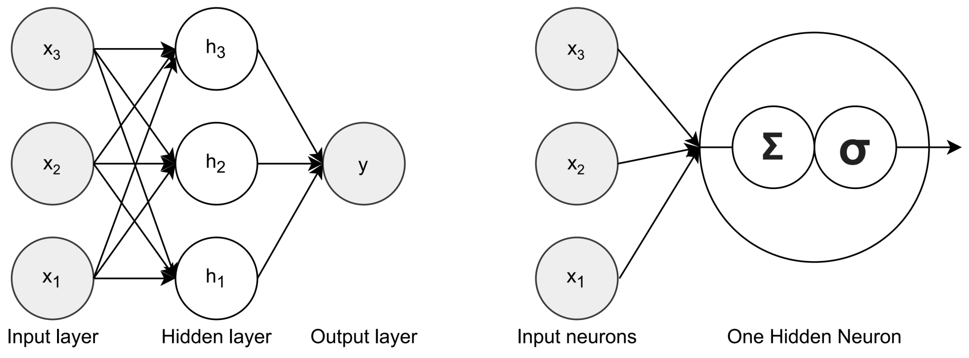


Figure 2.5: The left figure illustrates a feed-forward neural network with one hidden layer. The figure at the right illustrates the non-linear transformation of the input to a hidden neuron.

The process starts with passing the input vector (represented as described in Section 2.3.4) to the network, where each value of the vector enters a different input neuron. The input values are then passed through the network of hidden layers, where each input is multiplied by the output weight of that layer. The weighted outputs are passed as inputs to the next layer's neurons which repeats the process. The hidden neurons function as interventions between the input and output layers, where each output v_j of neuron j can be formalized as follows.

$$v_j = \sigma(\sum_{j'} w_{jj'} \cdot v_{j'})$$

Where σ is the non-linear transformation function, also called the *activation function*, and $w_{jj'}$ is the weight from neuron j' to neuron j (Lipton, 2015). The same process is performed for each hidden neuron in the network.

The main goal of a neural network used for classification is to predict the label of a given input instance. This is done by approximating a *target function* $f^*(x) = y$, where x is the input vector and y is the predicted label. f^* denotes the chained function $f^k(f^{k-1}(...(f^1(x))...)$, where k is the depth of the network corresponding to the number of layers. That means that each layer i has its own applied function as defined below.

$$f^i(x; W, b) = W^T x + b$$

Where x is the input vector of the previous layer, W^T is the transpose of the weights, and b is the linear bias. The bias is an additional parameter in the network used to adjust the output and the weighted sum of the inputs to the neurons. As with an intercept in a linear equation, the bias allows shifting the output value. For the transformation to be non-linear, one or more non-linear activation functions are applied to the chain's output which transforms the input to a defined output, inspired by how information is processed in biological neurons of the human brain. This enables the network to learn non-linear mappings of the data.

An activation function used in most of the Transformer-based models (described later in this chapter) is the Gaussian Error Linear Unit (GELU), which was defined by Hendrycks and Gimpel (2016) as follows.

$$\mathrm{GELU}(x) = 0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)))$$

GELU is a combination of functions (e.g., the hyperbolic tangent) and approximated numbers. When x is greater than zero, the output will be x, except when x ranges between 0 and 1, the transformation is slightly smaller. That is, inputs have a greater probability of being dropped as x decreases, such that the transformation is stochastic yet depends upon the input (Hendrycks and Gimpel, 2016).

Back Propagation and Batch Learning

A commonly used approach to train feed-forward neural networks is *back propagation* which proceeds in two phases; the forward and backward phase (Haykin, 2009). In the forward phase, the network's weights are fixed, and the input propagates through each layer of the network, confining the changes to the transformations performed by the network's neurons. A loss or error is calculated in the backward phase by comparing the output with the desired value. The loss is then propagated backward through the network's layers, adjusting the neuron weights. The weights are adjusted proportionally to the neuron's contribution to the loss. This is done using an optimization method such as gradient descent (Lipton, 2015) that aims to minimize a calculated loss function. This will be further described in Section 2.4.5.

The training process of the network is commonly performed using *batch learning* which adjusts the network weights after all instances in the batch are passed through the network (Haykin, 2009). The loss function that is minimized calculates the average loss contributions of all neurons after each batch. The training dataset can be divided into one or more batches, whereas the passing of all batches through the network constitutes one *epoch* of training. Training in batches reduces the computational effort required, however, often at the cost of a less accurate estimation of the loss when optimizing. This is due to less data being available at a time to generalize from. As the optimization process is iterative, it is common to perform several training epochs to minimize the loss. It is important to note that performing too many training epochs may lead to overfitting the model, meaning it does not generalize well to new data.

There is no formal approach to selecting the values for batch size or the number of epochs, which is dependent on how diverse the data is. However, some numbers have performed empirically better than others across several tasks for certain model architectures (Devlin et al., 2019).

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) (Rumelhart et al., 1988) is another class of ANNs derived from feed-forward neural networks but differs in the fact that they allow a type of memory storage. This lets the network exhibit temporal dynamic behavior using an internal state to process variable-length sequences. The storage mechanism is called *feedback* and lets the output of a neuron influence the input applied to the particular neuron, producing closed paths for information transmission (Haykin, 2009). There is no formal structure of RNNs, as essentially every neural network with at least one feedback loop allowing memory storage can be considered an RNN. An example of an RNN is a standard multilayer perceptron, except for adding at least one feedback loop.

As RNN's future outputs are dependent on past and previous decisions, time is taken into context, ranging from 1 to τ. The input sequences of RNNs contain vectors at different time steps, denoted as $x^{(t)}$. The information transmission can be expressed in the following recursive manner.

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

Where $h^{(k)}$ is the hidden state vector at time k, and θ is the model parameters. Each step of this procedure handles the computation and calculates the output and the loss, which depends on the particular network structure. The output can be used to produce a final classification of the sequenced input, and the network can be trained using an adjusted version of backpropagation, taking time into account. This is done by unrolling the network creating copies of the feedback connections. For instance, a single neuron x_1 with a connection to itself $x_1 \to x_1$ could be represented as two neurons $x_1 \to x_2$ making the graph acyclic and this way similar to a feed-forward neural network graph.

Although RNNs can handle sequenced inputs, their memory capacity is insufficient for handling long-term dependencies, requiring storing more than the previous state in memory. In the case of textual data, long-term dependencies include longer sentences and paragraphs.

Convolutional Neural Networks

An alternative to the RNNs is Convolutional Neural Networks (CNNs) (LeCun, 1989), which are regularized versions of Multilayer Perceptrons. CNNs take advantage of hierarchical patterns in the data such that more complex patterns can be learned using smaller and simpler patterns. The pattern between the nodes is inspired by the visual cortex of the brain. This works so that individual receptive fields of neurons overlap partially and combined cover the whole visual field. CNNs consist of an input layer, multiple hidden layers, and an output layer. The hidden layer typically incorporates a series of convolutional and pooling layers, fully connected to an ANN.

Convolution is the application of a filter to the input data, which results in activation. When a convolution is repeated, the input results in a map of activations indicating the locations and strengths of the input features. The convolutional layers perform the convolutions, linear transformations, of the input. As CNNs originally were developed for two-dimensional input, this transformation involves a multiplication between an array of input data and a two-dimensional array of weights, called the filter. The same filter is systematically applied across the input to detect specific features. Once the feature map is created, each value is passed to an activation function, much like what is done for a fully connected layer.

The pooling layers reduce the dimensions of the data by combining the outputs of neuron clusters to a single neuron in the next level. *Max pooling* uses the maximum value from the cluster of neurons at the prior layer, while *average pooling* uses the average value. In addition to reducing the dimensions of the data, this works as a form of regularization that helps prevent overfitting. The layers capture more local information, such as terms of a sentence or pixels of an image.

2.4.2 The Softmax and Argmax Functions

The produced output of the neural network is called logits and holds the prediction score of the model. A softmax function is often applied to the logits mapping each value into a probability between 0 and 1, normalizing the values to a more interpretable format. This is often performed during training before feeding the output values of a neural network to a loss function that is being optimized. The softmax function is defined as follows.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}}$$

Where z is a K-dimensional input vector, the resulting output is a value in the interval (0,1) which is normalized such that the summation of the output values is equal to 1. When the model is used for inference, an argmax layer typically replaces the softmax layer or is added on top to produce a discrete class label instead of a probability. The argmax function returns the arguments for the target function that returns the maximum value from the target function. Consider the function $g(x)$ where $g(1) = 1^3 = 1$ and $g(2) = 2^3 = 8$ and x is limited to the value 1 or 2. In this case, the argmax of g(x) is equal to 2. Note that the argmax is not the maximum value returned which would be 8, and it is also not the maximum value of the arguments, although in this case, they are the same. The armax function returns 2 because 2 is the function's argument when 8 is the produced output.

2.4.3 Regularization with Dropout Layers

Regularization techniques can be applied to machine learning classifiers to prevent model overfitting. A commonly used technique for regularizing neural networks is dropout layers which refer to the ignoring of certain neurons (along with their connections) in the network during training (Srivastava et al., 2014). In practice, this means that the dropped neurons are not considered in a backward or forward process, preventing them from co-adapting too much. The choice of what neurons to drop at each iteration is randomly selected with a probability p, also called the dropout probability. Applying dropout layers to a neural network has improved neural network performance in various application domains, including text classification tasks.

2.4.4 Cross-entropy Loss

When working with classification models, the loss function is used to optimize the model during training and estimates how far an estimated value is from the true value. Cross-entropy loss is a commonly used loss function for both binary and multiclass classification. Given two probability distributions, p, the true distribution, and q, the estimated distribution, the cross-entropy loss, $L(p, q)$ for a binary classification problem is given as follows.

$$L(p, q) = -p \log_2(q) + (1 - p) \log_2(1 - q)$$

The same formula is used for multiclass classification problems by summing each of the separate classes together. The cross-entropy loss increases as the predicted probability diverge from the true label, where predictions that are confident and wrong are penalized stronger. If all labels are correctly classified, the cross-entropy loss is equal to 0.

The function was initially designed to measure the loss of models with outputs ranging from 0 to 1, also called probability values. However, the produced output values holding the prediction scores of a neural network, logits, ranges below 0 and beyond 1. Although these values are not normalized, they still represent a probability distribution and can therefore be interpreted as probabilities. Because of this, the formula for cross-entropy loss can be used not only on normalized values but also directly on logits.

2.4.5 Optimizers and Learning Rate

As previously described in Section 2.4.1, neural networks are trained using an optimization method such as gradient descent, minimizing some empirical loss function L on the training data. The gradient descent algorithm depends on the first-order derivative of an objective loss function, also called the gradient, and calculates how to adjust the network weights to minimize this function (Haykin, 2009). The gradient is calculated at each training step for each weight and is altered in the opposite direction of the current gradient to approximate the minima. The gradient of the function f is denoted as ∇f and will point to the direction where the function increases most rapidly. Based on this, the new weights θ' can be calculated using the current weights θ as follows.

$$\theta' = \theta - \eta \nabla L$$

Where η is a small constant called the learning rate, which decides how much of the gradient vector will be used to adjust the current weights. A lower learning rate will lead to slower convergence but can help navigate the ravines in the search surface of the loss. In contrast, a larger learning rate may lead to larger steps that overshoot the minima with a non-convergent behavior. The learning rate can be selected in different manners. A common and straightforward way is to set the learning rate equal to a constant, but it is also possible to use a learning rate function that reduces the value over time. The latter reduces loss more quickly initially but avoids the unstable behavior when the value converges towards a solution. Gradient descent can guarantee convergence towards a local minima but requires a convex loss function to guarantee convergence to a global minima.

However, there are some challenges related to using gradient-based optimizers. Two problems may arise during training of a deep neural network, namely the *exploding gradient* and *vanishing gradient* problems. The exploding and vanishing gradient problems create unstable networks that cannot learn from the training data (Hochreiter and Schmidhuber, 1997). In the process of finding the gradient, i.e., partial derivatives of each layer, deeper layers go through continuous matrix multiplications to compute the derivatives. The problem of exploding gradients arises when the multiplication of numbers increases exponentially as the derivative propagates through the network. The vanishing gradient problem is essentially the opposite, where the numbers decrease converging to zero. Both problems create states in the network that update the weights to extreme values making the network incapable of learning from an overflow or vanishing gradient.

The ADAM Optimizer

The adaptive moment estimator, ADAM, is similar to gradient descent but works with momentums of first and second-order (Kingma and Ba, 2014). The algorithm uses squared gradients to scale the learning rate and the moving average of the gradients to take advantage of the momentum. The nth moment of a random variable x is defined as the expected value of the variable to the power of n, $E[x^n]$. As the gradient of a loss function, L usually is calculated with regards to a random batch of data; it can be considered a random variable. Adam updates the exponential moving average of the gradient, m_t, and the squared gradient v_t as follows.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla L_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\nabla L_t^2$$

Where ∇L_t is the gradient vector of the loss function, and β_1 and β_2 are hyperparameters for the exponential decay rates of the moment estimates, with recommended default values of 0.9 and 0.999, respectively. At the first iteration, the moving averages are initialized at zero. For the moment vectors not to be biased towards zero, they are corrected regarding this bias as follows.

$$\hat{m_t} = \frac{m_t}{1 - \beta_1}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2}$$

Based on these values, the model weights denoted as θ are updated using the following formula of the ADAM optimizer.

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon}$$

Where ϵ is a small constant typically equal to 10^{-6}.

The LAMB Optimizer

The Layerwise Adaptive Moments optimizer for Batch training (LAMB) (You et al., 2020) uses the ADAM optimizer from the previous section as a base algorithm in combination with the Layerwise Adaptive Rate Scaling (LARS) optimizer. The learning rate of LARS, and thereby also the LAMB optimizer, is adapted for each layer. This is performed by normalizing the gradients to decouple the magnitude of update from the magnitude of the gradient. The ratio of the weight norm to the gradient norm is called the *trust ratio* for each layer and allows the layers with larger weight norms to use a higher learning rate that converges faster without a performance loss.

The adaptivity of LAMB is two-fold: (1) per dimension normalization regarding the second moment's square root from ADAM, and (2) layerwise normalization obtained due to the layerwise adaptivity (You et al., 2020). Based on this, the LAMB optimizer forms an update of the model weights, $\theta_t^{(i)}$, where i is the layer at time step t as follows.

$$\theta_{t+1}^{(i)} = \theta_t^{(i)} - \eta_t \frac{\phi \left\| \theta_t^{(i)} \right\|}{\left\| r_t^{(i)} + \lambda \theta_t^{(i)} \right\|} (r_t^{(i)} + \lambda \theta_t^{(i)})$$

Whereas $r_t = \hat{m_t}$ from the ADAM optimizer, and λ is the weight decay rate with a default value of 0.01. η_t is the layerwise learning rate deciding how much the layer changes the model weights in one update, and ϕ which is the scaling term that scales the norm of the update into the same order as the model weight. You et al. found that ϕ typically ensures faster convergence and that a simple min-max function works well. The LAMB optimizer has proven to be more effective than the ADAM optimizer when training on larger batch sizes (You et al., 2020; Kummervold et al., 2021), although You et al. noted that the optimizer generally also works well for smaller batch sizes.

2.4.6 The Encoder-decoder Architecture

The *encoder-decoder architecture* is an architecture designed to handle inputs and outputs of variable-length sequences (Wu et al., 2016). The architecture has proven to be useful for machine translation and otherwise general sequence-to-sequence prediction tasks.

The architecture consists of two major components, the encoder, and the decoder. The encoder takes an input sequence of variable length, which it encodes into a state with a fixed shape. The decoder maps the encoded state of fixed shape into a sequence of variable length. This is often done using RNNs where both encoder and decoder are stacks of RNN units. In machine translation, a vector representing the input sequence is passed to the RNN encoder, which generates a list of vectors with one vector for every input token. The list of vectors is passed to the decoder RNN which produces one output token at a time until an end-of-sentence token is produced.

One challenge with the encoder-decoder architecture is that it often degrades as the length of the input sequence increases, a problem commonly known as the long-term dependency problem. One way to encounter this problem is by applying the attention mechanism described in the next section.

2.4.7 The Attention Mechanism

The *attention* mechanism initially proposed by Bahdanau et al. (2015) and Luong et al. (2015) introduced an improvement to the encoder-decoder architecture by considering the relative importance of terms of an input vector. There are two types of attention mechanisms: one that manages and quantifies the interdependence between input and output elements, called general attention, and one that quantifies the interdependence within input elements, called self-attention. Additionally, Vaswani et al. (2017) developed a mechanism called multi-head attention that will be described subsequently to general attention and self-attention.

General Attention

The general attention mechanism works by first passing all of the encoder's hidden states to the decoder and thereby making all information available simultaneously. The attention is then applied as a weighted average function that weights the importance of each term. The weight decides how the decoder should *attend* to each of the input terms, which relieves it from having to encode the entire input sequence to a fixed-length vector (Bahdanau et al., 2015). The decoder typically learns the weights using a feed-forward neural network where the sum of the weighted hidden states forms a *context vector*. Finally, the context vector can be passed through a feed-forward neural network producing the first output token, which again is passed to the network to produce a new hidden state, repeating the process until the final end-of-sequence token is produced. The mechanism resolves the long-term dependency problem, as the weights spread the information throughout the sequence of terms that can be selectively retrieved by the decoder (Bahdanau et al., 2015).

Self-attention

The self-attention mechanism fundamentally shares the same concepts as general attention but concerns how the terms of the input sentence correlate. The encoder can use this functionality to produce more meaningful representations of term sequences (Vaswani et al., 2017). Considering the sentence "The student did not study, because he was tired," the self-attention mechanism will capture that term *he* refers to *the student*. This is more difficult with simpler RNNs suffering from short-term memory loss as the sentence length increases.

More formally, the self-attention function maps every query and a set of key-value pairs to an output (Vaswani et al., 2017). This is obtained by initially representing the query, key, and value as three weight matrices learned during the training process. The three matrices are multiplied with the input word embedding vectors to obtain a vector for the query, key, and value for every input. The second step obtains the attention score for each term by multiplying the query of the current attending input with the key vectors from other inputs. These scores are then normalized using the softmax function and multiplied by the corresponding value vector. The sum of the final vectors represents the attention-encoded representation of the corresponding input sequence. The calculated self-attention scores obtain a higher value for terms of higher relevance that the model should "pay attention to" and a lower score for less relevant terms.

Multi-head Attention

Multi-head attention is simply a stack of self-attention that allows models to jointly attend to information from several representations at different sequence positions (Vaswani et al., 2017). This is obtained by running through the self-attention multiplication function h times in parallel. The independent attention "head" outputs are concatenated and linearly transformed into the desired dimensions. The formula for multi-head attention is given below, whereas Q, K, and V represent the query, key, and value matrices for the self-attentions.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The above Ws are learnable parameter matrices. The idea of the multi-head attention stack is that each attention head represents one property of the sentence, such as the nouns, whereas the others represent other property types such as pronouns, which combined produces a better representation of information of an input sequence.

2.4.8 Transformers

Whereas the attention mechanism described so far has been used to augment existing neural networks, the Transformer architecture proposed by Vaswani et al. (2017) is solely based on the self-attention mechanism in the format of multi-head attention to representing inputs and outputs. This approach has proven to achieve state-of-the-art results for most language tasks and is more efficient than the previous neural approaches for input and output representations (Vaswani et al., 2017). The improvement in efficiency is due to the absence of recurrence and convolutions, which lets the Transformer perform most of the calculations in parallel as opposed to previous approaches augmenting existing neural networks.

As illustrated in Figure 2.4.8, the Transformer architecture is composed of a stack of N identical encoder and decoder layers divided into two separate main modules. Each of the

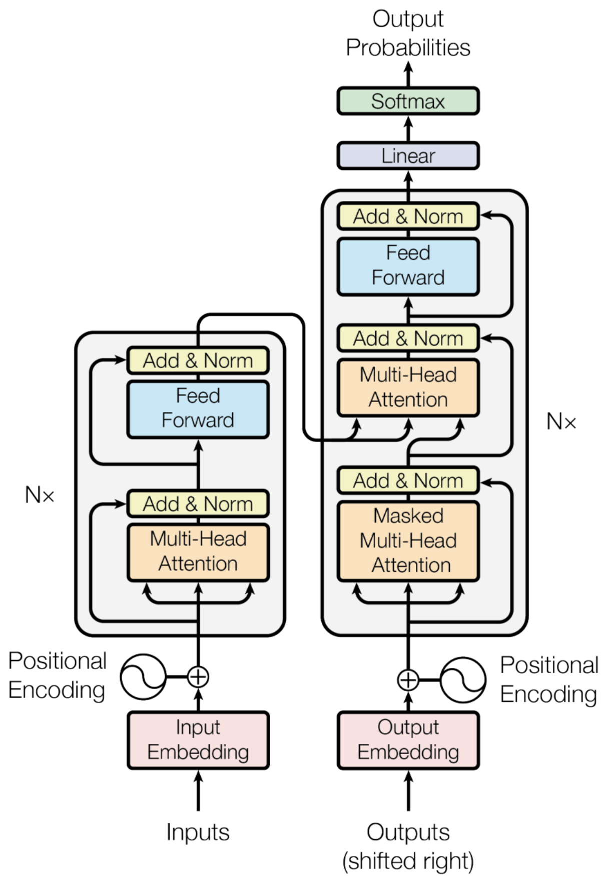


Figure 2.6: The attention-based architecture of a Transformer composed of an encoder (left) and a decoder (right). The figure is adopted from Vaswani et al. (2017) and reprinted with permission from Lion Jones.

encoder stack's layers has two sublayers: the first is a multi-head attention mechanism based on self-attention to encode input sequences, and the second is a simple feed-forward neural network. For each encoder layer, a residual layer is applied to aggregate the encoded and original input sequences by taking the sum of the two. This is followed by applying a normalization function to the aggregated sequence, which is finally passed to the feed-forward neural network. The two steps are noted as "Add & Norm" blocks in Figure 2.4.8. The output of the first encoder is passed to the second and so on until it has been passed through the N encoder layers and further to the decoder.

The decoder layers incorporate an additional layer performing multi-head attention to the output embeddings of the encoder stack. The outputs of the multi-head attention layer are aggregated similarly to the encoders and are also followed by layer normalization. The multi-head attention layer in the decoder is modified to prevent positions from attending to previous positions. The output embedding is offset by one position making it impossible for the predictions to depend on positions beyond their own. The decoder is in this way *masked* which means that it can only attend to words that are already produced. The term positions are obtained from the positional encoding, which gives every term a relative position in the sequence and is added to the embedded representation.

Another difference of the decoder layers is that the second multi-head attention layer takes both the aggregated and normalized output of the first masked multi-head attention layer and the encoder stack's output as inputs. Therefore, the encoded input sequence is used as input to all decoders at the same time as the output sequence is processed.

Finally, the output of the decoder is passed as input to a linear layer that produces an output vector of specified size, which for sequence classification tasks correspond to the number of classes. The predicted output corresponds to a probability distribution that can easily be converted to classes using the argmax function.

The attention function, when performed on matrices as described in Section 2.4.7, can be condensed into the following function proposed by Vaswani et al.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The denominator, $\sqrt{d_k}$, is the square root of the hidden dimension in the Transformer and works as a scaling factor. This factor was introduced by Vaswani et al. to prevent the softmax function from reaching regions with extremely small gradients, a problem previously described in Section 2.4.5.

The Transformer has been applied to several NLP tasks achieving state-of-the-art performance. One of the tasks is Neural Machine Translation (NMT), an approach to machine translation that uses artificial neural networks to predict the likelihood of a sequence of terms. The encoder of the Transformer takes the source language as input tokens and finds what output tokens of the target language correspond best to the source language. In cases with multiple target languages, that is, a model that can translate directly between several different languages, a special token can be added in the encoder indicating the source language and a special token in the decoder indicating the target language (Fan et al., 2020). The following section describes several other NLP tasks solved using Transformer-based architectures.

2.4.9 Bidirectional Encoder Representations from Transformers

The Bidirectional Encoder Representations from Transformers (BERT) developed by Devlin et al. (2019) is the first unsupervised, deep bidirectional model pre-trained on general language understanding. BERT is built to produce contextual sentence representations that can be used for several NLP tasks such as text classification and question answering. Devlin et al. trained the model on two separate learning objectives: masked language modeling and next sentence prediction. Masked language modeling randomly masks 15% of the input tokens. BERT is trained to predict the missing tokens based on the context it occurs. The second task of next sentence prediction is to predict whether a sentence is the following sentence of another. This allows BERT to understand better how sentences relate, which is essential knowledge in most NLP tasks.

Since the introduction of BERT, several models building upon the same principles have emerged. There are now multiple state-of-the-art Transformer-based models with different sizes and configurations of the standard BERT models, as will be mentioned in Chapter 4. As most of these models are not yet available trained on the Norwegian language, only the general BERT model will be described in this section. Furthermore, as this thesis concerns sentence classification tasks, this section will focus on how BERT is applied to classify sentences. Firstly, the model architecture will be described in detail, followed by a description of the particular tokenization and encoding techniques used to represent input documents. Finally, the process from model input to predicted output will be described.

BERT Model Architecture

The BERT model architecture used for text classification is mainly composed of a pre-trained Transformer-based architecture with an output layer on top used to fine-tune it for a particular downstream task (Devlin et al., 2019). The structure of attentions and Transformers are previously described in Sections 2.4.7 and 2.4.8.

To build a classification model, the BERT model goes through a pre-training and a fine-tuning stage. During the pre-training stage, the model is trained on a large corpus of various unlabeled data. The purpose is to learn the meanings of the terms and the contexts they occur in, that can later be used as transferred knowledge to a downstream task. The model is initialized with the pre-trained parameters for the fine-tuning stage, which are later adjusted with the labeled data for the specified task. The purpose of the fine-tuning stage is for the model to learn what language separates the different classes, for instance, neutral and offensive speech. The model already knows the meaning of most terms from the pre-training stage but is trained on what specific data represents each class. The pre-training process is computationally expensive and is a one-time procedure, as opposed to the fine-tuning stage, which has to be done individually on top of the pre-trained model for each downstream task.

Devlin et al. released the following two BERT model architectures, both of which are used in the experiments of this thesis.

- **BERT base:** 12-layer, 768-hidden, 12-heads, 110M parameters
- **BERT large:** 24-layer, 1024-hidden, 16-heads, 340M parameters

The *base* and *large* keywords denote the number of encoder layers in the model, determining the model's size. As the number of layers increases, so do the number of parameters (weights) and the number of attention heads in the model. BERT$_{base}$ has 12 encoder layers with 768 hidden units and twelve attention heads, whereas the BERT$_{large}$ model consists of 24 encoder layers with 1024 hidden units and 16 attention heads. Devlin et al. trained BERT using the GELU activation function, previously described in Section 2.4.1.

Furthermore, the model is either *cased* or *uncased* which determines what text format on which the model is trained. The *cased* keyword denotes that the model is trained on case-sensitive text, text combining upper- and lowercase letters, with accent markers preserved. Cased models have separate vocabulary entries for differently cased terms (e.g., the terms *Student* and *student* will be represented by different tokens, although they have the same meaning). In contrast, the uncased model is trained on lowercase text without accent markers. The cased model is often used for tasks where the case or accent markers have a predictive value; otherwise, the uncased model is used. However, not all models are available in both versions.

BERT Tokenization

For the model to understand the input documents, they have to hold an interpretable format, where the first step to achieve this is tokenization. The same pre-processing steps must be followed for the fine-tuning stage as for the pre-training stage. Therefore, the tokenizer used is also dependent on whether the model is *cased* or *uncased*, a configuration described in the previous section.

BERT tokenization is performed in two steps; general tokenization and WordPiece tokenization which does the following operations on the data.

- **General tokenization:** Performs normalization by converting all whitespace characters to spaces, followed by punctuation splitting, which is done by adding whitespace on each side of any punctuation character[1]. For uncased models, this step also converts all text into lowercase characters and removes accent markers.
- **WordPiece tokenization:** Produces meaningful context-independent token representations of terms. Each term is first split on whitespaces into separate tokens, which are then passed to the WordPiece algorithm, further described below.

The general tokenization makes it possible for the model to recognize terms without learning all combinations of them with different punctuation marks. The WordPiece algorithm is a technique to segment terms into subterms based on the principle that frequently used terms should stand as they are, while less frequently used terms are decomposed into meaningful subterms. For instance, the term *herrefotball* (translates to *men's soccer* in English) may be considered a less frequently used term that could be broken down into the tokens `herre` and `##fotball`. The double hashtags mark that the token is subsequent to the previous one[2]. The WordPiece representation allows the model to process previously unseen terms out-of-vocabulary (OOV) by breaking them down into subterms. This way, some of the original term's contextual meaning will be retained without representing it using a standard OOV token. This technique is particularly beneficial for agglutinative languages such as Norwegian, which has a vast number of different compound terms that the WordPiece algorithm handles.

[1]In this case punctuation characters include all characters that are not letters, numbers or spaces, such as $, which is technically not a punctuation character.

[2]Note that the WordPiece algorithm splits terms based on the vocabulary the model is pre-trained on, which may vary between different models. The examples given of WordPiece tokenization in this thesis are therefore illustrative and may not represent the same splits produced by the models used in this thesis.

BERT Encoding

Following the tokenization process, tokenized documents are encoded into a particular format interpretable for the BERT model. The BERT model takes specific length sequences as input, such as the encoders of the Transformer. The input representing the sequence of each document consists of the terms of the sentence, and the three additional tokens `[CLS]`, `[SEP]` and `[PAD]`, which represents the following meanings in the context of text classification.

- `[CLS]`: The classification token is a special token added in front of every input instance to mark the beginning of the document.
- `[SEP]`: The separation token is a special token that, in the case of classification tasks, is added at the end of each input instance to mark the end of the document.
- `[PAD]`: The BERT model requires all input instances to be of fixed length, which is set in the model configurations. For shorter lengths, the padding token adds extra length at the end of the tokenized text.

Instances longer than the fixed sequence length are truncated into the same size and format. Therefore, the tokens surpassing the length are removed, and some information may be lost. An example of a tokenized sentence is given below. The maximum sequence length of the sentence is set equal to nine for simplicity. The tokens represent the sentence "Studenten går på universitetet," which directly translates to "The student attends the university" in English.

[CLS]	Student	##en	går	på	universitet	##et	[SEP]	[PAD]

After the tokenization process is finished and the special tokens are added, all tokens are converted into their corresponding IDs from the BERT model's learned vocabulary. This way, the tokens are interpretable for the model to predict.

BERT Model Input and Output

The BERT input is represented by the input IDs of term sequences as described in the previous sections. These input IDs correspond to attention masks deciding what tokens should be attended to and which should not, i.e., the padding tokens. Each input representation is constructed by summing the corresponding *token*, *segment*, and *position* embeddings of the input ID elementwise into a vector representation. The three embeddings have the following functions for classification purposes.

- **Token embeddings:** Have the role of transforming the input tokens into a fixed-length vector representation of size (n,768) for the BERT$_{base}$, and (n,1024) for the BERT$_{large}$, where n corresponds to the length of the input sequence.

- **Segment embeddings:** Are originally used to distinguish between a pair of input sentences. However, the segment embeddings do not influence classification tasks. This layer is represented by a vector of zeros for classification tasks, not affecting the final summation value.
- **Position embeddings:** Hold information about the order of the input sequence, which is crucial for Transformer-based architectures. Without them, the representation corresponds to a simple bag-of-words model. The position embeddings are represented as a lookup table of size (512, 768) for BERT$_{base}$ and (512, 1024) for BERT$_{large}$, where 512 is the maximum length of BERT sequences. Each row represents the term on the position in the input sequence corresponding to the row index. The positional embeddings are similar for terms holding the same position in the input sequence.

After the summation, the resulting input representation of size (n, 768) is passed into BERT's encoder stack. A visualization of this process is displayed in Figure 2.7.

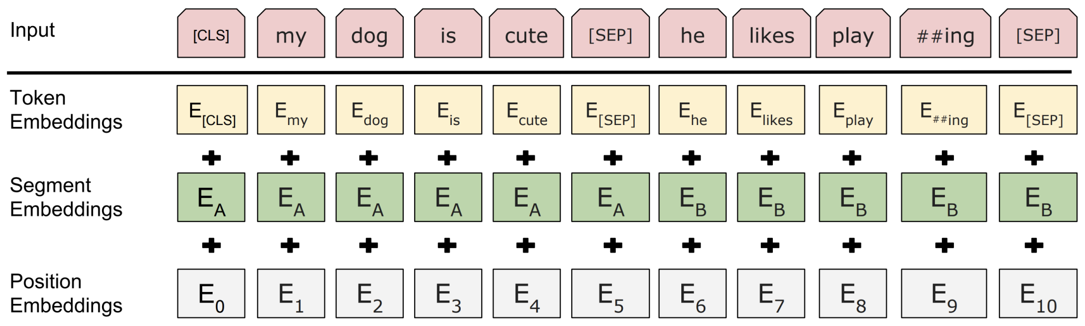

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2.7: The BERT model's input representation. The figure is adopted from Devlin et al. (2019) and reprinted with permission from Jacob Devlin.

Each token position of the input representation outputs a hidden vector of length 768 for BERT$_{base}$ and 1024 for BERT$_{large}$. In the case of classification, only the first position is passed onto the next layer. This vector represents the contextual word embeddings passed as input to the classification layer on top of the BERT model to predict the class of the input sequence. The classification layer can be defined in several different ways. Devlin et al. (2019) used a simple feed-forward neural network with one hidden layer and a softmax layer on top. This fine-tuning process of classification tasks is illustrated in Figure 2.8.

2.5 Tools and Libraries

Many available tools and open-source libraries can assist the implementation of the approaches described in this chapter. This section describes some easily accessible tools

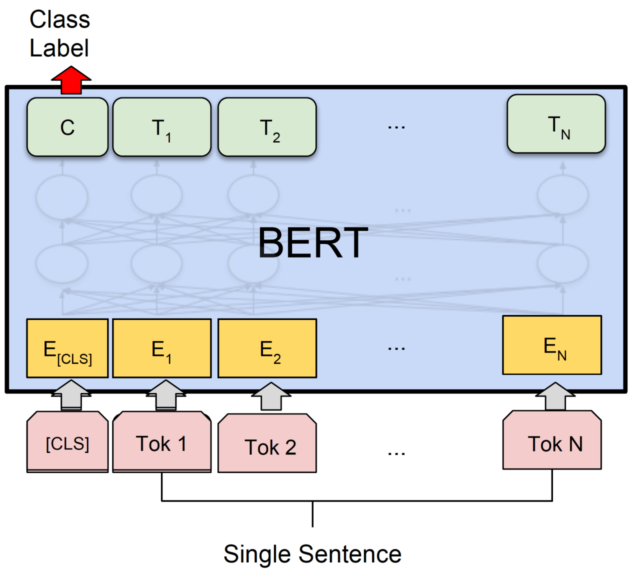


Figure 2.8: An illustration of the fine-tuning of BERT for single sentence classification tasks. The figure is adopted from Devlin et al. (2019) and reprinted with permission from Jacob Devlin.

and libraries used in this thesis. All code for the experiments is written in Python due to its extensive support for machine learning for text classification.

Poetry
Poetry (Poetry, 2018) is a tool for dependency management and packaging in Python. It allows declaring the libraries a project depends on, and it manages installations and updates in an activated virtual environment. Poetry supports reproducibility by obtaining all libraries and their corresponding dependencies in a `pyproject.toml` file. When all libraries and dependencies are installed, they are saved in a `poetry.lock` file, which locks the project to those specific versions. Using the same `poetry.lock` on different devices ensures that the project is built with the same dependencies and versions across all devices.

Pandas
Pandas (McKinney, 2010) is an efficient, flexible, and easy-to-use open-source data analysis and manipulation tool library for Python. The library enables easy manipulation and visualization of datasets through data structures called data frames.

Natural Language Toolkit
The Natural Language Toolkit (NLTK) (Loper and Bird, 2002) is an open-source library for handling natural human language data. The toolkit provides high-level functions for tokenization, stemming, stopword removal, punctuation removal, and more.

Hugging Face Transformers
Hugging Face (Hugging Face inc., 2021) is an NLP-focused startup with a large open-source community for Transformer-based architectures. Hugging Face's Transformer library exposes an API to download many versions of the different state-of-the-art Transformer-based approaches, such as BERT, for several languages and the tokenizers corresponding to these models. The library has support for PyTorch integrations, making it easy and efficient to use.

PyTorch
PyTorch (Paszke et al., 2019) is an open-source library that enables developing deep learning approaches in Python. PyTorch provides high-level implementations of more complex approaches and provides easy access to utilize available computational resources (such as multiple GPUs) in parallel. This makes the library efficient and easy-to-use. The Pytorch implementation of the BERT model has proven empirically to perform comparably to the original BERT implementation by Devlin et al. (2019).

Scikit-learn
Scikit-learn (Sklearn)(Pedregosa et al., 2011) is an open-source library providing simple and efficient tools for predictive data analysis. The library enables easy-to-use functionality for dataset handling, methods, and evaluation for machine learning.

Translatepy
Translatepy (Sekai and Roman, 2021) is an aggregation of multiple translation web APIs in Python. The different incorporated machine translator systems include Microsoft Bing Translator[3], Google Translate[4], Yandex Translate[5], Reverso[6], and DeepL[7], that support translation from Danish to Norwegian used in the experiments of this thesis. The library is made available through the Python Package Index (PyPi) (Python Software Foundation, 2021) which is an open-source repository for Python software.

EasyNMT
EasyNMT (Reimers, 2021) is a Python package providing state-of-the-art Transformer-based neural machine translation approaches. Several pre-trained models can be downloaded and used in a high-level and simple manner to translate between more than 150 languages. Three of the available models provide support for Danish-Norwegian translation and are used in the experiments of this thesis: OpusMT (Tiedemann and Thottingal, 2020), and M2M_100 (Fan et al., 2020). The OpusMT is a collective name for multiple models trained on translation for more than 150 pairs of languages based on a collection of translated texts from the web. The M2M_100 is a many-to-many

[3]`https://www.bing.com/translator`
[4]`https://translate.google.com/`
[5]`https://translate.yandex.com/`
[6]`https://www.reverso.net/text_translation.aspx`
[7]`https://www.deepl.com/translator`

multilingual machine translator system that translates directly between any pair of 100 languages. The model is trained on a dataset mined from CommonCrawl, an open repository of web crawled data, and is available trained with 418 million (M2M_100_418M) and 1.2 billion (M2M_100_1.2M) network parameters.

Wordcloud
Wordcloud (Mueller, 2021) is an easy-to-use word cloud visualization library in Python that enables visualization of high-frequency terms in texts.

3 Datasets

Creating an abusive language dataset requires an extensive amount of work and can pose some challenges. Therefore, the work in this thesis will build upon existing datasets. This chapter will cover some of the challenges of working with abusive language datasets and how they may cause unwanted effects to machine learning models. The chapter describes two abusive language datasets, one Danish and one Norwegian. Section 3.1 builds upon Section 3.1, and Section 3.2 builds upon Section 3.2.3 and 3.2.4 from the specialization project (Arntzen, 2020).

3.1 General Challenges with Abusive Language Datasets

Supervised classification tasks require labeled data to learn classifications. Collecting and annotating such data can pose some challenges. Section 3.1.1 describes challenges regarding the annotation process and how people's subjective perception of hate speech affect this task, while Section 3.1.2 looks into how the distribution of abusive language datasets are affected by the available documents online and how public media choose to moderate their content.

3.1.1 The Perceptions of Abusive Language Types

Annotating abusive language data is everything but trivial and requires a procedure and guidelines for the annotation process. It is common to use human annotators to label data, which can be done using expert annotators or amateurs. However, one significant challenge when annotating the data is that there is no formal definition of hate speech, as presented in Chapter 1. Hate speech is somewhat subjective and tends to be obfuscated with other abusive languages. The obfuscation of abusive language types has made it hard to set a boundary between them and for annotators to objectively comprehend them, leading to noise and bias in the datasets that further propagate to the models.

Ross et al. (2017) found a low agreement between annotators on what classifies as hate speech. Their research on how to reliably annotate a Twitter dataset surveyed two groups on tweets' abusiveness. They asked them whether a tweet is hateful if it should be filtered from Twitter and its degree of offensiveness. The degree of offensiveness was defined as a number from one to six, with an increasing degree of offensiveness. Only one group was given Twitter's definition of hateful conduct. Their result indicated that providing users with a definition led to them partially aligning their subjective views with the definition. It did not improve reliability, which was overall low. Their conclusion yields that hate speech is a vague concept and that it requires better definitions and guidelines for reliable annotations.

To further understand what cases make annotators disagree, Caselli et al. (2020) recently tested inter-annotator agreement on hate speech and found several areas with a substantial amount of disagreement. One area with disagreement concerns cases with specific characteristics of offensive messages, particularly messages with a subjective nature of sensitivity to the audiences. They noted that the distinction between negative stance and implicit expressions of abuse often is mistaken and gave an example of a tweet, as seen in the below example.

Example	@USER @USER I believe gun control should consist of guarding your firearms from thivery and kids.

The tweet contains a negative stance towards gun control, misinterpreted as implicit hate by annotators in the English OLID dataset by Zampieri et al. (2019b).

The second substantial amount of disagreement Caselli et al. found came from people's different perception of expressions in the borderline of swear words, such as the word *weirdo*. They noted that the degree of abusiveness is dependent on the context, and how annotators perceive it may depend on their background. The final area reported by Caselli et al. is annotators' understanding of implicit languages such as irony and sarcasm. In order to understand such cases, it may also be needed to have an understanding of the contextual circumstances. They noted that the lack of context might lead to mistakes in the annotation process, which propagate into noise in the dataset. Therefore, considering the context of occurrence is necessary to reduce biases in the data, particularly with jargons or dialects accepted within a community (Caselli et al., 2020).

Furthermore, Waseem (2016) found that expert annotations yield better results than amateur annotations, as amateurs are more likely to annotate a sample as hate speech than experts due to aligning their subjective idea of what hate speech is when annotating. However, Waseem also showed that using several amateurs having full agreement can achieve better results than expert annotators alone. Therefore, a method of reducing the expert annotators' workload is to use crowdsourcing for all documents and send the ones with disagreement to the expert annotator for review. Crowdsourcing is a process of obtaining labels from a large, relatively open group of people often considered amateurs. However, for the Norwegian language, finding platforms to outsource annotation jobs can be challenging. In contrast to English, there are fewer Norwegian native speakers, and the language is considered a minority language unknown to most annotators at global crowdsourcing platforms. Additionally, the average salary in Norway is high, making manual annotation of datasets costly.

The different perceptions of abusive language types also affect models' ability to generalize across datasets. As datasets use various definitions, models adapt to different boundaries when trained on separate datasets. Related literature shows that models trained on one dataset tend to perform well only when tested on the same dataset (Gröndahl et al., 2018; Swamy et al., 2019; Bulukin, 2021). Fortuna and Nunes (2018) argued that although models perform better on the data they are trained on, performance can be slightly improved by extending the training dataset with other social media data. Furthermore, Swamy et al. (2019) claimed that a model's performance on the dataset it is trained on is not indicative of how well it performs in real-world problems. Based on this, they suggested that the quality of an abusive language dataset should be assessed by measuring how it represents abusive language as a whole to increase generalizability.

3.1.2 The Unbalanced Nature of Abusive Language Datasets

As opposed to many text classification tasks, abusive language detection often suffers from a severe class imbalance issue. This not only originates in the complicated annotation process discussed in the previous section but is also due to the challenges of collecting the data. Abusive language, such as hate speech, occurs naturally less often in social media contexts than neutral speech. The estimated maximum number of derogatory documents on the social media platform Twitter is 3% according to Founta et al. (2018). This makes the identification and collection of such documents challenging, which leads to unbalanced classes in many cases. Classification using unbalanced datasets pose challenges as most machine learning algorithms assume an equal number of instances of each class. In many cases, the models tend to be biased towards the majority class, leading to poorer predictions for the minority class. In the case of abusive language, this is a problem as the minority classes, such as hate speech, are the most concerned classes.

Another challenge of collecting abusive language datasets is that social media platforms and public media sites remove abusive content violating their rules. This removal affects the availability of abusive content online, causing the number of available abusive documents to be even less than what naturally occurs. For instance, Twitter report taking action on hateful content violating their rules, such as targeted abuse and harassment (Twitter inc., 2021). When collecting abusive language datasets online, one can assume that some content has been removed beforehand, affecting the class distribution in the dataset.

Additionally, moderation affects the representative abusive documents online. One may assume that some abusive documents are immediately removed when posted, which puts some restrictions on the available data that can be collected. Such documents may include undoubtedly hateful documents that the public media platforms' algorithms detect before even being seen or reported by users of the platform. Therefore, the representative documents in abusive language datasets are limited to the documents that have not been filtered beforehand to the collection process, also limiting the scope of research to these documents. One may assume that without such moderation, more documents having a higher degree of abusive content would have been available for research.

3.2 Selected Abusive Language Datasets

This section describes the selected datasets used for abusive language detection in the experiments of this thesis, the collection process, structure, and statistics. The dataset by Svanes and Gunstad (2020) and Jensen (2020) described in Section 3.2.1, was chosen as it is the only available Norwegian dataset for abusive language detection at the time of writing. The dataset by Sigurbergsson and Derczynski (2020) described in the Section 3.2.2, was chosen due to being the most similar dataset to the Norwegian dataset when considering the labeling and the language similarities discussed in Section 2.1. The dataset is used in one of the experiments to extend the Norwegian dataset to increase the number of offensive and hateful documents.

3.2.1 A Norwegian Abusive Language Dataset

Svanes and Gunstad (2020) and Jensen (2020) collected the first available Norwegian abusive language dataset. The dataset consists of 41,145 documents from the social media platforms Twitter and Facebook and the uncensored Norwegian newspaper, Resett. Although Svanes, Gunstad, and Jensen produced the dataset, the statistics and descriptions in this thesis are based on the thesis by Svanes and Gunstad as their experiments and measures are closely related to the experiments of this thesis. The labels follow a mutual exclusive annotation scheme with the following categories according to Svanes and Gunstad:

- **Neutral:** Utterances that do not meet the criteria of other categories belong to this class. A neutral utterance contains neutral language and is a factual contribution to the debate.
- **Provocative:** Utterances that contain aggressive language to express an opinion or can be perceived as inappropriate. This includes the use of profane words, patronizing language, or the use of irony and sarcasm to lower the credibility of an opponent.
- **Offensive:** Utterances that contain hurtful, derogatory, or obscene language, either directed towards an individual or a group.
- **Moderately hateful:** Utterances that 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.
- **Hateful:** Utterances that 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.

The documents were collected in three steps, whereas the Resett documents were collected using web scraping on the 1,000 most recent articles, mainly discussions on controversial topics such as immigration, environment, and politics (Svanes and Gunstad, 2020). Svanes and Gunstad found that most of the abusive documents were related to immigration and Muslims and noted that their models struggled to detect abusive documents on other topics. Therefore, they extended the dataset with documents from Twitter and Facebook. The tweets were collected through Twitter's search API with a keyword search on common Norwegian inflamed terms to gather Norwegian tweets with a substantial amount belonging to the abusive classes. Lastly, the Facebook documents were collected manually from 65 different posts on public pages such as Norwegian newspapers and public persons. Most of the sources were also related to immigrant critical pages, due to this being a common subject in Norwegian hatred debates. All names in the dataset were anonymized. The documents originating from Twitter were anonymized with the `@USER` tag, whereas the remaining documents were anonymized using the `Navn` tag.

The documents were annotated by Svanes, Gunstad and Jensen and 20 external annotators, following the pre-defined annotator guidelines (presented in Appendix 2). Firstly, 2,500 documents were annotated by the three creators using the majority vote, and inter-annotator agreement was calculated, a process further described in the next paragraph. Svanes and Gunstad describes that during the annotation of the first 2,500 documents, they modified the annotation guidelines based on their discussions. Doing modifications to the annotator guidelines during the annotation process violates the principle of ensuring consistent research additionally to preventing reproducibility. However, for the remaining documents annotated by one annotator each (including the external annotators), the same annotator guidelines were discussed beforehand and used. The distribution of labels for each class is displayed in Table 3.1.

#	**Category**	**# documents**	**% of all**
1	Neutral	34,085	82.8
2	Provocative	4,737	11.5
3	Offensive	1,563	3.8
4	Moderately hateful	510	1.2
5	Hateful	250	0.6
1-5	**ALL**	**41,145**	**100**

Table 3.1: The distribution of labels in the Norwegian abusive language dataset by Svanes and Gunstad (2020).

Annotation agreement of the 2,500 first documents were calculated using Fleiss' kappa score, described in Section 2.3.2, and percentage agreement. The calculations are displayed in Table 3.2. Although they report a fear overall agreement with a Fleiss' Kappa score of 0.3869, the agreements amongst the abusive classes are not as coherent as for the neutral class. One can observe from Table 3.2 that the agreement is influenced by the high agreement for the neutral class, but that the annotators found it difficult to agree on which documents belong to the abusive categories, lowering the weighted score. Although this makes the abusive categories very interesting from a qualitative perspective, a poor inter-annotator agreement can make it difficult to distinguish between them in a quantitative valorization of the data.

	Neutral	**Prov.**	**Offensive**	**Mod. Hateful**	**Hateful**	**Total**
Full agreement	91%	3%	9%	13%	28%	89%
Majority vote	97%	26%	35%	9 %	39%	99.1%

Table 3.2: The annotation agreement of the Norwegian dataset. The table is adopted from Svanes and Gunstad (2020) and reprinted with permission from Marie Andreassen Svanes.

Although only one annotator annotated each of the remaining documents, the documents labeled within the abusive categories (2-5) were verified by one of the creators and relabeled in cases of disagreement. In cases of uncertainty, they involved the other creators and discussed and agreed upon the label to increase the labels' reliability[1].

3.2.2 The Offensive Language Identification Dataset

The first Offensive Language Identification Dataset (OLID) was created by Zampieri et al. (2019b) to target several aggressive content types hierarchically for the English language. The dataset has been used as the official dataset for the shared tasks OffensEval 2019 and 2020, which concerned identifying and categorizing offensive languages in social media (Zampieri et al., 2019a, 2020). For OffensEval 2020, datasets in several languages following the same annotation scheme and tasks were collected. This section first describes the OLID annotation scheme, followed by a description of the Danish OLID dataset used in the experiments of this thesis, collected and annotated by Sigurbergsson and Derczynski (2020).

The OLID Annotation Scheme

The OLID annotation scheme consists of three layers of annotations in which each layer corresponds to a specific task in OffensEval, as displayed in Figure 3.1. Each group's labels are mutually exclusive, such that a document cannot belong to multiple groups within one level. The labels within each level are defined by Zampieri et al. as follows.

- **Level A - Offensive language identification:** The first layer discriminates on whether the message is offensive (OFF) or not (NOT), in which an offensive message

[1]The source of this information is personal communication with Marie Andreassen Svanes, which clarified parts of the annotation process not explained in Svanes and Gunstad (2020).

is any message that includes insults and threats as well as any untargeted profanity, such as swear words.

- **Level B - Automatic categorization of offensive language types:** The second layer discriminates on whether an offensive message is targeted or not. A targeted offensive message (TIN) contains insults and threats to an individual, group, or others. An untargeted (UNT) offensive message contains general profanity, not targeting anyone.

- **Level C - Offensive language target identification:** The third layer discriminates on whether the targeted insult is targeted towards an individual (IND), group (GRP), or other (OTH). An individual is defined as a person that is a part of the conversation, while a group is defined as an entity of people based on ethnicity, gender or sexual orientation, political affiliation, religious belief, or other characteristics. The other category is the collection of documents that do not belong to the group or individual categories.

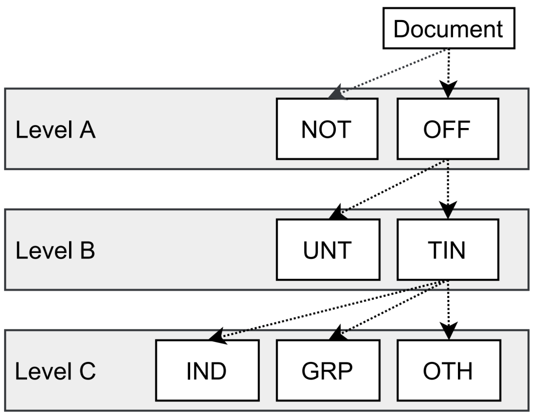


Figure 3.1: The hierarchical annotation scheme of the OLID dataset created by Zampieri et al. (2019b).

The annotation scheme builds upon the assumption that the target of offensive messages is essential in discriminating between types of abusive messages. According to Zampieri et al. insults and threats targeted at a group correspond to what is commonly understood as hate speech. In contrast, they report that insults and threats targeted at individuals are often defined as cyberbullying. Zampieri et al. does not exclude other types of abusive messages in these understandings as the OLID scheme's distinctions are on whether the documents are offensive and targeted and not whether they are hate speech or other forms of specified abusive language. However, if the goal is to detect specific subgroups of abusive language, the OLID scheme may present some challenges. For example, a threat targeted towards an individual based on identity factors such as religion or ethnicity will be labeled as GRP, although it targets an individual and therefore also belongs to IND, by definition. Although such statements will qualify as both IND and GRP, the groups are mutually exclusive, and the document will, therefore, receive only one label, introducing some vagueness to the dataset. With regards to this, Swamy et al. (2019) chose to report the number of hate speech examples as not applicable for this dataset.

The Danish OLID Dataset

The Danish version of the OLID dataset by Sigurbergsson and Derczynski (2020) consists of 3,600 documents collected from Facebook and Reddit and annotated using the OLID scheme. Facebook and Reddit were chosen as the sources by Sigurbergsson and Derczynski as Twitter, which was used by Zampieri et al. (2019b) in the original OLID dataset, has limited usage in Denmark. They, therefore, concluded that collecting data from Twitter would have resulted in low-quality data with many groups of interest being unrepresented. This is the first and currently only available Danish abusive language dataset. The label distribution of the dataset is displayed in Table 3.3.

Level A	**Level B**	**Level C**	**# documents**	**% of all**
OFF	TIN	IND	95	2.6
OFF	TIN	OTH	36	1.0
OFF	TIN	GRP	121	3.4
OFF	UNT		189	5.2
NOT			3,159	87.8
ALL			**3,600**	**100**

Table 3.3: The distribution of labels in the Danish OLID dataset by Sigurbergsson and Derczynski (2020).

Sigurbergsson and Derczynski manually collected 800 documents from the public Facebook page Ekstra Bladet, a Danish media company that, through an initial analysis, showed a high degree of variation in documents. They collected 2,800 documents using web scraping of two general Danish Reddit forums, in which a keyword search based on a crowdsourced lexicon compilation was performed at the end to obtain a sufficient proportion of offensive documents. The keyword was limited to half of the gathered data to ensure adequate data diversity. All user mentions were anonymized and replaced with the `@USER` tag, and documents containing any sensitive information[2] were removed.

The first hundred samples were annotated and evaluated by both authors to assess the similarity of their annotations. They calculated similarity using the percentage agreement

[2]Sigurbergsson and Derczynski (2020) define sensitive information as any information that can be used to uniquely identify someone by the following characteristics; racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, and biometric data.

for all three levels corresponding to 41.9% for level A, 39.1% for level B, and 42.8% for level C. The scores show that there were agreement for less than half of the annotations for all three levels, which indicates that the annotators found it challenging to agree on which documents belonged to what classes.

Sigurbergsson and Derczynski state that they discussed the cases of disagreement and found that most cases concerned documents that lacked context or documents using profanities, challenges presented in Section 3.1.1. They denoted that the annotation guidelines were refined to handle cases of disagreement. Following the refined guidelines, the remaining 3,500 were annotated only by the main author alone. As mentioned for the Norwegian dataset in Section 3.2.1, refining annotator guidelines during the annotation process is not good practice and prevent consistency and reproducibility. However, as this only included 100 comments for this dataset, the effect is likely negligible.

4 Related Work

This chapter covers related research in the field of abusive language detection and Transformer-based approaches. A structured literature search protocol is first described, followed by a presentation of related work to give an overview of the field of abusive language detection and clarify what gaps of research need to be filled.

4.1 Literature Review

This chapter's literature review aims to cover sufficient knowledge within the field of abusive language detection for the Norwegian language. This includes exploring high-performing models with the potential to achieve state-of-the-art results for abusive language detection in Norwegian. The literature review presented was initially guided by findings made in the specialization project (Arntzen, 2020) which surveyed several aspects of abusive language detection from traditional methods to more recent Transformer-based approaches in the field. This thesis will be concentrated around the more recent Transformer-based methods holding state-of-the-art results for abusive language detection tasks in several languages. To begin the specified study of this thesis, a structured literature search protocol is defined in Section 4.1.1, followed by some selection criteria defined in Section 4.1.2. The approach is adapted from Kofod-Petersen (2018). However, some adjustments specified in Section 4.1.2 were made to the process, as a folder of relevant research was provided by the supervisor of this thesis, Björn Gambäck.

4.1.1 Structure Literature Search Protocol

As a starting point *Google Scholar*[1] was selected as the search domain. Google Scholar is a freely accessible web search engine for scholarly literature across many publishing formats and disciplines. The literature is ranked based on the author's ranking, the number of references linked to the document and its relevance to other scholarly literature, and the publication's ranking of which the journal appears.

A set of search terms closely related to the specialization project's (Arntzen, 2020) research questions were defined to initiate the search. As the goal of this project was to give an overview of the state-of-the-art in hate speech detection and define future work for a following Master's thesis, the terms were general and covered more topics than what is included in the review of this thesis. The selected terms are defined in Table 4.1. They were defined to retrieve literature within the field for English and Norwegian, Swedish, and Danish. Swedish and Danish were selected due to limited research in the

[1]https://scholar.google.com/

area for Norwegian, as the three languages are mutually intelligible and have many similar characteristics, as previously described in Section 2.1.

	Group 1	**Group 2**	**Group 3**	**Group 4**
	Main topic	**Data domain**	**Specialized topic**	**Languages**
Term 1	Hate speech	Twitter	Machine learning	English
Term 2	Offensive language	Facebook	Classification	Norwegian
Term 3	Harmful speech	Instagram	Prediction	Swedish
Term 4		Reddit	Detection	Danish

Table 4.1: Terms used to define the search query of the literature review.

The groups and search query were defined as follows to retrieve relevant literature in the intersection of the groups, where n corresponds to the number of terms, and m corresponds to the number of groups.

$$Group_i = Term_1 \vee Term_2 \vee ... \vee Term_n$$

$$Query = Group_1 \wedge Group_2 \wedge ... \wedge Group_m$$

```
(hate speech OR offensive language OR harmful speech) AND
(twitter OR facebook OR instagram OR reddit) AND
(machine learning OR classification OR prediction OR detection) AND
(english OR norwegian OR swedish OR danish)
```

Table 4.2: Resulting query used in the structured literature search.

The query displayed in Table 4.2 resulted in a total of 29,000 papers in which the first 40 were collected for review. The gap between the amount of research targeting English abusive language detection versus the amount of research targeting Norwegian and otherwise Scandinavian abusive language detection, was as expected, large. Therefore, three additional searches were performed, which isolated the query for the Scandinavian languages Norwegian, Swedish and Danish, respectively, to ensure that no research got lost in the vast amount of English papers.

4.1.2 Quality and Selection Assessment

All papers were checked for the following criteria to maintain only relevant research in the review. The removal criteria were applied to filter papers out before verifying the inclusion criteria defining the requirements to be included in the review.

RC1 The article is a duplicate of another (keep the highest ranking)

RC2 The same study is published in different sources (keep the highest ranking)

IC1 The study's main concern is automatic classification of abusive language

IC2 The study is a primary study presenting empirical results

IC3 The study focuses on social media data from Twitter, Facebook, Instagram or Reddit

IC4 The study describes an implementation of a classification model

IC5 The study focuses on abusive language detection of the English, Danish, Swedish or Norwegian language

Following the above criteria resulted in a total of 22 papers. A comparison of these papers to the initially provided folder showed that 75% of the resulting papers matched. This confirmed that the folder contained relevant research. The structured search, therefore, initiated the literature review, which was supplemented with papers from the provided folder and the *snowball sampling* method, a method in which existing study subjects recruit future subjects among their acquaintances. As relevant research of high quality often refers to other relevant publications, this resulted in a good overview of abusive language detection.

4.1.3 Result

The specialization project (Arntzen, 2020) provided an overview of different aspects of abusive language methods from traditional to recent Transformer-based approaches currently holding state-of-the-art results. The work revealed that research within abusive language detection had grown rapidly over the past few years, leading to research beyond individual tasks. Several shared tasks on abusive language detection have been invented, providing a solid basis to compare models on equal terms resulting in an overview of current state-of-the-art models. The shared tasks OffensEval 2019 (Zampieri et al., 2019a) and OffensEval 2020 (Zampieri et al., 2020) are two examples of such tasks which laid much of the basis of the literature review in this chapter and the experiments in the remaining of this thesis.

4.2 Abusive Language Detection

This section presents relevant research in the field of abusive language detection, including a brief overview of state-of-the-art approaches in the field and an overview of available approaches for the Norwegian language. The research from the shared task, OffensEval will be presented first, followed by some research on domain-specific models. Finally, research on abusive language detection for the Norwegian language will be presented.

4.2.1 The Shared Task OffensEval

OffensEval is a shared subtask of SemEval, an ongoing series of evaluations of systems working on sentiment analysis problems (Zampieri et al., 2019a). The evaluations gather groups of researchers working on shared tasks using benchmark datasets, enabling easier comparisons of models. For OffensEval, the task concerns detecting different types of offensive languages based on the OLID dataset, previously described in Section 3.2.2. OffensEval was first organized in 2019 as the field over the past years has gained significant attention, evidenced in the rapid growth of research publications (Arntzen, 2020). The three tasks of OffensEval are closely related to the OLID dataset's hierarchical annotation scheme and are defined as follows.

- **Subtask A**: Offensive language identification. Classify instances as offensive or not offensive language.
- **Subtask B**: Automatic categorization of offense types. Classify offensive language as targeted or untargeted.
- **Subtask C**: Offense target identification. Classify targeted instances as targeted towards an individual, group, or other.

OffensEval 2019 concerned these tasks for the English language, whereas the most recent OffensEval 2020 expanded the task to include Subtask A for Arabic, Danish, Greek, and Turkish. As English is a high-resource language, the research represents several state-of-the-art approaches for abusive language detection. Danish is relevant to Norwegian abusive language detection due to the language similarities. Based on this, this section first describes OffensEval 2019, followed by the English and Danish track of OffensEval 2020.

OffensEval 2019

For OffensEval 2019, 800 teams signed up, whereas 115 submitted reports (Zampieri et al., 2019a). Zampieri et al. (2019b) created several baselines, with a CNN model holding the best results. The best baseline was surpassed with only 2.9 macro F1 points by the best performing model on Subtask A (Zampieri et al., 2020). The majority of the teams used deep learning approaches for this subtask, whereas 8% used the recently developed, Transformer-based BERT model (Devlin et al., 2019). The top five submissions for the same subtask used BERT models, demonstrating the power of introducing Transformer-based models to offensive language detection. The different approaches using BERT models mainly varied in pre-processing techniques and model configurations.

In contrast to Subtask A, where BERT models dominated, the top models for Subtask B were mainly ensembles. Several top teams used approaches to combine deep learning models, including BERT, with other non-neural models. Seganti et al. (2019) achieved the overall best macro F1 score across all subtasks with their ensemble model, which also included a model based on the Transformer, called Open AI GPT (Radford et al., 2019), and other traditional machine learners such as the SVM. They tested different data manipulation techniques to change the distribution of the dataset. In contrast, the traditional approaches performed better on the original unbalanced distribution, but the Transformer-based model performed better when trained on under- and oversampled data, balancing out the distributions.

The English Track of OffensEval 2020

OffensEval proved to still be popular the year after, with 528 teams signing up, 145 submitting their results, and 70 system papers created (Zampieri et al., 2020). Based on the promising results of Transformers in OffensEval 2019, such models gained even more attraction in OffensEval 2020.

The most popular models of OffensEval 2020 were all Transformer-based, with BERT and multilingual BERT (mBERT) among the most popular models. The mBERT will be further described in Section 4.3.1. Several of the other models used, such as ALBERT (Lan et al., 2019), ERNIE 2.0 (Sun et al., 2019), RoBERTa and XLM-RoBERTa (Conneau et al., 2019), are versions or extensions of the BERT model with different pre-processing, configurations, and sizes. The XLM-RoBERTa model is multilingual and trained on 100 different languages (including Arabic, Danish, Greek, and Turkish)[2]. Additionally to the emergence of several Transformer-based models, an extension of the OLID dataset, called SOLID, short for Semi-Supervised Offensive Language Identification Dataset, was also made available for the English track to improve model performance. The dataset contains over nine million English tweets labeled using the same annotation scheme as OLID, but in an unsupervised manner using machines (Rosenthal et al., 2020).

The top 20 submissions for Subtask A had less than one macro F1 point separating them, where the best team used an ALBERT ensemble model (Wiedemann et al., 2020). The top team for Subtask B and C, which also were second for Subtask A, used an ensemble of the XLM-RoBERTa base and large models for Subtask A. Furthermore, they used an ensemble of ALBERT and ERNIE 2.0 for Subtask B and C (Wang et al., 2020). Their models outperformed the second-best model for Subtask B with one macro F1 point and four macro F1 points for Subtask C. The results show that a wide range of Transformer-based models for the English language has been utilized to detect abusive languages and seems to have outperformed the more traditional approaches, which were more present among the top results for OffensEval 2019.

The Danish Track of OffensEval 2020

As described in Section 3.2.2, Sigurbergsson and Derczynski (2020) collected the first available Danish abusive language dataset, which was used in the Danish track of OffensEval 2020 working on Subtask A, offensive language identification. Sigurbergsson and Derczynski tested several different approaches on the dataset and achieved a macro F1 score of 0.699 using a non-neural linear approach for offensive language detection.

[2]The details of these architectures are left out as they are not yet available for the Norwegian language, which is the main focus of this thesis.

This result was later surpassed by a majority of the teams participating in OffensEval 2020.

A total of 72 teams registered to participate in this track, whereas 39 made official submissions (Zampieri et al., 2020). The best team for the Danish track detecting offensive language used the Nordic BERT model, which implements the standard BERT architecture customized for several Nordic languages. They used the version customized for Danish (Pàmies et al., 2020). After experimenting with different pre-processing techniques and finding that the BERT model performed well with almost no pre-processing, Pàmies et al. did little pre-processing except for reducing the orthographic lengthening to a maximum of two repeated characters and removing irrelevant punctuation marks. Their results surpassed the second-best team with one macro F1 point resulting in a macro F1 score of 0.812. The second-best team was the same as held the top results for Subtask B and C, and second-best results for Subtask A, for the English track, using the XLM-RoBERTa ensemble, as described in the previous section. This demonstrates how a successful approach for one language can transfer to others.

4.2.2 Domain-specific Abusive Language BERT Models

The advantage of using Transformer-based approaches pre-trained on large corpora, such as BERT, is that the model knows the meaning of terms and what context they occur in before the fine-tuning process (Devlin et al., 2019). Unlike traditional neural approaches, the model does not have to learn language during the fine-tuning process thoroughly, but only what separates the classes to be predicted. However, in cases where the fine-tuning dataset is represented by "non-standard" language, language different from what the BERT model is pre-trained on, problems may arise. An example is social media platforms with many domain-specific expressions used explicitly on these platforms. The meaning of such expressions may therefore be unknown for BERT models trained on general language understanding. Another challenge is grammar, as the BERT model is mainly pre-trained on pre-processed text such as what appears in books and newspapers. When classifying unprocessed user-generated data, jargon, grammar mistakes, and typos may appear. As input sequences of terms are represented as numerical word embeddings, terms with almost the same syntax will hold entirely different values. Therefore, the model may not be able to associate wrongly spelled terms with their correct representation unless it has been trained to do so.

Several different pre-processing techniques have traditionally been used to clean text, requiring extensive work. Another option to reduce the pre-processing needs is to continue the model's pre-training phase on domain-specific data, such as social media data. Caselli et al. (2020) introduced HateBERT, which is a BERT model pre-trained for abusive language detection for English. The corpus on which HateBERT is pre-trained is called RAL-E: Reddit Abusive Language English dataset. RAL-E is composed of 71.1 million tokens of user-created posts from communities at the social media platform Reddit that were banned due to being offensive, abusive, or hateful (Caselli et al., 2020). The model is customized for understanding social media data and the polarity and language jargon present in abusive languages. Caselli et al. showed that HateBERT outperformed

BERT for three benchmark abusive language datasets, including the OLID dataset used for OffensEval. However, the performance improvement was limited to around one to two F1 points for most tasks, which is not a large rise when considering the extensive work required to create HateBERT. Furthermore, re-training a model on domain-specific corpora requires a vast amount of domain-specific data and is computationally expensive (Devlin et al., 2019; Kummervold et al., 2021; Kutuzov et al., 2021).

Moreover, the technique has not always proven to be successful. Isaksen (2019) further pre-trained the general BERT model on unlabelled domain-specific data from several abusive language datasets, resulting in a file of nearly 170,000 lines (Isaksen, 2019). The further pre-trained model did not reflect any notable improvement to the general BERT model. Isaksen noted a flaw in a portion of the pre-training corpus composed of single sentences, which might have affected how the model learns language as BERT is trained on next sentence prediction. However, another reason may be that the pre-training corpora was too small for the model to learn high-quality representations of abusive language.

4.2.3 Norwegian Abusive Language Detection

The research available on Norwegian abusive language detection is currently limited to the research performed by Svanes and Gunstad (2020) and Jensen (2020). Their work collecting a Norwegian abusive language dataset was previously described in Section 3.2.1. Besides their work, there are other theses working on related topics in a parallel effort to the current thesis.

Mathias Wahl and Sondre Sjåstad at the Norwegian University of Science and Technology (NTNU) are writing a Master's thesis on detecting hateful utterances for the Norwegian language using an anomaly detection approach on the dataset by Svanes, Gunstad and Jensen. Furthermore, Jonas Nordhaug Myhre and Joakim Warholm at the Arctic University of Norway (UiT) are leading an initiative collecting a dataset on unhealthy conversations, where one of the attributes denoted is whether the language is hostile or not. Based on this dataset, Warholm is writing a Master's thesis on detecting unhealthy conversations using Transformer-based approaches. However, due to the results of these initiatives not being published yet, this section only describes the currently available research limited to Svanes and Gunstad (2020) and Jensen (2020) and the approaches they used to detect abusive Norwegian language.

Svanes and Gunstad trained various traditional linear and deep learning models to classify the five classes in their dataset ranging from neutral to hateful with different degrees of abusiveness. Additionally, the authors considered testing the Nordic BERT model, the same model as mentioned in Section 4.2.1 but customized for Norwegian. However, they had problems implementing the model for the Norwegian language and noted that several others had reported similar issues, resulting in them not proceeding forward with it. Moreover, Svanes and Gunstad used two different classification approaches for all models, where the first approach trained the models to classify all five classes in one step. The second approach divided the classification into two steps. The first step classified all instances as neutral or non-neutral, and the second step classified the correctly predicted non-neutral instances as the four categories within that group. All models struggled to discriminate between the different classes and were biased towards the neutral class due to the highly imbalanced dataset.

Svanes and Gunstad used accuracy score as the main measure to report model performance. They reported a top accuracy of 0.833 using the one-step approach and accuracies of 0.796 and 0.568 of the first and second steps using the two-step approach, respectively. Based on these measures, Svanes and Gunstad reported a Logistic Regression model as their best approach in the one-step approach and the Support Vector Machine for the two-step approach. A problem with their reported results is that the accuracy measure is biased towards the majority class of neutral instances. When examining their confusion matrices, it is evident that the Logistic Regression model cannot detect hateful and offensive instances. The model did not detect any hateful instances, whereas only one moderately hateful instance was detected. The scores were also deficient for the other abusive categories. When considering the classwise results of their confusion matrices, a Support Vector Machine outperformed the other models in both approaches[3].

As opposed to Svanes and Gunstad, Jensen tested an anomaly detection approach using a CNN model with word embeddings using the same dataset. The hateful and moderately hateful classes were merged into one class, resulting in a hate speech detection problem. Likewise to Svanes and Gunstad the results showed that the model struggled to detect the hateful instances and that it was biased towards neutral class. The best-reported accuracy was 0.93, whereas the best macro F1 score achieved was 0.515 based on their presented classwise F1 scores. Although the accuracy is high, the value is biased towards the neutral class, as Jensen reports that only 0.8% of the instances classified as hateful were correctly classified, yielding a severe low precision score. Furthermore, only 41% of the true hateful instances were detected as hateful, using the anomaly detection approach. Jensen 's overall results are only slightly above the majority class baseline, i.e., what can be achieved by labeling all instances as neutral.

4.3 The Transformer-based BERT Model

The emergence of Transformer-based models such as BERT has revolutionized models' ability to learn contextual representations in language. As Norwegian is transferring from a lower to a higher resource language, several state-of-the-art models previously only available for English and other languages have also recently been developed for the Norwegian language. Section 4.3.1 describes the available Norwegian BERT models, whereas Section 4.3.2 describes some challenges with the fine-tuning phase of these models. The concept of Transformers and BERT are previously described in Section 2.4.

[3]As accuracy is considered an inappropriate measure when evaluating model performance on a highly unbalanced dataset, the results by Svanes and Gunstad (2020) are re-calculated based on more representative measures in Chapter 7 for compatibility to the reported results from the experiments of this thesis.

4.3.1 Transformer-based BERT Models for Norwegian

The first Transformer-based model providing support for the Norwegian language is the multilingual BERT (mBERT) model. The mBERT was developed as an extension of the original BERT model (Devlin et al., 2019) trained on the entire Wikipedia[4] dump for each of the top 100 languages with the largest Wikipedia sites. This includes Norwegian Bokmål, Norwegian Nynorsk, Danish and Swedish. The Norwegian Bokmål version of this corpus corresponds to about 0.9 GB of data, whereas the Norwegian Nynorsk corresponds to about 0.2 GB (Kummervold et al., 2021).

Furthermore, the mBERT model has the same model configurations as BERT_{base}, and both the cased and uncased versions of the models were initially released. However, the uncased version has normalization issues for non-Latin alphabets, which include the Norwegian alphabet. During the normalization process, the tokenizer handles accent markers in the same step as it converts characters to lowercase, resulting in removing characters with a critical role in several languages. For instance, it converts letters such as *å* into *a* which in many cases will change a term's meaning. The word *mål* (translates to *goal* in English) will, for example, convert to *mal* (translates to *template* in English). The cased model does not have the same problems and is therefore recommended for use. Another limitation of mBERT is that the corpus is mainly objective and descriptive, lacking subjective and informal language, such as what is typically found in social media forums.

Besides the multilingual BERT model, the Danish company, BotXO, developed several monolingual BERT models for the Nordic languages, with the common name Nordic BERT (BotXO Ltd., 2021). The Norwegian version of this model is trained on 4.5 GB Norwegian text. Unfortunately, several researchers state that they were unable to reproduce this model for the Norwegian language due to a broken model vocabulary making the model unavailable for use (Svanes and Gunstad, 2020; Kutuzov et al., 2021; Kummervold et al., 2021).

Recently, two different initiatives have been working on monolingual Norwegian Transformer-based models. The Language Technology Group at the University of Oslo organized the Norwegian Large-scale Language Models (NorLM) initiative, introducing the Norwegian BERT (NorBERT) model, along with a viable alternative to Transformer-based models, namely the pre-trained neural architecture, NorELMo (Kutuzov et al., 2021). Kutuzov et al. showed that NorBERT outperformed NorELMo for all their NLP tasks but notes that the computational resources required to adapt NorELMo to a downstream task can be an order of magnitude less. However, as this thesis is concerned with BERT models, only NorBERT will be described here. NorBERT is trained on the corpus displayed in Table 4.3 which incorporates the Bokmål and Nynorsk Wikipedias and Norsk aviskorpus (Språkbanken, 2019) which is an extensive collection of Norwegian news texts, producing a greater foundation to understand the Norwegian language in several different contexts. The model has the same configurations as the $\text{BERT}_{base,cased}$.

Simultaneously to the development of NorBERT, the AI Lab of the National Library

[4]`www.wikipedia.org` is a free online encyclopedia, created and edited by volunteers.

Sources	**Period**	**Million words**
Bokmål Wikipedia	1998-2019	160
Norsk aviskorpus (Norwegian newspapers)	1998-2019	1,700
Nynorsk Wikipedia	2001-2019	40
Total		**1,900**

Table 4.3: The corpus of which NorBERT is pre-trained on (Kutuzov et al., 2021).

of Norway introduced the Nasjonabiblioteket BERT (NB-BERT) (Kummervold et al., 2021) (translates to National Library BERT), which is the largest available Norwegian Transformer-based model in terms of pre-training corpus. The model is available following both the BERT$_{base,cased}$ and the BERT$_{large,uncased}$ configurations, whereas the latter was released April 29th, 2021, concurrently with this work. The model is pre-trained on the Colossal Norwegian Corpus displayed in Table 4.4 which is about ten times larger than the corpus of NorBERT (Kummervold et al., 2021; Kutuzov et al., 2021). The sources are essentially pre-processed texts published in Norwegian books, newspapers, Wikipedia, and other formal legal and government reports, representing the Norwegian language historically and socially over the past 200 years (Kummervold et al., 2021).

As opposed to the corpora of mBERT and NorBERT, this corpus is also composed of some web crawled data, namely the Common Crawl OSCAR, that to some extent contains user-generated data crawled from the web. This type of data is more similar to what can be found in social media platforms, which may help the model learn language closer to the datasets used in the experiments of this thesis. However, some cleaning was initially performed on this part of the corpus, which includes removing all pages containing any term from the *List of Dirty, Naughty, Obscene, and Otherwise Bad Words* (Austegard, 2019). The removal of pages containing offensive terms is unfortunate for abusive language detection and may have removed some relevant data. Nevertheless, the list contains only forty terms and represents a smaller subset of bad terms used in social media contexts. Moreover, most of the sources from The Colossal Norwegian Corpus are collected using Optical Character Recognition (OCR)[5], which may have introduced some noise to the corpus. Kummervold et al. estimate that the resulting pre-training corpus is composed of 83% Norwegian bokmål, 12% Nynorsk, 4% English, and the remaining 1% is a combination of Danish, Swedish, and Sami and some traces from other languages.

Kummervold et al. and Kutuzov et al. reported the results of mBERT, NorBERT and NB-BERT for several NLP tasks. The results showed that NB-BERT significantly outperformed the other models for most tasks, with NorBERT being the second-best model. Kummervold et al. also showed that NB-BERT outperformed mBERT and NorBERT for tasks across Danish, Swedish, and English, demonstrating a decent multilingual understanding. Although mBERT has proven to perform well across several

[5]OCR uses technology to read and separate printed or handwritten text characters inside digital images of physical documents.

Sources	**Period**	**Million words**	**Text (GB)**
Books (OCR)	1814-2020	11,820	69.0
Newspapers Scans (OCR)	2015-2020	3,350	20.0
Parliament Documents[a] (OCR)	1814-2014	809	0.2
Common Crawl OSCAR	1991-2020	799	0.3
Online Bokmål Newspapers	1998-2019	678	0.6
Periodicals (OCR)	2010-2020	317	4.9
Newspapers Microfilms (OCR)	1961,1971,1981,1998-2007	292	5.1
Bokmål Wikipedia	1998-2019	140	1.8
Public Reports[b] (OCR)	1814-2020	91	4.0
Legal Collections[c]	1814-2004	63	1.9
Online Nynorsk Newspapers	1998-2019	47	0.4
Nynorsk Wikipedia	2001-2019	32	0.9
Total		**18,438**	**109.1**

[a] Stortingsforhandlingen. [b] Evalueringsrapporter. [c] Lovdata CD/DVD

Table 4.4: The composition of the Colossal Norwegian Corpus on which the NB-BERT model is pre-trained. The table is adapted from Kummervold et al. (2021) with permission from Per Egil Kummervold.

NLP tasks for multiple languages, the results from testing the model on NLP tasks for Norwegian Nynorsk and Bokmål indicates that the amount of Norwegian data may be inadequate for the model to learn more complex Norwegian representations (Pires et al., 2019; Kummervold et al., 2021; Kutuzov et al., 2021).

4.3.2 Training Instability of Fine-tuned Transformer-based Models

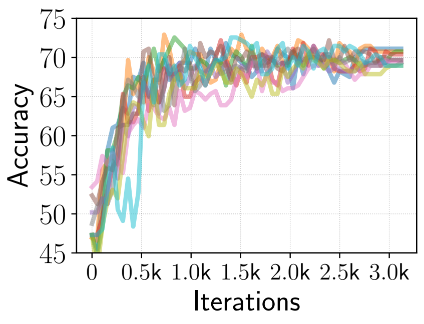
As previously described in Sections 4.2, Transformer-based models have showed a solid empirical performance surpassing traditional neural methods for several NLP tasks. However, fine-tuned Transformer-based approaches have proven to exhibit a significant training instability; when fine-tuning the model multiple times on the same dataset with random restarts, the standard deviation of the fine-tuning accuracy is large (Devlin et al., 2019; Li et al., 2020; Dodge et al., 2020; Liu et al., 2020; Mosbach et al., 2021).

As the Transformer-based architectures are more complex than traditional approaches, the origin of the instability is not apparent, and several different hypotheses have emerged. Devlin et al. (2019) claimed that the instability occurred when using smaller datasets. Based on this, they choose to run several random restarts and selected the best model on the development set. Devlin et al. also decided to report their results for several

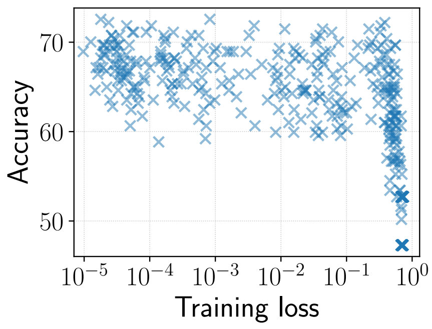
NLP tasks over five random restarts to get more representative results, less biased by the instability of each training. The same approach has also been adopted by other literature when reporting results of BERT models (Kutuzov et al., 2021).

Dodge et al. (2020) identified two factors causing instability when examining training and validation scores of BERT over 2,100 runs on four benchmark datasets. The two factors were influenced by choice of random restart state: weight initialization and training data order. Correspondingly to Devlin et al., Dodge et al. suggested small datasets to be a potential reason for the observed instability. In contrast, Lee et al. (2020) suggested catastrophic forgetting, which is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information. However, Mosbach et al. (2021) showed that both hypotheses fail to explain the observed instability.

Mosbach et al. suggested that the instability originates from optimization difficulties leading to vanishing gradients, a problem previously described in Section 2.4.1, additionally to the occurrence of generalization differences later in training. The differences in generalization is illustrated in Figure 4.1, where Figure 4.1a shows the development accuracy for 10 successful runs on 20 epochs, and Figure 4.1b shows the development accuracy versus training loss for 450 models using different hyperparameters.



(a) Development set accuracy over training



(b) Generalization performance vs. training loss

Figure 4.1: Development set accuracy of BERT for multiple fine-tuning runs. Figure (a) shows the models trained with 10 different random states, whereas Figure (b) shows the models at the end of training using different random states and hyperparameters. The figures are are adopted from Mosbach et al. (2021) and reprinted with permission from Marius Mosbach.

Based on this, Mosbach et al. proposed a simple strategy of using lower learning rates to avoid vanishing gradients early in training and increasing the number of iterations considerably to reach almost zero training loss, particularly for smaller datasets. They noted that this increases stability requiring less fine-tuning runs for reliable results. Furthermore, Mosbach et al. reported that the increasing number of iterations did not lead to a loss due to overfitting in their experiments and that the approach outperformed previous work.

5 Data Analysis

When using supervised learning to detect abusive languages, one of the main tasks is to find the boundaries separating the different classes. Previous literature has shown that several different features in the dataset can represent this boundary to some extent, ranging from word and hashtag co-occurrences to more contextual features such as word embeddings (Davidson et al., 2017; Zampieri et al., 2019a, 2020). This chapter is dedicated to building a better understanding of the datasets used in this thesis, identifying what separates the neutral class from other abusive classes, and exploring similarities and differences across the datasets that may affect how a model generalizes across them.

The Norwegian dataset previously described in Section 3.2.1 is the main dataset of this thesis and will also be the main focus of this chapter. In one of the experiments of the following chapters, the Danish dataset, described in Section 3.2.2 is used as an extension to the Norwegian dataset. For the model to generalize across different abusive language datasets, the boundary separating the classes should generalize across both datasets (as previously described in Section 3.1.1). Based on this, Section 5.1 describes a content analysis of the classes in the Norwegian dataset and how this compares to the classes of the Danish dataset.

Furthermore, Section 5.2 describes a quantitative and qualitative annotation analysis of a subset of the Norwegian dataset. This analysis aims to understand better the strengths and weaknesses of the Norwegian dataset in light of the challenges of abusive language datasets described in Section 3.1. Finally, Section 5.3 presents a refinement of the dataset's class generalization based on the findings from this analysis, which is used in the experiments of the following chapters.

5.1 Content Analysis

A term occurrence analysis is performed to understand the datasets' content and how it differs across classes. Figure 5.1 and Figure 5.2 visualize word clouds of the neutral and abusive document classes in the Norwegian dataset, respectively. A word cloud is a cluster of terms where larger and bolder terms appear more frequently in the documents. For all word clouds in this thesis, stop words are removed to obtain the more discriminative terms for each class.

The apparent first observation when comparing the neutral document class in Figure 5.1 to the abusive document classes in Figure 5.2, is that as the abusiveness of the documents increases, so does the occurrence of Islam-related topics. This indicates that the documents within the abusive categories have more topics related to Islam than the less abusive categories. Considering the definitions used by Svanes and Gunstad (2020)

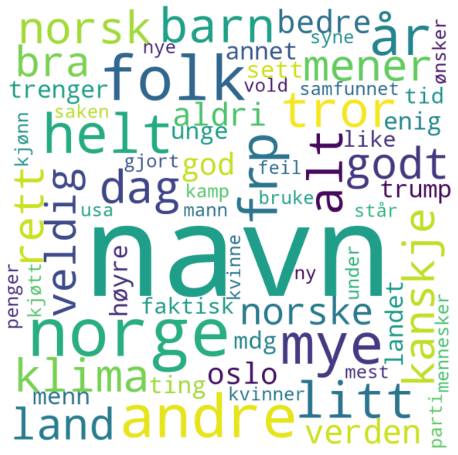


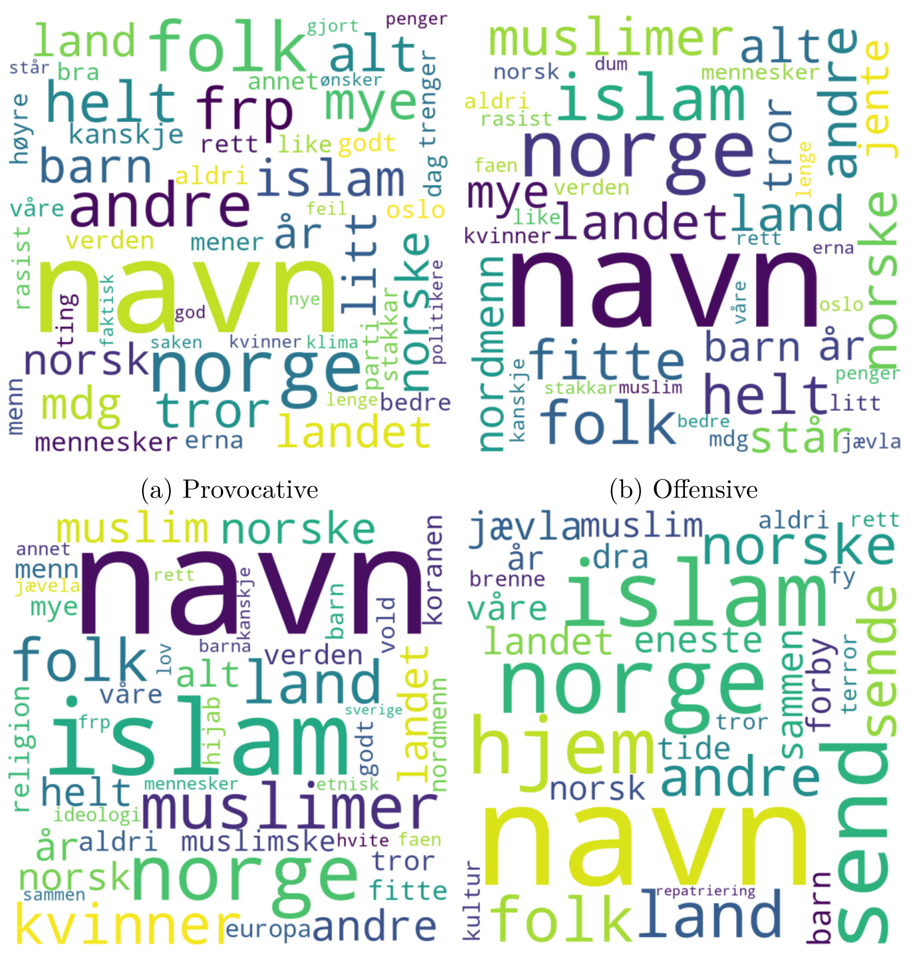
Figure 5.1: Wordcloud visualizing high frequent terms in the neutral class of the Norwegian dataset after removing stop words. Larger words represent words occurring more often.

and Jensen (2020) when annotating the dataset, it makes sense that the hateful classes contain more material directed at groups or individuals based on religion. However, the provocative and offensive documents also have a higher frequency of terms related to Islam than the neutral class, indicating that this is a central topic of discussion across the data collection sources.

Furthermore, one can observe that the terms in the neutral word cloud in Figure 5.1 have more diverse topics than the other classes, ranging from climate and children to gender and meat. In contrast, the provocative class (Figure 5.2a) seems to have a higher frequency of political-related terms such as *FrP*, *Høyre*, and *MDG*, which are Norwegian political parties, and *Erna*, the name of the prime minister of Norway. At the same time, the provocative class has fewer offensive terms than the more abusive classes, which correspond well to the definitions in Section 3.2.1. The offensive class seems to have the highest frequency of offensive terms, although there is also a frequent occurrence in the moderately hateful and hateful classes.

When considering the unbalanced nature of the Norwegian dataset and the few abusive instances, it would have been desirable to collect more documents. However, because of the extensive amount of work to collect and annotate documents, it was decided to extend the Norwegian dataset with another abusive language dataset, the Danish dataset by Sigurbergsson and Derczynski (2020). As described in Section 3.2.2 the dataset was chosen as the most similar available dataset to the Norwegian dataset. The word clouds representing the non-offensive, offensive, and targeted offensive documents are displayed in Figure 5.3.

The instant observation from the Danish word clouds in Figure 5.3 is that they are quite different from the Norwegian ones, both in terms of language and content. For instance, Islam and political parties that were central subjects in the Norwegian dataset do not seem to be frequent subjects of this dataset. Furthermore, the Danish dataset



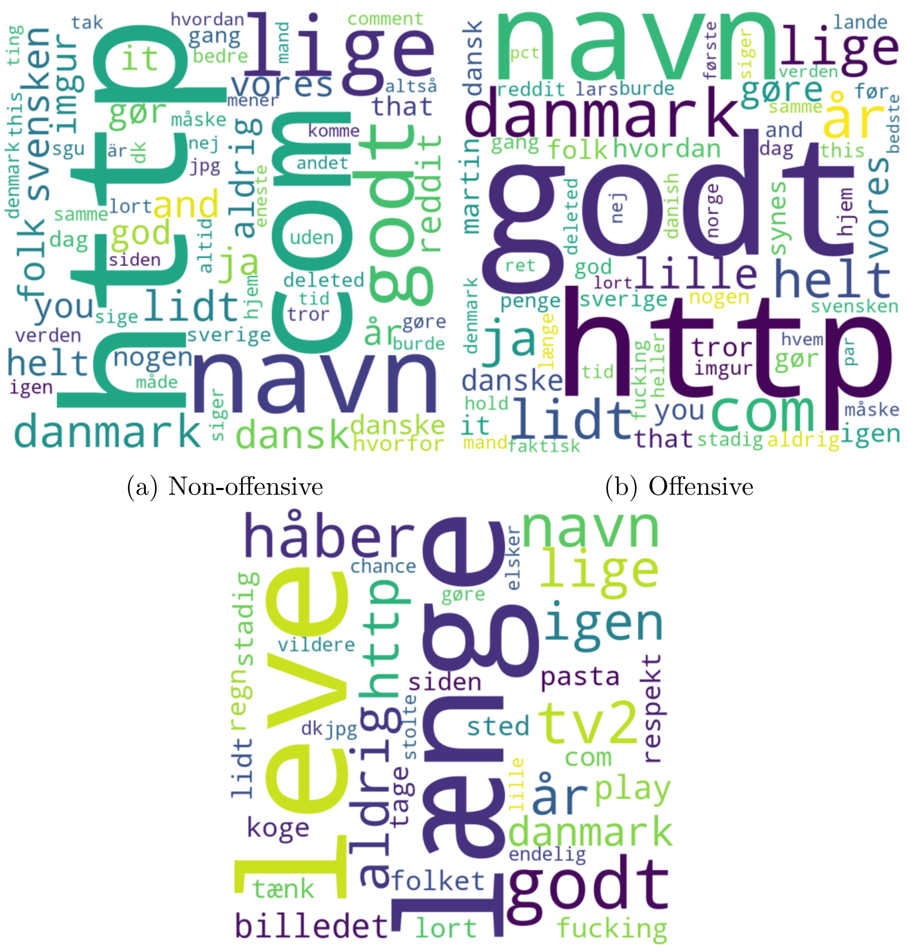
(a) Provocative (b) Offensive

(c) Moderately Hateful (d) Hateful

Figure 5.2: Word clouds visualizing high frequent terms in each of the five categories of the Norwegian dataset after removing stop words. Larger words represent words occurring more often.

contains more English terms and web-related terms such as *http* and *com*, which represent web links.

These differences may be due to the datasets being collected from different sources. The Norwegian dataset is collected mainly from immigrant-critical news, Twitter, and Facebook pages, while the Danish dataset is collected from more general Reddit pages. These sources may contain discussions of different topics by different people affecting the specific language used. Additionally, the culture affects how people express themselves, both when it comes to heated subjects and the selected terms and how English terms are incorporated into the language. Despite the syntactic similarities discussed in Section 2.1, it may be that the cultural differences between Norwegian and Danish affect how the abusive languages are expressed, leading to differences between the datasets.



(a) Non-offensive (b) Offensive

(c) Targeted insults

Figure 5.3: Word clouds visualizing high frequent terms in the non-offensive, offensive and targeted offensive documents of the Danish dataset. Larger words represent words occurring more often.

Besides the observed differences, there are also similarities between the word clouds of the two languages. For instance, there is an increasing occurrence of abusive and curse terms as the abusiveness of the document classes increases for both word clouds. Furthermore, one can observe that the frequencies of the abusive terms within the offensive group in Figure 5.3b are lower than for the Norwegian offensive class in Figure 5.2b. This is likely because the Danish offensive class includes profanities that are more similar to the provocative Norwegian class by definition. The Danish offensive class in Figure 5.3b also seems to correspond better to the provocative Norwegian class in Figure 5.2a.

Furthermore, *Danmark* (translating to Denmark) is consistently mentioned across the Danish documents, whereas *Norge* (translating to Norway) is consistently mentioned across the Norwegian documents. This indicates that many of the documents concern the country of origin of the pages they are collected. All Danish and Norwegian word clouds also mention the *navn* tag, which represents user mentions, a common feature in social media posts.

Although this analysis has shown considerable differences between the term occurrences of the two datasets, it is worth remembering that languages are composed of more than word occurrences. For instance, the context in which terms occur that the Transformer-based models previously discussed can help capture.

5.2 Annotation Analysis

A subset of the Norwegian dataset was annotated and analyzed to get an idea of the dataset's quality and otherwise possible ambiguity across classes beyond what is described in Section 3.2.1. The annotations were conducted in collaboration with the two other Master's students, Mathias Wahl and Sondre Grav Skjåstad, also working on abusive language detection with the same dataset. Section 5.2.1 describes the annotation experiment and the resulting inter-annotation agreements calculated from two subsets of the Norwegian dataset, followed by a description of some observations done during the process, in Section 5.2.2.

5.2.1 Quantitative Analysis of the Annotator Agreement

Due to the unbalanced class distributions of the dataset, the annotation experiment was divided into two separate steps annotating the following subsets.

1. **Representative subset:** Consists of 200 documents sampled using stratified sampling (as described in Section 2.3.3) to ensure a similar distribution to the original dataset with regards to the five classes.

2. **Abusive subset:** Consists of 100 documents with 25 random sampled documents from each of the four abusive categories.

It was selected to sample 200 instances for the representative subset as it was within our time constraints to annotate and considered representative of the dataset's distribution. The purpose of this subset was to represent a subset with similar distribution to the original dataset. However, as the number of instances from the abusive categories was small and not necessarily representative of each class, it was decided to sample the second subset of 100 documents from the four abusive categories, named the abusive subset. The abusive subset ensured that all categories were represented in the analysis.

Following Svanes and Gunstad, all annotators were familiarized with the guidelines (found in Appendix 2), and some discussion around the definitions and guidelines were done prior to the annotation process. Each annotator annotated all documents in the two subsets individually without discussing the annotations with others. Finally, the inter-annotator agreement was calculated for each subset using Fleiss' kappa and percentage agreement to measure the general agreement across classes and the separate agreement for each class, respectively. The approach and calculations follow Svanes and Gunstad (2020) which makes comparisons of results easier. The annotation agreement of the three annotators for the representative subset is displayed in Table 5.1.

	Neutral	**Prov.**	**Offensive**	**Mod. Hateful**	**Hateful**	**Total**
Full agreement	72%	15%	0%	0%	0%	74%
Majority vote	88%	51%	25%	0%	0%	99%

Table 5.1: The annotation agreement of three annotators when annotating 200 samples from the Norwegian dataset sampled using stratified sampling for the five categories.

The agreement is highest for the neutral class, although when comparing to the results of Svanes and Gunstad (2020) as previously displayed in Table 3.2 in Section 3.2.1, the agreement is significantly lower. This is also reflected in the total agreement, which is 15% lower than Svanes and Gunstad when considering the full agreement. Two reasons for this may be that the sample contains other variations in the documents than the subset annotated by Svanes and Gunstad, or that the annotators in this experiment agreed less on the documents due to aligning their subjective view when annotating.

Furthermore, the agreement of several of the abusive document classes is equal to zero, which indicates disagreement for these classes. However, as the sample sizes of the abusive document classes in this subset are small, this agreement is likely, not representative of the dataset but gives an idea of existing ambiguity in the dataset. The general agreement across classes for the representative subset using Fleiss' kappa was equal to 0.3591, slightly lower than Svanes and Gunstad, but still reflects a fair agreement.

The same calculations were performed on the abusive subset to investigate the abusive classes further. The annotation agreement of three annotators on the abusive subset is displayed in Table 5.1.

	Neutral	**Prov.**	**Offensive**	**Mod. Hateful**	**Hateful**	**Total**
Full agreement	0%	12%	9%	22%	13%	13%
Majority vote	22%	43%	42%	47%	52%	84%

Table 5.2: The annotation agreement of three annotators when annotating 100 samples from the Norwegian dataset sampling 25 random samples from each of the four abusive categories.

As can be observed from the table, the agreement is generally low for all classes, particularly for the neutral class, which is because the abusive subset was sampled from the abusive labeled classes of the original dataset. The majority vote for the neutral class of 22%, therefore, reflects that some documents were labeled neutral in the abusive subsample although none of them were originally labeled as neutral by Svanes and Gunstad and Jensen. The Fleiss' kappa score for the abusive subset was calculated to 0.2694, corresponding to the lower range of fair agreement. The value reflects that the annotators found it difficult to agree upon the class labels of the abusive categories.

Furthermore, there were some particularly repetitive patterns observed during the annotation process. When comparing the annotations of this experiment to the original labels, the provocative and offensive categories were often confused, and likewise between the hateful and moderately hateful groups. This experiment's annotations also annotated documents less abusive than what appeared in the original labels. This is likely due to the discussions of the definitions and guidelines beforehand affecting the subjective perceptions of the annotators. Despite the standard guidelines, discussions between annotators will affect how annotators perceive the definitions. The observations are interesting when considering a refinement of the class generalizations, which will be further described in Section 5.3.

Overall, the observations from this experiment confirm that annotating abusive languages, in particular, is a challenging task with a substantial amount of disagreement between annotators. Furthermore, using a fine-grained annotation scheme when annotating abusive language may introduce more vagueness to the dataset, as seen in the analysis of the inter-annotator agreement in the dataset.

5.2.2 Qualitative Analysis of Annotations

Besides the inter-annotator calculations performed in the previous section, an analysis of the documents with substantial disagreement in the dataset was performed to get an idea of what documents are more challenging to categorize. Some of the difficulties encountered during annotation were observed to correspond well to the general problems for abusive language datasets as previously described in Section 3.1.1.

The first example document listed below was annotated as neutral by two of the three annotators and as offensive by the last one.

Norwegian	er du **dyselektiker**?
English	are you **dyselexic**?

As the annotations range between two different classification levels, there is a conflict. A lack of context to emphasize the intention of the message explained the neutral label. However, when analyzing the document and the misspelling of the term *dyslexic*, the misspelling can be perceived as a technique to amplify an offensive message with a condescending tone. In this case, both neutral and offensive may seem reasonable depending on the subjective interpretation of whether the misspelled word signals a message in itself or just an innocent grammar mistake. The example demonstrates some of the difficulties encountered when annotating documents with the lack of further context and substantiates that more context is necessary for reliable labels, as previously emphasized as a general problem for abusive language datasets in Section 3.1.1.

Another example listed below demonstrates challenges with lack of context and annotators aligning their perceptions into annotations. The document was labeled as provocative, offensive, and moderately hateful by the three annotators, whereas the original label by Svanes and Gunstad and Jensen was moderately hateful.

Norwegian	Navn Navn Navn Du mener så at Syria skal ta seg av norske terrorister. Det gidder de selvfølgelig ikke, på samme måte som vi sender utenlandske kriminelle hjem til der de kommer fra
English	Name Name Name So you mean that Syria should take care of Norwegian terrorists. Of course, they do not bother, in the same way that we send foreign criminals home to where they come from ?

The annotators agreed upon the aggressive undertone but disagreed on the message's meaning. The aggressive undertone substantiated the provocative label. In contrast, the moderately hateful label was built upon the fact that Syria is compared to Norway in a way that does not consider that Syria is in a completely different situation than Norway. The annotator classified the comment as *whataboutism*, which is a common technique to suppress other people's concerns or opinions. It is often used to perpetuate racist attitudes in society. This was strengthened by the expression "home to where they come from," which according to the annotator is a well-known condescending and racist expression in several languages.

Other annotators argued that the negative attitude is directed at the handling of criminals and not towards an individual or group based on the characteristics defined in Section 3.2.1. On the other side, it was discussed that the fact that the criminals are foreign and that the country Syria is mentioned makes the message targeted at a particular group based on ethnicity. This demonstrates complex cases of annotating language where the perception of the annotators is central in the resulting label.

Several cases had a substantial disagreement, whereas the main causes seemed to be a lack of context, which led annotators to align their perceptions and assumptions when deciding on labels. It was also observed that the hateful documents with offensive terms had a higher agreement than the hateful documents without such terms. These observations are consistent with the challenges previously described in Section 3.1. Moreover, a majority of the hateful and moderately hateful instances concerned Islam-related topics, which may indicate a bias in the dataset that should be further analyzed in the predictions of the models. The following section uses the findings from these analyses to propose a new class scheme for the dataset.

5.3 Final Dataset Representations

Based on this chapter's previous analyses and the fact that the Norwegian dataset is highly unbalanced with few abusive instances, this section proposes some changes to the dataset used in the experiments of this thesis. The new representations aim to reduce the vagueness in the dataset and increase the number of instances within one class. First, the changes made to the Norwegian dataset will be presented in Section 5.3.1, followed by a description in Section 5.3.2 of how the Danish dataset was converted into the same format.

5.3.1 The Norwegian Dataset Representation

As a result of a low inter-annotator agreement within the abusive categories and the observations done when annotating, it was decided to reduce the number of categories into three levels, namely *neutral*, *offensive* and *hateful*. The three levels include the following categories.

- **Neutral:** Includes the neutral documents from the Norwegian dataset, which are all documents that do not belong to any of the other categories. The neutral category is, therefore, unchanged from the original dataset.
- **Offensive:** Includes all documents except those belonging to the neutral class. A document is offensive if it originally belonged to the Norwegian dataset's provocative, offensive, moderately hateful, or hateful classes.
- **Hateful:** Includes the moderately hateful and hateful documents of the original dataset.

Note that the neutral class is mutually exclusive to the other classes. Still, a notable change to the labels is that the offensive class includes all documents that are not neutral, including the hateful documents. The choices of generalization were made based on what classes were considered the most similar based on definition, which was also confirmed by the inter-annotator agreement calculated in Section 5.2.1. Additionally, this generalization makes it possible to see abusive language detection as a two-step classification problem of first classifying documents into the groups neutral or offensive and then classifying them further into offensive or hateful, as further explained in the next chapter. The resulting distribution of the new categories is displayed in Table 5.3.

5.3.2 The Danish Dataset Representation

The classes of the Danish dataset were converted to the same levels as the Norwegian dataset to construct compatible datasets that can be combined later in the experiments. The following generalizations were made to convert the classes.

- **Neutral:** Includes the non-offensive group (NOT) from the Danish dataset.

#	Category	# documents	% of all
1	Neutral	34,085	82.8
2	Offensive	7,060	17.2
3	Hateful	760	1.9
1-3	**ALL**	**41,145**	**100**

Table 5.3: The distribution of labels after converting the five categories of the Norwegian dataset into the three categories neutral, offensive, and hateful.

- **Offensive:** Includes all documents that belong to the offensive group (OFF) of the Danish dataset.
- **Hateful:** Includes all documents that belong to the insults targeted at groups (GRP) or individuals (IND) in the Danish dataset.

The offensive definition used for the Danish dataset includes any form of untargeted profanities. This corresponds well to the generalization done for the Norwegian dataset, as the offensive category incorporates the provocative category. A significant distinction from the work of Zampieri et al. is that the hate speech definition of this generalization includes insults targeted at individuals, as opposed to the suggestion by Zampieri et al. which only includes insults targeted at a group. This generalization was made based on the definitions of the Norwegian dataset presented in Section 3.2.1 as Svanes and Gunstad includes hateful messages targeted at individuals and groups in their definition. Additionally, the generalization avoids the problems of overlapping groups previously described in Section 3.2.2 as the IND and GRP are merged. The distribution of labels of the Danish dataset after generalization is displayed in Table 5.4.

#	Category	# documents	% of all
1	Neutral	3,159	87.8
2	Offensive	441	12.3
3	Hateful	216	6
1-3	**ALL**	**3,600**	**100**

Table 5.4: The distribution of labels after converting the three levels of categories into the three categories neutral, offensive, and hateful.

6 Architecture and Experimental Setup

This chapter describes the architecture and experimental setup of the systems used to predict offensive and hateful language in the experiments of the next chapter. The same overall system structure is used, with some variations to the BERT model architecture and dataset combinations used to solve two abusive language detection problems defined in the next chapter. The first section presents some remarks throughout the literature review in Chapter 4 which motivates the choices of architecture and experimental setup described subsequently. Section 6.2 describes the BERT model architectures selected for the experiments of this thesis, followed by Section 6.3 which describes the experimental setup, ranging from data and pre-processing to model configurations.

6.1 Motivation

Based on the material covered in Sections 4.2 and 4.3, this section is dedicated to giving a brief overview of remarks and motivation for the following choices of architecture and experimental setup. Section 6.1.1 describes some takeaways from the literature review on approaches used for abusive language detection, whereas Section 6.1.2 identifies possible improvements for Norwegian abusive language detection.

6.1.1 State-of-the-art Transformer-based Architectures

The most recent state-of-the-art models in abusive language detection include implementations of Transformer-based approaches, such as BERT. An advantage of such models is that they have proven to require less textual pre-processing than other traditional approaches (Pàmies et al., 2020). However, the models' complexity is evident based on the large pre-training corpora and number of layers, creating vast network parameters. Due to this complexity, the models require extensive computational resources and take longer to train than traditional approaches (Devlin et al., 2019; Kummervold et al., 2021; Kutuzov et al., 2021).

Furthermore, Section 4.3.2 showed how the network complexity of such models poses some challenges with variance in the fine-tuning process. Previous literature (Devlin et al., 2019; Kutuzov et al., 2021; Mosbach et al., 2021) have dealt with these challenges by averaging results over several runs and adjusting hyperparameters to obtain a more stable training. Mosbach et al. proposed increasing the number of iterations significantly and shrinking the learning rate. Both these adjustments will increase the training time of the model significantly due to how these hyperparameters influence the training process (as described in Section 2.4). The technique will be tested to the best-performing models in an extensive hyperparameter search, including larger numbers of epochs and lower learning rates.

Caselli et al. (2020) showed how training a BERT model on domain-specific abusive language data can help improve model performance for this particular task. However, the improvement in results was only slight. Additionally, Isaksen (2019) tried a similar approach by continuing the pre-training of a BERT model, which did not significantly improve performance. Adopting these approaches would require doing the process from scratch for Norwegian data, including collecting an abusive language corpus for Norwegian, as the available datasets for this task are limited. One option could be to further train a model on general social media data, to cope with the grammar challenges of general BERT models, and otherwise social media jargons, described in Section 4.2.2. However, a vast corpus is needed for BERT models to learn complex language representations, which, combined with the computational resources necessary for training, requires extensive work. Considering the scope of this task and without a strong foundation of it leading to significantly increased model performance, it was selected not to prioritize this task in the experiments of this thesis.

6.1.2 Identifying Potential Improvements for Norwegian Abusive Language Detection

When comparing the limited work on abusive language detection for Norwegian to English and Danish abusive language detection, it is evident that there is much potential for improvements. The classification approach taken by Svanes and Gunstad (2020) predicted more abusive instances than what was achieved with the anomaly detection approach by Jensen (2020). Therefore, the approach taken in this thesis is more similar to the approach by Svanes and Gunstad. An apparent challenge encountered by both Svanes and Gunstad and Jensen is the highly unbalanced dataset with very few abusive instances. Neither experimented with dataset manipulation techniques such as undersampling or expanding the dataset to change the class distribution, which, as previously discussed, improved the performance of the Transformer-based approaches in OffensEval 2019.

Furthermore, the recent introduction of Transformer-based models for Norwegian has made it possible to experiment with state-of-the-art approaches for abusive language detection. Additionally to taking context into account, BERT models also handle out-of-vocabulary terms, which is particularly beneficial for agglutinative languages, such as the Norwegian language. The general language understanding of these models should help improve abusive language detection for Norwegian, based on related literature (Zampieri et al., 2019a, 2020; Pàmies et al., 2020), and will lay the basis together with the other research mentioned on choices in the following sections.

6.2 The BERT Model Architecture

All models used in this thesis are versions of the BERT model, previously described in Section 2.4.9. BERT is composed of a pre-trained architecture with a linear classification layer on top. Whereas the pre-trained BERT models can be downloaded and used directly, the linear classification layer must be fine-tuned for each downstream task. The versions of the BERT models used in this thesis are listed below, all supporting the Norwegian language.

- **mBERT:** Multilingual BERT (Devlin et al., 2019)
- **NorBERT:** Norwegian BERT (Kutuzov et al., 2021)
- **NB-BERT:** Nasjonalbiblioteket BERT (Kummervold et al., 2021)

All the above models and their corresponding pre-training corpora are previously described in Section 4.3.1. The models included in this thesis experiments are mBERT$_{base,cased}$, mBERT$_{base,uncased}$, NorBERT$_{base,cased}$, NB-BERT$_{base,cased}$, and NB-BERT$_{large,uncased}$. The overall architectures of these models are the same as for the general BERT model, and the overall system architecture implemented for each of the separate models is illustrated in Figure 6.1. The linear classification layer on top corresponds to a feed-forward neural network with one hidden layer.

This paragraph describes the process for one BERT model, which is standard for all the various BERT models mentioned. The BERT model is first fine-tuned jointly with the linear classification layer on the training dataset to learn how to predict offensive and hateful language. Then, for each input document, the encoded sequence is passed into the model, and the produced hidden output of the first position (that received the special `[CLS]`-token) is fed into the linear classification layer. The input size of the linear classification layer is the same as BERT's hidden output, whereas the output size corresponds to the number of classes to predict, which is two for both offensive and hateful language detection in this thesis. Finally, the logits from the classification layer are passed through the `np.argmax` function, which calculates the final labels.

A dropout layer is added between the output of the BERT model and the linear classification layer to prevent overfitting to the training set. The dropout probability is 0.1 and functions by randomly dropping 10% of the connections between nodes in the network by setting them equal to zero. The classification error is calculated using cross-entropy loss during training, which is minimized using an optimizer algorithm. For mBERT, the optimizer selected is ADAM, whereas both NB-BERT and NorBERT use the LAMB optimizer, both described in Sections 2.4.5 and 2.4.5, respectively.

6.3 Experimental Setup

This section describes the technical setup of the systems which is needed to reproduce the experiments of the next chapter. The overall system structure is illustrated in Figure

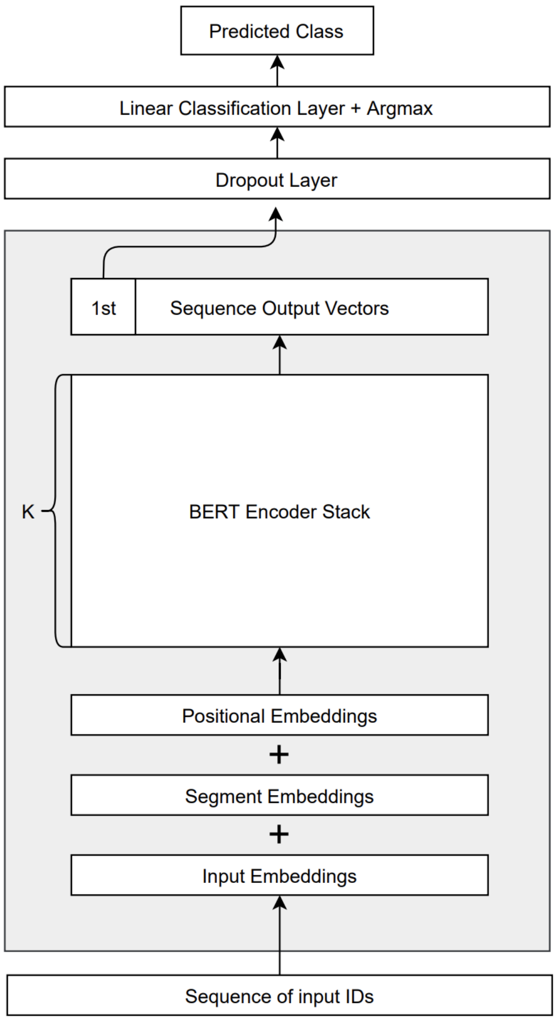


Figure 6.1: High level architecture of BERT with a linear classification layer on top used for single sequence classification tasks. K is the number of encoder-decoder blocks which is 12 for BERT base and 24 for BERT large.

6.2, and the different components will be described subsequently in the following sections. First, the data setup will be described, followed by pre-processing, model configurations, and optimization. Finally, the computational resources used to run experiments will be presented. The tools and libraries used are previously described in Section 2.5.

6.3.1 Datasets

The Norwegian dataset was provided upon request by Marie Andreassen Svanes as Comma-Separated Values (CSV) files, with document IDs, text, source, and corresponding labels.

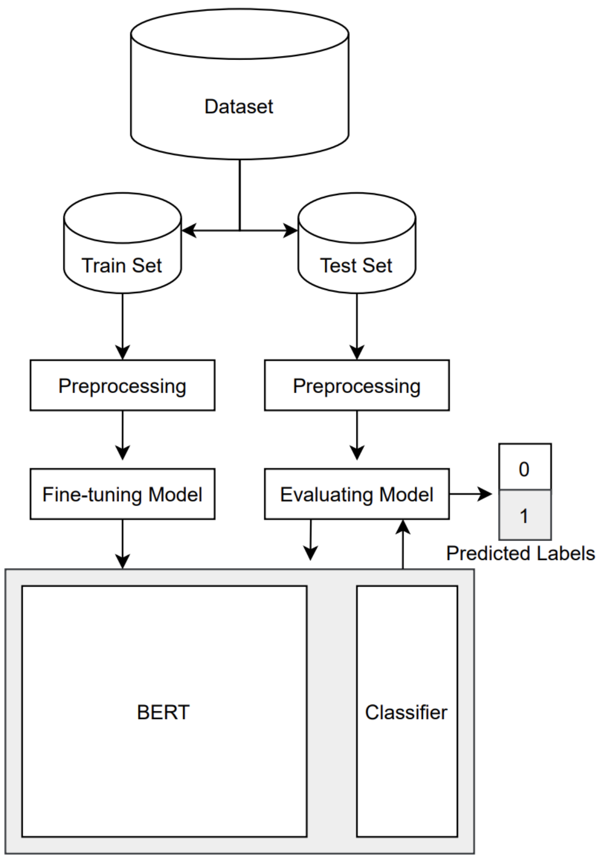


Figure 6.2: The overall system architecture used to detect offensive and hateful language in the experiments of this thesis.

The Danish dataset was openly available as an OpenDocument Spreadsheet (ODS) file[1] with document IDs and corresponding text, source, and labels for all categories. Some initial changes to the class labels are done to both datasets, as previously described in Section 5.3.

The Norwegian dataset was split using stratified sampling (described in Section 2.3.3) into a training and a test dataset with equal class distributions with regards to the five original classes described in Section 3.2.1. The training dataset constituted 70% of the complete dataset, whereas the remaining 30% was set aside for testing. The proportions were selected based on the highly unbalanced dataset to ensure that both test and training sets were representative of the original dataset variations and patterns. Additionally, the selected proportions correspond to Svanes and Gunstad (2020) which makes the results

[1] `https://figshare.com/articles/dataset/Danish_Hate_Speech_Abusive_Language_data/12220805`

more comparable. The splitting was performed using Sklearn's `train_test_split`[2] function with a random state of 200 for reproducibility.

The test dataset is consistent across all experiments, whereas some of the experiments modify the training dataset, further described in the next chapter's experimental plan. However, if not stated otherwise, the selected split and label distribution displayed in Table 6.1 is used for training and evaluation. Note that the hateful comments are also incorporated in the offensive comments.

Dataset	**# documents**	**% of all**	**# of offensive**	**# of hateful**
Train dataset	28,801	30	4,984	553
Test dataset	12,344	70	2,076	207
ALL	**41,145**	**100**	**7,060**	**760**

Table 6.1: The distribution of documents in the training and test datasets used in this thesis experiments, and the corresponding number of instances belonging to the offensive and hateful classes for each dataset.

6.3.2 Pre-processing

The process of pre-processing transformed the document texts into interpretable and predictable forms. The advantage of using the BERT model is that it has proven to perform well with less textual pre-processing compared to other traditional approaches that are more dependent on term occurrences (Zampieri et al., 2020; Pàmies et al., 2020). Therefore, the textual pre-processing in this thesis was limited to a few general steps to reduce noise typically present in social media texts while at the same time retaining context. Finally, a specific tokenization and encoding process for BERT models was also performed.

As there is currently no research available on Norwegian abusive language detection with Transformer-based models, the textual pre-processing steps were mainly based on Pàmies et al. which hold state-of-the-art results for Danish abusive language detection. This choice was based on the underlying assumption that considering the language similarities discussed in Section 2.1, successful pre-processing steps for the Danish language transfer to the Norwegian language.

As noted in Section 3.2.1, one portion of the Norwegian documents was anonymized by Svanes and Gunstad (2020) using the `@USER` tag, whereas the others were anonymized using the `Navn` tag. The term *navn* translates to *name* in English, which is also used in other contexts than for user mentions. Using this term may confuse the model to some extent, as the term ends up representing both the term's meaning and the marking of user mentions. Changing `Navn` tags to a more representative term, such as the Norwegian term

[2]`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

for *username* or *user*, could not be done automatically due to the possibility of changing the term in contexts not representing user mentions. Due to time constraints, it was not an option to manually configure all user mentions present in the 41,145 documents to a common representation. Therefore, it was decided to convert all `@USER` tags into the common `Navn` representation using Python's library for regular expressions, `re`[3], for both the Norwegian and Danish datasets.

Furthermore, the orthographic lengthening was reduced to a maximum of two repeated characters, and irrelevant punctuation marks were removed using the `string`[4] library in Python. Other textual pre-processing steps such as stopword removal, stemming, and lemmatization, as described in Section 2.2, could have been applied. However, as the BERT model takes sentences as input, the pre-processing steps were centered on retaining as much context as possible and creating complete sentences. It was also decided not to remove emojis or emoticons as one of the models used, namely NB-BERT, is partly trained on web-collected data and may understand such content to some extent.

For the Danish data exclusively, the `translatepy`[5] and `EasyNMT`[6] libraries were applied to translate the Danish text into Norwegian. The machine translator systems were selected as the only free, publicly available automatic systems that could be implemented for Danish-Norwegian translations in Python. Google Translate's Python API[7] was not used due to limited expenses, but several unofficial Python libraries implementing the Google Translate web API[8] were tested. This included `googletrans`[9], `pygoogletranslation`[10], and `pyGoogleTranslate`[11]. However, a change to the Google Translate API seemed to have caused exceptions with open issues on GitHub[12] making these libraries unavailable for use. Four different versions of the translated Danish text using the functional machine translation systems were obtained to be tested for detection of offensive and hateful language additionally to the raw Danish text in the experiments of Chapter 7.

The subsequent pre-processing steps included tokenizing and encoding all documents using BERT's specialized tokenizer. A tokenizer corresponding to the particular model, and therefore also based on the same vocabulary, was downloaded using the `BertTokenizer`[13] from Hugging Face's library. The tokenizer performed general and WordPiece tokenization of the input, followed by encoding the tokenized documents into corresponding input IDs, as described in Section 2.4.9.

[3]`https://docs.python.org/3/library/re.html`
[4]`https://docs.python.org/3/library/string.html`
[5]`https://pypi.org/project/translatepy/`
[6]`https://github.com/UKPLab/EasyNMT`
[7]`https://cloud.google.com/translate/docs/reference/libraries/v3/python`
[8]`https://translate.google.com/`
[9]`https://pypi.org/project/googletrans/`
[10]`https://pypi.org/project/pygoogletranslation/`
[11]`https://pypi.org/project/pyGoogleTranslate/`
[12]`https://github.com/ssut/py-googletrans/issues/234`
[13]`https://huggingface.co/transformers/model_doc/bert.html#berttokenizer`

6.3.3 Model Implementation

The same model implementation was used across all experiments. The code used to fine-tune and evaluate the BERT models was adopted from Kummervold et al. (2021) and is publicly available on GitHub[14]. The code was re-structured from notebook files to Python files with minor changes to customize it for running on the IDUN clusters (Själander et al., 2019) as will be further described in Section 6.3.6. The changes included turning off statistics reportings to other platforms such as Weights & Biases (wandb)[15], which required obtaining an open API connection to display training statistics that slowed down the training process. All model configurations were also re-structured in separate JavaScript Object Notation (JSON) files to allow different setups and hyperparameters for different models. The code is publicly available on GitHub[16] with incorporated comments in all places that adopt code from Kummervold et al. with any changes to the code or configurations denoted and described. All BERT models were downloaded using the Hugging Face inference API and implemented using the PyTorch[17] implementation.

The evaluation metrics and confusion matrices used to evaluate the models were implemented using Sklearn's library for model evaluation[18]. All reported results were similar to Devlin et al. (2019) averaged over five fine-tuning sessions to ensure representative reported values. This required five times the computational effort than training once for each experiment. Nevertheless, considering the general variance in training for BERT models as discussed in Section 4.3.2, it was considered a necessary step to ensure reliable result reporting. Based on this, the results reported are less prone to outlier training sessions with a drop or rise in performance.

6.3.4 Selected Hyperparameters

The model hyperparameters refer to any parameter that is set before the learning process takes place. The different hyperparameters and their functions are described in Section 2.4.1. Table 6.2 displays the selected hyperparameter values and ranges used in this thesis, which are based on a set of values defined by Devlin et al. (2019) to work well across several NLP tasks. Devlin et al. defined set values for the maximum sequence length and warmup proportion but stated that the other values depend on the task.

The warmup proportion, which defines the proportion of steps of which the linear decay from the learning rate to zero is performed, was set equal to 10%, similar to Devlin et al. and Vaswani et al. (2017). An initial analysis of the dataset was performed to select the maximum sequence length. The analysis showed that all the Norwegian documents were within a 256 token limit, whereas 0.1% of the documents from the Danish dataset exceeded 512, which is the BERT models' maximum sequence length. Therefore, it was initially decided to set the maximum sequence length to 512 for all models to retain as

[14]`https://github.com/NBAiLab/notram`
[15]`https://wandb.ai/site`
[16]`https://github.com/vildera/abusive_language_detection`
[17]`https://pytorch.org/`
[18]`https://scikit-learn.org/stable/modules/model_evaluation.html`

much information as possible and obtain simplicity across all experiments. However, due to memory limitations encountered during the second experiment using both datasets, the maximum sequence length was reduced to 256 for the remaining experiments (including Experiments 2 and 3). The reduction did not affect the Norwegian documents, but 0.7% of the Danish dataset had to be truncated, and some information was lost. Nevertheless, this is considered a small amount that should not have significantly impacted the results.

Hyperparameter	**Values**
Batch size	16, 32
Learning rate	2e-5, 3e-5, 5e-5
Number of epochs	2, 3, 4
Maximum sequence length	256, 512
Warmup proportion	10%

Table 6.2: The hyperparameter values used in the models of the next chapter's experiments.

For the remaining hyperparameter values, batch size, learning rate, and the number of epochs, a range of different values were searched for each model to find the best combination for each dataset. The values in Table 6.2 were used as base values for the hyperparameter optimization described in the next section unless stated otherwise.

6.3.5 Hyperparameter Optimization

All models were optimized through an exhaustive search over the batch size, learning rate, and the number of epochs defined in Table 6.2 unless stated otherwise. The approach was adopted from Devlin et al., where five random restarts of the fine-tuning were performed for each parameter combination, and the average macro F1 score was calculated and reported from 5-fold cross-validation.

Stratified cross-validation was used to split the training set into five different data splits, ensuring that the validation block for each fold was different and that each dataset had a representative distribution, as described in Section 2.3.7. Note that the optimization process is a part of model selection and that only the training set is used in this process (as previously illustrated in Figure 2.4). The stratified sampling was done based on the original five categories from Svanes and Gunstad (2020) to ensure a variation of documents in each set beyond the two categories used for classification. The splitting was performed using Sklearn's `StratifiedKFold`[19] function with a specified random state of 200 for reproducibility. A random data shuffling was performed for each round, and the best hyperparameters combination averaged over the five splits was chosen.

The process of optimizing Transformer-based models is more complex than for simpler neural networks and linear approaches due to the instability in the fine-tuning process.

[19]`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html`

As previously described, several random restarts are necessary due to the variance in different fine-tuning runs of the BERT model, a process that requires vast amounts of computational resources running over several days. The instability is due to the models being more complex than general approaches. The first steps in the fine-tuning process significantly affect the final solution, a process that is described in more detail in Section 4.3.2. However, when considering context, one may speculate that this is due to the language having many different "solutions" in this level of interpretation.

Despite the extensive computational resources required, the optimization of hyperparameters was considered profitable. This thesis aims to accurately detect and distinguish neutral, offensive, and hateful language, which optimized models can significantly help improve. A summary of the selected optimized hyperparameters for the models used in the experiments of the following chapter is displayed in Table 6.3. The corresponding hyperparameter optimization tables for the five BERT architectures fine-tuned on the Norwegian dataset can be found in Appendix 1.

Offensive Language Detection				
Model	**Experiment**	**Batch Size**	**Learning rate**	**# Epochs**
mBERT$_{based,cased}$	1	16	2e-05	3
mBERT$_{based,uncased}$	1	32	5e-05	3
NorBERT$_{based,cased}$	1	32	5e-05	3
NB-BERT$_{based,cased}$	1	16	3e-05	2
NB-BERT$_{large,uncased}$	1	32	3e-05	3
NB-BERT$_{large,uncased}$	2a	32	3e-05	3
NB-BERT$_{large,uncased}$	2b 1)	32	3e-05	2
NB-BERT$_{large,uncased}$	2b 2)	32	3e-05	2
Hate Speech Detection				
Model	**Experiment**	**Batch Size**	**Learning rate**	**# Epochs**
mBERT$_{based,cased}$	1	32	5e-05	3
mBERT$_{based,uncased}$	1	16	5e-05	3
NorBERT$_{based,cased}$	1	16	5e-05	4
NB-BERT$_{based,cased}$	1	16	2e-05	3
NB-BERT$_{large,uncased}$	1	32	3e-05	3
NB-BERT$_{large,uncased}$	2a	16	3e-05	3
NB-BERT$_{large,uncased}$	2b 1)	32	2e-05	4
NB-BERT$_{large,uncased}$	2b 2)	32	2e-05	4

Table 6.3: The resulting hyperparameters from the hyperparameter search of the best models for experiments 1 and 2.

6.3.6 Computational Resources and Setup

All model experiments required extensive hardware resources due to their high complexity and were run on the NTNU IDUN High-performance Computer Cluster (Själander et al., 2019). The steps taken to connect to the IDUN clusters and set up scheduled jobs followed the instructions available on their web page[20]. The implemented libraries described in the previous section have automatic support for multi-GPU use. When several GPUs are detected, the batches are automatically split over the available GPUs without any further specifications in the code. The BERT$_{large}$ model took about 40 minutes to run on the complete Norwegian dataset for offensive language detection and about 10 minutes to run on the offensive part of the dataset for hate speech detection when utilizing two Nvidia P100 GPUs using the optimized hyperparameters. The translation process of the Danish dataset into Norwegian using the Transformer-based EasyNMT machine translator system took about 40 minutes to run on the same resources. For the merged experiment, the 32GB RAM Nvidia V100 GPUs had to be used for sufficient memory capacity. It took about 1.5 hours for offensive language detection and 15 minutes for hate speech detection. When adjusting the hyperparameters, particularly for larger numbers of epochs and lower learning rates, the fine-tuning process significantly increased, about two to three times the initial training time. The data pre-processing and results analysis were performed using a personal computer. All hardware resources used to run the experiments can be found in Table 6.4.

Server / Type	**Processors**	**# Cores**	**Memory**	**GPUs**
Dell PE730	2x Intel Xeon E5-2650 v4	24	128 GB	2x Nvidia Tesla P100 16 GB VRAM
Dell PE740	2x Intel Xeon Gold 6132	28	768 GB	2x Nvidia Tesla V100 32 GB VRAM
Personal Computer	1x Intel i9 2300K	8	16 GB	None

Table 6.4: The hardware resources used to run the experiments in Chapter 7.

[20] `https://www.hpc.ntnu.no/idun`

7 Experiments and Results

The research conducted in this thesis is partly carried out through experiments implementing the systems described in the previous chapter. These experiments are carried out following the experimental plan described in Section 7.1 which is mainly guided by the research questions with the aim to achieve the goal of this thesis. Finally, the experimental results will be presented in Section 7.2 following the same structure as the experimental plan.

7.1 Experimental Plan

The experimental plan is primarily guided by the thesis's goal: "*Investigate how to accurately detect and distinguish neutral, offensive, and hateful Norwegian language in social media.*" The two binary classification problems defined below are used to detect abusive language for all experiments in this thesis to achieve this goal.

- **Offensive language detection:** The task of separating offensive language from neutral language. A document is either offensive or neutral.
- **Hate speech detection:** The task of separating hateful language from the other offensive language types. A document is either hateful or (only) offensive. The neutral documents are excluded from this task.

This means that all of the following experiments are performed for both offensive language detection and hate speech detection, requiring training and evaluation for all models on two separate subsets of the data. The models are trained and evaluated for offensive language detection on neutral and offensive documents (corresponding to the complete datasets). In contrast, for hate speech detection, the models are trained and evaluated only on the offensive documents (including the hateful documents). All models are also optimized through an exhaustive search described in Section 6.3.5 to possibly improve performance further. The following steps of the experimental plan are guided by the research questions designed to achieve the goal of this thesis.

7.1.1 Experiment 1 - Norwegian BERT Models

The first experiment aims to answer the first and second research questions concerning how well the Norwegian BERT models can detect offensive and hateful Norwegian language and how their performance compares to one another in these tasks. The models follow the architectures summarized below, as described in Chapter 6, and are trained and evaluated on the Norwegian dataset.

- **mBERT base, cased:** pre-trained on the 100 largest Wikipedia sites.
- **mBERT base, uncased:** pre-trained on the 100 largest Wikipedia sites.
- **NorBERT base, cased:** pre-trained on the Norwegian Wikipedia and Norsk Aviskorpus.
- **NB-BERT base, cased:** pre-trained on Norwegian books, newspapers, Wikipedia, and other formal reports.
- **NB-BERT large, uncased:** pre-trained on Norwegian books, newspapers, Wikipedia, and other formal reports.

The results will be used to evaluate the effect of transferring knowledge from pre-trained Norwegian language models to the downstream tasks of detecting offensive and hateful language for Norwegian. The different versions of the BERT models will be compared, and the optimized model with the best performance will be used in the following experiments.

7.1.2 Experiment 2 - Dataset Manipulation Techniques

The second experiment aims to answer the third research question: "*What are the effects of applying different data manipulation techniques to the Norwegian dataset when detecting offensive and hateful social media language?*". The best-evaluated model architecture from Experiment 1 will be trained on different variations of the Norwegian and Danish datasets. First, the undersampling technique will be tested as described in Experiment 2a, followed by testing the effect of concatenating the Danish and Norwegian datasets as described in Experiment 2b.

Experiment 2a - Undersampling

This sub experiment investigates whether undersampling, i.e., removing instances, from the majority class when detecting offensive and hateful language can help improve model performance. This is performed by creating a training dataset with 67% samples from the majority class and 33% from the minority class using random sampling. That is, all instances from the minority class are obtained, whereas the majority class is undersampled. The proportions were selected to reduce the majority class bias and simultaneously retain a skewed distribution. This is necessary because, in addition to learning the language, the models learn the class distribution, which will affect their label choice in cases of uncertainty.

Experiment 2b - Expanding the Norwegian Dataset

This sub experiment consists of two steps. The first step tests the different versions of the Danish dataset previously described in Section 6.3.2, and the second step tests the effect of training the model on the concatenated Norwegian and Danish datasets, and the Danish dataset only. The four versions of the Danish documents listed below will be used for training the model, followed by evaluating it on the Norwegian test dataset.

- **No machine translation:** The pre-processed document texts without any machine translation performed.
- **Translatepy:** The document texts translated using the `translatepy` library based on several known web machine translator systems described in Section 2.5.
- **EasyNMT:** The document texts translated using the three versions OpusMT, M2M_100_1.2B and M2M_100_418M of the Transformer-based machine translation system, EasyNMT, also described in Section 2.5.

For this step, the best-performing model from the previous experiment is used along with the optimized hyperparameters. In order to select the best version of the Danish dataset, the model is fine-tuned on the Norwegian dataset concatenated with the listed versions of the Danish dataset. Note that the evaluations of the Danish dataset version will indicate what version is better for this particular task when training a model on offensive and hateful language detection tasks and not whether the actual machine translations are better, which would have to be evaluated differently. Therefore, the selected machine translator system is solely based on which system creates a better representation of the Danish dataset for offensive and hateful language detection. The best representation selected from the above list will be used in the remaining part of the experiment.

For the second step of this experiment, the following data combinations will be used for fine-tuning the model for offensive and hateful language detection.

1. **Full Danish and Norwegian datasets:** The training dataset will be extended with all instances from the danish dataset.
2. **Full Danish dataset:** Only the full Danish dataset will be used to train the model, meaning that the Norwegian training instances will not be used in this step.

The steps will investigate the generalizability from the Danish to the Norwegian dataset and whether the two datasets generally and in terms of language have enough similarities to help the model learn patterns distinguishing between the different classes. The first step investigates if the Danish and Norwegian datasets can be combined to better learn abusive language types, and the second step investigates the ability of the model to generalize from the Danish to the Norwegian dataset, which may say something about the similarities of the two datasets. The best-performing model will be compared to the models from the previous section, and the best model across both experiments will be used in the final experiment.

7.1.3 Experiment 3 - Final Model Optimization

Guided by the goal of this thesis, the final experiment aims to optimize the best models for offensive language detection and hate speech detection across the previous experiments.

This includes an extensive hyperparameter search in the same manner as performed for all previous models (described in Section 6.3.5), but with a larger hyperparameter space requiring more computational power and time. The selected hyperparameter space searched is listed in Table 7.1. The selected values expand the general optimization search for all models (described in Section 6.3.5) with a higher number of epochs and lower learning rate. These values are selected based on the research by Mosbach et al. (2021) which proposed this as an approach to stabilize the training instability of BERT models and possibly avoid bad training sessions, as described in Section 4.3.2.

Hyperparameter	**Values**
Batch size	8, 16, 32
Learning rate	1e-5, 2e-5, 3e-5, 5e-5
Number of epochs	2, 3, 4, 5, 10, 15, 20
Maximum sequence length	512
Warmup proportion	10%

Table 7.1: The hyperparameters used in the final optimization of the best models across all experiments.

7.2 Experimental Results

This section presents the results from each experiment described in Section 7.1. All results are presented for the offensive language detection and hate speech detection tasks presented in Section 7.1. The metrics described in Section 2.3.7 which includes F1 score (F1), precision (P), and recall (R), are used to evaluate models. Note that for binary classification, the micro F1 score is equal to the accuracy score. Furthermore, the macro F1 score is used for model selections due to the unbalanced nature of the datasets used. The experimental results follow the same structure as the experimental plan, presenting the results of the first, second, and third experiments subsequently.

7.2.1 Experiment 1 - Norwegian BERT Models Results

The overall results for offensive and hateful language detection using the Norwegian BERT models are presented in Table 7.2. As the experiments in this thesis solely explore the Transformer-based BERT models, the top results from Svanes and Gunstad (2020) using more traditional non-transformer approaches (previously described in Section 4.2.3), are added to the tables for comparisons. The results representing Svanes and Gunstad were calculated with reference to their confusion matrices to get the same class generalizations and metrics representations used in this thesis. Svanes and Gunstad reported accuracy as their main measure for model selection, which in this thesis was considered unsuitable based on the highly unbalanced data as previously discussed. A re-evaluation of their results was therefore performed with regards to the macro F1 score. The reported results are based on their SVM model, which achieved the best results for both offensive and hateful language detection in terms of macro F1 score. The majority class baseline is also added to the overall results in Table 7.2 for comparison.

	Neutral/Offensive		**Offensive/Hateful**	
Model	**Macro F1**	**Micro F1**	**Macro F1**	**Micro F1**
mBERT$_{base,cased}$	0.698	0.850	0.630	0.876
mBERT$_{base,uncased}$	0.694	0.846	0.614	0.886
NorBERT$_{base,cased}$	0.720	0.857	0.626	0.880
NB-BERT$_{base,cased}$	0.738	**0.867**	0.664	0.887
NB-BERT$_{large,uncased}$	**0.754**	0.866	**0.680**	0.896
Majority Class Baseline	0.454	0.832	0.474	**0.900**
Svanes and Gunstad (2020)	0.657	0.796	0.622*	0.806*

*The denoted results are calculated from a two step classification approach where the instances classified firstly were correctly classified as offensive.

Table 7.2: Reported overall results for all models tested in Experiment 1 for offensive and hateful language detection. The best results are marked in bold.

The table shows that the NB-BERT$_{large,uncased}$ outperforms the other models for both tasks with a Macro F1 score of 0.754 for offensive language detection and 0.680 for hate speech detection. These results are in line with Devlin et al. (2019) which showed that the BERT$_{large}$ architecture outperformed the BERT$_{base}$ architecture for all their experiments for different downstream NLP tasks, which was also confirmed by Kummervold et al. (2021) for the Norwegian BERT models for several downstream tasks[1]. NB-BERT$_{base,cased}$ is the second-best model and nearly reaches the results of NB-BERT$_{large,uncased}$, while the other three models are a few steps behind. For offensive language detection, all BERT models exceeded Svanes and Gunstad, whereas for hate speech detection, all models except mBERT$_{base,uncased}$ exceeded Svanes and Gunstad. Considering the problems with the tokenizer corresponding to the model as described in Section 4.3.1, these results were not unexpected.

Nevertheless, as described in Section 4.2.3, the approach by Svanes and Gunstad is based on a two-step approach predicting only the 979 correctly predicted offensive instances, compared to the more representative test set of 2076 instances used in the experiments of this thesis. The results are therefore not fully comparable, as one can assume that several of the more challenging documents to classify did not proceed to the second step in Svanes and Gunstad's approach. One may, therefore, assume that it is more challenging to improve the results on the test dataset used for hate speech

[1]This information is based on personal communication with Per Egil Kummervold, one of the creators of NB-BERT.

detection in this thesis than the subset used by Svanes and Gunstad in their two-step approach. Based on this, it is evident that the Transformer-based models in this thesis have significantly improved the results of Svanes and Gunstad for both offensive and hateful language detection.

Furthermore, the classwise results for both offensive language detection and hate speech detection are displayed in Table 7.3. The first apparent observation from the table is that all models were able to classify the majority classes quite well, the neutral class for offensive language detection and the offensive class for hate speech detection. The NB-BERT$_{large,uncased}$ obtained the best results for the hateful class with precision, recall, and F1 scores of 0.474, 0.371, and 0.416, respectively, and achieved the best recall and F1 score of 0.573 and 0.589, respectively, for the offensive class. The classwise results also reflect that NB-BERT$_{base,cased}$ almost approached the results to NB-BERT$_{large,uncased}$, whereas the other models are further behind.

	Offensive Language Detection					
	Neutral			**Offensive**		
Model	**P**	**R**	**F1**	**P**	**R**	**F1**
mBERT$_{base,cased}$	0.889	0.937	0.912	0.574	0.418	0.484
mBERT$_{base,uncased}$	0.888	0.931	0.909	0.554	0.422	0.479
NorBERT$_{base,cased}$	0.896	0.937	0.916	0.599	0.465	0.523
NB-BERT$_{base,cased}$	0.902	**0.943**	**0.922**	**0.634**	0.491	0.554
NB-BERT$_{large,uncased}$	**0.915**	0.925	0.920	0.606	**0.573**	**0.589**
Svanes and Gunstad (2020)	0.889	0.863	0.875	0.411	0.470	0.438

	Hate Speech Detection					
	Offensive			**Hateful**		
Model	**P**	**R**	**F1**	**P**	**R**	**F1**
mBERT$_{base,cased}$	0.924	0.934	0.932	0.359	0.302	0.328
mBERT$_{base,uncased}$	0.919	**0.958**	0.938	0.384	0.236	0.279
NorBERT$_{base,cased}$	0.922	0.947	0.934	0.372	0.280	0.318
NB-BERT$_{base,cased}$	0.931	0.944	0.938	0.421	0.364	0.390
NB-BERT$_{large,uncased}$	**0.932**	0.954	**0.943**	**0.474**	**0.371**	**0.416**
Svanes and Gunstad (2020)	0.891*	0.880*	0.886*	0.346*.	0.370*	0.358*

*The denoted results are calculated from a two step classification approach where the instances classified firstly were correctly classified as offensive.

Table 7.3: Reported classwise results for all models tested in Experiment 1 for offensive and hateful language detection. The best results are marked in bold.

For offensive language detection, 61% of the instances classified as offensive were correctly classified, and 57% of the true offensive documents were labeled correctly. For hate speech detection, 47% of the instances classified as hateful were correctly classified, while only 37% of the true hateful instances were labeled correctly, yielding a lower recall for this task. The NB-BERT$_{base,cased}$ and NB-BERT$_{large,uncased}$ developed by Kummervold et al. (2021) outperformed the other models with a significant gap in macro F1 scores to the other models for the tasks of detecting offensive and hateful language. Considering the fact that these models are pre-trained on a much larger Norwegian corpus, the results correspond to the expectations.

7.2.2 Experiment 2 - Dataset Manipulation Results

The results from training NB-BERT$_{large,cased}$ on the manipulated datasets in Experiments 2a and 2b are displayed in Tables 7.4 and 7.5 presenting overall and classwise results, respectively.

	Neutral/Offensive		**Offensive/Hateful**	
Dataset	**Macro F1**	**Micro F1**	**Macro F1**	**Micro F1**
Undersampling	0.747	0.836	0.648	0.824
Danish & Norwegian	**0.758**	**0.868**	**0.662**	**0.892**
Danish	0.566	0.840	0.395	0.489

Table 7.4: Reported overall results for all models tested in Experiment 2 for offensive and hateful language detection. The best results are marked in bold.

Considering the overall results from testing undersampling, the model detecting offensive language achieved almost the same results as the best model from Experiment 1 in terms of macro F1 score. In contrast, the performance for hate speech detection dropped slightly with 3.2 macro F1 points compared to the best results from Experiment 1. The more significant change in the performance of hate speech detection is not surprising, considering the fact that the distribution change for the dataset used for this task was more drastic than for offensive language detection, resulting in a larger effect on performance.

From Table 7.5 one can observe the classwise results for the same experiments, which compared to the results from Experiment 1 shows a significant change in the classwise performance when using the undersampling technique. More specifically, there is a large rise in recall and a drop in precision for offensive and hateful language detection, demonstrating the trade-off between the two measures. For offensive language detection, undersampling led to 51% of the instances classified as offensive being correctly classified and 72% of the true offensive instances being classified correctly. Only 30% of the instances classified as hateful were correctly classified for hate speech detection, but 59% of the true hateful instances were classified correctly. These results show that as the model increases the number of instances detected, these detections become less precise.

For experiment 2b, the results from training NB-BERT$_{large,cased}$ with the optimized hyperparameters from Experiment 1 on the concatenated Norwegian and Danish dataset

	Offensive Language Detection					
	Neutral			**Offensive**		
Dataset	**P**	**R**	**F1**	**P**	**R**	**F1**
Undersampling	**0.938**	0.860	0.897	0.509	**0.718**	**0.596**
Danish & Norwegian	0.916	0.927	**0.921**	**0.614**	0.578	0.595
Danish	0.850	**0.981**	0.911	0.600	0.144	0.222
	Hate Speech Detection					
	Offensive			**Hateful**		
Dataset	**P**	**R**	**F1**	**P**	**R**	**F1**
Undersampling	**0.949**	0.850	0.897	0.304	**0.587**	**0.400**
Danish & Norwegian	0.928	**0.954**	**0.941**	**0.446**	0.335	0.382
Danish	0.907	0.483	0.616	0.106	0.549	0.174

Table 7.5: Reported classwise results for all models tested in Experiment 2 for offensive and hateful language detection. The best results are marked in bold.

versions were used as the basis to select the version of the Danish dataset to proceed with. As previously described in Section 6.3.4, the expansion of the dataset led to out-of-memory issues on the available devices. Due to this, the initial sequence length was reduced from 512 to 256, and the GPUs with 32 GB RAM had to be used for all experiments. This ultimately led to the experimental plan being significantly delayed.

The results are presented in Table 7.6 and show that the variations in the performance of the machine translation systems were only slight. The use of no machine translation outperformed the other versions for hate speech detection and performed equally well to the OpusMT machine translation system for offensive language detection in terms of macro F1 score. The results match the fact that the NB-BERT models are trained on old Norwegian and some Danish text and seem to understand raw Danish better than machine-translated Norwegian text. Based on these results, the Danish text without any machine translation was used in the remaining parts of Experiment 2b.

The results from Experiments 2b in Tables 7.4 and 7.5, show that expanding the Norwegian dataset with the Danish data had no significant effect for offensive language detection and resulted in a slight performance drop of two F1 macro points for hate speech detection. When considering the sizes of the Norwegian training datasets for both tasks, the Danish dataset makes up a much larger proportion when training for hate speech detection than for offensive language detection, explaining the different effects on performance for the two tasks. Furthermore, the results from training on only the Danish dataset show that the models were not able to generalize from the Danish to the Norwegian data for offensive and hateful language detection.

	Neutral/Offensive		Offensive/Hateful	
Machine Translator	**Macro F1**	**Micro F1**	**Macro F1**	**Micro F1**
No machine translation	**0.757**	0.866	**0.642**	**0.897**
OpusMT	**0.757**	**0.867**	0.612	0.896
M2M_100_1.2B	0.754	0.865	0.635	0.893
M2M_100_418M	0.754	0.866	0.635	0.896
Translatepy	0.751	0.865	0.623	0.890

Table 7.6: Reported overall results for the NB-BERT$_{large,uncased}$ model with the same parameters as in Experiment 1 trained on the Norwegian dataset merged with the different versions of the Danish dataset.

7.2.3 Experiment 3 - Final Model Optimization Results

The best model across both experiments corresponded to the NB-BERT$_{large,uncased}$ for offensive and hateful language detection. The resulting optimal hyperparameters from the broader hyperparameter search for both tasks using this model are displayed in Table 7.7. Furthermore, Tables 7.9 and 7.10 present the average macro F1 scores from 5-fold cross-validation for all hyperparameter combinations for offensive and hateful language detection, respectively. The results reported in these tables are, as described in Section 6.3.5, averaged over each fold of the training set, which is completely separated from the test set to ensure an unbiased selection of hyperparameters.

Task	Batch size	Learning rate	#Epochs
Neutral/Offensive	16	5e-5	3
Offensive/Hateful	32	3e-5	3

Table 7.7: The resulting optimal hyperparameters from the final optimization of the best models across the previous experiments.

For offensive language detection, the resulting hyperparameters differed from the narrower search in Experiment 1 but were within the same ranges. The shift in selected hyperparameters is not surprising when considering the small variations between various hyperparameter combinations in Table 7.9 combined with the instability in fine-tuning for BERT models (described in Section 4.3.2). When evaluating NB-BERT$_{large,uncased}$ using the optimal hyperparameter combination from Table 7.7 for offensive language detection, the resulting macro and micro F1 scores were 0.753 and 0.866, which is approximately equal to the results achieved for the same model in Experiment 1. For hate speech detection, the resulting optimal hyperparameters correspond to those selected in the narrower search in Experiment 1, and the evaluated results are, therefore, the same as presented in Section 7.2.1. Overall, the final hyperparameter search did not improve model performance.

The fact that the optimal hyperparameters for offensive and hateful language detection were within the ranges defined for the narrower search in Table 7.10 conform with Devlin et al. (2019) which found that these ranges are optimal for several general NLP tasks. At the same time, this indicates that increasing the number of epochs and lowering the learning rate, proposed by Mosbach et al. (2021), did not improve model performance on the Norwegian dataset for offensive or hateful language detection. Considering the size of the dataset used for hate speech detection, the results for this model were unexpected as Mosbach et al. proposed the approach, particularly for smaller datasets.

Furthermore, the results from Tables 7.9 and 7.10 show that for all batch sizes, the optimal number of epochs were equal or less than four. The learning rates were typically also within the values defined by Devlin et al., except when using a smaller batch size of eight, which resulted in lower optimal learning rates (1e-5 for offensive language detection and 2e-5 for hate speech detection). When observing the variations in macro F1 cross-validation scores from the two tables, one can see that the values for offensive language detection vary with about three macro F1 points, whereas the values for hate speech detection vary with about 20 macro F1 points. This reflects that hyperparameter choice for the offensive language detection model had less impact on the results than it had for the hate speech detection model. This is consistent with the fact that the latter is trained on a much smaller dataset and that Devlin et al. found the BERT$_{large}$ model to sometimes be unstable on small datasets.

The overall best results from all three experiments after evaluating the optimized models on the test dataset are presented in Table 7.8 for offensive and hateful language detection. The best model for both tasks corresponded to NB-BERT$_{large,cased}$ and the results reported were achieved in Experiment 1 from optimizing this model on the Norwegian dataset for the respective tasks searching the hyperparameters previously presented in Table 6.2, i.e., using the narrower hyperparameter search.

Task	**Model**	**Macro F1**	**Micro F1**
Neutral/Offensive	NB-BERT$_{large,cased}$	0.754	0.866
Offensive/Hateful	NB-BERT$_{large,cased}$	0.680	0.896

Table 7.8: The best overall results across all experiments for offensive and hateful language detection.

Batch Size: 32				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.740	0.742	0.744	0.739
15	0.745	0.742	0.745	0.740
10	0.743	0.745	0.743	0.745
5	0.746	0.749	0.750	0.754
4	0.755	**0.759**	0.757	0.757
3	0.750	0.753	0.757	0.756
2	0.753	0.752	0.756	0.742
Batch Size: 16				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.738	0.740	0.747	0.744
15	0.744	0.742	0.746	0.743
10	0.737	0.744	0.743	0.745
5	0.750	0.747	0.746	0.751
4	0.748	0.752	0.755	0.760
3	<u>**0.762**</u>	0.758	0.759	0.757
2	0.749	0.755	0.758	0.750
Batch Size: 8				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.737	0.738	0.744	0.745
15	0.741	0.744	0.744	0.744
10	0.738	0.742	0.743	0.745
5	0.745	0.746	0.747	0.750
4	0.752	0.752	0.750	0.755
3	0.752	0.755	0.755	**0.760**
2	0.750	0.750	0.758	0.753

Table 7.9: Results from final hyperparameter optimization of the best offensive language detection model. The values are macro F1 scores averaged over 5-fold cross-validation. The best values for each batch size are marked in bold, and the best overall value is marked with an underline.

Batch Size: 32				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.657	0.666	0.662	0.665
15	0.652	0.671	0.655	0.658
10	0.662	0.674	0.671	0.665
5	0.655	0.657	0.659	0.625
4	0.676	0.673	0.671	0.549
3	0.676	<u>**0.693**</u>	0.683	0.543
2	0.603	0.514	0.508	0.471
Batch Size: 16				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.658	0.655	0.663	0.661
15	0.644	0.646	0.649	0.667
10	0.652	0.661	0.649	0.667
5	0.651	0.663	0.661	0.652
4	0.659	0.680	0.678	0.650
3	**0.682**	0.677	0.677	0.629
2	0.638	0.645	0.574	0.470
Batch Size: 8				
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**	**1e-5**
20	0.586	0.657	0.650	0.658
15	0.605	0.657	0.658	0.663
10	0.574	0.613	0.653	0.670
5	0.580	0.670	0.670	0.663
4	0.663	0.684	0.681	0.675
3	0.604	0.670	**0.687**	0.665
2	0.624	0.668	0.655	0.520

Table 7.10: Results from the final hyperparameter optimization of the best hate speech detection model. The values are macro F1 scores averaged over 5-fold cross-validation. The best values for each batch size are marked in bold, and the best overall value is marked with an underline.

8 Evaluation and Discussion

The experiments and results of the previous chapter left several outstanding questions to be settled. Section 8.1 evaluates the previous chapter's experiments, whereas Section 8.2 discusses the findings with regards to the research goal and questions defined in Section 1.2.

8.1 Evaluation

The experiments of this thesis were carried out following the experimental plan described in Section 7.1. The plan was composed of three separate experiments testing various BERT architectures, data manipulation techniques, and hyperparameter optimization searches, for the tasks of offensive and hateful language detection. This section evaluates the use of the Transformer-based BERT models and discusses the dataset and how its characteristics affect the predictions. Finally, an error analysis is presented investigating what types of instances are successfully predicted by the best models, and what types are not.

8.1.1 Model Selection and Comparison to State-of-the-art

From the results of the experiments presented in the previous chapter, it is evident that the introduction of Transformer-based models had a positive effect on detecting Norwegian offensive and hateful language. The models outperformed the Svanes and Gunstad (2020) significantly, which besides Jensen (2020) is the only research available detecting abusive languages for Norwegian. The best performing model for offensive language detection outperformed Svanes and Gunstad with ten macro F1 points. Additionally, the best model for hate speech detection outperformed their model with six macro F1 points. The latter is also a large rise in performance, considering that Svanes and Gunstad evaluated their model on already correctly classified non-neutral instances. This likely makes the test dataset easier to classify, as several incorrectly predicted instances were withdrawn before the evaluation.

Although the BERT models generally outperformed Svanes and Gunstad, the performance of the different BERT models varied largely. The range in performance of the models aligned well to what was experienced by Kummervold et al. (2021) and Kutuzov et al. (2021). NB-BERT achieved the best results, followed by NorBERT, and mBERT with the poorest performance among the three. The ranking of the models corresponds to the increasing sizes of their pre-training corpora. Lastly, the NB-BERT model is the only model pre-trained on data crawled from the web, which is more similar to social media data and, therefore, may have contributed to a better language understanding of such content.

The large version of NB-BERT outperformed the base version for offensive and hateful language detection. This also corresponds to the results reported in the original BERT paper, Devlin et al. (2019), and the creators of NB-BERT[1], which both showed that the large BERT version outperforms the base versions for most NLP tasks. The increased performance when using the large NB-BERT model compared to the base model is likely due to the increased number of parameters in the model, allowing for more complex pattern representations of the language.

The increased performance comes at the cost of increased model complexity, as previously described in Chapter 6. The large model complexity of BERT models requires extensive computational resources. The experiments of this thesis heavily relied on the IDUN HPC Cluster (Själander et al., 2019) and could not have been performed without available GPU resources with sufficient memory capacity. Moreover, the BERT models showed a large variance when trained to detect offensive and hateful language, particularly hateful, constituting a smaller dataset. The instability in fine-tuning is consistent with Devlin et al. and the other relevant literature described in Section 4.3.2. Based on Devlin et al., the instability was coped with by averaging five fine-tuning runs for each reported result, increasing the computational effort extensively. This led to more work than initially assumed; however, the increased computational effort was considered necessary to report reliable and representative results.

All reported results were based on optimized hyperparameters unless stated otherwise. The optimization procedure, previously described in Section 6.3.5, was an extensive procedure that ran over several days to weeks depending on the hyperparameters searched and the model used. Despite the dedicated computational effort to the extensive search in Experiment 3, the search did not lead to increased performance. This was slightly unexpected for hate speech detection, particularly, when considering the proposed approach by Mosbach et al. (2021) of using lower learning rates and increased number of epochs for more stable results and increased performance on smaller datasets. The results showed that the narrower search based on the proposed hyperparameters by Devlin et al. was sufficient and resulted in the best overall performance. Based on this, the extensive search in Experiment 3 was unnecessary regarding model performance which is an interesting result in itself when considering how to prioritize the computational effort in future work.

When considering the instability observed during fine-tuning of the models, the reliability of the hyperparameter searches can also be discussed. Although precautions were taken by averaging the results over five runs using stratified cross-validation, the instability could still have affected the outcome. One can observe from Tables 7.9 and 7.10 for offensive and hateful language detection, respectively, that the performance for several hyperparameters are close in values. This is particularly emphasized for offensive language detection as the performance of different combinations ranges with a maximum of three macro F1 points across all combinations. One unstable run can therefore easily affect the hyperparameter choice in the search, which is also likely why

[1]This information is based on a discussion with Per Egil Kummervold, one of the creators of NB-BERT.

the final hyperparameter run for offensive language detection in Experiment 3 had a different outcome than the narrower search within the same search space of Experiment 1. An approach to limit the variance is to increase the number of fine-tuning runs beyond five until the performance converges towards a value. However, this requires extensive computational resources and must therefore be weighed against the possible return in results.

Based on the small differences in the choice of hyperparameters for offensive language detection, and that the instability in fine-tuning on larger datasets is less prominent according to Devlin et al. and Mosbach et al. (2021), such extensive searches might not be advantageous for offensive language detection using the dataset by Svanes and Gunstad and Jensen. For hate speech detection, which is based on a smaller dataset, increasing the number of fine-tuning runs could decrease the instability and therefore raise the reliability of such searches.

8.1.2 Datasets and Data Manipulation Techniques

As for other machine learning tasks, offensive and hateful language detection heavily rely on the dataset used. The specialization project (Arntzen, 2020) revealed a lack of Norwegian abusive language datasets, which led to using the only available dataset by Svanes and Gunstad (2020) and Jensen (2020) in the experiments of this thesis. As previously discussed in Sections 3.2.1 and 4.2.3, this dataset is highly unbalanced with few instances in the abusive categories, which has negatively affected the ability of models to detect abusive languages in previous work (Svanes and Gunstad, 2020; Jensen, 2020).

Two dataset manipulation techniques were tested in Experiment 2 to cope with the biased predictions towards the neutral class. The results from undersampling demonstrated the trade-off between precision and recall and reflected how undersampling could significantly change the classwise performance for offensive and hateful language detection. The detection for the positive classes (hate speech and offensive language) increased at the cost of a drop in performance for the negative classes. This can be explained by how the model obtains knowledge. Recall that the model's objective is to learn how to best separate between the positive and negative classes. During training, the classifier obtains two types of knowledge: knowledge about the language and knowledge about the distribution. Knowledge about language concerns what types of language typically represent the positive class and what types of language represent the negative class. In contrast, knowledge about the distribution tells how often instances from the different classes occur. In cases of doubt, the distribution knowledge will influence the model's guess, reflected in the shift in class performance. What predictions are preferred depends on the circumstances. When monitoring content, a higher degree of detected hateful content and wrongly predicted non-hateful content would obtain a safer environment at the cost of less freedom of expression. In order to maintain freedom of expression and the overall best performance, undersampling was not considered beneficial and, therefore, not further used in the experiments.

Experiment 2b expanded the Norwegian dataset with instances from a Danish dataset. First, several different machine translator systems were tested to see if they could help the model learn patterns from the Danish dataset. The results showed that the translation had a negative or no positive effect on the NB-BERT model's ability to detect offensive and hateful language. This might be because NB-BERT is pre-trained partly on Danish text and on a large portion of old Norwegian text that almost corresponds to Danish, and thereby has a decent understanding of the Danish language. This aligns well with the results from Kummervold et al. (2021) which showed that NB-BERT performed relatively well for the Danish language and outperformed mBERT in their Danish experiment. Furthermore, when examining a subset of the translated Danish dataset, it was evident that the machine translations were not accurate for a large proportion of documents. An example of a translated sentence with multiple errors marked in bold is given below.

Danish	**Minder** mig om det 25 årige black army medlem jeg mødte på OBC for nogle år tilbage, han var også **stukket i låret**, og samtidig på flugt fra politiet på vej til Kbh. **Det vildt det stadig foregår**.
Norwegian	**Miner** meg om det 25 år gamle black army medlem jeg møtte på OBC for noen år tilbake, var han også **stuktet i lart**, og samtidig på flukt fra politiet på vei til Kbh. **Det vil fortsatt skje**.
English	**Reminds** me of the 25 years old black army member I met at OBC some years back, he was also **stabbed in the thigh**, and at the same time on the run from the police on the way to Kbh. **It wild it's still going on**.

Table 8.1: Example of Danish Document translated to Norwegian using the OpusMT machine translator system. The errors are marked in bold.

The first error is the translation of the Danish term *Minder* (*Reminds* in English), into *Miner* (*Mines* in English), a type of explosives. The Danish term *Minder* is a common term that the NB-BERT model most likely knows from the Danish or old Norwegian parts of the pre-training corpus, making the translation unfortunate. The following two errors are the translations of the terms *stabbed* and *thigh*, which originally are the same in Norwegian and Danish, namely *stukket* and *låret.* However, the terms were translated into two non-existing terms in the Norwegian vocabulary, demonstrating how the machine translator system can introduce errors in places that initially were correct.

The last error in the above sentence is the translation of "It wild it's still going on" into "It will continue to happen," changing the complete meaning of the sentence. Note that the sentence is missing an *is* before *wild* to bind together the sentence. This may be one reason why the machine translator system changes the meaning of the text, as the machine translator is also trained on general language understanding with the lack of such errors. These errors in the machine translations affect the BERT model's ability to understand language and confuse it by changing the context and meaning of the sentence. As previously discussed, BERT models pre-trained on general language understanding are not robust towards grammar mistakes due to the numerical word embeddings representing slightly different terms with entirely different numerical values.

Using the machine translator system introduced several errors resulting in an overall information loss for several of the documents in the corpus, which may explain the performance loss using machine translator systems.

Moreover, the results from training NB-BERT$_{large,cased}$ on the concatenated Norwegian and non-translated Danish dataset showed no significant effect for offensive language detection and a slight drop in performance for hate speech detection. It was also evident that the model could not generalize well from the Danish to the Norwegian dataset when being trained on the Danish dataset solely. For hate speech detection, this could be due to the limited number of instances in the training dataset (441 offensive instances and only 216 hateful instances), which may be insufficient for the model to generalize from. However, the fact that the model could not generalize well from one dataset to another, combined with the fact that training on both datasets led to a drop in performance for hate speech detection, may indicate that the Danish data added noise to the Norwegian training data. Further pre-processing could have been applied to possibly improve generalizability across the datasets. For instance, the analysis in Chapter 5 revealed that the Danish dataset contained more web links than the Norwegian dataset, which could have been removed to get greater compatibility. However, based on the fact that NB-BERT, as mentioned, has proven to perform well for Danish tasks generally, a hypothesis is that the lack of ability to generalize goes beyond the general language differences and lack of pre-processing.

The analysis in Chapter 5 reflected several differences between the two datasets. For instance, the abusive classes of the Norwegian dataset seemed to be biased toward immigrant- and Islam-related topics, which did not seem to be the case for the Danish dataset, having entirely different term occurrences than the Norwegian. As mentioned in Chapter 5, reasons for this may include the fact that the datasets were collected from different social media platforms with different users and content, and that they were annotated using different annotators and annotator guidelines. An influencing factor may also be culture, affecting how abusive language is expressed across platforms and countries and what is considered an abusive language by the annotators from the respective countries. Although Norway and Denmark are geographically close and share a common history, how abusive language is expressed in Denmark may not correspond to how it is expressed in Norway. This includes subjects common in hatred debates and how abusive language is linguistically expressed and perceived.

The annotators of the different datasets may have different perceptions of what should be labeled as neutral, offensive, and hateful, based on what is culturally accepted. Additionally, the two datasets used different labeling schemes that might not generalize despite being converted into common representations. Considering the challenges with abusive languages and how there are no standard definitions or perceptions, the assumption that a Danish dataset will generalize to a Norwegian dataset, may have been naive. Based on these reflections and the increase in computational resources required to train on both datasets (described in Section 7.2.2), the disadvantages outweigh the advantages. Therefore, it was decided not to proceed with further pre-processing and testing with the Danish dataset.

Although the experiments with the Danish dataset did not significantly improve model performance on the test dataset, it does not necessarily mean that the model is less successful on real-world abusive language detection. The test dataset used for all experiments is a subset of the Norwegian dataset. Therefore, it is not surprising that expanding the dataset with documents from other datasets introduces noise. Recall that Swamy et al. (2019) stated that a model's performance on the same dataset it is trained on is not indicative of how it performs in real-world problems. Furthermore, they suggested that models should be evaluated on a test dataset representing abusive language as a whole. It can be discussed whether this thesis's test dataset represents real-world abusive language detection problems, as its content is restricted to the sources from which it was collected. Therefore, the experiments in this thesis are limited to reflect how the models predict offensive and hateful language represented in the Norwegian dataset. Evaluating the models' ability to detect abusive language as a whole would require collecting a broader test dataset that was not possible for this thesis due to limited time and resources.

8.1.3 Error Analysis

In general, the results from the experiments indicate that distinguishing neutral, offensive and hateful language is a challenging task, particularly for the abusive classes. The predictions from the overall best performing models for offensive and hateful language detection are presented in the confusion matrices of Tables 8.2 and 8.3, respectively. From Table 8.3 one can observe that the hateful class has the overall most significant proportion of misclassifications of 59%, indicating that the model confuses hateful and offensive language to a large extent. In fact, the model wrongly classifies more of the hateful instances as offensive language than it correctly detects as hate speech. The second-largest misclassification can be observed in Table 8.2 for offensive language detection, where 40% of the offensive instances are wrongly classified as neutral. The misclassification for offensive language is also significant. The large proportions of misclassifications in the predictive classes underlines the difficulty of separating neutral, offensive, and hateful language. For the majority classes, the performance is relatively good for both offensive and hateful language detection.

		Predicted Label	
		Negative	Positive
True Label	Negative	9382	886
	Positive	826	1250

Table 8.2: Confusion matrix for NB-BERT$_{large,uncased}$ trained to detect offensive language. Positive corresponds to the offensive class, and negative corresponds to the neutral class.

To better understand what types of language the best model, NB-BERT$_{large,cased}$,

		Predicted Label	
		Negative	Positive
True Label	Negative	1758	111
	Positive	122	85

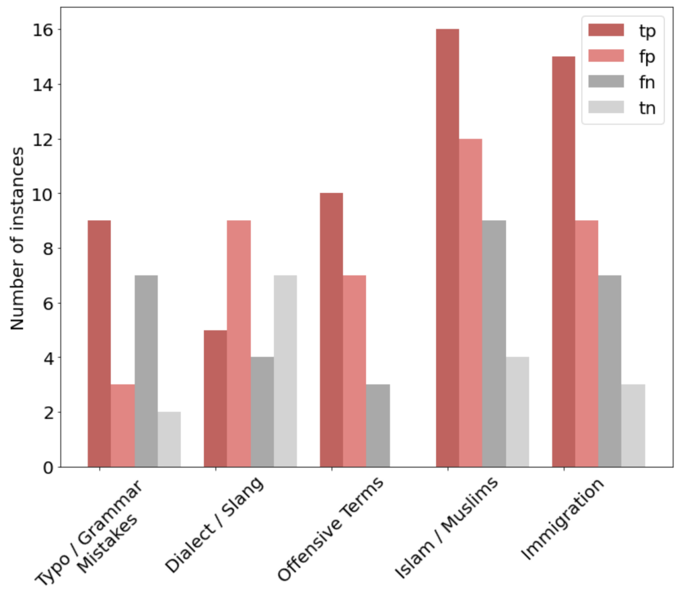
Table 8.3: Confusion matrix for NB-BERT$_{large,uncased}$ trained to detect hate speech. Positive corresponds to the hateful class, and negative corresponds to the offensive (but not hateful) class.

correctly predicts and what it struggles to predict, 85 instances were randomly sampled from all the predictive groups and analyzed for offensive and hateful language detection. The intention is that, as these instances are randomly sampled, they are representative of the typical model predictions.

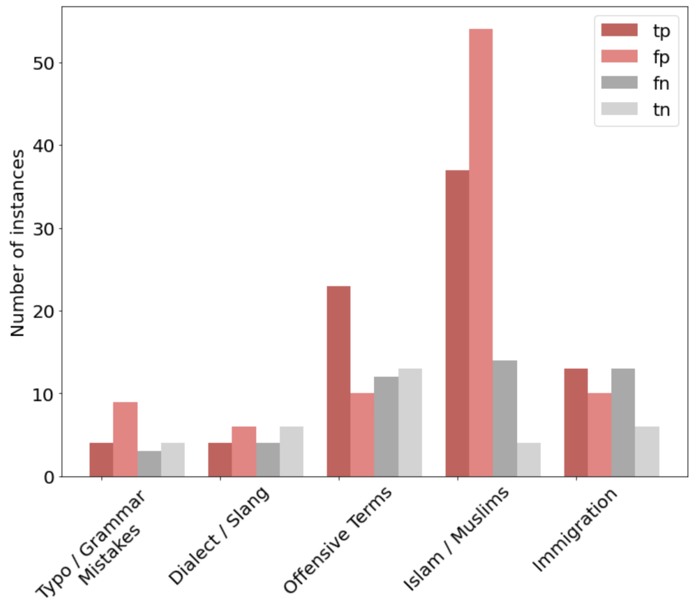
The analysis builds upon the findings from Chapter 5, which left several outstanding questions about the dataset, its content, and possible biases. Five groups of characteristics were defined, and each instance from the predictive classes was analyzed to investigate if it could be categorized within one or multiple groups. The first group is whether an instance contains typos or grammar mistakes, whereas the second is whether it contains dialect or slang beyond Norwegian Bokmål or Nynorsk. One initial hypothesis was that people writing hateful and offensive content are less concerned about their writing form and have more non-standard language, such as grammar mistakes and typos, incorporated into their language. Additionally, as general BERT models are not robust against non-standard language, it is interesting to investigate if any of the predictive groups features such language and whether it affects the predictions.

The following two classes are content-based, with regards to the frequently occurring words from the analysis of Section 5.1. The third class examines whether the instances contain any offensive terms, and the fourth and fifth classes investigate whether the instances are related to Muslims/Islam and immigration, respectively. This is interesting in order to further understand what features categorize the different predictions and how the model can generalize across these patterns. The analysis is also helpful to understand possible biases in the model. Figure 8.1 illustrates the number of instances within the true positives (tp), false positives (fp), true negatives (tn), and false negatives (fn) for offensive and hateful language detection for each category.

The first obvious observation from Figure 8.1 is that a large portion of the predicted hateful instances, in Figure 8.1b, contains topics related to Islam. The fact that the true hateful instances contain such topics is not surprising considering the analysis in Chapter 5 which revealed that this is a common topic of the hateful classes of the Norwegian dataset. However, more than 60% of the 85 instances that were wrongly predicted as being hateful contained Islam-related topics. When considering this degree of misclassifications combined with the findings from the analysis in Chapter 5, it cannot be disregarded that there is a bias from the Norwegian abusive language dataset built into the model towards classifying instances related to Islam as hateful. This observation



(a) Offensive Language Detection



(b) Hate Speech Detection

Figure 8.1: Different characteristics of predicted instances for offensive (a) and hateful (b) language detection. The red bars illustrate instances predicted as positive classes, whereas the gray are negative predictions. The darker contrasts are predictions that originally belong to the positive class, whereas the lighter tones are predictions that originally belong to the negative class.

is further supported by the fact that almost none of the correctly predicted non-hateful instances are related to Islam.

Only 553 hateful instances were present in the training dataset, which may have represented hateful instances towards Muslims to a greater extent than hateful instances as a whole. In that case, the dataset is insufficient for the model to generalize what categorizes hate speech and other offensive language in general. This is a problem originating from the dataset and not the model in itself. The model's behavior of learning these patterns is, in fact, "correct" as it picks up on characteristics in the dataset. However, the bias is rather a problem of the dataset as it does not represent a true pattern of all hateful language. Although such biases can increase detected hateful instances in some cases, they are problematic as they discriminate on what topics are accepted in the public debate and can, therefore, limit the freedom of expression.

Another reason for the confusion between the different classes is the lack of clear separations between hate speech, offensive and neutral language. An analysis of predictions showed that 52% of the originally provocative labeled instances were mistaken as neutral. In contrast, only about 20% of each of the other classes were mistaken as neutral for offensive language detection. This conforms with the observations in the analysis of Chapter 5, where a large portion of the provocative instances also was mistaken as neutral, substantiating some of the vagueness of this class. As the misclassification concerns half of this class, a change in class generalization, i.e., excluding the provocative class from the offensive class, would likely also result in a large number of misclassifications.

For hate speech detection, as much as 65% of the originally moderately hateful instances and 47% of the hateful instances were wrongly predicted as offensive (and not hateful). This emphasizes the confusion between the abusive classes, which was also observed in practice during the qualitative investigation of the predictions. An example is shown below, exemplifying the Norwegian and English versions of an instance which was labeled as offensive by the annotator and incorrectly labeled as hate by the model.

Norwegian	Min mening er jo at mesteparten av danskene burde være enige i at muslimene absolutt burde sendes hjem.
English	My opinion is that most of the Danes should agree that Muslims should definitely be sent home.

The author of the sentence wants to send people out of the country based on their religious beliefs. Similar expressions have been considered hate by annotators of the same dataset, such as the below example which holds a very similar message but was originally labeled as hateful by the annotator and the model.

Norwegian	Muslimer hører ikke hjemme her.
English	Muslims do not belong here.

The lack of a clear distinction in annotations makes it hard for the model to learn patterns in the data. This is consistent with the low inter-annotator agreement in the abusive classes found by Svanes and Gunstad (2020), and the analysis described in Section 5.2.1 which confirmed these results. One may ask how a model can separate well between abusive languages if not even human annotators can separate them well. A way to possibly handle this problem could have been to re-annotate the dataset using expert annotators or crowd-sourcing with full agreement, which accordingly to Waseem (2016) yield better results than using amateurs. However, due to restricted funds and time, this was not prioritized in the work of this thesis.

An observation that is not as prominent but seems to discriminate the predicted categories is the use of offensive terms. Figure 8.1b shows that more of the correctly predicted hateful instances contain offensive terms than the other groups, which may indicate that it is easier to detect hateful language containing offensive slurs than hateful language that do not. This is consistent with Caselli et al. (2020) which emphasized the difficulty of separating implicit abusive language types compared to more explicit offensive slurs.

The same trend is seen to a smaller extent for offensive language detection in Figure 8.1a, which reveals that the true offensive instances contains more offensive terms and that they contain more topics related to immigration and Islam than the other predictions. This makes sense when considering the word clouds presented in Section 5.1 which showed an increased occurrence of immigration- and Islam-related topics as the abusiveness of the documents increased. One can also observe a substantial amount of false negatives for offensive language detection with offensive terms. As most offensive terms would be defined as provocative or offensive by Svanes and Gunstad and Jensen, these observations typically concerns terms in the borderline of swear words such as *idiot* or *weirdo*, which during the investigations of the comments were considered as offensive terms. As previously described in Section 3.1.1, Caselli et al. (2020) noted this as one of the aspects making separation of hate speech and offensive language difficult due to how such terms are perceived differently by annotators. Another example of implicit offensive language that the model was not able to predict is given below.

Norwegian	Navn Navn Ville kanskje vurdert å sterilisere meg snart, om jeg var deg! Men det er jeg jo ikke
English	Name Name Would probably consider sterilizing myself soon, if I were you! But I'm not

Although the sentence at first glimpse has a positive tone, there is an underlying offensive message. The model has likely not understood that the term *sterilizing* in this context is associated with something negative or offensive.

Another interesting observation from Figure 8.1a is that the true offensive instances have a higher occurrence of typos and grammar mistakes, which means that these are more typical features in offensive language than neutral language. This does, to some extent, confirm the initial hypothesis that people writing offensive expressions online have more incorporated errors in their language. Likewise, one can see that the true neutral instances have a higher occurrence of dialect and slang than the true offensive instances.

This was slightly unexpected as the initial thought was that most non-standard language was present in the offensive class, and not a typical feature of neutral language. These observations are useful to understand what features separate the different classes. Despite these observations, the model does not seem to have learned these features, as there are almost as many true as false predictions within these categories. This might indicate that the model is less prone to predict non-standard language correctly, which makes sense considering the fact that the model is pre-trained mainly on standard language. Other types of social media language the model was not able to predict correctly include instances that manipulated words to a non-standard representation such as writing *Muslim* as a mouse emoji + lim at the end, or exchanging letters in offensive terms with other symbols such as "f!!**ck", which is non-standard forms that the pre-trained language understanding does not assist detecting.

This error analysis reveals that several of the problems noted by Svanes and Gunstad and Jensen regarding the dataset are still present, despite the introduction of Transformer-based approaches. This includes the confusion between neutral and offensive and offensive and hateful language and the biased predictions towards the neutral class and towards Islam-related topics. Moreover, the analysis confirms with the research by Caselli et al. which showed that explicit abusive language is easier to predict than implicit. Additionally, several interesting observations were made regarding the use of non-standard language and how grammar mistakes and typos seem to be a feature of documents containing offensive and hateful language, and how dialects and slang were more typical features of neutral language. Despite this, the NB-BERT model struggled to detect this feature based on its pre-trained corpus, motivating further investigations on this topic.

8.2 Discussion

In the beginning of this thesis, Section 1.2 formulated an overall research goal and three research questions designed to approach the goal. The three research questions are discussed below, followed by an evaluation of the goal of this thesis.

Research question 1 *How well does the Bidirectional Encoder Representations from Transformers (BERT) – pre-trained on a large Norwegian corpus – detect offensive and hateful Norwegian language in social media?*

The first research question laid the basis for the experiments of this thesis as essentially all experiments conducted used a BERT model pre-trained on a large Norwegian corpus. Experiment 1 trained five different BERT models on the tasks of separating between neutral and offensive language and hateful and other offensive language. The top results achieved from using the Transformer-based BERT models significantly outperformed existing approaches for both offensive and hateful language detection. The results indicate that the general language understanding obtained during pre-training of BERT models helps the model recognizing more offensive and hateful instances than the traditional neural and non-neural methods.

Despite the increased performance compared to previous approaches on the dataset, the results for the predictive classes revealed that the model still tends to confuse offensive and hateful language and, in several cases, offensive and neutral language. For the best overall system, the recall for the hateful class was poor and showed that only 37% of the true hateful instances were detected, a recurring problem for abusive language detection. However, based on the analyses conducted, this seems to be a problem originating from the dataset rather than model.

Furthermore, the error analysis in Section 8.1.3 revealed that the model seemed to be biased towards classifying instances related to Islam as more abusive. This also originates from a bias of the Norwegian abusive dataset and ambiguity of what separates hate from other offensive language. Such biases pose problems when applying the model in real-world abusive language detection, as it will filter out content that is not necessarily unintended which does not preserve the freedom of expression.

Research question 2 *How do the different BERT models pre-trained on the Norwegian language compare when applied to detecting offensive and hateful social media language?*

The second research question was defined to select the model to proceed with for the remaining research of this thesis. The results from Experiment 1 showed that the NB-BERT$_{large,cased}$ model, without doubt, outperformed the other models for both offensive and hateful language detection. This is likely because of its large pre-training corpora compared to the other models and the larger number of network parameters, allowing it to learn more complex patterns in the data. The second-best model proved to be NorBERT, followed by mBERT, which is consistent with the decreasing size of their pre-training corpora. The model ranking is the same as what was overall achieved by both Kummervold et al. (2021) and Kutuzov et al. (2021) for several other NLP tasks.

Research question 3 *What are the effects of applying different data manipulation techniques to the Norwegian dataset when detecting offensive and hateful social media language?*

To cope with the unbalanced nature and the few abusive instances of the Norwegian dataset, two data manipulating techniques were tested for both offensive and hateful language detection. The first technique tested undersampling the dataset by reducing the number of instances in the majority class to change the dataset's class distribution. Undersampling significantly increased the performance of the predictive classes at the cost of a drop in overall system performance. The second technique tested expanding the Norwegian dataset with instances from the Danish dataset, which showed no significant increase in performance for offensive language detection and a drop in performance for hate speech detection. The error analysis combined with previous observations of the dataset revealed several differences between the Norwegian and Danish datasets, which was likely why the model did not generalize well across them.

Based on the three research questions and the research conducted to answer them, the research goal can finally be addressed.

Goal *Investigate how to accurately detect and distinguish neutral, offensive, and hateful Norwegian language in social media.*

A literature review was conducted to get a broad perspective on the field of abusive language detection generally and for Norwegian. Based on successful existing English and Danish abusive language detection approaches and the recent introduction of Norwegian Transformer-based models, multiple BERT models were fine-tuned to detect Norwegian offensive and hateful language. As a result, the NB-BERT model significantly outperformed the existing systems on the Norwegian dataset with ten macro F1 points for offensive language detection and more than six macro F1 points for hate speech detection, which demonstrates an overall successful approach considering the quality of the dataset used.

Although the approach outperformed existing approaches, there was still a large portion of misclassification, particularly within the offensive and hateful categories for offensive and hateful language detection, respectively. The approach can thus not be said to detect and distinguish neutral, offensive, and hateful Norwegian language *accurately*, as was the goal of this thesis. Nevertheless, the research conducted in this thesis significantly improved upon existing approaches and identified areas with potential improvement, which motivates further research on Norwegian abusive language detection using Transformer-based approaches.

9 Conclusion and Future Work

This chapter concludes upon the work conducted in this thesis and presents the main contributions and their significance to the field of abusive language detection. The final section presents ideas to motivate further research in the field and possible improvements to the work conducted in this thesis.

9.1 Conclusion

As social media platforms have grown in terms of user-generated content, there is a rising need to develop automatic systems that can assist the monitoring of hateful content. This thesis aimed to investigate how neutral, offensive, and hateful Norwegian language can be accurately detected and distinguished. The literature review conducted show that the interest in abusive language detection has multiplied over the past years. The introduction of Transformer-based models has improved the language understanding of models and thereby their ability to detect and distinguish abusive language types. However, due to the lack of available datasets and technologies, little research has been conducted to detect abusive Norwegian language. Only one abusive language dataset is currently available for Norwegian. This dataset is highly unbalanced, with few abusive instances. Based on this dataset, previous research by Svanes and Gunstad (2020) and Jensen (2020) showed to be biased towards the neutral class and struggled to detect abusive instances partly due to the skewed distribution and the few abusive instances of their dataset.

Based on successful Danish and English abusive language detection approaches, this thesis developed a system to detect offensive and hateful Norwegian language using the recently introduced Transformer-based BERT models. The results show that the NB-BERT$_{large,cased}$ model with the most extensive pre-training corpus and network parameters significantly outperformed the other BERT models, and the only existing solutions by Svanes and Gunstad and Jensen with more than ten macro F1 points for offensive language detection and six macro F1 points for hate speech detection.

Despite the improved performance, the error analysis conducted in Chapter 8 shows that the approach struggled to detect and distinguish offensive and hateful language. The most pressing issues seem to be related to the dataset, which has a low inter-annotator agreement and is biased towards Islam-related topics, particularly for the hateful instances – leading to biased predictions. Based on the ambiguity between the abusive language types, one may question how it can be expected for computational models to distinguish between abusive language types if human annotators have difficulties doing the same. Moreover, the error analysis emphasize several challenges of social media language, making it difficult to separate the categories, including implicit abusive language and non-standard language such as typos, grammar mistakes, and abbreviations. Although introducing Transformer-based models to Norwegian abusive language detection is a step in the right direction, the performance is insufficient to bring the system into practical use. It is evident from the experiments that there is still a way to go before the neutral, offensive, and hateful Norwegian language is accurately distinguished using computational approaches to detect hate speech.

9.2 Contributions

This section presents the most significant contributions of this thesis and builds upon the list that can be found in Section 1.4.

Quantitative and Qualitative Dataset Analysis
The dataset analysis of Chapter 5 presented several characteristics of the Norwegian and Danish datasets used in this thesis to outline features and challenges of the datasets. An annotation experiment performed on a subset of the Norwegian dataset, revealed high disagreement between the three participating annotators. The analysis substantiated the need for a higher-quality Norwegian abusive language dataset with more abusive instances and a more apparent separation between the abusive classes. A new label scheme was proposed based on the analysis to cope with the vagueness of the dataset.

Optimized BERT Models for Norwegian Abusive Language Detection
An architecture overview and implementation of the Norwegian BERT models were presented with a corresponding ranking and comparison. This is the first implementation of BERT models for Norwegian offensive and hateful language detection. The best model, NB-BERT$_{large,cased}$ outperformed previous approaches significantly. Based on the literature review and the observation of variance in the models, all results were averaged over five fine-tuning sessions. This increases reliability and substantiates reproducibility of the results for future research. Additionally, all models were optimized using an extensive search, where all optimized hyperparameters were presented and can be used in future work. A broader hyperparameter search was also performed for offensive and hateful language detection, which confirmed that searching a broader hyperparameter space outside the values proposed by Devlin et al. (2019) was unnecessary for the Norwegian dataset.

Prediction and Error Analysis
The discussion of Chapter 8 presented a throughout prediction error analysis of the NB-BERT$_{large,cased}$ model for offensive and hateful language detection. The analysis revealed biases propagating from the dataset to the model used and what features categorized the different predictions and dataset classes. This corresponded to the initial qualitative and quantitative analysis findings and further substantiated challenges with

the dataset. Additionally, the analysis contributed to understanding what types of content the NB-BERT model struggled to predict and what types were easily predictable.

9.3 Future Work

The final section of this thesis proposes future work to possibly improve upon the work conducted in this thesis. The work is based on the literature review, experiments, and analyses conducted and concern the improvement of the dataset and models used in this thesis.

9.3.1 Improving the Norwegian Abusive Language Dataset

Previous work (Svanes and Gunstad, 2020; Jensen, 2020) and analyses conducted in this thesis revealed several challenges with the Norwegian abusive language dataset by Svanes and Gunstad and Jensen (2020). The main challenges can be summarized as 1) few abusive instances, particularly for the hateful class, 2) abusive instances are biased towards Islam-related topics, and 3) low inter-annotator agreement for the abusive categories resulting in confusion among them. The first problem can be solved by expanding the dataset by collecting more instances (including abusive). As discussed in Section 8.1.2 changing the class distribution of the training set will affect how the model predicts instances and should therefore be considered carefully. Maintaining a natural distribution equal to what typically occurs in social media is often preferred for the model to do a more educated guess in cases of uncertainty.

The second challenge requires ensuring that Islam-related topics are present in all classes. That is, as Islam is not a direct indicator that a message contains hate, more discussions around Islam with a neutral tone should be added to the dataset. This can be done by collecting data from a more diverse specter of sources that not only concerns immigrant- and Islam-critical debates. Moreover, for the model to better recognize other hateful content than what is related to Islam, hateful content within other topics will likely also be collected when considering diverse sources. Finally, the confusion between different classes is a common problem as described in Section 3.1.1. Therefore, the annotation process should be done from scratch optimally by expert annotators or multiple amateurs with full agreement, according to Waseem (2016). This can, for instance, be done by hiring crowdsource workers or a student group with multiple annotators for each instance.

9.3.2 Further Pre-training NB-BERT on Social Media Data

The prediction and error analysis of Chapter 8 revealed that the best NB-BERT model struggled to understand certain types of non-standard social media content. The analysis showed that offensive content had more typos and grammar mistakes than neutral content and that neutral content used more dialects and slang than offensive content. However, the model did not learn these features, which is likely due to the lack of understanding of non-standard content, meaning content on which the model is not pre-trained. To extend the language understanding of NB-BERT it could be beneficial to further pre-train it on social media data consisting of more typos, grammar mistakes, slang, and dialects. As described in Chapter 4, Caselli et al. (2020) showed that this could be beneficial when training on enough data. Training the model on general social media data and not abusive language data exclusively also increases the model's application beyond abusive language detection to other NLP tasks concerning social media data.

9.3.3 Multi-Genre Natural Language Inference with NB-BERT

Multi-Genre Natural Language Inference (MNLI) is the task of predicting whether a premise entails or contradicts a hypothesis (Wang et al., 2018). In certain cases, predicting whether a document belongs to an exact class can be difficult and sometimes even impossible. Therefore, an alternative is to reformulate the classification question into an MNLI hypothesis, a method proposed by Yin et al. (2019). The creators of NB-BERT (Kummervold et al., 2021) recently developed an NB-BERT$_{base}$ model fine-tuned on Norwegian machine-translated MNLI, which could be interesting to investigate for the case of detecting Norwegian abusive language. One approach is to formulate MNLI Hypotheses that feature some of the characteristics of abusive language, such as "this text contains offensive terms," if yes, then the text is offensive, and so on. The model is available through Hugging Face's inference API[1].

9.3.4 Implementing Other Transfer-learned Approaches and Ensemble Models

This thesis focused solely on BERT models with support for the Norwegian language. As mentioned in Section 4.3.1, another pre-trained neural architecture, NorELMo, was recently released by Kutuzov et al. (2021) and could be interesting to test for Norwegian abusive language detection. Kutuzov et al. (2021) reported that the model achieved state-of-the-art results for several NLP tasks with the advantage of using an order of magnitude less computational resources. There is no doubt that the experiments performed in this thesis required extensive computational resources and that reducing this requirement can increase the resources available for work on other aspects of detecting abusive language detection.

Moreover, the literature review presented several successful Transformer-based approaches which have outperformed the BERT model for several NLP tasks for English and other languages, such as ALBERT, ERNIE, and Open AI GPT. Possible future work is to implement these models pre-trained on a Norwegian corpus to investigate if they can outperform BERT also for Norwegian abusive language detection.

Another suggestion is to combine the BERT models used in this thesis into an ensemble model. Research from the shared tasks OffensEval 2019 and 2020 (Zampieri et al., 2019a, 2020) showed that ensembles of Transformer-based approaches, including BERT, were successful for offensive language detection and target identification. Combining the strength of several different models can produce superior predictions with less bias and variance by combining the patterns learned from each model.

[1]`https://huggingface.co/NbAiLab/nb-bert-base-mnli`

Bibliography

Vilde Roland Arntzen. A Literature Review on Detecting Hateful Language in Social Media. Specialization Project Report in TDT4501, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, 2020.

Ron Artstein. Inter-annotator Agreement. In *Handbook of Linguistic Annotation*, pages 297–313. Springer Netherlands, Dordrecht, Netherlands, June 2017. ISBN 978-94-024-0879-9 978-94-024-0881-2. doi: 10.1007/978-94-024-0881-2_11. URL `http://link.springer.com/10.1007/978-94-024-0881-2_11`.

Oskar Austegard. List of Dirty, Naughty, Obscene or Otherwise Bad Words. `https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words/blob/master/no`, 2019. Accessed: 29.03.2021.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, San Diego, CA, USA, May 2015. URL `http://arxiv.org/abs/1409.0473`.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research 3*, 3(null): 1137–1155, March 2003. ISSN 1532-4435. URL `https://dl.acm.org/doi/10.5555/944919.944966`.

BotXO Ltd. Nordic BERT: Pre-trained Nordic Models for BERT. `https://github.com/botxo/nordic_bert`, 2021. Accessed: 27.05.2021.

Samuel Brody and Nicholas Diakopoulos. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 562–570, Edinburgh, Scotland, UK., Jul 2011. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D11-1052`.

Peter Cook Bulukin. Classifying Hateful and Offensive Language Across Datasets and Domains. MSc Thesis, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, Jun 2021.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. HateBERT: Retraining BERT for Abusive Language Detection in English. *ArXiv*, art. arXiv:2010.12472, Oct 2020. URL https://ui.adsabs.harvard.edu/abs/2020arXiv201012472C.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, Inga Kartoziya, and Michael Granitzer. I Feel Offended, Don't Be Abusive! Implicit/Explicit Messages in Offensive and Abusive Language. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6193–6202, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.760.

Christopher Cieri, Mike Maxwell, Stephanie Strassel, and Jennifer Tracey. Selection Criteria for Low Resource Language Programs. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4543–4549, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1720.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. *CoRR*, abs/1911.02116, 2019. URL http://arxiv.org/abs/1911.02116.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, May 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *CoRR*, abs/2002.06305, 2020. URL https://arxiv.org/abs/2002.06305.

Helga Eggebø and Elisabeth Stubberud. Hatefulle Ytringer Delrapport 2: Forskning på Hat og Diskriminering. *Rapport 2016:15*, 2016. URL https://samfunnsforskning.brage.unit.no/samfunnsforskning-xmlui/handle/11250/2442447.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. Beyond English-Centric Multilingual Machine Translation. *CoRR*, abs/2010.11125, 2020. URL `https://arxiv.org/abs/2010.11125`.

Paula Fortuna and Sérgio Nunes. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, 51(4), July 2018. ISSN 0360-0300. doi: 10.1145/3232676. URL `https://doi.org/10.1145/3232676`.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large Scale Crowdsourcing and Characterization of Twitter Abusive behavior. *CoRR*, abs/1802.00393, 2018. URL `http://arxiv.org/abs/1802.00393`.

Google LLC. YouTube Community Guidelines Enforcement. `https://transparencyreport.google.com/youtube-policy/removals?hl=en`, 2020. Accessed: 24.03.2021.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. All You Need is "Love": Evading Hate Speech Detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, AISec '18, page 2–12, New York, NY USA, 2018. Association for Computing Machinery. ISBN 9781450360043. doi: 10.1145/3270101.3270103. URL `https://doi.org/10.1145/3270101.3270103`.

Simon S. Haykin. *Neural Networks and Learning Machines.* Pearson Education, Upper Saddle River, NJ, third edition, 2009. URL `https://cours.etsmtl.ca/sys843/REFS/Books/ebook_Haykin09.pdf`.

Dan Hendrycks and Kevin Gimpel. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units. *CoRR*, abs/1606.08415, 2016. URL `http://arxiv.org/abs/1606.08415`.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Hugging Face inc. The AI community building the future. `https://huggingface.co/`, 2021. Accessed: 05.06.2021.

Vebjørn Isaksen. Detecting Hateful and Offensive Language with Transfer-Learned Models. MSc Thesis, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, Jun 2019. URL `https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2634455/no.ntnu:inspera:36079153:36099951.pdf?sequence=1`.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013. URL `https://faculty.marshall.usc.edu/gareth-james/ISL/`.

Maria Hilmo Jensen. Detecting hateful utterances using an anomaly detection approach. MSc Thesis, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, Jun 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014. URL http://arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Anders Kofod-Petersen. How to do a Structured Literature Review in Computer Science. Technical report, Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, 2018. URL https://research.idi.ntnu.no/aimasters/files/SLR_HowTo2018.pdf.

Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, New York, NY, 2013. ISBN 9781461468493 1461468493 1461468485 9781461468486. URL http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/.

Kulturdepertamentet. Hatefulle ytringer. *Regjeringen: Likestilling og Inkludering*, Sep 2019. URL https://www.regjeringen.no/no/tema/likestilling-og-inkludering/likestilling-og-inkludering/regjeringens-arbeid-mot-hatefulle-ytringer/id2510986/. Accessed: 27.05.2021.

Per Egil Kummervold, Javier de la Rosa, Freddy Wetjen, and Svein Arne Brygfjeld. Operationalizing a National Digital Library: The Case for a Norwegian Transformer Model. *CoRR*, abs/2104.09617, 2021. URL https://arxiv.org/abs/2104.09617.

Andrey Kutuzov, Jeremy Barnes, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. Large-Scale Contextualised Language Modelling for Norwegian. *CoRR*, abs/2104.06546, 2021. URL https://arxiv.org/abs/2104.06546.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for Self-supervised Learning of Language Representations. *CoRR*, abs/1909.11942, 2019. URL http://arxiv.org/abs/1909.11942.

J. Richard Landis and Gary G. Koch. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1), 1977. doi: 10.2307/2529310. URL https://www.jstor.org/stable/2529310.

Yann LeCun. Generalization and network design strategies. *Connectionism in perspective*, 19, 1989. URL https://ci.nii.ac.jp/naid/10008946620/en/.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkgaETNtDB.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the Hyperparameters for Fine-tuning. *CoRR*, abs/2002.11770, 2020. URL `https://arxiv.org/abs/2002.11770`.

Likestillings- og diskrimineringsombudet. Hatefulle ytringer i offentlig debatt på nett. `https://www.ldo.no/globalassets/_ldo_2019/arkiv/publikasjonsarkiv/fb-og-hatytring/ldo-fb_rapporten_2018.pdf`, 2018. Accessed: 24.03.2021.

Zachary Chase Lipton. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR*, abs/1506.00019, 2015. URL `http://arxiv.org/abs/1506.00019`.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the Difficulty of Training Transformers. *CoRR*, abs/2004.08249, 2020. URL `https://arxiv.org/abs/2004.08249`.

Edward Loper and Steven Bird. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118108.1118117. URL `https://doi.org/10.3115/1118108.1118117`.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL `https://www.aclweb.org/anthology/D15-1166`.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a. URL `https://conference.scipy.org/proceedings/scipy2010/pdfs/mckinney.pdf`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc. URL `https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf`.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong baselines. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=nzpLWnVAyah`.

Andreas Mueller. Wordcloud: A Word Cloud Generator in Python. `https://github.com/amueller/word_cloud`, 2021. Accessed: 05.06.2021.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 145–153, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883062. URL `https://doi.org/10.1145/2872427.2883062`.

Marc Pàmies, Emily Öhman, Kaisla Kajava, and Jörg Tiedemann. LT@Helsinki at SemEval-2020 task 12: Multilingual or language-specific BERT? In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1569–1575, Barcelona (online), December 2020. International Committee for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.semeval-1.205`.

Endang Wahyu Pamungkas, Valerio Basile, and V. Patti. Stance Classification for Rumour Analysis in Twitter: Exploiting Affective Information and Conversation Structure. *ArXiv*, abs/1901.01911, Jan 2019. URL `https://arxiv.org/pdf/1901.01911.pdf`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. URL `https://papers.nips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Oliver Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

Telmo Pires, Eva Schlinger, and Dan Garrette. How Multilingual is Multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL `https://www.aclweb.org/anthology/P19-1493`.

Poetry. A Tool for Dependency Management in Python. `https://python-poetry.org/`, 2018. Accessed: 05.06.2021.

Python Software Foundation. Python Package Index - PyPi. `https://pypi.org/`, 2021. Accessed: 05.06.2021.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 2019. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

Nils Reimers. EasyNMT - Easy to use, state-of-the-art Neural Machine Translation. `https://github.com/UKPLab/EasyNMT`, 2021. Accessed: 05.06.2021.

Sarah T. Roberts. *Behind the Screen: Content Moderation in the Shadows of Social Media*. Yale University Press, 2019. ISBN 9780300235883. URL `http://www.jstor.org/stable/j.ctvhrcz0v`.

Guy Rosen. Facebook Community Standards Enforcement Report. `https://about.fb.com/news/2020/08/community-standards-enforcement-report-aug-2020/`, Aug 2020. Accessed: 24.03.2021.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. A Large-Scale Semi-Supervised Dataset for Offensive Language Identification. *ArXiv*, abs/2004.14454, Apr 2020. URL `https://arxiv.org/pdf/2004.14454.pdf`.

Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. Measuring the Reliability of Hate Speech Annotations: The Case of the European Refugee Crisis. *CoRR*, abs/1701.08118, Jan 2017. URL `http://arxiv.org/abs/1701.08118`.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976. URL `https://www.nature.com/articles/323533a0`.

Alessandro Seganti, Helena Sobol, Iryna Orlova, Hannam Kim, Jakub Staniszewski, Tymoteusz Krumholc, and Krystian Koziel. NLPR@SRPOL at SemEval-2019 Task 6 and Task 5: Linguistically enhanced deep learning offensive sentence classifier. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 712–721, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2126. URL `https://www.aclweb.org/anthology/S19-2126`.

Anime Sekai and Zhymabek Roman. Translatepy: An Aggregation of Multiple Translation APIs. `https://github.com/Animenosekai/translate`, 2021. Accessed: 05.06.2021.

Gudbjartur Ingi Sigurbergsson and Leon Derczynski. Offensive Language and Hate Speech Detection for Danish. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3498–3508, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://www.aclweb.org/anthology/2020.lrec-1.430`.

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *CoRR*, abs/1912.05848, 2019. URL `http://arxiv.org/abs/1912.05848`.

Språkbanken. Norsk aviskorpus, 2019. URL `https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-4/`. Accessed: 22.04.2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A Continual Pre-training Framework for Language Understanding. *CoRR*, abs/1907.12412, 2019. URL `http://arxiv.org/abs/1907.12412`.

Marie Andreassen Svanes and Tora Seim Gunstad. Detecting and Grading Hateful Utterances in the Norwegian Language. MSc Thesis, Deptartment of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, Jun 2020.

Steve Durairaj Swamy, Anupam Jamatia, and Björn Gambäck. Studying Generalisability across Abusive Language Detection Datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 940–950, Hong Kong, China, Nov 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1088. URL `https://www.aclweb.org/anthology/K19-1088`.

Carl-Erik Särndal, Bengt Swensson, and Jan Wretman. *Model Assisted Survey Sampling (Springer Series in Statistics)*. Springer, 2003. ISBN 0387406204. URL `http://www.amazon.com/Assisted-Survey-Sampling-Springer-Statistics/dp/0387406204/sr=8-1/qid=1172587067/ref=pd_bbs_sr_1/103-2111122-6886251?ie=UTF8&s=books`.

Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – Building open translation services for the World. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, November 2020. European Association for Machine Translation. URL `https://www.aclweb.org/anthology/2020.eamt-1.61`.

Twitter inc. Rules Enforcement. `https://transparency.twitter.com/en/reports/rules-enforcement.html#2019-jul-dec`, Aug 2020. Accessed: 25.03.2021.

Twitter inc. About offensive content. `https://help.twitter.com/en/safety-and-security/offensive-tweets-and-content`, 2021. Accessed: 02.04.2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In

Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Erik Velldal, Lilja Øvrelid, and Petter Hohle. Joint UD Parsing of Norwegian Bokmål and Nynorsk. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 1–10, Gothenburg, Sweden, May 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W17-0201`.

Lars S. Vikør. Norwegian: Bokmål vs. Nynorsk, 2015. URL `https://www.sprakradet.no/Vi-og-vart/Om-oss/English-and-other-languages/English/norwegian-bokmal-vs.-nynorsk/`. Accessed: 09.04.2021.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL `https://www.aclweb.org/anthology/W18-5446`.

Shuohuan Wang, Jiaxiang Liu, Xuan Ouyang, and Yu Sun. Galileo at SemEval-2020 task 12: Multi-lingual Learning for Offensive Language Identification Using Pre-trained Language Models. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1448–1455, Barcelona (online), December 2020. International Committee for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.semeval-1.189`.

Zeerak Waseem. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas, Nov 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-5618. URL `https://www.aclweb.org/anthology/W16-5618`.

Tor Ragnar Weidling and Magne Njåstad. Norge under dansk styre (1537-1814), 2020. URL `https://snl.no/Norge_under_dansk_styre_-_1537-1814`. Accessed: 09.04.2021.

Gregor Wiedemann, Seid Muhie, and Chris Biemann. UHH-LT at SemEval-2020 Task 12: Fine-Tuning of Pre-Trained Transformer Networks for Offensive Language Detection. In *Proceedings of the International Workshop on Semantic Evaluation (SemEval)*, Apr 2020. URL `https://arxiv.org/abs/2004.11493`.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. *CoRR*, abs/1909.00161, 2019. URL http://arxiv.org/abs/1909.00161.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes, 2020. URL https://arxiv.org/abs/1904.00962.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA, Jun 2019a. Association for Computational Linguistics. doi: 10.18653/v1/S19-2010. URL https://www.aclweb.org/anthology/S19-2010.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420, Minneapolis, Minnesota, Jun 2019b. Association for Computational Linguistics. doi: 10.18653/v1/N19-1144. URL https://www.aclweb.org/anthology/N19-1144.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of the International Workshop on Semantic Evaluation (SemEval-2020)*, Sep 2020. URL https://arxiv.org/abs/2006.07235.

Appendices

Appendix 1 presents the results from the hyperparameter searches performed in this thesis, whereas Appendix 2 presents the annotator guidelines created by Svanes and Gunstad (2020) and Jensen (2020). The annotator guidelines were used to annotate the Norwegian dataset in this thesis. The guidelines are adopted and reprinted with permission from Maria Hilmo Jensen.

1 Hyperparameter Optimization Tables

The following tables show the results for each hyperparameter combination for the different BERT models trained on the Norwegian abusive language dataset. Firstly, all tables for offensive language detection are presented, followed by the tables for hate speech detection. The values are macro F1 scores averaged over 5-fold cross-validation. The results reported in these tables are, as described in Section 6.3.5, averaged over each fold of the training set, which is completely separated from the test dataset to ensure an unbiased selection of hyperparameters. The best values for each batch size are marked in bold, and the best overall value is marked with an underline.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.678	0.685	0.689
3	0.691	**0.693**	0.688
2	0.686	0.668	0.675
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.682	0.685	0.686
3	0.687	0.691	<u>**0.696**</u>
2	0.683	0.691	0.685

Table 1: Offensive language detection - hyperparameter search mBERT$_{base,cased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.691	0.685	0.696
3	**<u>0.700</u>**	0.696	0.694
2	0.690	0.688	0.682
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.687	0.695	0.692
3	0.697	0.697	**0.698**
2	0.693	0.693	0.692

Table 2: Offensive language detection - hyperparameter search mBERT$_{base,uncased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.709	0.718	0.716
3	**<u>0.719</u>**	0.716	0.712
2	0.708	0.704	0.694
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.709	0.711	**0.718**
3	0.710	**0.718**	0.716
2	0.699	0.703	0.702

Table 3: Offensive language detection - hyperparameter search NorBERT$_{base,cased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.730	0.736	**0.737**
3	0.731	0.735	**0.737**
2	0.735	0.736	0.731
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.726	0.733	0.734
3	0.730	0.736	0.741
2	0.737	<u>**0.742**</u>	0.740

Table 4: Offensive language detection - hyperparameter search NB-BERT$_{base,cased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.748	0.755	0.751
3	0.753	<u>**0.761**</u>	0.757
2	0.748	0.754	0.759
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.744	0.746	0.754
3	0.748	0.692	**0.756**
2	0.751	0.754	0.755

Table 5: Offensive language detection - hyperparameter search NB-BERT$_{large,uncased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.612	0.620	0.597
3	**<u>0.624</u>**	0.620	0.604
2	0.577	0.544	0.513
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.599	0.610	0.621
3	0.556	**0.624**	0.614
2	0.479	0.546	0.569

Table 6: Hate speech detection - hyperparameter search mBERT$_{base,cased}$.

Batch Size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.629	0.627	0.638
3	0.593	0.620	0.555
2	0.531	0.533	0.499
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.624	0.624	0.633
3	0.645	0.628	0.644
2	0.488	0.571	0.571

Table 7: Hate speech detection - hyperparameter search mBERT$_{base,uncased}$.

Batch size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	**0.644**	0.596	0.562
3	0.627	0.585	0.566
2	0.563	0.522	0.471
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	<u>**0.648**</u>	0.617	0.618
3	0.639	0.632	0.610
2	0.614	0.590	0.544

Table 8: Hate speech detection - hyperparameter search NorBERT$_{base,cased}$.

Batch size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.634	0.643	0.646
3	0.616	0.653	**0.654**
2	0.502	0.484	0.501
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.608	0.640	0.654
3	0.645	0.655	<u>**0.656**</u>
2	0.604	0.578	0.610

Table 9: Hate speech detection - hyperparameter search NB-BERT$_{base,cased}$.

Batch size: 32			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.667	0.685	0.645
3	0.674	<u>**0.692**</u>	0.681
2	0.642	0.553	0.513
Batch Size: 16			
# Epochs / Learn. Rate	**5e-5**	**3e-5**	**2e-5**
4	0.670	0.678	0.674
3	0.675	**0.679**	0.677
2	0.649	0.642	0.620

Table 10: Hate speech detection - hyperparameter search NB-BERT$_{large,uncased}$.

2 Annotation Guidelines

This appendix presents the guidelines provided to the outside annotators as a part of the user-based annotation of the Norwegian dataset. The guidelines are written in Norwegian, but a brief English version is given in Svanes and Gunstad (2020) and Jensen (2020).

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	Alt annet

Eksempler på nøytrale kommentarer:

1. Artig at det å påpeke hva som skjer i våre gater har blitt en forbrytelse. Også en smule morsomt at 1.7 milliarder muslimer kalles en minoritet mot Norges 4 millioner nordmenn
2. blir forbanna over at listhaug sier at spagetti er en norsk verdi eller noe sånt
3. Tenker begge kjønn ghoster men at menn som regel er de som kommer med møkkete svar. Uansett føles det vondt å bli ghostet, man er jo tross alt håpefull når man matcher med noen. Jeg er hvertfall det.
4. Gud hjelpes for en gjeng

Mange kommentarer kan ofte oppleves å ikke være nøytrale hvis man tenker på kontekst eller andre underliggende faktorer. Det er viktig i dette prosjektet å analysere og annotere hver enkelt kommentar separert, og heller ikke ilegge forfatteren meninger som ikke står eksplisitt. Med dette mener vi ikke at ironi og sarkasme alltid skal tolkes nøytralt, men at kommentarer ikke skal tolkes i verste mening hvis den hadde blitt rettet mot en bestemt type gruppe eller i en viss setting. Les den som den står med andre ord.

2	Provoserende	En ytring blir definert som provoserende om den inneholder et aggressivt språk for å uttrykke en mening eller kan oppfattes upassende. Dette inkluderer bruk av banneord, hersketeknikk, sarkasme og ironi for å dekredibilitere motstanderen.

Eksempler på provoserende kommentarer:

1. Oi så flink du er til å forenkle fremstillinger. Er det derfor du ellers ser samfunnet svart/hvitt? Media skriver om noe = løgn Gjevjon/Steigan etc skriver noe = ufeilbarlige sannheter... Mester i nyanser og dyptenkning er du vel ikke
2. @USER Hvordan er det mulig å hate mennesker så intenst? Hva i all verden har disse menneskene gjort denne gjengen? FRP er jo et råttent og superegoistisk parti. Det er høyreekstreme som står for terroren i dette landet ikke muslimene. Håper at folk begynner å se det nå
3. Ehhh ... om du er religiøs er du hjernevasket. Islam er en eneste stor sekt, hva annet? Sekten holder 1,5+ milliarder mennesker nedtrykt i patriarkalske strukturer, underutvikling og uvitenhet, ekstrem fattigdom ... hva godt gjør d for menneskeheten å sitte å pugge koranvers?
4. Nok en gang viser det seg at islamkritikk ikke tåles, og er dødsens farlig. Og at feminister og venstresiden™ heller bøyer seg for muslimer, enn å forsvare ytringsfrihet og likestilling.,#JeSuisMila

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

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

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

Eksempler på støtende kommentarer:

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

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

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

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.

Vilde Roland Arntzen

Detecting Norwegian Abusive Language in Social Media with Transformer-based Models

NTNU
Norwegian University of
Science and Technology