Master's thesis

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

Peter Cook Bulukin

Classifying Hateful and Offensive Language Across Datasets and Domains

A Comparison of Various Transformer-Based Models

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

NTNU
Norwegian University of
Science and Technology

Peter Cook Bulukin

Classifying Hateful and Offensive Language Across Datasets and Domains

A Comparison of Various Transformer-Based Models

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

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

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis explores the compatibility of popular offensive language datasets while utilizing models pre-trained on large textual corpora. Training on different datasets iteratively is found to be very difficult due to models forgetting previous knowledge, while merging and shuffling datasets yield results comparable with normal in-domain training. It is difficult to tell if good performance on the merged dataset is due to models learning to identify different datasets, or if the definitions of hateful and offensive language used in the individual annotation methodologies are actually compatible. Recent models' abilities to separate hateful from offensive language using in- and cross-domain data were investigated in two phases. A preliminary study was conducted, where a system based on A Lite BERT (ALBERT) was implemented to separate offensive and normal language in order to test the effects of reducing memory consumption from the lately popular Bidirectional Encoder Representations from Transformers (BERT). This was followed by experiments utilizing five implemented systems, of which two were based on a model that retains factual knowledge called Enhanced Representation through kNowledge IntEgration (ERNIE), and the remaining three were based on ALBERT. Experiments were conducted using three datasets, where all annotation schemes were mapped to 'Hateful', 'Offensive', or 'Neither' in order to facilitate for cross-dataset training and evaluation. The results suggest that the knowledge-integrating models perform better than the relatively light model on both in- and cross-domain experiments. An extensive discussion of the compatibility between the various annotation schemes was carried out with respect to definitions presented in the research papers published in conjunction with the datasets.

Automatic hate speech detection using machine learning is important to keep social media healthy. It is problematic that no common definition of hate speech and offensive language exists, and the subjective nature of the topic makes it hard to create coherent datasets. Different data compilation methodologies for different datasets and changing trends on social media introduce topical differences between datasets. If the goal is to consistently have a good model for practical use, a combination of datasets may be needed for models to obtain knowledge about the subtle characteristics that turn a social media post hateful.

Sammendrag

Denne oppgaven utforsker kompatibiliteten mellom populære datamengder som inneholder støtende språk ved bruk av språkmodeller forhåndstrente på store skriftsamlinger. Å trene iterativt på forskjellige datamengder viser seg å være vanskelig på grunn av at modeller glemmer tidligere kunnskap, mens sammenslåing og stokking av datamengder gir resultater som kan sammenlignes med trening og testing av modeller på data fra kun ett domene. Det er vanskelig å si om den gode ytelsen på de sammenslåtte datamengdene kommer av at modellene lærer å identifisere forskjellige datamengder, eller om definisjonene på hatefullt og støtende språk som blir brukt i de individuelle annoteringsmetodikkene faktisk er kompatible. Nylige modellers evne til å se forskjell på hatefullt og støtende språk i data fra både ett og flere domener ble utforsket i to faser. Et fordypningsprosjekt ble først utført, hvor et system basert på A Lite BERT (ALBERT) ble implementert til å skille støtende og normalt språk for å teste effektene av å redusere minnebruken fra en nylig populær modell kalt Bidirectional Encoder Representations from Transformers (BERT). Deretter ble fem systemer implementert, hvor to var basert på en modell som kan "huske" faktakunnskap. Denne modellen kalles Enhanced Representation through kNowledge IntEgration (ERNIE). De resterende tre var basert på ALBERT. Eksperimenter ble utført på tre datamengder, hvor alle annoteringsstrukturene ble synkronisert til tre kategorier: "Hateful", "Offensive", eller "Neither" for å gjøre kryssevaluering av modeller på forskjellige datamengder mulig. Resultatene antyder at de faktaintegrerende modellene yter bedre enn modellene med relativt lavt minneforbruk på data fra både ett og flere domener. De forskjellige annoteringsstrukturenes kompatibilitet ble diskutert inngående med utgangspunkt i definisjonene som ble brukt i datamengdenes respektive artikler.

Automatisk oppdagelse av hatefullt språk ved bruk av maskinlæring er et viktig virkemiddel for å ha et sunt miljø på sosiale medier. Det er problematisk at det ikke finnes noe felles definisjon på hatefullt og støtende språk, og temaets subjektive natur gjør det vanskelig å lage datamengder uten logiske brister mellom annoteringer og definisjoner. Varierende metodikker i bruk når datamengder settes sammen og trender på sosiale medier i konstant endring introduserer tematiske forskjeller mellom datamengder. Hvis målet er å kontinuerlig ha en god modell for praktisk bruk, så kan det være at en kombinasjon av flere datamengder er nødvendig for å få nok kunnskap om subtile trekk som gjør et innlegg på sosiale medier hatefullt.

Preface

This Master's Thesis was written as the final part of a Master's degree at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. I want to thank my supervisor Björn Gambäck for great advice, valuable feedback, and for encouraging me to follow topics I found extremely interesting. I also want to thank my father Atle Bulukin, bonus father Morten Hvaal, and my fellow student Ingrid Volden for giving me much appreciated feedback on language, and my girlfriend Natalia Alacuart Vichier-Guerre for being there with me through the challenging time of Covid, which without her would have been a much more isolating endeavour. I would also like thank the publishers of datasets used in this thesis, especially Marcos Zampieri who answered my questions regarding data and research papers related to OffensEval 2020.

Peter Cook Bulukin
Trondheim, 1st February 2021

Contents

List of Figures

List of Tables

1. Introduction

A significant proportion of the world's population is frequently active on social media. Facebook reported having 2.5 billion monthly active users in 2020, a 9% increase over the previous year. Twitter (2020c) at the same time reported 152 million daily active users. From estimates done by Solutions (2018), the number of tweets has been relatively stable since it hit a ceiling of 500 million tweets per day in 2014 (Twitter, 2014). Increasing legal demands from several countries are driving companies like Twitter to increase their focus on moderating their content. Because of the vast amount of data, Twitter is already resorting to technologies in addition to human editorial moderating. In 2019, 50% of the tweets removed for being abusive surfaced through technologies rather than human editors or users. In total, 584,429 individual accounts were reviewed because of hateful conduct in 2019 (Twitter, 2019). Machine learning methods have shown promise on Natural Language Processing (NLP) tasks and have been a part of the recent development of Twitter (Twitter, 2018). The recent inclusion of transfer learning methods in NLP is pushing hate speech detection in social media even further and provides better separability between different levels of offensive language. A considerable amount of research has been conducted in this field, with a recurring theme being the incompatibility of different datasets and their annotations. This thesis will focus on the recent models' abilities to distinguish between hate speech, offensive, and neutral language whilst being trained using several datasets. This combination of datasets will not be done in order to improve the results from using specific datasets, but to explore to what extent these datasets are compatible, and in what areas they differ and have similarities.

1.1. Background and Motivation

Social media platforms are generally perceived to be tools for creating and sharing ideas and information. Unfortunately, not all these ideas are constructive, but rather derogatory and meant to humiliate or in other ways cause harm to individuals or groups. In order to allow everyone to share their opinions and beliefs freely, it is essential to moderate the discussion so that everyone can feel safe to participate.

Locating the exact line between hate speech and acceptable free speech can be difficult due to the lack of a universal definition of hate speech. Davidson et al. (2017) define it as language that expresses hate towards a targeted group, or language intended to derogate, humiliate, or insult members of the group. Twitter has their own definition of hateful conduct, focusing on violence, threats, and direct attacks against people based on race, ethnicity, national origin, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease (Twitter, 2020b). With growing pressure and focus on moderation of content on social media, it becomes increasingly important for platforms to show that they put serious effort into solving the issue.

Hateful conduct online does not just result in restricted expression but can also motivate hate crimes in the physical world. Müller and Schwarz (2019) suggest that US President Donald Trump's tweets were highly correlated with anti-Muslim hate crimes. They argue that Trump's tweets caused the number of racist hate crimes to increase, more anti-Muslim activity amongst his followers, and a higher focus on Muslim-related topics in cable news, such as Fox News. They include an extensive analysis of causality and establish that Trump's tweets were the real cause of these effects. This highlights the importance of moderating hate speech on Twitter, not just as a means to maintain a healthy discussion but also to prevent a rise in real-world crimes. On the other hand, Müller and Schwarz (2019) caution against using their findings as grounds for implementing policies that restrict online communication, referring to the trade-off between moderating and free speech.

Traditionally, social media platforms have used people to detect and remove unwanted content. They have relied on users to report undesirable posts and then go manually through the reported content. Because users are not necessarily aware of the terms and conditions of different platforms, some cases can be unnecessarily reported, and some cases can fail to be reported at all. This can result in large workloads for moderators and difficulties in keeping up with the rapid increase in content on social media platforms. The task of going through vast amounts of negative and hateful content can also be mentally damaging for the moderators (Roberts, 2019; Newton, 2019; Koebler and Cox, 2018).

> *"She began to put in more time than her contract called for, and her mental energy was eaten up by worry and upset caused by what she experienced in the the forums she moderated."*
>
> \- Roberts (2019, p. 158)

Automatic detection of hate speech can thus help remove unwanted content, even before it is discovered by the people it is supposed to harm, creating a healthier online community.

1.2. Goals and Research Questions

Goal *Investigate the effects of training models using combined datasets with overlapping labels, where the datasets separate between several categories of offensive language.*

This goal aims to answer how training models using multiple datasets affects performance. This can be done by fine-tuning a model on one dataset and evaluating this model on multiple datasets, before continuing to train on another dataset and evaluating again. It can also be achieved by training a combination of several datasets by shuffling the data. The goal is to achieve this using the best possible models, and will thus be aided by a literature review, preliminary study, and experiments aimed at finding a model that performs well. The pursuit of the goal will further be driven by the research questions below.

Research question 1 *How much do irreconcilable properties of different offensive language datasets impact performance of deep learning models?*

When different annotators, methodologies, and definitions are used in compiling datasets in offensive language detection, there are bound to be incompatible differences in annotations. Some of the labels have to be very well defined, and they require a great deal of attention to be consistent. This research question aims to answer how large these differences are, and whether all the differences lie in the annotations, or a part of the problem is that different datasets have data from different topics, such as politics and racism. To answer this and find good candidate models for further exploration, a review of related work and different datasets will be conducted. The datasets will then be used in experiments, where they are trained using different permutations of the datasets to see how this impacts performance when the models are being tested using their respective test sets.

Research question 2 *Which models and parameters can best capture the structures of combined offensive language datasets?*

While there might be properties of different datasets that are incompatible, this research question aims to answer which models and which parameters of these models are best able to capture the patterns of several datasets simultaneously, and thus lead to higher performance in general and less deterioration from continued training. To answer this research question, a literature review of models used in offensive language detection and general natural language processing will be conducted to gain further insights on the best performing models. The highest-scoring models in workshops like OffensEval (Zampieri et al., 2019a, 2020) will be a guide for which models are eligible. There will be a focus on transformer-based models, especially BERT and its variations, since they have achieved state-of-the-art on important NLP benchmarks.

Research question 3 *Can the use of knowledge-training objectives during pre-training of a transformer improve on the performance of hate speech and offensive language identification?*

Transfer learning strategies in machine learning models intended for textual representations were first aimed at syntax and then semantics. This research question aims to answer whether the incorporation of factual knowledge in these models can be helpful in predicting hate speech and offensive language.

1.3. Research Method

Research question 1 was answered experimentally using datasets that had their labels mapped to a common annotation scheme. Since this research question references properties of datasets that are difficult to measure, several experiments were designed in an incremental manner in order to systematically gain insight on the underlying reasons of the results. In order to answer research question 1, a thorough review and discussion on how different datasets define different types of offensive language was also conducted. Answering research question 2 was aided by surveying existing research in the field of offensive language identification. Additionally, research on state-of-the-art language models used in other NLP tasks was reviewed in order to find suitable alternatives for further experimentation. Little previous work has been conducted on using three labels when conducting cross- and combined-domain experiments with hateful tweets, and the second research question was therefore answered experimentally. Research question 3 was answered by looking at the recent use of ERNIE in the OffensEval 2020 workshop and by further using the model in experiments. The experiments were conducted using three popular offensive language datasets, and the results were compared to experiments done with ALBERT in this thesis and with other related research on the same datasets.

1.4. Contributions

C1 A concise but thorough description of variations on the BERT language model.

C2 A proposed label mapping for synchronizing several annotation schemes including the hierarchical scheme of OLID.

C3 In- and cross-domain experiments using ALBERT and ERNIE based language models.

C4 Experiments on the the possibility of combining datasets for optimal performance in real-world scenarios.

1.5. Thesis Structure

Chapter 2 introduces background theory needed to understand the concepts used in the work done in this thesis and in related work.

Chapter 3 describes the datasets used in this thesis and other interesting datasets that are relevant.

Chapter 4 presents work related to the models used in this thesis, both in general NLP and with focus on hateful and offensive language identification.

Chapter 5 presents the preliminary study that was conducted leading up to this thesis. It includes descriptions of the data used, models, experiments, and results.

Chapter 6 describes the architectures that were utilized. It includes how the data was obtained and processed, and the model architectures used in the experiments.

Chapter 7 presents details on the experiments conducted and the results of these. The chapter includes the experimental plan, setup, and results.

Chapter 8 evaluates the methodology and decisions that were made, and discusses the results found in the experiments.

Chapter 9 presents a conclusion to the thesis, what contributions it brings to the field of hate speech detection, and propositions for future work.

2. Background Theory

This chapter will present some of the techniques, tools, and methods mentioned in this report. First, an introduction to classical machine learning models will be given, then different techniques and models that are categorized as deep learning. Common to a lot of deep learning models is that they all need some optimizer to train them. Different optimization techniques have been introduced with the growing size of machine learning models. Some of them will be described here along with some of their advantages and disadvantages. This chapter will also describe techniques for dealing with issues in machine learning data, such as unbalanced data. An introduction to the different tools and libraries used in the project will also be covered. Finally, a description of different metrics for evaluating the model performance will be given. Section 2.1 with minor additions and parts of section 2.2 are taken from the preliminary project leading up to this thesis.

2.1. Classical Machine Learning Models

When a computer system has the task of classifying data, two different approaches exist, as well as a combination of the two. The first type of system is rule-based, and the second is machine learning-based, where hybrid systems are the combination of the mentioned systems. Rule-based systems require a lot of manual work, need high levels of expert knowledge, and have to be crafted with great care. To overcome these restrictions, machine learning-based systems can learn directly from data. This can be done by learning associations between input data and corresponding labels on each example, also known as supervised learning. The goal is to capture structures hidden in the data. The problem is thus transformed into crafting models with architectures that are able to learn these structures. Not all machine learning models are labeled. Some models try to capture structures in data without specifically mapping it to some label, known as unsupervised learning. Hate speech detection is mostly issued as a supervised learning problem due to the use of labeled data, but unsupervised techniques are also utilised. The following sections will introduce some of the models successfully used for text classification.

2.1.1. Naïve Bayes

Naïve Bayes (NB) Classifier is a popular method based on basic probability theory. It is a very simple approach but has proven very powerful, especially with limited data. It has been used for Natural Language Processing (NLP) classification tasks with great success. It builds upon Bayes' Theorem on conditional probability.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Here we are able to compare the probabilities of all classes given a document. The approach is said to be naïve because it assumes that all features are conditionally independent. This is a very strong assumption and is not likely to be true in real-world scenarios. However, it simplifies the problem significantly, making it possible to calculate the probabilities, and has shown to work well in practice.

Bayesian optimization is not limited to classification tasks and can be used in several domains of artificial intelligence and optimization. Bayesian optimization can for example be used as a technique when searching for optimal hyperparameters. These are parameters of machine learning algorithms that have to be set manually, often just by a combination of knowledge of what usually performs well, and trial and error. Grid search is a normal method for testing how different parameters perform together. This searches through all the combinations of parameters, which can rapidly multiply because of the curse of dimensionality. Bayesian optimization makes it possible to search within the same space of hyperparameters for a combination that optimizes a chosen metric.

2.1.2. Support Vector Machine

Support Vector Machines (SVMs; Cortes and Vapnik, 1995) are non-probabilistic models which in its simplest form are able to do linear classification, and in that case also known as Support Vector Classifiers (SVCs). An SVC works by fitting a threshold based on training data by maximizing the margin, which is the distance from the threshold to the closest points from either class. To allow for outliers, Cortes and Vapnik implemented the use of soft margins, so that the model is less sensitive to wrongly annotated data. This increases the bias a little but lowers the variance, which is beneficial for generalizing. One-dimensional data would be separated by a point, 2-dimensional data by a line, and general data would be separated by a hyperplane.

Data cannot always be classified directly due to complex data structures. To alleviate this, Support Vector Machines use kernels to map data into a relatively higher-dimensional space, where the data might be better separable by a hyperplane using an SVC. Transforming data into higher dimensions can be very computationally expensive, and with some kernels, such as the Radial Basis Function (RBD), it is practically impossible. SVMs use the *kernel trick* to solve this, which calculates the relationships between data points in the higher dimensional space without actually mapping the values to a higher dimension.

2.1.3. Logistic Regression

Logistic Regression (LR) is a technique for statistical modeling that can also be used in machine learning. It is closely related to linear regression but is more directed towards discrete outputs. This makes it useful for fitting machine learning models for classification. Instead of using the least squares method as is done in linear regression, logistic regression tries to find the best model by changing parameters so that the *log(likelihood)* of the data given the mode is maximized. The model is optimized using parametric optimization techniques, such as Stochastic Gradient Descent (SGD).

2.2. Deep Learning

As data has become more available and computing power is increasing, deep learning has grown in popularity. Artificial Neural Networks (ANNs) are basic structures in deep learning, and they can take many forms depending on the task they are designed for. This section is going to focus on the most common techniques used for NLP. ANNs build on concepts from the biological brain at a very abstract level. Inputs are propagated through a weighted graph, where each node, often called a Perceptron (Rosenblatt, 1958), sends out a signal if the input becomes big enough. More advanced node structures have also been used, providing more functionality to a single node. An example of this is in Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) cells, which will be elaborated below.

2.2.1. Convolutional Neural Network

Convolutional Neural Networks (CNNs) were originally designed to work on images and the structure is abstractly inspired by how neurons are structured in the visual cortex (LeCun et al., 1999). They are able to encode spatial and shape related concepts from raw image data. This spacial awareness has also shown to be useful in language processing tasks (Collobert and Weston, 2008), but the reasoning behind the structure is best explained with visual processing as reference task. CNNs extract local information from an image by mapping a local area with pixel values through an operation convolution. Each convolutional node in the network can be interpreted as a filter, where filter weights are learned from data. This enables the CNN to learn local features like shapes and edges. In addition to the convolutional layer, there is also pooling layers. These layers reduce the dimensionality of the data, which makes it possible to reduce the computational complexity. When using a CNN for text processing, one-dimensional filters are used instead of two-dimensional.

2.2.2. Recurrent Neural Network

When using techniques such as bag-of-words or inputting a sequence of words into a neural network, we lose relationships given by the order of the words. Recurrent Neural Networks (RNNs) use recurrence to capture these relationships. The success of the original version of these networks was limited, and they tend to be hard to train as well as having other issues, such as the vanishing gradient problem which can happen if many activation functions are used on top of each other due to low values being multiplied together. The problems of the RNNs are dealt with in the Long Short Term Memory networks. The LSTM is a version of RNNs that uses a forget gate and cell state, enabling the network to learn what to remember and what to discard of information. This makes it possible to learn relationships between words in sequences, and to learn dependencies that span several iterations of recurrent loops.

2.2.3. Encoder-Decoder Attention

The attention mechanism was first proposed by Bahdanau et al. (2015) and was first applied to a RNN sequence-to-sequence model. When translating a sentence from one language to another, a human translator would probably not read the whole sentence, memorize it, and then read it out again in another language. He would probably pay attention to different words and translate little by little. This is the concept utilized in the attention mechanism. It is a structure that learns how to associate parts of the input sequence with parts of the output sequence.

An encoder-decoder network encodes an input sequence to a fixed size vector, before decoding it into the output sequence. This structure was very popular in neural machine translation. Decoding a sequence correctly from this latent vector can be a difficult task, so with the attention mechanism the input sequence is encoded to a sequence of annotations or hidden states. The weighted sum of these annotations is then used as input to each step of the decoder RNN, where the weights are based on both the annotations and the previous state of the decoder RNN. In this way, each output can selectively choose which annotations to pay attention to.

2.2.4. Transformer

The transformer was introduced by Vaswani et al. (2017) in "Attention is all you need". At the time of this paper, attention mechanisms had become quite popular for use in NLP tasks. A problem with the sequential structure of the encoder-decoder attention mechanisms built on RNNs is that they are very hard to parallelize and therefore hard to scale. Vaswani et al. argued that the attention mechanism was enough to capture relationships between tokens and that the underlying recurring structure was not needed. By removing this, the order of words and sequential structure are lost, which is resolved by adding a positional encoding to the input sequence. The transformer is built from several multi-headed self-attention layers which will be elaborated in section 2.2.5, and feed-forward blocks organized in an encoder and a decoder. Multi-head in this context means to project input to the self-attention h times, apply self-attention to all these projections, before concatenating the outputs. Around all self-attention and feed-forward layers are residual connections, enabling the signals early in the model to be preserved throughout the deep model structure. Tokens at each position in the attention blocks of the decoder stack are not allowed to attend to subsequent positions, allowing the model to only depend on "seen" tokens when decoding, making language modeling possible by shifting the outputs one step to the right.

2.2.5. Self-Attention

Self-attention was introduced by Vaswani et al. (2017) as a part of the Transformer. It avoids the sequential properties that original encoder-decoder attention employs. It splits the input into three parts through weight matrices that are learned: queries, keys, and values. For a specific token in the input sequence, the query of this token is multiplied with keys of all tokens in the sequence including itself. The outputs of all these equations are divided by the square root of the length of the keys to stabilize the gradients, before a softmax is applied for the values to be positive and sum up to 1. The output matrices of the softmax can be interpreted as compatibility scores between the specific token and all the tokens of the sequence. Each score is then multiplied with its corresponding value matrix before all matrices are added together to form the output of the specific token. These operations are done for all tokens. The formula for self-attention is given by:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

Self-attention comes with the cost of reducing the effective resolution of the data, which is counteracted by applying Multi-Head Attention by projecting the queries, keys and values h times to different representation subspaces where attention is applied in parallel. The outputs of all the attention-heads are concatenated and projected back to the dimensionality of a single output.

2.3. Optimization

Neural networks are parametric functions. It is practically impossible to search through all parameters the model can have, and a guided search for these parameters must therefore be conducted. This section will cover some relevant optimization algorithms that are used to solve this. Section 2.3.1 is from the preliminary project.

2.3.1. Stochastic Gradient Descent

A problem a lot of classical machine learning models have is that they are not able to train on some data, stop, and then train on some more data, at least not in regular implementations. Stochastic Gradient Descent (SGD) is a method to optimize a parametric function through following the gradient of the parameter space from a randomly initialized point. The basic method is Gradient Descent, which uses the gradient of a parameter space, combined with a learning rate, to take steps in a direction improving a loss function based on all parameters and examples. In SGD, this is approximated, by calculating the gradients for one example or a subset of examples every training step, to be able to train on large amounts of data. In practice a subset of examples, also known as mini batches, are commonly used. By using this method, it is possible to train numerous classical machine learning models on batches of data. SGD can be used to optimise both classical machine learning methods and neural networks.

2.3.2. Adam and AdamW

SGD takes a step towards larger models in terms of dealing with higher computational complexity. The Adam (Kingma and Ba, 2015) optimizer utilizes adaptive learning rates based on moments of the individual gradients. The moments are exponential running averages of the gradients, making the learning rate for a particular parameter dependent on the recent magnitude of its gradients. By doing this, Adam improves the convergence rate because parameters with little change will move sooner into a space where more rapid learning is possible. The step sizes of Adam are approximately bound by a hyperparameter to avoid too large step sizes that might lead to divergence. Adam requires four hyperparameters:

1. $\beta 1$: used to decay the running average of the gradient.
2. $\beta 2$: used to decay the running average of the square of the gradient.
3. α: step size parameter.
4. ϵ: used to prevent division by zero.

Adam improves the convergence rate of the models, but with the sacrifice of not generalizing as well as SGD with momentum, (Wilson et al., 2017). For this reason SGD has still been preferred in academic research, where the training time is not an issue.

A reason for the better performance by SGD might be the use of L_2 regularization to force the weights to be smaller, and thus generalizing better. L_2 regularization has been applied to Adam as well, but Loshchilov and Hutter (2019) noted that it was applied on the cost function of Adam in a manner that affected the moving averages of the moments. They proposed a corrected version of Adam with weight decay called AdamW, which has shown to generalize better than Adam and be competitive with SGD. For SGD, L_2 regularization and weight decay is logically the same thing.

2.3.3. LAMB

You et al. (2020) proposed LAMB, a new optimization algorithm tackling the long training time of BERT, which originally utilized Adam. LAMB was later used to pre-train ALBERT. It tackles the problem of larger batch sizes leading to less generalization by adding layerwise adaptive learning rates to Adam and normalizing each update to unit L_2-norm. While SGD in many cases are preferred over adaptive algorithms, Adam has been show to outperform SGD in certain situations, for example in attention models such as BERT and ALBERT (Zhang et al., 2019a). While You et al. (2020) focus on improving training time with the possibility of using large batch sizes without losing accuracy, they note that LAMB is a general optimizer that works well on smaller batch sizes. They do not actually refer to any experiments regarding this, where the lowest batch size in the paper is 512. They also write that the normalization of each layer to unit L_2-norm leads to a biased gradient update, but that this bias is small with larger batch sizes. This indicates that the bias might be larger with smaller batch sizes.

2.4. Training techniques

2.4.1. Overfitting

Machine learning models try to fit a model so that it predicts well on unseen data. This is the reason why test sets are kept away from the training process. Overfitting happens if the model performs well on data it has already seen, but fails to generalise well and perform well on new unseen data. Models are often so big that they can, if trained long enough, learn the exact mapping from examples to predictions. This is not good since the model will then utilise features that do not work well in general. Early stopping and dropout are often used to counteract this, and often the best way of tackling overfitting is by getting more data.

2.4.2. Over- and Undersampling

Imbalanced datasets pose some difficulties when training machine learning models. The issue is prevalent in classical machine learning (Guo et al., 2008) and deep learning (Buda et al., 2018). Several approaches have been explored to deal with class imbalance, but the most prevalent is oversampling. This method naïvely samples an increased number of samples from minority classes, that is classes with relatively fewer examples than the majority class. More advanced heuristic methods have also been explored, where for example SMOTE (Chawla et al., 2002) uses interpolation techniques to infer examples that lay in between the samples in the dataset. This is done to reduce overfitting caused by resampling the exact examples over and over again. According to Buda et al. (2018), resampling causes less overfitting in convulutional neural networks than classical machine learning methods, which might mean that other deep learning models are less prone to overfit due to oversampling as well.

2.5. Metrics

Choosing a good metric is crucial to evaluate the desired performance of models. Choosing the wrong metric can give seemingly good results even when a model is not performing well. Some commonly used metrics are covered in this section. Metrics are also integral for evaluation of inter-annotator agreement in datasets which will be used in chapter 3. The Fleiss' Kappa will therefore be covered in this section, together with accuracy, precision, recall, and F1.

Fleiss' Kappa is a statistical measure of inter-annotator agreement. The measure is is closely related to Cohen's Kappa which deals with a maximum of two annotators, while Fleiss' Kappa can have an unlimited fixed number of annotators. The measure compensates for agreement that is due to chance, and thus shows agreement on a scale from 0 to 1. It is given by

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{2.3}$$

The denominator represents the agreement that is possible to achieve above chance, and the numerator represents the agreement that is actually achieved above chance. $\bar{P}$ is the mean of label-wise annotator agreements, while $\bar{P}_e$ is the mean expected agreement.

Accuracy is one of the simplest methods to evaluate performance of models and is given by $\frac{CorrectPredictions}{AllPredictions}$. It works well when datasets are balanced, but can give the impression of a model being better than it actually is when dealing with unbalanced datasets. If a large proportion of data in a binary dataset belongs to label A, a model which only predicts label A would achieve good accuracy. This does not necessarily mean that the model is good, because the model would perform worse if tested on data where a larger percentage of examples had label B.

There exists smarter strategies of evaluating datasets that handle unbalanced datasets and can be tuned to favor bias towards specific labels. The basis for some of these metrics are true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). Precision is a metric that, as the name implies, measures how many predictions for a specific label are correct. It is given by

$$\frac{TP}{TP + FP} \tag{2.4}$$

Another metric that is often used in relation to precision is recall. This metric measures how many labels a model was able to predict correctly in relation to the total number of times the label appears in the dataset. This metric is given by

$$\frac{TP}{TP + FN} \tag{2.5}$$

These two metrics are valuable to measure two aspects of performance of a model. They are usually used side-by-side, and the harmonic mean of them is called the F1-score. A more general score is the $F_\beta - Score$. The F1-score is most commonly used, since usually no preference of recall versus precision is present. The F1-score is given by

$$2 \times \frac{precision \times recall}{precision + recall} \tag{2.6}$$

Metrics based on positives and negatives are originally meant for binary evaluation. For multi-class datasets, equation 2.5 is modified to $\frac{TP}{N_P}$, where N_P is the number of examples of the positive class.

The metrics are used from the reference point of a “positive” class. It is often interesting to evaluate an average of all , but this can be done in several ways. Three common averages are: macro average, micro average, and weighted average. Macro average works by calculating a metric for one label at the time and then averaging the scores for all labels. This is a good strategy when a dataset is unbalanced, but all labels have the same importance regardless of how many times they appear. Micro and weighted averages are sensitive to the label distribution, where micro average aggregates the contributions of all labels before calculating the final metric. By summing the positives and negatives, a minority label will be given less importance. A weighted average infuses the label distribution in the metric by averaging the score of each label, weighted by the number of times each label occurs in the dataset. Weighted and micro averages are usually very close to each other, while macro average can differ more from the two other metrics depending on the label distribution. Weighted and micro average are suitable if more importance should be given to the majority class. A weighted average can also be used to weight each label to preference.

There exists several other metrics, but these are the metrics most often used in offensive language prediction, and the ones that are most often used in the work referenced in this thesis.

3. Data

Progressing in research on methods for detecting and categorizing offensive language has been suffering the lack of consensus on definitions and annotation practices. Most datasets that are published on the topic have their own definitions on the classes they use in their dataset. Several categorization schemes are present, with offensive language, abusive language, cyberbullying, hate speech, sexism, and racism being some of them. The task of annotating a dataset requires considerable attention to how the dataset is annotated. Whether using experts, crowdsourcing, or experienced annotators, making sure the annotations are stable and coherent is crucial to make a good dataset. A problem arises when trying to compare work done on different datasets, where the practices usually differ, leading to difficulties in generalizing between several datasets. Gröndahl et al. (2018) showed that system performance decreases dramatically when testing systems across datasets, using train- and test-sets from different publications. Swamy et al. (2019) did a study on evaluating systems on different datasets than what they were trained on, using state-of-the-art models. Their results indicate the same as Gröndahl et al. (2018), with decreasing performance on cross dataset evaluation. It is probable that this is due to differing annotation practices, and it might be reasonable to assume that the inter-dataset performance would increase if the different datasets were annotated by the same people. In the same way the performance using a single dataset would decrease if different annotators were used on different sections of the data without the utmost care.

Especially thin is the line between offensive speech and hate speech, where the former is not to be moderated, while the latter is regarded as a criminal offense in some countries (Khan, 2013). Due to the lack of agreement between different datasets, care must be taken when approaching different datasets and the interactions between them. This chapter will give an overview of some popular datasets on the topic of offensive language, where some of them are used in this thesis, and comment on their data collection practices and annotation methods. The parts of sections 3.1, 3.3, and 3.4 relating to the structures of these datasets are mostly taken from the preliminary project, while the discussions and comments on the coherency and quality of the datasets are novel to this thesis.

3.1. Davidson et al. (2017)

The first dataset to be reviewed is from Davidson et al. (2017). They addressed the issue of separating between offensive and hate speech by compiling a dataset through the Twitter API. Davidson et al. (2017) have their own definition of hate speech based on the policies of Twitter and Facebook, and definitions found in the laws of different countries regarding hate speech:

Total	Not hateful	Hate speech	Offensive
24,783	4,163 (17%)	1,430 (6%)	19,160 (77%)

Table 3.1.: Distribution of labels in Davidson et al. (2017).

language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group.

So in addition to targeting speech, Davidson et al. (2017) include the authors' intention to incite harm as a qualifier for hate speech. While giving a clear definition of hate speech, they do not give a similar definition of offensive language. It can be understood though, that offensive language are phrases containing offensive words, where the phrase in total does not classify as hate speech. The dataset was compiled by searching terms from Hatebase.org, giving a sample of tweets from 33,458 Twitter users. All tweets by these users were then gathered in a set with a total of 85.4 million tweets, from which they randomly sampled 25k tweets, given that they contained words in the lexicon from Hatebase.org. The tweets were then manually coded by CrowdFlower workers. It is no surprise that the dataset consists of mostly offensive language, due to the fact that the data was compiled by searching offensive terms. Under the definition of a tweet being offensive if it contains an offensive word, all tweets in the dataset should be offensive due to the way the data was collected. Either extra tweets without offensive words were added, the CrowdFlower workers did not regard the all words in the Hatebase lexicon as offensive, or the definition of offensive is not that it contains an offensive word. The latter makes a lot of sense, due to the fact that offensive words can be present in very positive sentences. This is problematic though, because it leaves no clear definition of offensive speech, at least not presented by Davidson et al. (2017).

In the final dataset are 24,783 total tweets, with 19,190 being offensive, 1,430 labeled as hate speech, and 4,163 being neither. Hate speech only makes up 5% of the tweets, probably due to Davidson et al. (2017)'s strict hate speech criteria, but also that hate speech might not be as prevalent as expected. The low percentage of hate speech is challenging, but can be solved through for example over- and undersampling. A matter of more concern is that only 1.3% of tweets were unanimously labeled as hate speech. The intercoder-agreement given by CrowdFlower is presented as being 92%, which may seem good. The low percentage of tweets unanimously labeled as hate speech indicate that this agreement is dominated by the large amount of offensive tweets. This makes it hard to conclude whether the hate speech label is coherently labeled or not. Tweets were labelled by a minimum of 3 workers, while only a small number were labelled by more than 3. This makes the 1.3% of unanimously hate speech labeled tweets indicate that the agreement on hate speech labels were very low.

3.2. Founta et al. (2018)

Several difficulties with compiling a quality dataset on hate speech is addressed by Founta et al. (2018). They apply a three-step methodology in order to find the sweet spot when balancing economic resources and quality of annotations. How to conduct sampling of tweets in order to combat the issue of abusive tweets being sparse, the number of annotators to use, and how much to pay crowdsourcing workers are parameters Founta et al. (2018) tries to optimise in the face of limited economic resources.

- **Step 1:** The first step is to collect a sufficiently large amount of data and to combat the issue of abusive tweets being sparse in the ocean of total Twitter traffic. Founta et al. (2018) estimate the percentage of abusive tweets to be between 0.1% and 3%. To raise the number of offensive and hateful tweets in the dataset, a section was collected with sampling techniques aimed at raising the probability of tweets being offensive. In terms of training machine learning models, it is convenient to have a balanced relationship between labels. Models are not supposed to learn what appears most of the different labels, but how to distinguish them, since the distribution of tweets could always be changed through i.e. a filter.

  The collection of tweets was done through the Twitter Stream API, where a total of 32 million tweets were collected. Metadata was extracted from all tweets and tweets that should not be annotated, like spam and tweets without text, were removed. When sampling tweets from the full collected set, 12.5% were sampled using boosted sampling (BS), while the rest were sampled using random sampling. The BS utilised metadata, sentiment analysis, and counts of offensive words found in two dictionaries of hateful or offensive language[1] to bias the sampling towards more offensive tweets. The random sampled tweets were then mixed with the boosted tweets, and 80k tweets were set aside for use in the final dataset, while 300 were sampled to be used for analysis in subsequent annotation iterations.

- **Step 2:** The second step is to explore the impact of choices like wage and number of annotators, type and number of labels, trust in users, etc. The 300 tweets set aside in the previous step were used to fine-tune these parameters, in addition to the filters and boosting techniques used to increase the number of examples in the minority classes. The tweets were first categorized into spam, normal, and inappropriate. The inappropriate tweets were then sent to another panel on CrowdFlower for further annotation into more specific categories. The tweets were annotated in three rounds, where the two first rounds were used to gain insights into correlation between labels. They applied Pearson, Spearman, and Kendall Tau Correlation Coefficients to assess which labels to keep and which to discard. Their initial set of labels L was:

$$L = \{Offensive, Abusive, Hateful, Aggressive, Cyberbullying, Spam, Normal\}$$

[1]https://www.hatebase.org and https://www.noswearing.com/dictionary

Label	Hateful	Abusive	Normal	Spam	Total
# Tweets	4,948 (5%)	27,037 (27%)	53,790 (54%)	14,024 (14%)	99,799

Table 3.2.: Label distribution of Founta et al. (2018)

This set was then reduced to

$$L' = \{Abusive, Hateful, Normal, Spam\}$$

by merging 'Abusive', 'Offensive', and 'Aggressive' into 'Abusive', and eliminating 'Cyberbullying' which is scarcely used.

- **Step 3:** The third step is to annotate the 80k tweets for the final dataset. They created profiles on all annotators, showing that most of them were from Venezuela (48%). Many of them had a Bachelor Degree (48.4%), and 57% had an income level below €10k a year. A high percentage of tweets were labeled as spam in the large annotation round due to a higher percentage of tweets being randomly sampled than in the exploratory rounds.

The original dataset had a total of 80k tweets, where abusive tweets accounted for 11%, hateful 7.5%, normal 59%, and spam 22.5%. This yields around 8,800 abusive tweets, 6,000 hateful tweets, 47,200 normal tweets, and 18,000 tweets labeled as spam. The paper has later been extended, where Founta et al. said they would add more boosted data, since that seemed to be the most valuable. The total number of examples in the dataset can be seen in table 3.2.

During the validation round of step 2, almost 70% of tweets were annotated with an overwhelming agreement, which is defined as more than 80% agreement. For the validation round this means that 4 or 5 out of 5 workers annotated the same for almost 70% of tweets. Almost 30% of the tweets had a strong agreement, meaning more than 50% agreement. There is thus a simple majority only on a small percentage of the total tweets. On the full dataset they reached an overwhelming agreement on $\sim 55.9\%$ of the tweets, $\sim 36.6\%$ reach strong agreement, while $\sim 7.5\%$ had a simple majority with two annotators. Founta et al. provide a large, high-quality dataset for distinguishing between offensive/abusive and hateful language. Their methodology is clearly explained, and their iterations gives important insights into how to compile the dataset. The explanation of the methodology makes it possible for others to continue their work and create more quality annotations in the future.

3.3. Waseem and Hovy (2016)

Another widely used dataset is by Waseem and Hovy (2016). In contrast to other datasets, this only categorises racist or sexist remarks, separating the two. Similarly to Davidson et al. (2017), they used a bootstrapping approach, starting with a manual search on hate speech related terms. They also targeted specific hashtags and user accounts from

	Total	**None**	**Hate speech**	
			Racism	**Sexism**
Original	16,914	11,559 (68%)	1,972 (12%)	3,383 (20%)
Available	11,112	8,185 (74%)	17 (0%)	2,910 (26%)

Table 3.3.: Distributions of labels in Waseem and Hovy (2016) as given by Swamy et al. (2019)

the search results. They made sure that their methodology resulted in the dataset containing non-offensive use of potentially offensive words. Waseem and Hovy (2016) annotated the dataset manually themselves with the help of expert knowledge in each label. The dataset is provided as tweet IDs, making other feature information on the author available, such as gender and location, even though a lot of tweets could not have their extra features identified. Out of 136,052 tweets, 16,914 tweets were annotated, with 5,455 being regarded as hate speech through being labeled as sexist or racist, and 11,559 tweets not being regarded as hate speech. When comparing datasets later, Swamy et al. (2019) reported that several tweets, especially those labeled as racism were removed from the dataset, yielding a total of 2,927 tweets being hate speech by being labeled as either sexist or racist, and 8,185 being labeled as not hate speech. The inter-annotator agreement on the original data was $\kappa = 0.84$, and 85% of all disagreements appeared in annotations of sexism. Most of the cases where the expert review disagreed with the original annotations resulted in the tweet being labeled as neither sexist or racist. In contrast to Davidson et al. (2017), Waseem and Hovy (2016) note that a lot of hate speech do not contain any hateful slurs. To be able to detect hate speech consistently, they used a list of conditions for a phrase to be labeled as hate speech, writing that a tweet is offensive if it inhabits one of the points on the list.

3.4. Zampieri et al. (2019b)

Zampieri et al. (2019b) build on ideas of former papers suggesting that we need a better typology for offensive language, in the dataset Offensive Language Identification Dataset (OLID). They therefore introduced a novel hierarchical labeling scheme for annotating offensive language datasets. The hierarchy is based on three levels: **A**, **B**, and **C**, and is visually represented in figure 3.1. OLID contains 14,100 total tweets, where 9,460 of them are labeled as 'NOT' (Not offensive), and 4,640 are labeled as 'OFF' (Offensive). Of the offensive tweets, 4,089 are labeled as 'TIN' (Targeted Insult) and 551 are labeled as 'UNT' (Untargeted). The targeted insults are again categorised into 'IND' (Individual) with 2,507 tweets, 'GRP' with 1,152 tweets, and 'OTH' (Other) with 430 tweets. The details of the label distribution can be seen in table 3.4.

Level **A** categorizes the data into:

- **Not Offensive (NOT):** Tweets that are not offensive.

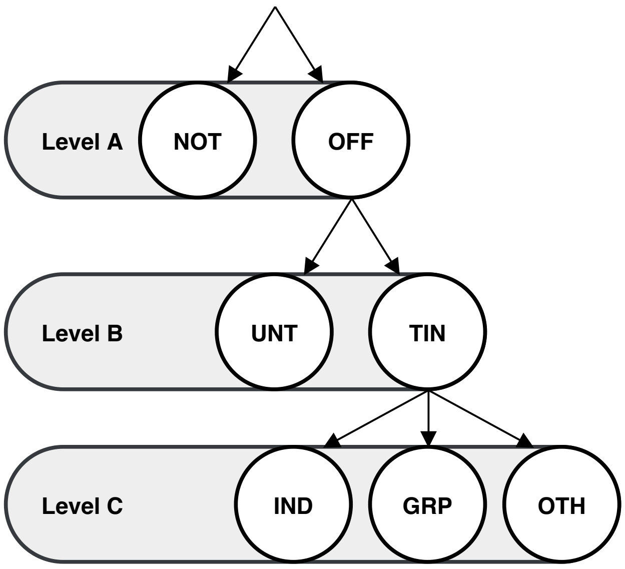

Figure 3.1.: The hierarchical annotation scheme of OLID

- **Offensive (OFF):** Tweets that contain any form of offensive language, i.e. swear words.

Level **B** then categorizes the tweets that were classified as offensive into:

- **Targeted Insult (TIN):** Tweets where the offensive comment is targeting a group, individual, or others.
- **Untargeted (UNT):** Tweets that go within category **A** but are not targeted.

Level **C** then categorizes the tweets that were categorized as **TIN**, as:

- **Individual (IND):** Tweets that fall into **TIN** and are targeting an individual. This could be regarded as cyberbullying.
- **Group (GRP):** Tweets that fall into **TIN** and that are targeting a group. This could be regarded as hate speech.
- **Other (OTH):** Tweets that fall into **TIN** but does not fall into neither **IND** nor **GRP**.

Through this hierarchy it could be possible to classify hate speech into several different categories matching other datasets. More precisely, the hierarchy labels the data into non-offensive, offensive, hate speech, and cyberbullying. Zampieri et al. (2019b) argue that the combination of ‘OFF’, ‘TIN’, and ‘GRP’ corresponds to what is usually regarded as hate speech, and ‘OFF’, ‘TIN’, and ‘IND’ as cyberbullying. This matches partly with other datasets, but a tweet that is offensive and targeting an individual based on

Task A		**Task B**		**Task C**	
OFF	4,640 (33%)	TIN	4,089 (88%)	IND	2,507 (61%)
				GRP	1,152 (28%)
				OTH	430 (11%)
		UNT	551(12%)		
NOT	9,460 (67%)				

Table 3.4.: Label distribution of OLID

some quality they possess, for example their ethnicity or sexual orientation, would be regarded as both hate speech and cyberbullying, so the typology has some drawbacks. The insult being based on ethnicity or sexual orientation might make the tweet being labeled as targeted towards a group, even though the insult was regarding an individual. This distinction is not explicitly stated by Zampieri et al. (2019b). Swamy et al. (2019) therefore choose to report the number of hate speech examples as not applicable.

To test out their hierarchy and create a gold standard with test questions for later annotators, Zampieri et al. (2019b) created a trial dataset of 300 tweets that was annotated by 6 experts. They report the Fleiss' kappa agreement on task A to be $\kappa = 0.83$ on 21 tweets pulled from the trial dataset, which they say was annotated by "the five annotators". There is no mention of why only five of the six annotators were included in the agreement evaluation, and why the Fleiss' kappa agreement was only calculated on 21 of the 300 tweets. There is also no mention of the Fleiss' kappa agreement on task B and C, so the agreement on these might be small. This trial dataset was only a starting point for what would become the final dataset. This was annotated by crowdsourcing, using workers from what was then called Figure Eight, formerly known as CrowdFlower, and now called Appen[2]. They only used experienced annotators and test questions to discard annotations from annotators that did not meet certain standards. For task A, if the 21 tweets used to calculate the Fleiss' kappa agreement was used, or the agreement of these tweets reflects the full trial dataset, the gold standard questions can be assumed to be of a good quality. There is however no certainty that the gold standard questions given to the Figure Eight workers were of sufficient quality, due to no details on the agreement for task B and C being given.

When manually inspecting the official dataset, several incoherencies surface. Examples of tweets about gun control can be seen in table 3.5. Here, the tweets either attack gun control or show support for gun control. Perhaps someone would be offended because they disagree with the views of the author of the tweet, but a tweet should not be classified as an insult just because it supports or opposes certain political views. Without extended context it is unclear whether or not some of the tweets are sarcastic, but the target of the tweets is still gun control. It can be argued that the people opposing gun control is a group, but with this logic all political statements would offend some group, namely the people with the opposite standpoint. This would imply that all political statements are offensive, which would be detrimental to the consideration of free speech. The first

[2]https://appen.com/

Tweet	**A**	**B**	**C**
It's not my fault you support gun control	NOT	-	-
Gonna suck with all that gun control	OFF	UNT	-
I mean it worked for gun control right? URL	OFF	TIN	GRP
Perfect example of a country with gun control. I bet those guys were happy the criminals didn't have guns.	OFF	TIN	GRP
Gun control kills 3 more people.	OFF	TIN	GRP
This is why we need gun control	OFF	TIN	OTH

Table 3.5.: Examples of tweets from Zampieri et al. (2019b) on gun control. @USER tags in the beginning of tweets have been removed.

Tweet	**A**	**B**	**C**
Liberals are all Kookoo !!!	OFF	TIN	OTH
liberals got to crap all over everything.	OFF	TIN	GRP

Table 3.6.: Examples of tweets from Zampieri et al. (2019b) on liberals. @USER tags in the beginning of tweets have been removed.

example in table 3.5 shows gun control not being labeled as offensive, while in the last example the support of gun control is labeled as offensive. This shows the incoherency in the annotations. Another example of this can be seen in table 3.6, where liberals are regarded as both 'GRP' and 'OTH'. With the word "liberals" appearing over 1000 times and being one of the ten keywords used to compile the dataset, one would expect that an annotator would be able to keep to one categorisation, which puts the quality control of the dataset under scrutiny. Regardless of these observations, as it will be elaborated on in chapter 4 on Related Work, machine learning models have been able to get good results on the dataset, indicating that some coherency must be present between examples with similar labels.

3.5. Rosenthal et al. (2020)

The dataset by Rosenthal et al. (2020) follows the same hierarchical structure as OLID, and is called **S**emi-Supervised **O**ffensive **L**anguage **I**dentification **D**ataset (SOLID). It was distributed by the organizer of OffensEval 2020 (Zampieri et al., 2020). It is called semi-supervised because it is not manually annotated, but created with democratic co-training of an ensemble of models, including PMI (Turney and Littman, 2003), FastText (Joulin et al., 2017), LSTM (Hochreiter and Schmidhuber, 1997), and BERT (Devlin et al., 2019).The tweets in the dataset was collected by querying common English stop words and filtering all but English tweets. The ensamble was trained on OLID, and then used to label the new dataset with originally over 12 million tweets, where around 9 million were published and 3 million kept private for further future analysis. One of the reasons for collecting such a big dataset is to handle the issue of labels further down in the hierarchy having very low representation. Of the published dataset of 9 million tweets, only around 7,000 is labeled as OTH and around 22,000 labeled as 'GRP'. A detailed overview of the label distribution in SOLID can be seen in table 3.7. Since the dataset is annotated through an ensamble, the labels are given as confidence scores with standard deviation instead of hard labels. This makes it possible to make decisions on the use of the tweets based on these values. The next paragraph mentions how Rosenthal et al. (2020) used these scores to create several evaluation sets, separating hard from easy examples. When using SOLID in this thesis all examples were used, and the highest scoring label was chosen.

To test their new dataset, Rosenthal et al. used both an analysis set from the semi-supervised collected data, and a manually annotated set. For the latter, all co-authors firstly annotated 48 tweets that were predicted 'OFF' by the models, reaching an inter-annotator agreement (IAA) of 0.988 for task A, 0.818 for task B, and 0.630 for task C, using

$$P_0 = \frac{agreement\_per\_annotation}{total\_annotations \cdot num\_annotators}$$

for the IAA. After seeing the IAA being sufficiently high, they annotated 1,397 tweets for task A, and 1,927 tweets for task B and C. Due to the dataset being intended for use in OffensEval 2020 (Zampieri et al., 2020), the tweets for task A, and task B and C were different. They also split tweets into HARD and EASY, where EASY was more explicit offensive speech, while HARD more implicit, but used the ensamble models' confidences to categorise this. Since they manually annotated offensive tweets into 'OFF' and NOT, they added extra 'NOT' labeled tweets randomly sampled. From their experiments with the effects of using SOLID in addition to OLID, they found that especially the weaker models gained a boost by being trained on SOLID. BERT were in all cases the best performing model. The use of SOLID did not improve the performance on task A for BERT, indicating that the amount of data for task A in OLID is enough. The use of SOLID in task B and C resulted in improvements, where OLID was suffering from a lack of training data. For task A, initial training with SOLID before fine-tuning with OLID gave the best result, while the opposite was the case for task B and C. Rosenthal et al. explain this by OLID being sufficient for task A, therefore giving worse results by fine-tuning on SOLID which is just an approximation. For task B and C though, the data in OLID is not sufficient, leading the model to overfit if using OLID during the final fine-tuning. They therefore got better results by initially setting the path by pre-training on OLID, before fine-tuning on SOLID which has more examples.

3.6. Caselli et al. (2020)

The dataset by Caselli et al. (2020) is not on new data, but a new layer of annotations on top of OLID (Zampieri et al., 2019b). They are targeting the issue of implicit and explicit offensive language, that few other datasets regard in their annotations. Rosenthal et al. (2020) did some work on the topic through their easy and hard labels on the analysis set of SOLID, but this set is mostly meant to be used as a test or analysis set more than a

Task A		Task B		Task C	
OFF	1,448,861 (16%)	TIN	149,550 (79%)	IND	120,330 (80%)
				GRP	22,176 (15%)
				OTH	7,043 (5%)
		UNT	39,424 (21%)		
NOT	7,640,279 (84%)				

Table 3.7.: Label distribution of SOLID

training set. Caselli et al. (2020) comment, as many have previously, on the contradictory guidelines and changing definitions in various works, leading to incoherent annotations. To combat this they propose newly adopted annotation guidelines for the hierarchy in OLID. Their main contribution is still the added annotation layer. They name this novel resource AbuseEval v1.0. Here they also look into which tweets need more context to reliably be annotated, and their explicitness.

They include an in depth analysis of OLID task A, the prediction of offensive or non-offensive language, containing statistics on words in the dataset, message lengths, and offensive prior which is calculated based on a weighted offensive dictionary (Wiegand et al., 2018). By ranking words per class by a TF-IDF approach, they found that several of the words appear in both offensive and non-offensive sentences, even though they are more apparent in the offensive ones. They found that dictionary approaches were competitive against the other solutions of OLID, showing that a high degree of the data in OLID is explicit. They also annotated OLID as explicit and implicit offensive language, using Urban Dictionary[3] as a guide on unknown words, as none of the authors are native English speakers. Their annotation rule was annotating a tweet as explicit if it contained one or more profanities or slurs. Only tweets labeled as offensive in OLID were considered. Their annotations resulted in around 66% of tweets in the training set of OLID being explicit. They also found that a perfect dictionary with all slurs, with all possible spellings would result in a F1 score on the offensive label of 0.783, which is 3.17 points above the best scoring model.

They comment on the distinctions between abusive and offensive language, the connection between targeted and untargeted insults used in OLID, and abusive language. They find through definitions that abusive language is more concerned with the intention of inciting harm, while offensive language is concerned with the feelings of the receiver, or general profanities in the tweets. They connect this to the targeted and untargeted insults of OLID, noting that targeting can be a form of intention. They interpret through these definitions that abusive language should be labeled as targeted insults, and non-abusive, but offensive language should be labeled as untargeted. In their dataset they define abusive language as

> *"hurtful language that a speaker uses to insult or offend another individual or a group of individuals based on their personal qualities, appearance, social status, opinions,*

[3]https://www.urbandictionary.com/

		OFF	**TIN**	**UNT**	**NOT**	**TOTAL**
Train	EXP	2,023	1,887	136	0	4,046
	IMP	726	668	58	0	1,452
	NOTABU	1,651	1,321	330	8,840	12,142
Test	EXP	106	103	3	0	212
	IMP	72	70	2	0	144
	NOTABU	62	40	22	620	744

Table 3.8.: Label distribution of AbuseEval v1.0

statements, or actions.“

They label the data as ‘EXP’ (explicit abuse), ‘IMP’ (implicit abuse), or ‘NOTABU’ (not abusive). Their ‘NOTABU’ label distinguishes itself from other datasets by including all tweets that are not interpretable without additional information or context. They were also sure to exclude tweets that were not real utterances, and therefore not fully interpretable, and made sure that abusive quotes in a sentence does not automatically make the sentence abusive, depending on the support or opposition to the quote. If a message is inside a quote and no additional information is given, the tweet was labeled as not abusive. This might have detrimental effects on learning, since the quotes are such a small part of the sentence. Further, the full tweet in itself seems to be abusive when seen without quotes, and could have been interpreted as such. To distinguish explicit abusive content from implicit, profanities were used as markers.

Their inter-annotator agreement was calculated by annotating 100 random tweets from OLID, reaching a Fleiss’ kappa of $\kappa = 0.61$ after the first round of annotation. Discussions revealed that a considerable number of the disagreements came from the inherently subjective notion of what is offensive and not. Another issue was the distinction between negative stance on a topic versus implicit abuse. Furthermore, disagreements arose through what is regarded as profanity, where certain words are around the threshold between profanity and not, will by some be placed on the offensive side, and by some on the other. After discussions and a new evaluation they reached a Fleiss’ Kappa of $\kappa = 0.86$. The label distribution of AbuseEval v1.0 can be seen in table 3.8. They note that there are a substantial number of tweets labeled as ‘UNT’ in OLID, that are labeled as ’NOTABU’ in the new dataset. By their definition, all abusive language is always targeted. This points to a probable mismatch between what is regarded as targeted or not between the two datasets.

4. Related Work

In this chapter, a review of earlier work done on Natural Language Processing (NLP) with special regard to hate speech detection will be presented. First, some techniques for transfer learning in general language processing will be addressed, before getting into more specific work done on hate speech detection, offensive language, and other related topics. This includes workshops and shared tasks that have become very popular in recent years and the models and techniques used in them. Transfer learning in the form of transformers have seen a surge in recent hate speech detection, so an elaboration on the models successfully used in later workshops will also be given prior to information on the workshops themselves. Section 4.1.1 and 4.1.2, describing traditional forms of transfer learning in NLP, and section 4.2, describing models used in hate speech detection before the introduction of transformers, are mostly taken from the preliminary project. While this section presents the architectures used in hate speech detection, related work regarding cross-dataset evaluation will be discussed in chapter 8.

4.1. Transfer Learning in Natural Language Processing

The advantages of transfer learning have been utilised in image processing for a long time. The use of pixels has allowed the mapping of almost all input to the same format. This issue has been more problematic for text data, where each value is not a pixel with one or three brightness levels, but one of several tokens in a vocabulary defined from task to task. Clever techniques have been put in use to solve this. This section will present some of these techniques, and how they have been utilised to improve Natural Language Processing. Transfer learning has become unavoidable in language processing tasks, and many of the high-ranking models are based on the Transformer (Vaswani et al., 2017).

4.1.1. Word Embeddings

In neural networks and other machine learning methods, words were originally represented as integers in the order they were discovered when creating the vocabulary. Word embeddings are more advanced methods of representing words as dense context vectors. These incorporate both the syntactic and semantic meaning of each word. They are often trained from large amounts of unlabeled data, both on highly general data or more domain specific data. A novel neural approach for representing words as distributed vector representations was proposed by Bengio et al. (2003) in the Neural Network Language Model (NNLM). Later, Mikolov et al. (2013a) introduced another model that has gained huge popularity, and is usually mentioned by the name *word2vec*. This model comes in two formats, Skip-Gram and Continuous Bag of Words (CBOW). Both of the formats build on the concept of creating a dense vector of continuous values from the context of each word. They differ in the way they are trained, where CBOW intend to predict each word given its context, while skip-gram intends to predict the context given the word. Another popular model called *Glove* was introduced by Pennington et al. (2014). Since these vector representations capture the context of each word, two words that appear in similar contexts will also have similar vectors. Because of this, word embeddings can be pre-trained on large corpora of general purpose text data, and thus capture the syntactic and semantic relationships between words before continuing to train on downstream tasks. Word embeddings therefore represent an early version of transfer learning in NLP, and has been very popular in conjunction with neural models like Recurrent Neural Networks. These new architectures received a lot of hype in the beginning, and proved to live up to their reputation when tested against the more traditional counting methodology of Distributional Semantic Models (DSMs; Baroni et al., 2014), and have also proved to be extendable from words to longer structures such as sentences (Mikolov et al., 2013b). Cer et al. (2018) also created general sentence embeddings with the Universal Sentence Encoder. Because of the complexity of language incorporated in sentence embeddings, they will be further addressed in section 4.1.2 on Language Models.

An issue with word embeddings is that words that are not contained in the vocabulary of the pre-trained vectors will be counted as out-of-vocabulary. To address this problem, Bojanowski et al. (2017) created fastText that represents each word as the sum of character n-grams in the word. This incorporates sub-word information, and does therefore aid performance on metrics such as cosine similarities between similar words, especially the words that are complex, technical and infrequent. Another issue with word embeddings are that they do not differentiate between ambiguous words with different semantic meanings such as (river)bank and a (financial) bank. This problem is addressed by Peters et al. (2018), where word embeddings are not created as fixed vectors, but as functions of the word with the context surrounding it, this will also be elaborated on in section 4.1.2.

4.1.2. Language Models

The concept of transfer learning found in word embeddings has similarities to the use of Language Models (LMs) and some of the techniques, especially the ones that work on sentences, might actually be categorized as pre-trained LMs. They both share the goal of enhancing word processing tasks by learning structures and semantics of natural language. In NLP, all types of models could be said to be some sort of LM due to their common property of having to incorporate an understanding of natural language to process the data. The type of LMs issued here are pre-trained LMs that can further aid other tasks such as classification. Venugopalan et al. (2016) improved an LSTM-based Video Description system by incorporating an LM trained on unlabeled data from Gigaword, BNC, UkWaC, and Wikipedia, inspired by Gülçehre et al. (2015) that used an LM trained on Gigaword to improve Neural Machine Translation. Both Venugopalan et al. (2016) and Gülçehre et al. (2015) pre-trained their own LMs and did not use an already available one. This makes their technique similar to other two-step approaches, but the concept is inherently the same as if the LM was already pre-trained. Gülçehre et al. (2015) used the LM proposed by Mikolov et al. (2011) as a tool for improving language modeling tasks. These tasks use the LM to aid in generating proper output sentences. This differs from the task of using pre-trained models as input to aid in classification. It shows, nevertheless, the advantage of separating models into language modelling and the main task of the network, and how LMs can incorporate large monolingual, unlabeled datasets when data sparsity is present in the main task.

While the above models have aimed at improving text output in sequence-to-sequence models using transfer learning in decoding, there are models that aim to improve other NLP tasks such as text classification in sentiment analysis, author profiling, next sentence prediction, hate speech detection, and more. One of them is Embeddings from Language Models (ELMo; Peters et al., 2018). This model could be classified as word embeddings, but can also be classified as an LM. ELMo addresses the problem that a word might have different meanings in different contexts, and should therefore have the possibility to be represented with different vectors given the context. To obtain this they represent each word as a function of the whole input sentence instead of saving fixed word vectors. They use a bidirectional Language Model (biLM) as basis for ELMo. The biLM is a bidirectional version of a standard LM predicting the next token t_k given the preceding ones $(t_1, \ldots, t_{k-1})$. This is a way of capturing the context of the word. In addition to the LMs, the biLM also incorporates a context-independent embedding obtained through a form of token embedding or as a CNN over the characters in the token. ELMo embeddings are then created by collapsing the biLM, computing a weighted sum of all layers. Peters et al. reported great improvements in NLP benchmark tests, especially noticing the model's abilities in Word Sense Disambiguation.

4.1.3. Transformer based models

After the Transformer was introduced by Vaswani et al. (2017), many models have been proposed based on the concepts it introduced. The use of only the encoder has deemed very popular, and several models have achieved state-of-the-art results on popular NLP benchmarks. Some of the most popular models will be presented in this section.

Universal sentence encoder

While ELMo balances between word embeddings and LM, other methods have worked purely on sentences. The Universal Sentence Encoder (Cer et al., 2018) tries to encode sentences into embedding vectors. They note that this is especially for transfer learning purposes, and compare their model with both word embedding transfer learning models and baselines without any transfer learning at all. They develop two versions of their model, one based on the transformer (Vaswani et al., 2017), and the other on Deep Averaging Network (DAN; Iyyer et al., 2015). The transformer computes a representation for each word incorporating context information such as order, and a sentence embedding can thus be regarded as an element-wise sum of the tokens in the sequence divided by the length so that sentence length is taken out of the picture. The model is trained on several down stream tasks from the start, using multi-task learning to train the embeddings. The DAN model is used primarily as a trade-off between model complexity and performance, having a compute time linear to the input sequence length, averaging input embeddings and using a feed forward network to train the embeddings. The transformer model obtains the best results on all benchmark tests from Word Embeddings Association Tests (WEAT; Caliskan et al., 2017), with the DAN model also beating the baseline CNNs and DANs that use pre-trained and randomly initialized word embeddings.

BERT

Another model that is based on the Transformer architecture (Vaswani et al., 2017) and has gained massive popularity is Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019). Vaswani et al. (2017) proposed the Transformer as an architecture to be used for Neural Machine Translation (NMT) specifically, while BERT does similarly to General Pre-Training (GPT; Radford et al., 2018) and trains on large amounts of unlabeled data in order to publish the model weights for further fine-tuning on downstream tasks. While GPT uses a stack of transformer decoder layers, the architecture of BERT is a stack of transformer encoder layers. One of the contributions of BERT is its novel strategy for language modeling, namely Masked Language Model (MLM). They argue that the standard way of computing LMs, by predicting the next word, lacks the deep bi-directional properties needed to fully represent the context at a specific token, even when using a bi-directional LM. According to Devlin et al., the strategy of using two unidirectional LMs on top of each other can be regarded as a shallow bidirectional model, and does not capture the total context in a deep manner. The MLM therefore masks random words during training for the model to predict them. This removes the uni-directional properties often seen in traditional LMs. In a sense, this is quite similar to the CBOW method of word embeddings.

In addition to MLM, BERT is also pre-trained on the Next Sentence Prediction (NSP) task to learn long term dependencies and connections between sentences. The model is fed with two sequences during pre-training, and the goal is to predict whether the two sequences are subsequent. Devlin et al. (2019) argue that the use of this training objective allows BERT to score better on tasks like question answering.

When using transformers, the order of tokens does not get encoded as in other LMs. BERT therefore uses positional embeddings that they add together with the token embeddings obtained from WordPiece (Wu et al., 2016) which solves the issue of out-of-vocabulary tokens. BERT achieves state-of-the-art performance on several benchmark tasks in The General Language Understanding Evaluation (GLUE) framework (Wang et al., 2018). More details on BERT will be given in section 6.4.1.

ALBERT

A problem with BERT has been its huge architecture, which can easily lead to memory issues. Both ALBERT and BERT come in several sizes, where the model size is dependent on the size and number of layers. $BERT_{large}$ seemed to have better performance in the experiments of Devlin et al. (2019) when compared to $BERT_{base}$, even on smaller datasets. Lan et al. (2020) utilise the large architecture of BERT, while scaling down the memory consumption by reducing the total number of parameters. They do this by factorized embedding parametrization and sharing weights between the different layers of the model. The factorized embedding parameterization is based on decoupling the embedding size E and the hidden size H into two matrices instead of one. In BERT, H is tied with E, such that increase in H results in large increase of the embedding matrix size, due to the vocabulary size V of NLP tasks usually being quite big (around 30k in the case of BERT). The number of parameters in the passage from vocabulary to the hidden layer is thus reduced from $O(V \times H)$ to $O(V \times E + E \times H)$. The reduction in the number of parameters due to this can be significant when $H >> E$. The number of parameters are thus reduced from around 23M in $BERT_{base}$ to around 4M in $ALBERT_{base}$. The second way ALBERT manages to reduce the memory consumption is to share parameters across layers. Different modes of sharing parameters are possible, such as sharing all weights, or sharing only the weights of the attention layers. Lan et al. (2020) found that sharing all parameters hurt performance, and that most of this performance drop comes from the sharing of feed-forward layers. Lan et al. (2020) still use the mode of all-shared as default to keep the memory consumption low, and the performance drop is compensated by faster computations times, lower memory requirements, and thus the possibility to train on larger structures. Lan et al. (2020) noticed that when no parameters are shared, using the embedding size of $E = 768$ leads to best performance. This no-shared mode can be regarded as the same setting BERT uses. When using the all-shared mode applied in the final distribution of ALBERT, the smaller embedding size of $E = 128$ gave the best results.

RoBERTa

When trying to replicate and evaluate BERT, Liu et al. (2019b) found that the model was significantly undertrained and proposed an improved method for pre-training BERT. This included training longer, on bigger batches, with more data, and modifying the training objective. The new model is published under the name RoBERTa. The training objective was modified by removing the next sentence prediction (NSP) task and replacing it by training on the other objectives with blocks of text spanning several sentences and documents. This made the model learn longer dependencies without having to use more training objectives. Using sentences inside a single document yielded slightly better results, but required dynamically altering the batch size, so the approach packing sentences from multiple documents was selected. They also noted that training BERT is very sensitive to the epsilon term of the Adam optimizer, showcasing BERT's, and probably its descendants' sensitivity towards optimizer choice. They thus achieved better results by tuning this parameter. They also used larger batch sizes than BERT by utilizing gradient accumulation, training with batches up to 8,000. The WordPiece (Wu et al., 2016) approach used for encoding text in BERT was exchanged with a byte-level approach. When these changes were applied independently, they yielded either comparable results, or marginal improvements. The full effect was seen when combining all the new design choices into RoBERTa, where large improvements could be observed. A large improvement was also achieved through using additional training data and training longer. RoBERTa achieved state-of-the-art results on GLUE, RACE, and SQUAD.

ERNIE

While pre-training of language models have been crucial for the latest improvements in NLP, most of the training have been with the goal of modeling generic language, without much focus on the goal of retaining knowledge. Sun et al. (2019) approached this problem with new training objectives through the model ERNIE (Enhanced Representation through kNowledge IntEgration), aimed at making the model retain some of the knowledge it comes over in pre-training. They use *Harry Potter* as an example, where in former pre-trained models, a relationship like *J. K. Rowling* being the author of the novel *Harry Potter* would not be retained. They argue that even though language models are to be used for specific downstream tasks, retaining general knowledge like this is advantageous for accurate language modelling. While it is arguable if these goals are still within the scope of language modelling, the addition of knowledge into pre-training could clearly be advantageous in downstream NLP tasks.

To retain knowledge, Sun et al. changed the masking scheme proposed by Devlin et al. (2019), used for pre-training of the transformer (Vaswani et al., 2017) encoder model. They introduced entity masking, where all the words of entities consisting of multiple words are masked, and phrase-level masking, where a conceptual unit of words is masked as a whole. The first publication of ERNIE is more aimed at Chinese NLP tasks, but the ideas can be extended to the English domain. This is done in ERNIE 2.0 presented in the last paragraph of this section. ERNIE achieved state-of-the-art results on Chinese NLP tasks. A lot of standard NLP benchmark tests do not really require prior knowledge, and it is thus possible that the model has advantages in downstream tasks that are not visible in the benchmarks. The different masking schemes are used in multiple stages, where a basic masking strategy, similar to the one used in BERT, is used in the first stage. Lexical analysis tools are used in the second stage to find phrases, selecting a few phrases in each example, where the words in the phrase were masked out. The third stage consists of masking out entities, which includes people, locations, organizations, products, etc. By masking these entities, the model is forced to remember what relationships exist between entities. As an important part of the pre-training, both encyclopedic data, news, and informal discussion forum data are included. To model the relationships between sentences, a Query-Response structure similar to the one of BERT's next sentence prediction is incorporated, but with the added possibility of several turns of queries and responses, resulting in longer dialogues. Masks are still applied to words, similar to the Masked Language Model (MLM) of BERT, and the model is made to predict whether any of the queries or responses are exchanged with random ones that are out of context.

By coincidence, just about one month before ERNIE from Sun et al. (2019) was submitted, **E**nhanced Language **R**epresentatio**N** with **I**nformative **E**ntities was submitted by Zhang et al. (2019b), yielding the same acronym. It is noted that these do no not have any connection, even though they both deal with knowledge integration in transformer based models. Sun et al. (2019) are all from the Chinese technology company Baidu, while Zhang et al. (2019b) are from Tsinghua University of Beijing. While Sun et al. (2019) only use modified training objectives on a BERT architecture, Zhang et al. (2019b) use knowledge graphs to incorporate knowledge into training. They use BERT as a textual encoder and add a knowledge encoder on top of the token outputs of BERT, aligning entities extracted through knowledge embedding algorithms. They use the same training objectives as BERT, but add a new training objective masking token-entity relations and also switching out the entity with a random one, making the model predict whether a wrong entity has been used. The last step is to correct for possible errors in the knowledge embedding extraction phase, making the model be aware that the entity inputs might be faulty. Their version of ERNIE performs better than BERT on two relational datasets, FewRel (Han et al., 2018) and TACRED (Zhang et al., 2017), while still being comparable with BERT on traditional NLP benchmarks like GLUE. This indicates that the model is able to incorporate knowledge information while still retaining the general language understanding of BERT, which might aid in downstream tasks that can gain from utilizing relational knowledge outside of what is possible to extract from only training on heterogeneous text data.

Sun et al. (2020) extends upon the work of Sun et al. (2019), incorporating continual multi-task learning. They add several pre-training tasks, categorized in word-, structure-, and semantic-aware. By using big data and weakly-supervised datasets, they are able to train on large amounts of data in a incremental manner. To enable the model to differentiate between different tasks, they add a task embedding to the inputs, similar to the sequence embeddings of BERT, denoting which task is being performed. They note that the textual encoder of their model can be both a transformer and a type of recursive neural network, even though they use BERT with the exact same parameters for accurate comparison. Their system beats BERT on all tasks in GLUE. This might be due to the pre-training issue noted by Liu et al. (2019b), and that they have trained on much more data, but by the time of writing this thesis, ERNIE is still number 5 on the GLUE leaderboard, only surpassed by ensembles containing i.e. ALBERT (Lan et al., 2020) and StructBERT (Wang et al., 2020b).

4.2. Models in Hate-Speech Detection

Gaydhani et al. (2018) used n-grams and TF-IDF to detect hate speech on Twitter, following the popular path of differentiating between levels of toxic language, using three classes, *hatefull*, *offensive*, and *clean*. They trained models based on Logistic Regression, Naïve Bayes, and Support Vector Machines and found that LR had the best results. They also used their own definition of offensive language, based on a review of definitions from other papers, related work, or sources:

> *"any speech that attacks a person or group on the basis of attributes such as race, religion ethnic origin, national origin, gender, disability, sexual orientation, or gender identity."*

They defined offensive language as

"the text which uses abusive slurs or derogatory terms."

Even though they had their own definitions, they used the dataset of Davidson et al. (2017), so these definitions did not have any actual effect on annotations. The definition of offensive language is heavily based on the use of specific terms, and might therefore be easily captured by n-gram features. They tuned the feature parameters, n in n-grams, and the use of L1 and L2 regularization through grid-search with SVM, NB, and LR. For the n-grams they used ranges of n, with $n \in \{1\}$, $n \in \{1, 2\}$ or $n \in \{1..3\}$. After tuning on feature parameters, NB performed best, closely followed by LR. All models performed best with L2 regularization and $n \in \{1..3\}$. They further tuned LR and NB on hyperparameters relevant to the two models. The best final model was LR, with F1 scores on all categories higher than 95%. These are remarkable results, but Isaksen (2019) have noted that the results have been hard to reproduce, even with their distributed code, and that 74% of the test data is either duplicate or already present in the training data.

Instead of experimenting with n-grams on different models, Nobata et al. (2016) used n-grams and other features and feature extractors like word2vec which were all inputs to an SVM. Of single features, character n-grams had the best performance, with token n-grams following closely behind. The best performance was obtained by using all features. They only used the features as input to an SVM, but note that the model outperforms deep learning models. They do not explicitly show any comparisons with deep learning models though, and the datasets used are extracted by themselves, so this has to be taken with a grain of salt. Additionally, their data is split into financial data and news data from Yahoo, which implies that the data has a limited domain. They also experimented with how data changes over time, testing whether the performance declined with temporal change of commenting trends. They found that recent data is more important than the amount of data.

When data becomes more available, deep learning models become more relevant. LSTMs gained a lot of traction in NLP when its ability to learn long-term-dependencies was discovered. The same cannot be said about its application in hate speech detection. It can be found in several papers as a feature extractor for i.e. SVMs or other shallow learners, but as the main model it is mostly used in papers of lesser importance, though there are some exceptions. When used in comparative studies, it rarely comes out as the best performing model. Badjatiya et al. (2017) had great success with a combination of an LSTM and Gradient Boosting Decision Tree (GBDT), but the results have been hard to reproduce and Mishra et al. (2018) comment that it might have to do with the fact that the model was trained on the full dataset including the test set with no early stopping, which might have led to overfitting. Meyer and Gambäck (2019) is one of few that use LSTMs successfully, using a combination of inputs through word embeddings in a word component and character inputs through a character component. The word input component utilised an LSTM and word2vec (Mikolov et al., 2013a) as well as GloVe (Pennington et al., 2014) embeddings. They tested several different system configurations, having the most success with 64 convolutional filters, 100 LSTM cells in the character component, and 150 LSTM cells in the word component. On the configurations of the convolutional components, two layers with 64 filters and length 4 and 3 respectively gave the best F1 score. The F1 score also improved with the use of GloVe embeddings compared to word2vec. The authors attribute this to the removal of stop words in word2vec, which might have been detrimental to the short average sequence length of the dataset. The system outperformed several state-of-the-art systems in terms of macro-, weighted-, and micro average score on the dataset from Waseem and Hovy (2016). Gao et al. (2017) also used an LSTM based system, combining it with a system detecting slurs. They annotated their own Twitter based dataset using the definition of hate speech from Waseem and Hovy (2016), which makes it hard to compare with other results. Their performance is much lower than other papers, but this might have to do with their dataset. Gao et al. also published Gao and Huang (2017) at the same time as Gao et al. (2017), based on a self annotated dataset from Fox News. There they provide an example of hate speech, which is an attack on religion. An attack on religion is by many definitions not considered as hate speech, because the comment has to be targeted at people at the base of a religion, not at the religion itself to be regarded as hate speech (Fortuna and Nunes, 2018). This means that Gao et al. (2017) might differ in annotations from other datasets.

The LSTMs younger brother Gated Recurrent Unit (GRU; Cho et al., 2014) has seen a little more appliance with good results. Pavlopoulos et al. (2017) employ an RNN system with the use of GRU cells on English Wikipedia comments and a Greek news portal. They report obtaining state-of-the-art results on the Wikipedia dataset (Wulczyn et al., 2017), while the Greek dataset is novel to their paper. Zhang et al. (2018) and Founta et al. (2019) both used GRU-based RNNs, where Zhang et al. (2018) used a combination of a CNN and GRU, while Founta et al. (2019) concatenated a GRU component working on text data and a Deep Feed Forward component with 6 dense layers working on metadata input, before feeding this into a classifying layer. Both papers report results using text only for the Waseem and Hovy (2016) dataset, which at the time of writing outperformed state-of-the-art, but were both later surpassed by Meyer and Gambäck (2019) using a combination of CNN and LSTM.

Some of the papers already mentioned use CNNs in conjunction with other model architectures. Meyer and Gambäck (2019) combined a character based CNN with a word based LSTM, but also experimented with disabling the word based LSTM. The system still performed fairly well without the LSTM, beating the baseline. The highest precision was found with this configuration. Gambäck and Sikdar (2017) used a CNN-only network trained on the dataset by Waseem (2016), experimenting with the use of word vectors, both randomly initialized, with word2vec, character 4-grams, and a combination of these. They report that character n-grams result in higher precision, but word2vec gives the highest F1 score. This corresponds with the findings done later by Meyer and Gambäck, where character features seem to increase precision.

Isaksen (2019) did a deep review of the available datasets for hate speech, and experimented with how well a transfer learning model is able to separate hate speech, offensive language, and normal speech. BERT$_{\text{base}}$ and BERT$_{\text{large}}$ (Devlin et al., 2019) were used to conduct the experiments on the datasets of Davidson et al. (2017) and Founta et al. (2018), due to those being the only available datasets that separated hate speech and offensive language. The results were also compared to classical forms of machine learning like Logistic Regression, SVM, and Naïve Bayes. In OffensEval 2019 (Zampieri et al., 2019a) the top scoring models used BERT with differing pre-processing steps. Isaksen applied basic preprocessing using a tokenizer specialized for Twitter, called `TweetTokenizer`, to remove Twitter specific tokens. He then used the `BertTokenizer`, specially targeted to create inputs to BERT, by utilising WordPiece tokenization (Wu et al., 2016). This is less pre-processing than the top scoring models in OffensEval 2019, but keeping the preprocessing simple might be an advantage for research purposes. In the evaluation section, there is a table of predicted and annotated examples (Isaksen, 2019, p. 76), where the first tweet is an example of the difficulty of annotating correctly. The tweet is: "the guy in the White House had an entire election campaign selling fear. But I digress...". This is annotated as hateful. From the other tweets in the dataset, this could be interpreted as being a comment on President Trump's polarizing election campaign. As Isaksen comments, it is hard to see why this tweet was annotated as hateful, when a comment on the campaign of an upcoming or currently sitting president can hardly be categorized as hateful, and is an important example of free speech. Isaksen therefore also comments on the challenges of annotating such a large dataset, and how it is prone to mislabeling. As for the results, the BERT system performs a lot better than the traditional machine learning systems, and Isaksen also argues that the system outperforms others on the dataset of Founta et al. (2018), by comparing to the baselines set by Lee et al. (2018), and also being comparable with state-of-the-art by Davidson et al. (2017). The most difficult task for the system seems to be predicting hateful speech, which mostly is predicted as normal or offensive. The poor performance on hateful speech might come from the low percentage of hateful tweets in the training data, only 6% and 5% in Davidson et al. (2017) and Founta et al. (2018) respectively, and Isaksen not using any techniques like over- or undersampling, or weighted loss to combat this issue, resulting in a bias towards offensive and normal speech.

Using BERT as a standalone model have resulted in many good results. Zahiri and Ahmadvand (2020) tried to improve on the task of abuse detection by using class representations in an architecture called Class Representation Attentive BERT (CRAB). They used BERT to create contextualized input embeddings, but extended this with another layer of logic to incorporate class representations. They used a concatenation of all token-level outputs of BERT as input to a token-wise class representation layer, which computes multi-headed matching scores between input embeddings and class representations. The classes are represented as $A = (A_0, \dots, A_m), A_i \in R^{c \times |k|}$, where m is the number of class representation heads, c is the number of classes, and $|k|$ is the embedding dimension. The matching scores are obtained by taking the dot product of A and the concatenated output of BERT, before aggregating the output of the dot product through a series of fully connected layers, both before and after the class representation heads are concatenated. Sentence-wise class representation matching scores are calculated in the same manner but without the aggregation. The token- and sentence-wise matching scores are then normalized, added, and the final label probabilities are obtained using a softmax activation layer. They hypothesize that minority classes gained most from the architecture. Their best performing architecture beat their BERT baseline model, and a comparison with other systems will be presented in chapter 8.

4.3. Workhops

This section will present two popular workshops on offensive language detection. Workshops allow several participants to submit models, and are therefore excellent platforms for comparing solutions where the terms are equal for all participants. The two workshops included here are yearly iterations of the same workshop, which can provide insight on how popular models have evolved the latest years.

4.3.1. OffensEval 2019

Due to its increasing popularity, the challenge of automatic hate speech detection has in recent years found its way into highly popular workshops and shared tasks. One of them is the 6th task of SemEval 2019, also know as OffensEval (Zampieri et al., 2019a), where 800 teams signed up and 115 teams submitted reports. The shared task was based on the newly created dataset Offensive Language Identification Dataset (OLID; Zampieri et al., 2019b), compiled by the same author. This uses a new hierarchical annotation schema with three levels, intended to capture different granularities of offensive language. The three levels each represented a separate task in the competition, where task A is offensive language detection, task B is categorizing targeted and untargeted offensive speech, and task C is categorizing of the target. Baselines on OLID were set by Zampieri et al. (2019b), where they experimented with SVMs, BiLSTMs, and CNNs. The CNN had the best baseline for all tasks, except tying with the BiLSTM on task C. The CNN baseline was beaten by 2.9 percentage points on the macro averaged F1 score by the top scoring model on task A. BERT was used by all top 5 submissions on task A, with the sixth submission using an ensemble of CNN, BiLSTM, BiGRU, and word2vec (Mikolov et al., 2013a). The differences in the submissions using BERT were mostly in preprocessing techniques and different BERT configurations. For task B, pure BERT systems did not do as well as in task A, but several of the highly ranked results used ensembles consisting of deep learning models such as BERT and other non-neural machine learning algorithms. Seganti et al. (2019) had the best average macro F1 score on all tasks. They used an ensemble including another pre-trained LM, also based on the Transformer (Vaswani et al., 2017), called OpenAI GPT (Radford et al., 2018), as well as pre-trained embeddings from ELMo (Peters et al., 2018), fastText (Bojanowski et al., 2017), and Universal Sentence Encoder (Cer et al., 2018). The ensemble also included two classical machine learning models: random forest and SVM. They experimented with different combinations of the ensemble on development data and reported that GPT had best results on unbalanced data, while the Transformer model performed best on over- and undersampled data.

4.3.2. OffensEval 2020

OffensEval 2020 (Zampieri et al., 2020) was still popular the year after with 528 participants, 145 teams with official submissions, and 70 system papers. The workshop utilised the newly compiled dataset called SOLID (Rosenthal et al., 2020) which is covered in section 3.5. The best scoring model on task A scored very similarly to the other top 20 submissions and used a RoBERTa ensamble in the final submission (Wiedemann et al., 2020). The team on second place on task A and first on task B and C was Galileo (Wang et al., 2020a). They used XLM-R (Conneau et al., 2019) on task A and ensambles of knowledge distillation networks based on ERNIE 2.0 (Sun et al., 2020) on task B and C, but also experimented with ALBERT XXLarge. They beat second place with 1 percentage point on task B and 4 percentage points on task C. The last result is quite large in terms of model comparison. Task B and especially C are more important in terms of separating hate speech from offensive language, where the granularity is approximately the same. Most top scoring teams used some pre-trained Transformer model.

5. Preliminary Study

This chapter will describe the preliminary study leading up to this thesis. This work was done parallel to OffensEval 2020 (Zampieri et al., 2020), and investigated how ALBERT performed on task A of the workshop, namely differentiating offensive language from normal language. Some classical machine learning models were also used to provide baselines. The preliminary study gave a lot of insight into what techniques were popular in state-of-the-art hate speech detection. The first part of this chapter will explain the implementation of the system used, and the second part will present the experiments and results.

5.1. Data

This section will describe the data used in the preliminary study. The details of the datasets are given in chapter 3 since the same data sources were used in this thesis, and this section will describe the difference between the data of the thesis and the preliminary study. The training and testing data used was that of Rosenthal et al. (2020), while Zampieri et al. (2019b) was used as development data. The dataset of Rosenthal et al. will be denoted **S** and the one of Zampieri et al. **O**. The train and test sets of **S** will be marked with subscripts on **S**. These datasets are also used in this thesis, and more information about them can be found in section 3.4 and 3.5. While this thesis focuses on a conversion of the data structure, utilising all three levels of the OLID hierarchy, the preliminary study only included level A, namely differentiating offensive language from non-offensive language. This task has much coarser granularity than differentiating hate speech and offensive language. It is thus an easier task, and served as an introduction to the field of automatic hate speech detection.

5.2. Implementation

This section will present how the project was implemented, from processing the data to running the experiments. The project used both classical machine learning methods and ALBERT, which have somewhat different requirements for preprocessing, and the classical machine learning methods and their preprocessing will therefore be presented before ALBERT.

5.2.1. Classical Machine Learning Methods

The classical machine learning methods intended for the project was Naïve Bayes, Support Vector Machine, and Logistic Regression. They were intended for use as baselines since the results of the submissions of OffensEval 2020 were not published at the time of writing the preliminary study. To implement these models, a machine learning library called Scikit Learn (Buitinck et al., 2013) was used. The tweets were first tokenized by the TweetTokenizer of the NLTK library, before being mapped to a vector space by applying a TF-IDF approach. When going from a smaller dataset for developing the code, to the full dataset $\mathbf{S}_{train}$, the methods did not work due to their inability to work on mini-batches. To solve this, Stochastic Gradient Descent was used in combination with these methods. The Scikit Learn implementation of Naïve Bayes still had memory issues, even when training on small mini-batches, and was therefore discarded.

5.2.2. ALBERT

Since ALBERT is used in this thesis, the architecture of the language model itself will be given in chapter 6, while this section will focus more on the implementation and architecture of the full classifier system in the preliminary project.

The input of ALBERT is restrained to be used with SentencePiece (Kudo and Richardson, 2018), a slight variation on WordPiece (Wu et al., 2016). These methods intend to improve the problem of out-of-vocabulary words when the model is used for inference on the test set or in practical use. They bridge the gap between word and character representations, utilising the best of both, getting semantic and syntactic information in the longer words, while still being able to form new words through word pieces or characters. The “unknown” token can still appear though, in for example symbols that are lacking in the vocabulary.

Before applying SentencePiece tokenization, some preprocessing was done using the same TweetTokenizer as for the classical machine learning models. The processing of this tokenizer was used in a limited manner, and its main use was to replace html entities with their Unicode symbol. SentencePiece works on text strings when tokenizing, so the TweetTokenizer was not actually used as a tokenizer, where the resulting tokens were joined together into full strings again before feeding them through SentencePiece for final tokenization. An elaboration on SentencePiece is given in section 6.3.

In addition to normal preprocessing, ALBERT also requires a specific input format, starting every document with the “[CLS]” token and ending it with the “[SEP]” token. This is also elaborated in section 6.3. $\mathbf{S}_{train}$ is a very big dataset, and it was therefore not desirable to do tokenization every time an experiment was to be performed. TFRecords, files that are specially made to hold data for use with TensorFlow, were therefore created before training. The data in the TFrecords were then ready to load directly into the model. The architecture of the full classifier can be seen in figure 5.1. The 12 transformer blocks have shared parameters, and the first token of ALBERT’s output layer, named “pooled output”, was used as input to a final linear classifier layer, mapping the latent vector space into probability values for the two classes. No hidden layers were added except this output layer.

The implementation of ALBERT was done based on the official distribution on GitHub[1], written in raw TensorFlow, without any abstraction layers such as Keras. The system was built based on the `classifier_utils` file of the distribution, to add an output layer, compute the loss and apply the training step with the given optimizer.

5.3. Experiments and Results

The experiments done in the preliminary work was done to get familiar with the area of hate speech, but some research questions and hypothesises were also formulated to get a direction for the experiments. These questions were directly linked to the performance of ALBERT compared to the submissions of OffensEval 2020 (Zampieri et al., 2020), which would then be the state-of-the-art on $\mathbf{S}_{test}$. The project aimed to see how well a model based on the Transformer (Vaswani et al., 2017), specifically ALBERT, compared to the highest ranking systems. Zampieri et al. (2020) report that most of the highest scoring models of OffensEval 2020 were Transformer based, and most used pre-training techniques through models based on BERT, so the implementation of the preliminary work was actually compared mostly to models of the same class.

5.3.1. Experimental Setup

The output layer was fixed to the same size as the first output token, which is 768 for ALBERT Base and 1,024 for ALBERT Large. The training was run both with early stopping and without. The final models reported were ALBERT_b Hate, ALBERT_b Hate*, and ALBERT_l Hate, where the subscripts b and l denote base and large respectively. The asterisk denotes the model being trained with over- and under sampling and the use of early-stopping. ALBERT_l Hate used re-sampling but not early stopping. A grid search on the hyperparameters were not conducted, and the parameters for ALBERT_b Hate were chosen to be similar to the ones used in Lan et al. (2020) when fine-tuning for the SST-2 task in GLUE, due to this being a binary classification task and thus similar to the task at hand. The chosen hyperparameters can be seen in table 5.1, while the parameters that differ between the models can bee seen in table 5.2. The number of training steps were chosen to approximate three epochs with the batch size of 128, the change of batch size was done to speed up training on the large dataset, and the sequence length was set to 128 to save memory. Early in the exploration phase the sequence length was set to 512, but this required the batch size to be set to 4, giving very unstable training that did not converge. ALBERT_b Hate was trained on $\mathbf{S}_\text{train}$ for 1 day and 10 hours on two NVIDIA V100 GPUs with 32GB memory each, while ALBERT_b Hate* was trained for 2 hours and 20 minutes, terminating from early stopping with comparable results.

[1]https://github.com/google-research/albert

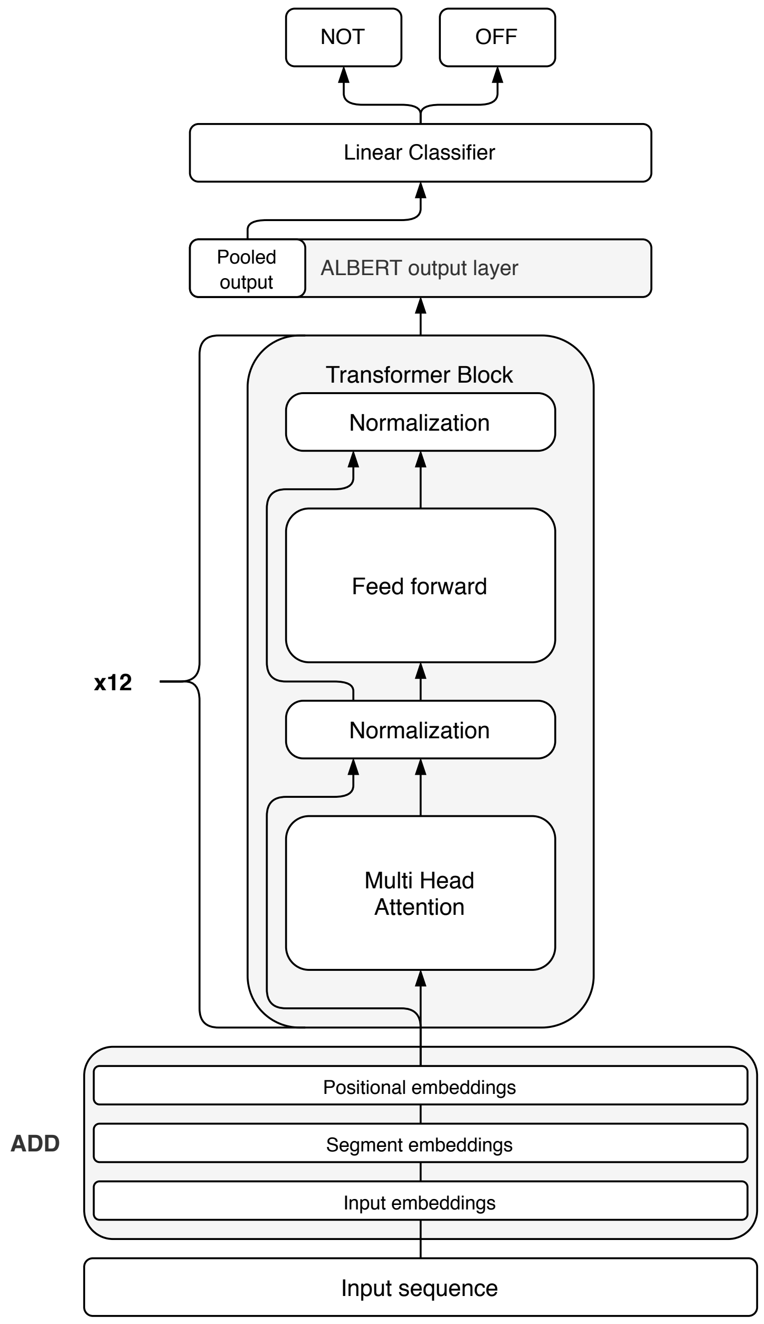

Figure 5.1.: Architecture of ALBERT$_{\text{base}}$ used for Offensive language detection. The difference between ALBERT and BERT is sharing of weights and different pre-training techniques, therefore the architecture is visually the same.

	Lan et al. SST-2	ALBERT_b Hate
Learning Rate	1.00E-05	1.00E-05
Batch Size	32	128
ALBERT Dropout	0	0
Classifier Dropout	0.1	0.1
Training Steps	20,935	212,705
Warmup Steps	1,256	1,024
Max Sequence Length	512	128
Early Stopping	Yes	No

Table 5.1.: Hyperparameters of Lan et al. (2020) for fine-tuning on SST-2 and ALBERT_b Hate.

	MS	BSZ	TS	ES	RS
ALBERT_b Hate	base	128	212,705	No	No
ALBERT_b Hate*	base	128	14,420	Yes	Yes
ALBERT_l Hate	large	32	19,292	Yes	No

Table 5.2.: Differing parameters of the ALBERT based models. The abbreviations used are MS for Model Size, BSZ is Batch Size, TS is Training Steps, ES is early stopping, and RS is re-sampling[2].

5.3.2. Results

This section will present the results of the experiments. Precision, recall, and F1 will be used separately for the two classes, and a macro average of the classes will also be presented. The results of the preliminary project are presented in table 5.3, and the confusion matrix for the best performing model, ALBERT_b Hate, is presented in table 5.4. While it is interesting to compare the results with the baselines of classical machine learning methods, a comparison with state-of-the-art through the results of OffensEval 2020 (Zampieri et al., 2020) can be seen in table 5.5.

The results show that the different configurations are very close in terms of performance, showing that the modifications done had little effect on ALBERT's ability to learn. The models using ALBERT performed considerably better than the classical machine learning models. The best scoring model was ALBERT_b Hate, with a macro F1 score of 0.911 as seen in table 5.3. The confusion matrix for this model can be seen in table 5.4. This was the only model that did not terminate training due to early stopping. The models trained on $\text{ALBERT}_\text{base}$ were more prone to classify non-offensive tweets as offensive, yielding a high recall on offensive tweets by sacrificing some precision. The model trained on $\text{ALBERT}_\text{large}$ was a bit more balanced, but did not achieve as good an F1 score. For comparison, the $\text{ALBERT}_\text{base}$ models had 3 and 5 false negatives respectively, while the $\text{ALBERT}_\text{large}$ model had 84, yielding a high precision on the non-offensive label. All models were trained with the same parameters, except batch size, total training steps,

Model	Offensive			Not Offensive			Macro avg.		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Logistic regression	**0.8996**	0.4648	0.6129	0.8264	**0.9800**	0.8967	0.8630	0.7224	0.7548
SVM	0.8427	0.6796	0.7524	0.8853	0.9512	0.9171	0.8640	0.8154	0.8347
ALBERT_b Hate	0.7857	0.9954	**0.8782**	0.9980	0.8953	**0.9439**	**0.8918**	**0.9453**	**0.9110**
ALBERT_b Hate*	0.7791	**0.9972**	0.8748	**0.9988**	0.8910	0.9418	0.8890	0.9441	0.9083
ALBERT_l Hate	0.7960	0.9222	0.8544	0.9680	0.9089	0.9375	0.8820	0.9155	0.8960

Table 5.3.: Results of different model configurations. ALBERT_l Hate was run using $\text{ALBERT}_\text{large}$ and ALBERT_b Hate(*) with $\text{ALBERT}_\text{base}$. Details of hyper-parameters can be seen in table 5.2.

	Predicted OFF	Predicted NOT
Actual OFF	1,074	5
Actual NOT	293	2,505

Table 5.4.: Confusion matrix for ALBERT_b Hate

the use of early stopping, and the use of re-sampling techniques. The parameters of ALBERT_b Hate can be seen in table 5.1 and an overview of the differences in parameters between the models are given in table 5.2, where we can see that ALBERT_b Hate had a lot more training steps than the other models due to not using early stopping. The number of training steps corresponds to 3 epochs of $\mathbf{S}_\text{train}$. ALBERT_b Hate has seen 27,226,240 examples in total, while ALBERT_l Hate has seen only 617,344 examples in comparison. For ALBERT_l Hate, the total examples are fewer than the dataset, so not even all examples were seen during training.

All versions of ALBERT scored better than the LR and SVM baseline models. A comparison with the results of OffensEval 2020 can be seen in table 5.5. If ALBERT_b Hate would have been submitted to OffensEval 2020, it would have positioned itself on 19th place out of 81, only 0.94 percentage points behind the top scoring model.

1	0.9204	6	0.9151	11	0.9134	16	0.9119
2	0.9198	7	0.9146	12	0.9132	17	0.9115
3	0.9187	8	0.9139	13	0.9129	18	0.9114
4	0.9166	9	0.9136	14	0.9129	19	**0.9110**
5	0.9162	10	0.9135	15	0.9128	20	0.9105

Table 5.5.: Macro F1 scores from the participants of OffensEval 2020 on task A. The score of ALBERT_b Hate, which were not actually submitted to the workshop, is added to the list on place 19 based on the score. There were a total of 81 results on the English dataset for task A for OffensEval 2020.

6. Architecture

Many experiments using Transformer-based models for classifying hateful language have been carried out. This chapter will describe the full architectures chosen for this thesis. BERT has been a very popular model, but the lighter ALBERT and knowledge-inducing ERNIE have also shown promise in workshops like OffensEval described in section 4.3.2. ALBERT was a natural continuation from the work in the preliminary work, and ERNIE might be able to push hate speech even further with the help of its knowledge on facts about the real world. The first sections will describe handling of data, while the last sections will describe ALBERT and ERNIE models in more detail, extending upon what has already been described in sections 4.1.3 and 4.1.3.

6.1. Data selection

The research goals of this thesis require the use of several datasets. Numerous experiments have been done on OLID (Zampieri et al., 2019b) and SOLID (Rosenthal et al., 2020) with recent state-of-the-art models. The full datasets will also be referenced as **O** and **S** respectively, with subscripts marking subsets used for training, evaluation and testing. The datasets being included in OffensEval 2019 and 2020 (Zampieri et al., 2019a, 2020) gives them a wide platform for comparison with results of other teams participating in the workshops. The hierarchical structure of the two datasets also provides a good starting point for exploration of definitions on different categories of offensive language in other datasets. Parts of the datasets were used in the preliminary study, making them readily available. A third dataset outside of this hierarchical structure is Davidson et al. (2017), which is denoted as **D**. The differentiation of offensive language and hate speech makes it a good candidate for comparison with SOLID and OLID, where certain paths in the hierarchy have suggested similarities with hate speech, and the top layer is offensive itself. Founta et al. (2018) provides a large dataset for differentiating hateful, abusive and normal language, as well as a spam label. This dataset will be denoted as **F**. They describe abusive language as something very similar to offensive language, so this dataset also has the potential of fitting into the three label annotation scheme of Davidson et al. (2017), namely 'Hateful', 'Offensive' and 'Neither'. More information on the datasets is given in chapter 3.

6.2. Data collection

The development policy of Twitter has certain guidelines on how data from Twitter should be handled. **O** and **S** were already available due to them being used in the preliminary

Version	Hateful	Offensive	Neither	Spam	Total
Original	4,948 (5%)	27,037 (27%)	53,790 (54%)	14,024 (14%)	99,799
Isaksen	2,941 (4%)	14,202 (21%)	41,784 (61%)	9,372 (14%)	68,299
Available	2,271 (4%)	10,460 (18%)	36,193 (63%)	8,090 (14%)	57,014

Table 6.1.: The number of available tweets at different times for the dataset of Founta et al. (2018).

work. They were published as a part of OffensEval 2020 (Zampieri et al., 2020), and the text of the tweets were published as part of the dataset. This makes the use of the dataset very convenient without loss of any data if tweets are deleted, but it also contradicts the Twitter policy for development (Twitter, 2020a), where only tweet ids should be published. The labels in **R** are not given as discrete values, but as mean and standard deviation of the probability values given by the models used to annotate the dataset. The dataset has three levels: A, B and, C, where A and B are binary and come with a single probability of being a positive value, while level C has three classes and a probability distribution over the three classes are given for each example. All labels were chosen to be the highest probability, without any bias towards any value. The standard deviation values were ignored, and all examples were included in the dataset.

F is published as a file with comma separated values (CSV), with only tweet ids and a majority label. The text of the tweets must therefore be retrieved using the Twitter API. Due to the tweets being abusive and therefore prone to deletion, several tweets had been removed and were thus not available to download. This was also a problem for Isaksen (2019), where many of the hateful and offensive tweets were removed. At the time of writing this thesis, even more tweets were removed, making it hard to compare the results. **F** was therefore obtained by Isaksen providing the dataset he was able to collect when writing his Master's Thesis. Later it was discovered that Founta et al. had been made aware of the problems regarding tweet retrieval, and therefore provide the possibility of obtaining the dataset by request. This was discovered after all experiments were conducted, so no attempt to obtain the full original dataset was made. Using data utilised by Isaksen (2019) also give the possibility of comparing results with his BERT based approach. The different label distributions of the tweets available at different times can be seen in table 6.1.

Dataset **D** is published as a file with comma separated values (CSV). They provide the class majority label for each example, the text of the tweets, and the votes from the annotators for the different labels on each example. Only the text and the final majority labels were used in this work.

S already comes with a test set, and the use of $\mathbf{S}_{train}$ as training set, **O** as development set and $\mathbf{S}_{test}$ as test set is a natural split of the data. Dataset **F** and **D** come with all examples in one document, requiring the data to be split into training, development, and test sets. The data of both **F** and **D** were split into training, development, and test sets with 60, 20, and 20 percent of the data respectively, using the `split()` function of

NumPy[1] after using the `DataFrame.sample()` function of Pandas[2] with random seed 42 to shuffle the data.

6.3. Preprocessing

The text in tweets are often messy, filled with emojis, links, hashtags, user mentions, and other details that make it different from standard written English. The models in use are trained on natural language, and it is therefore desirable to process and clean the text. Initial processing was done by the `TweetTokenizer` of NLTK, mainly to replace 'http' entities with their proper Unicode symbol. '&amp;' were for example replaced with '&'. The tokens were joined together to strings again, to continue preprocessing before using model dependent tokenization in the end. Dataset **D** had some tweets beginning with long series of exclamation marks. These were replaced with a single '!'. In datasets **S** and **O** the user tags are anonymized to '@USER'. The number of trailing user tags were limited to three. Several trailing space characters were replaced with a single space character. URLs were replaced with 'http' in the case of full URLs in **D** and **F**, and in **S** and **O** where URLs were already exchanged with the 'URL' token, this token was also replaced with 'http'. This was due to 'url' not being a part of the vocabularies of the pre-trained models in use, while http existed as a token. Hashtags that consisted of several English words were split into separate words by using the wordsegment[3] library, inspired by the two top scoring models in OffensEval 2019 (Zampieri et al., 2019a; Liu et al., 2019a; Nikolov and Radivchev, 2019) on task A. Several tweets are actually retweets, and the 'RT' marker used to signify a retweet was also removed due to the assumption that the meaning of a tweet does not change if retweeted and no additional comment is made on the content of the tweet. The tweets were demojized using the emoji[4] library, meaning that emojis were replaced with English descriptions of the emojis, within colon delimiters. This was done due to emojis not being part of the vocabularies of the pre-trained models, and also the technique being used by Liu et al. (2019a) who reached the top score on OffensEval 2019 task A. Words in the output are joined by underscores in the emoji library, so these were replaced with spaces because the vocabulary of ALBERT does not contain underscores. Another character that is not included in the ALBERT vocabulary is '@'. Since this is regularly used in tweets as user tag, all '@'s were replaced with '[at]'. Data intended to be used with ERNIE was then exported to TSVs with text and labels as columns. The framework used to run ERNIE, implemented by Sun et al., comes with the class `FullTokenizer`, that first cleans the data and performs punctuation splitting and lowercasing with the use of the `BasicTokenzier`, that also splits the text into tokens, before a WordPiece (Wu et al., 2016) based tokenizer splits each token into sub tokens.

WordPiece mentioned in the last paragraph, and SentencePiece (Kudo and Richardson, 2018) mentioned in section 5.2.2 are two very similar algorithms for creating vocabularies

[1]https://numpy.org/
[2]https://pandas.pydata.org/
[3]https://pypi.org/project/wordsegment/
[4]https://pypi.org/project/emoji/

with fixed length, that have a combination of characters, sub-word units, and full words included. These techniques aim to tackle the problem of out-of-vocabulary words and avoid having too large dictionaries that results in large memory consumption. WordPiece and SentencePiece utilise a technique called Byte Pair Encoding (BPE; Gage, 1994) that was used by Sennrich et al. (2016) to split words for use in Neural Machine Translation (NMT). The algorithm is greedy, beginning with a set of characters before merging them into longer sequences depending on their frequency. Normal words will therefore exist as full tokens, while rare words will be split into sub-word units and eventually characters. SentencePiece also supports the use of a unigram language model (Kudo, 2018), which is more efficient by using a binary heap for merged symbols. These improvements allow SentencePiece to work on full sentences, while WordPiece requires initial tokenization by a basic tokenizer before being applied to each word individually. This is most important for Asian languages that do not utilise any space at all, but it is convenient when working with English as well. For ALBERT, the SentencePiece model was loaded with the 30k token vocabulary it was pre-trained with, and for ERNIE the WordPiece model was loaded with an English vocabulary containing around 30k tokens, where some are Chinese characters but most are English tokens. The vocabulary of ERNIE also contains a lot of symbols, including i.e. Sanskrit. This vocabulary also contains '@', which the vocabulary of ALBERT lacks.

The first dataset used was SOLID. Due to the dataset having around 9M examples, an architecture where the dataset needed to be processed only once was utilised. The text in the datasets were therefore processed and saved in the TFRecords format, specialised for efficient data processing in TensorFlow. The data was stored in numeric form, ready to be used in the model. This architecture was used for all datasets that were to be used with ALBERT. An example of the preprocessing of data to be used in ALBERT can be seen in 6.1. The implementation of ERNIE[5] is written in PaddlePaddle, and tokenization is built into the pipeline of this implementation. The data for ERNIE were therefore preprocessed in advance, but not tokenized, and saved to a TSV file in the format suitable for the `run_classifier.py`.

6.4. BERT and Transformer Based Architectures

Both ALBERT and ERNIE build upon the framework proposed by Devlin et al. (2019) in BERT, which again builds on the Transformer model proposed by Vaswani et al. (2017) and described in section 2.2.4. Due to ALBERT's and ERNIE's natural inheritance of the basic structure of BERT, the first part of this section will describe the BERT architecture, and thereafter describe the modifications and additions that lead to the architectures found in ALBERT and ERNIE.

[5]https://github.com/PaddlePaddle/ERNIE/tree/release/r2.1.0

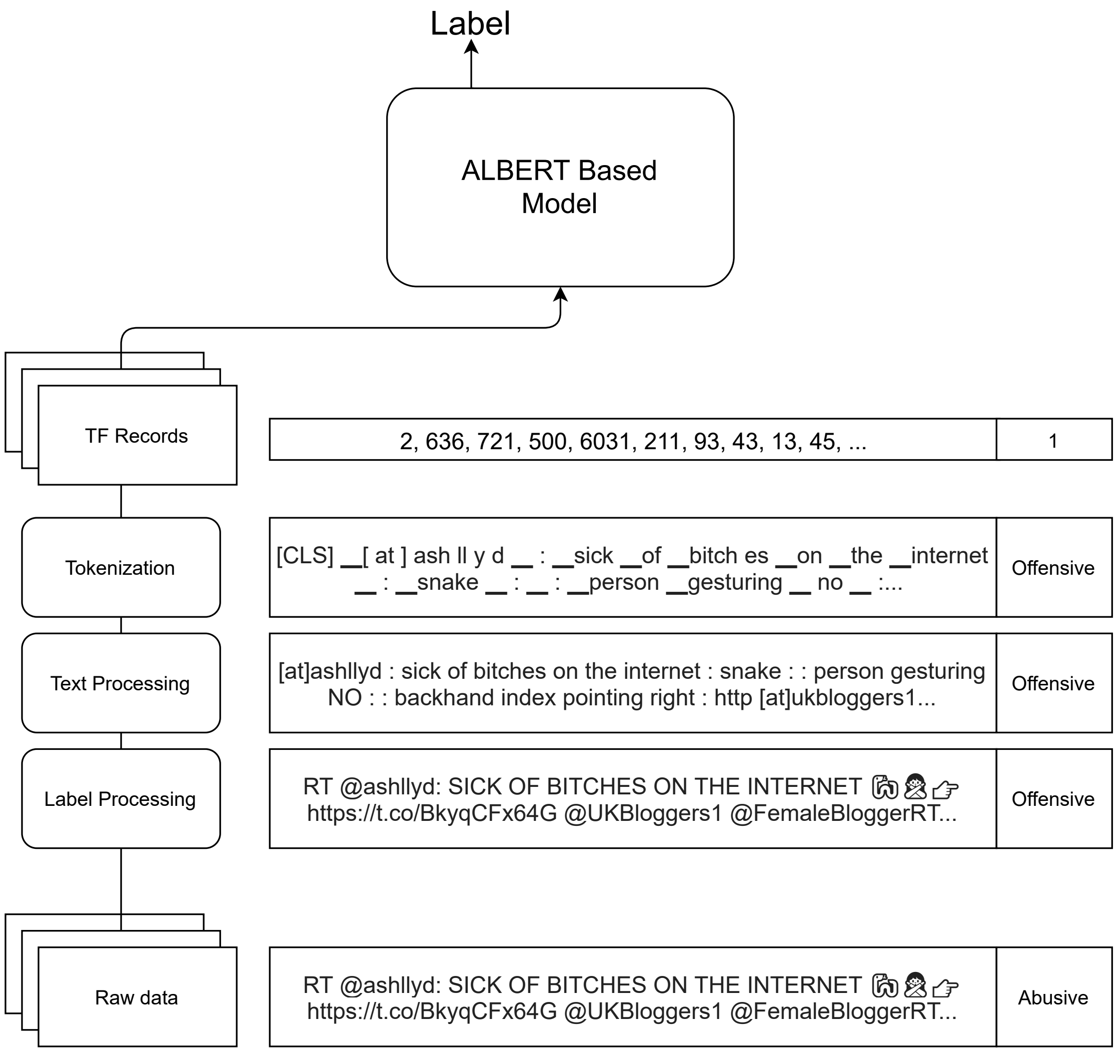
Label
ALBERT Based Model
TF Records
2, 636, 721, 500, 6031, 211, 93, 43, 13, 45, ...
1
Tokenization
[CLS] __[at] ash ll y d __ : __sick __of __bitch es __on __the __internet __ : __snake __ : __ : __person __gesturing __ no __ :...
Offensive
Text Processing
[at]ashllyd : sick of bitches on the internet : snake : : person gesturing NO : : backhand index pointing right : http [at]ukbloggers1...
Offensive
Label Processing
RT @ashllyd: SICK OF BITCHES ON THE INTERNET 🐍🙅👉 https://t.co/BkyqCFx64G @UKBloggers1 @FemaleBloggerRT...
Offensive
Raw data
RT @ashllyd: SICK OF BITCHES ON THE INTERNET 🐍🙅👉 https://t.co/BkyqCFx64G @UKBloggers1 @FemaleBloggerRT...
Abusive

Figure 6.1.: Pre-processing architecture of ALBERT.

6.4.1. Bidirectional Encoder Representations from Transformers (BERT)

The architecture of BERT (Devlin et al., 2019) can be divided into the initial layers that encode specific properties upon the input, the main model architecture, and the training objectives that are used during pre-training to make the model learn the desired representations. The training goals are covered in section 4.1.3. This section will therefore describe the input structure of BERT and the transformer encoder structure that is responsible for most of the parameters in the model.

The input structure of BERT is made to encode certain properties upon the input. To enable the model to process several sequences and to clearly mark where the separation between the sequences lie, BERT requires the use of a special '[SEP]' token. This token is also used in the end of the input, even when there is only one sequence. To mark the start of a document, each document begins with a '[CLS]' token. More importantly, this also makes room for the corresponding pooled output index described below. After special tokens have been added and the input tokens have been converted to numerical form, an input mask is applied by multiplying some randomly selected input words with 0. The use of the input masks is only relevant when pre-training the model, and during fine-tuning all values in the mask are set to 1. After the input mask is applied, segment ids are added to the input. This is 0 and 1 for the first and second sequence respectively. All segment ids are 0 if only one sequence is used. Because no recurrence is present in Transformer based models, they need to apply positional embeddings to give the model a sense of which words come first in the sequence. Vaswani et al. (2017) solved it by adding positional encodings based on sinusoids, while BERT uses a randomly initialized matrix that is learned during pre-training.

The output of the input section is then fed into the main part of the architecture. This is built from a series of Transformer encoder blocks proposed by Vaswani et al. (2017). BERT Base and Large use 12 and 24 of these encoder blocks respectively. Details of the Transformer and self-attention is given in sections 2.2.4 and 2.2.5. The first output of the last layer in BERT has an additional fully connected layer on top of it. The output of this layer is referenced as the pooled output, and is intended for use in downstream classification tasks. This pooled output corresponds to the `[CLS]` token in the input described earlier. The rest of the output is used for MLM during pre-training, but also other downstream tasks that require sequence output during fine-tuning. It is also possible to project the sequence output down to a lower dimension through an additional layer in order to do classification, but this is not the intended use of the model.

6.4.2. A Lite BERT (ALBERT)

The work in this thesis utilises the pre-trained model ALBERT (Lan et al., 2020) that is closely related to BERT. The changes applied in terms of reducing the number of parameters are described in section 4.1.3. The input of ALBERT, all Transformer encoder blocks, and the output structure are identical to the ones in BERT described in the previous section. The official implementation of ALBERT is almost an exact copy of the implementation for BERT, where only small changes are made to facilitate for parameter sharing and the factorization of the embedding size. The embedding size E is invariable between the different model sizes of ALBERT, $E = 128$, while the number of Transformer blocks, the size of the hidden layers, and the size of the intermediate layers all change between the different model sizes. The intermediate layer is one of two fully connected layers inside of the feed-forward blocks, while the second fully connected layer has dimensionality equal to the hidden size. The intermediate layer is in all cases 4 times the size of the hidden size. Both ALBERT Base, Large, and Xlarge were used in this project, and the variable parameters for the different models can be seen in table 6.2. These parameters give ALBERT Base, Large, and Xlarge, 12M, 18M, and 60M parameters respectively for the all-shared versions. BERT Base and Large have 108M and 334M parameters for the Base and Large versions.

Experiments were conducted with using the pooled output directly, using more dense layers on top, and projecting the sequence output down to a single layer for use in classification. None of these techniques seemed to have much impact on the models' abilities to learn, and only the single dense classification layer in the pooled output was used. The architecture of the ALBERT based model can be seen in figure 6.2.

6.4.3. ERNIE

A second model that is closely related to BERT is ERNIE 2.0 (Sun et al., 2020) which is an extension of ERNIE 1.0 (Sun et al., 2019). The changes introduced to both versions of ERNIE are mainly different training goals and training methodologies, and these changes are described in section 4.1.3. Since the model architecture in terms of model structure is the same between the two versions, this section will refer to both ERNIE 2.0 and ERNIE 1.0 as ERNIE, unless the distinction between them is relevant. The model weights used in this thesis are the ones from ERNIE 2.0. While ALBERT utilises a slight change in model structure, ERNIE utilises the same internal architecture as BERT. ERNIE comes in model sizes Base and Large, where both versions were utilised in this thesis. Their parameters are the same as for the same model sizes of BERT and ALBERT, and can be seen in table 6.2. Only the Base and Large columns are valid for ERNIE, where only these two configurations are available. It must be noted that ALBERT has the

	Base	Large	Xlarge
Hidden Size	768	1,024	2,048
Intermediate Size	3,072	4,096	8,192
Number of Attention Heads	12	16	32
Number of Hidden Layers	12	24	24

Table 6.2.: Variable model parameters for the different model sizes. The parameters for the Base and Large versions are the same for BERT, ALBERT and ERNIE, and the Xlarge values are only valid for ALBERT, because ALBERT is the only mode which is available in that size.

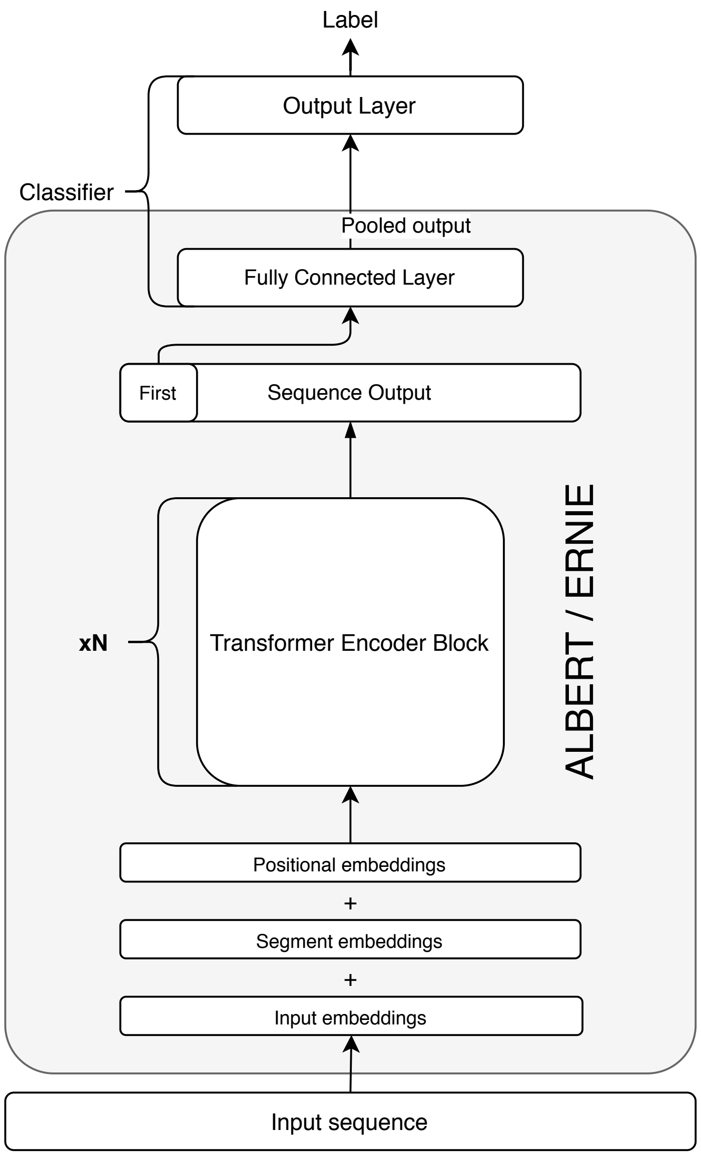

Figure 6.2.: High level architecture of both ALBERT and ERNIE based models. In ALBERT, the N transformer blocks have shared variables.

added embedding size parameter which BERT and ERNIE do not employ, but all the parameters listed in table 6.2 are the same for ERNIE and ALBERT except the Xlarge column.

While ALBERT and BERT add a randomly initialized variable, which then is learned to be the position embeddings, ERNIE has an additional input of explicit position ids that are input to the positional embedding layer. This difference does not really have an impact on how the model works, just on how the input to the model has to be formatted. While the range of 0 to sequence length has to be generated in ERNIE for use in an embedding lookup, ALBERT just adds a variable on top of the input and let the indices determine the position. The input structure of ERNIE is similar to the one of ALBERT and BERT except for this difference. The same special tokens are used, and the output of ERNIE can therefore either be the sequence output, or the pooled output corresponding to the '[CLS]' token in the input. Similarly to ALBERT, the pooled output has a fully connected layer added on top of it to aid in classification tasks.

6.5. Optimization

During training the classification error is calculated as cross entropy loss. The model is then optimized by minimizing this loss through an optimizer. Having an efficient optimizer is an import part of training a deep learning model. SGD has often been chosen by researchers due to a time-performance trade-off. In an academic setting, time is usually not as pressing as in a production environment. Optimizers like Adam shortens training time by taking bolder steps when gradients are gentle and shorter steps when the gradients are steep. This lets the network reach an optimum faster, but might lead to worse results than SGD. Adaptive optimizers such as Adam have shown to be better suited for Transformer networks such as BERT and ALBERT, yielding better results.

ALBERT was optimized with LAMB and the official code on Github[6] comes with the AdamW and LAMB optimizers, which are both described in sections 2.3.2 and 2.3.3. Both AdamW and LAMB was used in the hyperparameter optimization, but the LAMB optimizer that came with the official distribution of ALBERT had to be rewritten, because it was not compatible with multi GPU training. Both optimizers were utilised, LAMB being used with ALBERT Base and Large, while AdamW was used with ALBERT Xlarge. These choices were made due to observations in the exploratory rounds. All ERNIE models used AdamW with a scheduled decaying learning rate.

[6]https://github.com/google-research/albert/

7. Experiments and Results

This chapter will present the experiments that were conducted and the results of these. The goals and research questions of the thesis require several experiments to be conducted, and the strategy used when performing experiments will be described in the first section which is the experimental plan. Parameters used in the models and how the experiments were set up will be presented in the experimental setup. Detailed results of in-domain and cross-domain experiments will then be presented on a dataset-by-dataset basis in the final sections.

7.1. Experimental Plan

Combining several datasets, hyperparameters and models of this thesis requires an elaborate experimental plan. It will progress in complexity, with initial experiments establishing performance on a single dataset at a time, while the second part will experiment on different permutations and combinations of datasets. Light versions of BERT: ALBERT Base, ALBERT Large, and ALBERT Xlarge and the knowledge incorporating versions of BERT: ERNIE Base, and ERNIE Large will be used in all experiments. The initial experiments are related to Research Question 3 in section 1.2. Here, both ALBERT and ERNIE models will be used to investigate their abilities compared to each other and to former work on the selected datasets. These experiments will also serve as baselines when moving on to the second part of the experiments. Here, the models will be tested for cross-dataset performance and aim to answer Research Questions 1 and 2. First, the performance of a model trained on a dataset will be tested on all other datasets without further training. Datasets that will be used are SOLID (Rosenthal et al., 2020) denoted as **S**, OLID (Zampieri et al., 2019b) denoted as **O**, Founta et al. (2018) denoted as **F**, and Davidson et al. (2017) denoted as **D**. Inspired by Isaksen (2019), initial experiments on **F** will be done with all annotations: ‘Hateful’, ‘Abusive’, ‘Normal’, and ‘Spam’, while the rest of the experiments will be done with ‘Spam’ removed. The resulting dataset will be denoted **F***. Both **O** and **S** utilise a hierarchical annotation structure, separated in three levels: A, B, and C as described in 3.4. While introductory experiments can be done on the three levels separately, matching the datasets with each other requires mapping these levels into a single dataset. This mapping will be elaborated later in this section. When only a certain level of **O** or **S** is used, it will be denoted with a subscript marking the level, i.e. $\mathbf{S}_A$. To restrict the total number of experiments, only permutations of **F** and **D** will be used. Since **S** is annotated by models using **O** as seed, and that **S** has many more examples than the other datasets, $\mathbf{S}_{\text{test}}$ will be used as test set, and $\mathbf{O}_{\text{train}}$ will be used as training set. These will be regarded as coming from the same distribution. $\mathbf{S}_{\text{test}}$ will be used as representative for both **S** and **O**, due to the possibility this provides of comparing with participants of OffensEval 2020 (Zampieri et al., 2020), where this test set was used. Finally, experiments will be conducted on a dataset combining **O/S**, **D**, and **F**. This dataset will be denoted **C**.

The different experiments can be summarized as such:

1. Baseline and high performance experiments on **O/S**, **D**, **F** and **F***
2. Cross-dataset evaluation of all permutations of **O/S**, **D**, **F** and **F***
3. Models trained on **D** will be further trained on **F***, and models trained on **F*** will be further trained on **D**
4. Models from step 3 will be tested on test sets of **S**, **D** and **F***
5. Models will be trained and tested on a combined dataset **C**, and tested on **S**, **D** and **F***

The second part of the experiments deals with the merging and permuting of datasets. This requires all datasets to use the same annotation structure. 'Hateful', 'Offensive', and 'Neither' form a convenient annotation schema used by dataset **D**. **F***, with annotations 'Hateful', 'Abusive', and 'Normal' can simply be mapped to the other structure by mapping 'Abusive' to 'Offensive' and 'Normal' to 'Neither'. A mapping like this can be justified with Founta et al. (2018) stating that abusive language is "Any strongly impolite, rude or hurtful language using profanity, that can show a debasement of someone or something, or show intense emotion." This definition matches well with how Davidson et al. (2017) describe offensive language as the prevalence of strong language of any kind, while hate speech requires the speech to actually be intended as humiliating, degrading, or similar. When mapping **O** and **S** to the same annotation space, the hierarchical structure is exploited. Tweets marked as 'NOT' will be mapped to 'Neither', tweets marked as 'OFF-TIN-GRP' will be mapped to 'Hateful', and all other combinations of the different annotation levels will be mapped to 'Offensive'. A clear representation of the label mappings can be seen in table 7.1. This mapping will be chosen based on the notion that an offensive tweet can be regarded as hateful if it is targeted towards a group (Zampieri et al., 2019b). A tweet can be hateful and targeted towards an individual as well, but this usually involves targeting the individual based on some group this person belongs to, i.e. religion, gender, or sexual orientation. Some experiments will be done during the exploration phase to assess whether offensive tweets targeted at individuals should be included in the hateful category.

7.2. Experimental Setup

Experiments on both models were done using their official code distributions on Github. Experiments with ALBERT were conducted first and in a familiar framework, resulting in a much more extensive parameter search. Final hyperparameters and model configurations

TaskA	**Task B**	**Task C**	**Resulting Label**
OFF	TIN	IND	Offensive
		GRP	Hateful
		OTH	Offensive
	UNT	-	Offensive
NOT	-	-	Neither

Table 7.1.: Label mapping from the label scheme of SOLID to Davidson et al. (2017)

chosen for individual models will be presented in this section. To make the models more comparable, ERNIE Base and ERNIE Large have the same model configuration as BERT, yielding 110M and 340M parameters respectively. ALBERT Base, Large, and Xlarge have reduced the number of parameters from BERT and have 12M, 18M, and 60M parameters respectively. ERNIE was trained with the parameters from ERNIE 2.0, incorporating several additional pre-training objectives in Sun et al. (2020)'s continual pre-training framework described in section 4.1.3. Using the sequence output of ALBERT was tried during the exploration phase, projecting the sequence dimension down to a single value instead of just using the first output which is standard. Experiments were also conducted using different numbers of linear layers on top of the transformer structure. None of these techniques seemed to have much effect, so all final models were trained with the basic models and a linear output layer on top of the language models, predicting the output.

The experiments were run on two 32GB NVIDIA V100 GPUs using a mirrored distribution strategy. A copy of the model was therefore placed on each GPU, and the variables were synchronized using an allreduce algorithm. ALBERT Base used batch sizes between 100 and 128 with the LAMB optimizer (You et al., 2020), while nearly all versions of ALBERT Large and Xlarge used batch sizes of 64 and 32 respectively because of memory requirements of the larger model sizes. Due to You et al. (2020) stating that LAMB introduces a biased gradient update, but that this bias is small at larger batch sizes, AdamW was used as optimizer for ALBERT Xlarge. ALBERT Large used LAMB in some cases, but AdamW was used later in the experiments due to instances where running ALBERT Large with LAMB lead to the model predicting only a single label.

Some exceptions were made to the general settings, due to hyperparameter optimization being done on some of the datasets. It is therefore hard to show a compressed view of the hyperparameters, and they are therefore presented in detail in three tables: The hyperparameters used for all model sizes of both ERNIE and ALBERT are presented in table 7.2, the hyperparameters used for ERNIE but differed between datasets are presented in table 7.3, and the ones used for ALBERT but differed between datasets are presented in table 7.4. The hyperparameter optimization done for ERNIE was much smaller than the one for ALBERT. The only parameter that was tested was the learning rate, where ERNIE was better on smaller learning rates on the larger datasets. Due to framework limitations with mostly Chinese documentation in PaddlePaddle, the hyperparameter optimization was also just done with accuracy as metric. As can be seen in table 7.3, ERNIE was trained with a learning rate of 1e-5 on the large datasets, while

2e-5 was used on the smaller ones. The warm-up steps are different between the datasets, due to the parameter controlling them being set to 10% of the total steps. For continued training the hyperparameters were the same as in the base-models.

	TD	BSZ	EPOCHS	OPT	RS
ERNIE	0.1	32	4	Adam	No
ALBERT	0	-	4	-	Yes

Table 7.2.: Shared Hyperparameters
Hyperparameters for ERNIE and ALBERT that was shared by all model sizes. TD is Transformer Dropout, BSZ is Batch Size, OPT is Optimizer, and RS is Resampling.

	Davidson	Founta	Founta*	Solid	Combined
Learning Rate	2.0E-05	1.0E-05	1.0E-05	2.0E-05	1.0E-05
Warmup Steps	124	342	295	83	502

Table 7.3.: Hyperparameters ERNIE
Hyperparameters for ERNIE that was different between the datasets.

7.3. Experimental Results

This section will present the results of the experiments. In-domain experiments will be presented first, where results on **D**, **F**, **F***, and **S** will be presented separately. The results will be presented as precision, recall, and F1 for all labels. Weighted and macro average scores will also be presented to be able to compare with results from other publications. Macro average is used by OffensEval and many other publications, and the unbalanced datasets make macro average a more suitable metric. Some publications use weighted average, so that will be included as well. The results on the combined dataset **C** will then be presented, as well as the cross-dataset results. The results on **C** will be presented similarly to the results on the other in-domain experiments, while the cross-dataset experiments will be presented as one table for ERNIE Base and one for ALBERT Base, where the macro F1 score for all permutations will be presented. Only macro F1 is presented due to the large number of experiments, but the details of all experiments can be seen in the appendix.

7.3.1. Results on dataset D

The first results to be presented are the ones of the Davidson et al. (2017) dataset. As can be seen in table 7.5, the 'Offensive' and 'Neither' classes are very similar when comparing the best ALBERT and ERNIE results with exactly the same F1 score at three decimal points precision. That leaves 'Hateful' as the only class that differs between the model frameworks. Here ERNIE achieves a F1 score around two percentage points above the best scoring ALBERT based model. This naturally leads to ERNIE having the best

	Model Size	LR	BSZ	OPT	WS
Davidson	Base	3E-05	102	Lamb	324
	Large	1E-05	32	AdamW	300
	Xlarge	1E-05	32	AdamW	300
Founta	Base	4E-05	128	Lamb	300
	Large	2E-05	64	AdamW	300
	Xlarge	1E-05	32	AdamW	300
Founta*	Base	4E-05	128	Lamb	300
	Large	1E-05	64	AdamW	300
	Xlarge	1E-05	32	AdamW	300
Solid	Base	1E-05	32	AdamW	400
	Large	1E-05	32	AdamW	400
	Xlarge	1E-05	32	AdamW	300
Combined	Base	2E-05	128	Lamb	300
	Large	1E-05	64	AdamW	300
	Xlarge	1E-05	64	AdamW	300

Table 7.4.: Hyperparameters ALBERT
Hyperparameters for ALBERT that differ for the different size configurations and datasets. LR is Learning Rate, BSZ is Batch Size, OPT is Optimizer, and WS is Warmup Steps.

macro averaged F1 score, and as can be seen in table 7.5, also the best weighted average F1. ERNIE has slightly lower variance for each metric, calculated from the two and three variations of model size used for ERNIE and ALBERT respectively. The results of ERNIE Base are in most cases better than the ones of ERNIE Large, but the different ALBERT variations all score best on each their class, having Base, Large, and Xlarge score best on 'Hateful', 'Offensive', and 'Neither' respectively. This means that comparing a single ERNIE model with a single ALBERT model will increase the performance gap between the two models on two of the classes. From the ALBERT based models, ALBERT Base is the one yielding the best macro averaged F1 score, while ALBERT Large has the best weighted average F1 score due to its better performance on the larger 'Offensive' and 'Neither' classes. For the 'Hateful' class, it can be observed that while the ERNIE based models keep a balance between precision and recall, ALBERT seems to sacrifice one over the other to a higher degree.

7.3.2. Results on dataset F

The next results to be presented are from experiments on the dataset of Founta et al. (2018). Differing from the rest of the experiments, these results have four labels and are not merged under the label scheme used by the other experiments. ERNIE Base has the highest score on most of the metrics, the highest F1 score on the 'Abusive' and 'Normal' classes, while ERNIE Large and ALBERT Base have the best performance on

		Ernie		Albert		
		Base	Large	Base	Large	Xlarge
Hateful	Prec	**0.467**	0.416	0.372	0.431	0.284
	Rec	0.415	0.403	0.469	0.314	**0.484**
	F1	**0.439**	0.409	0.415	0.363	0.358
Offensive	Prec	0.947	0.946	0.951	0.944	**0.957**
	Rec	0.955	0.952	0.936	**0.959**	0.908
	F1	**0.951**	0.949	0.944	**0.951**	0.932
Neither	Prec	0.906	**0.911**	0.889	0.883	0.893
	Rec	0.901	0.894	0.883	0.891	**0.913**
	F1	**0.903**	0.902	0.886	0.887	**0.903**
Weighted Average	Prec	**0.915**	0.912	0.911	0.907	0.912
	Rec	**0.918**	0.913	0.903	0.914	0.887
	F1	**0.917**	0.913	0.907	0.910	0.897
Macro Average	Prec	**0.773**	0.757	0.738	0.753	0.711
	Rec	0.757	0.750	0.763	0.721	**0.769**
	F1	**0.765**	0.753	0.748	0.734	0.731

Table 7.5.: Results on **D** given as precision, recall, and F1 for each label, and their weighted and macro average.

the 'Hateful' and 'Spam' classes respectively. In terms of F1, the difference between ERNIE and ALBERT is around one percentage point in the 'Spam' class where ALBERT has the upper hand, while in all other classes the best ERNIE model has around two percentage points better score than the best ALBERT based model. This leads to ERNIE also being about two percentage points ahead in both weighted and macro average F1 scores. As with the results on **D**, the F1 scores on all metrics are around two percentage points higher on the highest scoring ERNIE result, compared to the highest ALBERT based result. In the end, ERNIE Base has the best F1 score on both weighted and macro average F1. Here too, the variances of the different metrics on the different classes, as well as the averages, are lower with ERNIE than for ALBERT. When inspecting the recall and precision scores of the different labels, it can be observed that both ERNIE Base and ERNIE Large always lean the same way in terms of favouring recall over precision. Both ALBERT Large and Xlarge seem to lean the same way as ERNIE Base and Large, while ALBERT Base consistently leans towards favouring the opposite of all the other models. On the 'Hateful' class, both ERNIE and ALBERT have less balanced relationships between precision and recall scores compared to the other classes, with ERNIE favoring precision.

7.3.3. Results on dataset F*

The results on the dataset which is a subset of the one from Founta et al. (2018), with the 'Spam' class removed and 'Abusive' changed to 'Offensive' fall into the same tendencies as the former experiments. ERNIE Large has the best F1 score on the 'Hateful' class, while ERNIE Base has the highest F1 scores on the two other classes. From the ALBERT based models, ALBERT Base scores the highest F1 score on the 'Hateful' and 'Offensive' classes, while being closely behind ALBERT Large on the 'Neither' class. The ERNIE and ALBERT based models are close in terms of performance, having only a slight difference of about 2 percentage points difference on the 'Offensive' label. The variance of the different models are also comparable, with some metrics favouring ALBERT while others favour ERNIE. The F1 score on the 'Hateful' class has actually decreased when going from dataset **F** to **F***, even though the averaged scores have increased. Similarly to the results on **F**, the relationship between recall and precision is less balanced on the 'Hateful' class than on the other classes, with ERNIE Base favoring precision similarly to the experiments on **D**.

7.3.4. Results on dataset S

The results on on the dataset of Rosenthal et al. (2020) converted to the label scheme of Davidson et al. (2017) can be seen in table 7.8. As with the other models, ERNIE Base seems to have the best results on most of the metrics. ALBERT Xlarge performs best on all metrics among the ALBERT based models. The F1 score of 'Hateful' is 1.8 percentage points better on the best scoring ERNIE model compared to the best scoring ALBERT model, 0.7 percentage points better on 'Offensive', and ALBERT and ERNIE scores similarly on the 'Neither' class. This results in the macro average of ERNIE being 0.8 percentage point better than for ALBERT. The variances of all metrics on both ERNIE and ALBERT are similar and low.

7.3.5. Results on dataset C

This section presents the results on dataset **C**, which was obtained by merging **D**, **F*** and **O***. It can be observed that also these results favour ERNIE, and in most cases ERNIE Base. ERNIE Large has the best F1 score on the 'Hateful' label, while ERNIE Base has the best results on the two other labels, similarly to the results on dataset **F***. The differences between the best ERNIE model results and the best ALBERT model results are 4, 3, and 1 percentage points for 'Hateful', 'Offensive', and 'Neither' respectively. ERNIE Base scores the highest macro averaged F1 score, with a 2.5 percentage point lead on the best ALBERT based model. Of the ALBERT based versions, ALBERT Large has the best F1 score on all labels. ERNIE has low variance on all of the metrics, while the variance of the ALBERT models is larger, with ALBERT Base and Xlarge lagging behind ALBERT Large. The ERNIE based models have a more unbalanced relationship between precision and recall compared to the other results, and the ALBERT based models continue their trend of having the precision recall relationship be unbalanced.

7.3.6. Cross-dataset results

This section presents results when testing models on other datasets than the ones they were trained on, as well as training models on several datasets. The results can be seen in table 7.10, where the dataset or datasets used for training are represented as rows, and the test set is given in each column. The upper part of the table shows the results when using ERNIE and the lower when using ALBERT. The base version was used for both models. The scores are given as macro averaged F1. The diagonal in the top 3x3 part of each table will be the macro average scores in tables 7.5, 7.7, and 7.8, and are the results from training and testing with train and test sets that were intended to be used together. Training on several datasets will be represented by combining the symbols for each dataset. The training on first **F***, and then **D** will therefore be represented as **F*D**.

ERNIE has slightly better performance than ALBERT with all training sets when testing on **D** and with all training sets except **F*D** when testing on **S**. When testing on **F***, the scores are more similar, with the highest difference between the two models' scores being on the training set **F*D**, at 0.043 in favour of ALBERT.

When testing the models that were trained on only one dataset for cross-dataset performance, it can be observed that **F*** and **S** have better cross dataset performance between each other than when testing on **D**. The model trained on **D** performs similarly on both **F*** and **S**, with a drop of around 12 percentage points from testing on **D**. Both the model trained on **F*** and the model trained on **S** perform similarly on $\mathbf{D}_{\text{test}}$, with a drop of 17 and 23 percentage points respectively from the baseline.

There are still drops in performance when training the models on several datasets but not as big as the cross-dataset performance with single training sets. Training on **F*** after training on **D** gives a slight improved performance when testing on **F*** compared to the baseline. This improvement is very small though, but there is at least not a drop in performance.

The performance drop is smaller when training with dataset **C**, or the performance gain is similar, compared to training with **F*D** or **DF***, and testing on all test sets. The performance gain on **F*** when initially training with **D** can also be seen when the datasets are combined. For dataset **D** the performance drop is small, around 1 percentage point, the same as with initialising training with **F***. The performance drop on the **S** dataset is around 5 percentage points. The training on dataset **C** hurts the ERNIE model less than the ALBERT model, while the performance drop with continued training has a slightly smaller drop in performance for the ALBERT model.

		Ernie		Albert		
		Base	Large	Base	Large	Xlarge
Abusive	Prec	0.855	0.863	**0.874**	0.852	0.838
	Rec	**0.911**	0.898	0.836	0.880	0.870
	F1	**0.882**	0.880	0.855	0.866	0.854
Hateful	Prec	**0.500**	0.471	0.320	0.452	0.448
	Rec	0.353	0.372	**0.510**	0.307	0.318
	F1	0.414	**0.416**	0.393	0.365	0.372
Normal	Prec	0.863	0.864	**0.880**	0.837	0.842
	Rec	**0.879**	0.873	0.808	0.873	0.867
	F1	**0.871**	0.868	0.843	0.854	0.854
Spam	Prec	**0.592**	0.575	0.523	0.527	0.531
	Rec	0.542	0.556	**0.641**	0.460	0.483
	F1	0.566	0.565	**0.576**	0.491	0.505
Weighted Average	Prec	**0.807**	0.805	0.803	0.779	0.779
	Rec	**0.814**	0.811	0.777	0.790	0.789
	F1	**0.810**	0.808	0.787	0.783	0.783
Macro Average	Prec	**0.702**	0.693	0.649	0.667	0.665
	Rec	0.671	0.675	**0.699**	0.630	0.634
	F1	**0.683**	0.682	0.667	0.644	0.646

Table 7.6.: Results on **F** given as precision, recall, and F1 for each label, and their weighted and macro average.

		Ernie		Albert		
		Base	Large	Base	Large	Xlarge
Hateful	Prec	**0.527**	0.451	0.343	0.389	0.426
	Rec	0.322	0.372	**0.493**	0.316	0.294
	F1	0.400	**0.408**	0.404	0.349	0.348
Offensive	Prec	0.860	0.858	**0.883**	0.852	0.835
	Rec	**0.920**	0.910	0.858	0.871	0.890
	F1	**0.889**	0.883	0.870	0.861	0.862
Neither	Prec	0.960	0.962	**0.963**	0.952	0.953
	Rec	**0.965**	0.955	0.942	0.958	0.954
	F1	**0.963**	0.959	0.953	0.955	0.953
Weighted Average	Prec	**0.914**	0.911	0.913	0.900	0.898
	Rec	**0.921**	0.914	0.899	0.904	0.905
	F1	**0.916**	0.912	0.905	0.902	0.901
Macro Average	Prec	**0.782**	0.757	0.730	0.731	0.738
	Rec	0.736	0.746	**0.764**	0.715	0.713
	F1	**0.751**	0.750	0.743	0.722	0.721

Table 7.7.: Results on **F*** given as precision, recall, and F1 for each label, and their weighted and macro average.

		Ernie		Albert		
		Base	Large	Base	Large	Xlarge
Hate	Prec	**0.608**	0.470	0.342	0.423	0.509
	Rec	0.564	0.475	0.380	**0.676**	0.642
	F1	**0.586**	0.472	0.360	0.520	0.568
Off	Prec	0.782	**0.837**	0.769	0.789	0.788
	Rec	**0.941**	0.809	0.881	0.870	0.915
	F1	**0.854**	0.823	0.821	0.828	0.847
Neither	Prec	0.988	0.933	0.977	0.986	**0.990**
	Rec	0.907	**0.945**	0.910	0.905	0.906
	F1	0.946	0.939	0.942	0.944	**0.946**
Weighted Average	Prec	**0.913**	0.886	0.891	0.906	0.912
	Rec	**0.902**	0.886	0.879	0.885	0.897
	F1	**0.904**	0.886	0.883	0.892	0.902
Macro Average	Prec	**0.793**	0.747	0.696	0.733	0.762
	Rec	0.804	0.743	0.724	0.817	**0.821**
	F1	**0.795**	0.745	0.708	0.764	0.787

Table 7.8.: Results when training on **O*** and testing on $\mathbf{S}_{test}$ given as precision, recall, and F1 for each label, and their weighted and macro average.

		Ernie		Albert		
		Base	Large	Base	Large	Xlarge
Hate	Prec	**0.486**	0.448	0.244	0.285	0.293
	Rec	0.328	0.352	**0.598**	0.481	0.395
	F1	0.392	**0.394**	0.347	0.358	0.337
Off	Prec	0.888	0.889	**0.925**	0.907	0.872
	Rec	**0.940**	0.932	0.816	0.859	0.883
	F1	**0.913**	0.910	0.867	0.882	0.877
Neither	Prec	**0.962**	0.960	0.961	0.958	0.961
	Rec	**0.953**	0.947	0.914	0.934	0.924
	F1	**0.957**	0.954	0.937	0.946	0.942
Weighted Average	Prec	0.911	0.908	**0.912**	0.905	0.894
	Rec	**0.917**	0.912	0.862	0.883	0.882
	F1	**0.913**	0.910	0.881	0.893	0.887
Macro Average	Prec	**0.779**	0.766	0.710	0.717	0.709
	Rec	0.740	0.744	**0.776**	0.758	0.734
	F1	**0.754**	0.753	0.717	0.729	0.719

Table 7.9.: Results on **C** given as precision, recall, and F1 for each label, and their weighted and macro average.

		Test		
	ERNIE	Davidson	Founta*	SOLID
	Davidson et al.	0.765	0.638	0.636
	Founta et al.*	0.593	0.751	0.704
Train	OLID*	0.520	0.658	0.795
	Davidson et al. Founta et al.*	0.638	0.755	0.715
	Founta et al.* Davidson et al.	0.750	0.660	0.652
	Combined	0.751	0.756	0.739

	ALBERT	Davidson	Founta*	SOLID
	Davidson et al.	0.748	0.645	0.605
	Founta et al.*	0.535	0.743	0.689
Train	OLID*	0.489	0.666	0.787
	Davidson et al. Founta*	0.599	0.752	0.694
	Founta et al.* Davidson et al.	0.737	0.703	0.679
	Combined	0.670	0.734	0.690

Table 7.10.: Cross-dataset results for **D**, **F*** and **S** with ERNIE Base and ALBERT Base as models. The rows indicate which dataset the model was trained on, and the columns show which dataset the model was tested on. The top table is with ERNIE as model and the bottom one is with ALBERT as model.

8. Evaluation and Discussion

The first part of this chapter evaluates the results and experiments of chapter 7, as well as choices made leading up to and during these experiments. The label mappings used when transforming **O** and **S** into the label space of **D** will be thoroughly discussed. A comparison of the results with state-of-the-art systems and other literature will also be given. The second part will discuss the findings up against the goals and research questions in section 1.2.

8.1. Evaluation

The experiments of this thesis were conducted on two model architectures written in two languages, where each architecture had two and three size modes each, resulting in a total of five systems. The results on these experiments and the experiments themselves will be evaluated in this section. Both in-domain, and cross-dataset experiments have been conducted.

8.1.1. Choice of datasets

The different datasets used in this work are all described in detail in chapter 3, and some reasons for selecting the ones used were further elaborated in section 6.1. The choice of datasets were explained in terms of convenience in mapping the datasets into the same label-space. The datasets used were Founta et al. (2018), Davidson et al. (2017), Zampieri et al. (2019b), and Rosenthal et al. (2020), denoted as **F**, **D**, **O**, and **S** respectively. **O** and **S** were regarded as closely linked datasets, where $\mathbf{S}_{\text{train}}$ is annotated by models seeded with **O**. The biases of these models might therefore manifest themselves in the annotations. Not all of the data was annotated by the models, $\mathbf{S}_{\text{test}}$ was the test set of OffensEval 2020 (Zampieri et al., 2020), and therefore naturally had to be annotated by people.

OffensEval 2020 can be regarded as a good test arena for Zampieri et al. to test whether the use of the large generated dataset would be useful. Since **S** was used in the preliminary work described in chapter 5, it naturally followed into the exploration phase of this thesis, using the part of **O** originally used for training, $\mathbf{O}_{\text{train}}$ as development set. Some of the hyperparameters might therefore not have been optimal, but the close connection between the two datasets makes it a reasonable assumption that the hyperparameters might be similar. The top model in OffensEval 2020 did not use $\mathbf{S}_{\text{train}}$ on task A, while many others did with scores very similar to that of the top scoring model. $\mathbf{S}_{\text{train}}$ was used for task B and C, where there are fewer examples and thus the extended dataset had better use. The use of **O** as training data instead of **S** became reasonable when it became apparent that the amount of training data was extremely large compared to the limited potential benefit it yielded. The use of combined datasets later in the experiments would also have completely overpowered the data of the other datasets, and the use of **O** in all experiments was therefore a more manageable decision. A better solution would have been to use the resulting 'Hateful' and maybe 'Offensive' labels from the converted dataset. One could for example use $\mathbf{S}_\mathrm{B}$ and $\mathbf{S}_\mathrm{C}$ but not $\mathbf{S}_\mathrm{A}$. This would have given much more data for the models to train with, and still kept the number of labels reasonable. This was the strategy of Galileo (Wang et al., 2020a), the top scoring team in OffensEval 2020 on task B and C.

When setting aside 20% of **D** and **F** for test data and using the test data of **S**, the amount of training data is 19,826, 68,620, and 14,100 examples for **D**, **F**, and **O** respectively. This results in **F** being much larger than the two other datasets, and could introduce a bias. All datasets have around 5% hateful speech, while **D** has a surplus of offensive language, and **F** and **O** have a surplus of normal data. This evens out the skewed relationship between the 'Offensive' and 'Neither' labels, but keeps the 'Hateful' label as a minority.

8.1.2. Label Mapping

When trying to compare effects of training a model on one dataset and testing on another, or to combine several datasets, the issue of synchronising the labels arise. This task has to be made through analysing the similarity of definitions used for different labels. In some cases such a mapping from one label-space to another does not really exist, and one must resort to the use of sub-optimal label mappings. Swamy et al. (2019) experiment with cross-dataset performance, but do so with binary labels, only separating classes into positive and negative. The importance of separating between hateful and offensive language, as stated by Isaksen (2019), thus disappears. This binary separation is an easier task than separating hate speech from offensive language, but is a valuable baseline that surfaces the lack of generalizability across datasets. Swamy et al. also did experiments with annotating both **D** and **F** with labels from the hierarchical structure of OLID described in 3.4, using a BERT based model trained on **O** to annotate. From the tweets labeled as 'Hateful' in **D**, 63.22 percent were annotated as targeted towards an individual, while only 24.61 percent were annotated as targeted towards a group. The percentages were 75.56 and 10.42 for labels 'IND' and 'GRP' when tested on **F**. This signals that mapping tweets labeled as 'GRP' in **O** and **S** to hate speech might not be reconcilable with the data.

To be able to properly assess the compatibility of datasets, some of the definitions used are given here. **D** and **F** have very clear and similar definitions. Here is the definition of hate speech in Davidson et al. (2017):

> *"Language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group."*

The definition of Founta et al. (2018) is very similar:

"Language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnic origin, sexual orientation, disability, or gender."

The difference between the two is that in **F** hatred can be expressed towards an individual, not just a group, and the insults have to be based on specified attributes. Insults targeted at an individual will therefore never be regarded as hate speech in **D**, while **F** can have hate speech targeting an individual. Insults targeting a group will in **D** always be hateful, while **F** only regard a subset of insults targeting groups to be hateful. Zampieri et al. (2019b) do not include a formal definition of hate speech, but give an indication of what is regarded as such. This is their definition of the 'GRP' class:

"Posts targeting a group of people considered as a unity due to the same ethnicity, gender or sexual orientation, political affiliation, religious belief, or other common characteristic. Many of the insults and threats targeted at a group correspond to what is commonly understood as hate speech."

This shows that Zampieri et al. approximate hate speech as a subset of insults against a group. This corresponds to the last part of the definition given by Founta et al. but without the first part about targeting an individual. From the definitions one would therefore expect that both **D** and **F** are more biased towards the 'Hateful' class than **S**. This contradicts the findings of Swamy et al., where several of the hateful tweets in **D** were labeled as targeted towards an individual.

It may be that some bias was present in **O**, leaning towards the 'IND' label. A tweet could conceivably be targeted towards an individual based on their group, and thus be eligible for annotation into both 'GRP' and 'IND'. There are also much more tweets in the 'IND' class, further enabling a bias towards 'IND'. All confusion matrices on cross-domain experiments show a majority of hateful tweets being mislabeled. This indicates that the bias towards 'Offensive' is due to the difficulty of predicting hateful tweets. There are many factors in play, and it is probable that there is some skewed overlap between targets, hate speech, and offensive language. The results on experiments involving datasets with converted labels should therefore be analysed carefully and with great caution, where different annotation methodologies, annotators, and definitions have been used. Certain trends seem to be present, but the difficulty of clearly defining the labels makes it necessary to do more tests and to explore the data more thoroughly.

Violation of the assumptions made when mapping the labels can make interpretation of results difficult. The next paragraphs will try to shine some light on what effects might be present due to such violations. As a reminder, the label scheme of OLID, used in **S** and **O**, was mapped to the label scheme of Davidson et al. (2017). The details of these mappings can be seen in section 7.1, but the most controversial mappings, and therefore most important to evaluate, are 'IND' to 'Offensive' and 'GRP' to 'Hateful'. These mappings result in that all offensive speech targeted at individuals will be regarded as just offensive, while all offensive tweets targeted at a group will be regarded as hateful. The following evaluation will regard a scenario where a model has been trained on **D**, and will be tested on **S**. Both datasets are regarded as gold standards of the original label scheme they employ, so a tweet annotated as 'Hateful' in **D** is regarded as being truly hateful, and a tweet annotated as 'IND' in the original dataset $\mathbf{S}_{\mathrm{C}}$ is regarded as really targeted towards an individual. A tweet annotated as 'Hateful' in the transformed dataset **S** is not regarded as really hateful, but assumed hateful through the label mapping.

Another assumption is that these underlying probabilities are the same for both datasets:

- $P(Hate|Ind)$: The probability of a tweet being hateful given that it is targeted towards an individual.
- $P(Hate|Grp)$: The probability of a tweet being hateful given that it is targeted towards a group.

The violation of this assumption is not part of this evaluation, and the assumption is regarded as true. The distribution of hate speech conditioned on the target might be different in different datasets, due to different boosted sampling methods used to increase the amount of hate speech when collecting the datasets. These differences are hypothesised to be small, and the probabilities will therefore be regarded as equal in this analysis. The probabilities evaluated here, are these:

$$\begin{aligned} P(Hate|Ind) &= 0.0 \\ P(Hate|Grp) &= 1.0 \end{aligned} \tag{8.1}$$

They describe the assumptions, that no fraction of offensive language targeted towards individuals is hateful, and that all offensive language targeted at a group is hateful. A plausible scenario that breaks these assumptions is tweets in **D** being hateful but targeted towards individuals. This implies that **S** also contains some tweets that are hateful and targeted towards individuals, and thus wrongly annotated as 'Offensive'. If the model then correctly predicts one of these tweets to be hateful, it would be regarded as wrong by the test set. Violation of the assumptions contribute to a lower score when testing on **S** in this case. A model can also misclassify. If the same tweet is wrongly classified as offensive, it would be regarded as correctly predicted by the test set and would thus aid in improving the performance for the model. A large proportion of hateful labels being targeted at individuals combined with many poor classifications based on i.e. misunderstanding of language would then lead to good performance but not for the desired reason. On the contrary, if many hateful tweets are targeted at individuals and the model is very good at predicting real hateful tweets, this would lead to low performance. The consequence is thus that the model is rewarded for being good if the assumptions are correct, and rewarded for being bad if the assumptions are violated. The amount and direction of change in metrics are thus affected by the model's ability to correctly classify. The probabilities representing this ability is given by:

- $P(Correct|Off)$: The probability of a model predicting correct on an offensive tweet.

- $P(Correct|Hate)$: The probability of a model predicting correct on a hateful tweet.

The equation for macro average F1 score was tried analysed to see how these probabilities affect the scores. The equations became pretty messy and were hard to interpret due to many variables being involved. A widget with sliders was therefore implemented, where the effects of changing the different probabilities were observed. The widget can be seen in the appendix, section A.1, together with an in depth explanation of the calculations and assumptions that were made. It was found that if

$$\frac{P(Correct|Off) + P(Correct|Hate)}{2} > 0.5 \tag{8.2}$$

any incline in $P(Hate|Ind)$ and any decline in $P(Hate|Grp)$ will lead to decrease in macro averaged F1. This strengthens the evaluation that a better than average model will be punished for violation of the assumptions. The 'Hateful' class can be pretty hard to predict, but from the experiments on **D** and **F*** it seems probable that inequality 8.2 holds in most experiments if we exclude the neutral label due to good performance on the 'IND' label. This suggests that the macro F1 scores of **S**, seen in the experimental results, are lower boundaries. If one would correct for the wrongly mapped labels, one would expect the macro F1 scores to increase.

The inclusion of the neutral 'Neither' label will impact the metrics. So will the fact that OLID has a more complicated annotation scheme than utilised here. The rest of the labels are assumed to be correctly mapped, so they would not affect this analysis. These assumptions might also be violated, but through the definitions used for offensive language and hate speech, these assumptions probably introduce little error. This analysis has made quite a lot of assumptions, and a proper mathematical analysis of the full label-mapping is suggested as future work, in order to properly assess whether OLID is compatible with datasets such as **D** and **F**. A general analysis on how making wrong assumptions when mapping fine-grained labels affects performance might exist, but is not know to the author. The mapping from a hierarchical structure to a three label annotation schema, such as done in this thesis, is a very specific scenario, and a useful analysis might be easier to do by restricting the scope of the analysis.

8.1.3. Model Comparison on In-Domain Experiments

The results show ERNIE Base scoring consistently better on all datasets compared to ERNIE Large and all ALBERT models. The different size configurations of ALBERT are less stable in terms of F1 performance, where the base version scores best on **D**, **F**, and **F***, the xlarge version on **S**, and large on **C**. The differences in macro average F1 scores between ERNIE Base and Large are 0.011 and 0.050 on **D** and **S** respectively. A margin of 5 percentage points on these tasks are worth paying attention to. On **D**, **F**, and **S** the differences between ALBERT and ERNIE are around 2 percentage points, while the differences are around 1 percentage point for the remaining datasets. This is in line with the results of Galileo (Wang et al., 2020a), who had the best results on OffensEval 2020, task B and C, using a knowledge distillation network based on ERNIE. They also experimented with a knowledge distillation network based on ALBERT, but used ERNIE as the final model. Two of the team members of Galileo were also co-authors of Sun et al. (2020), giving them an incentive to use ERNIE.

Lan et al. (2020) showed that ALBERT increased performance when increasing the model size, even when the parameters were shared. This reflects the findings of Devlin et al. (2019), where increased model size also lead to increased performance. The unclear difference between different model configurations of the ALBERT based models used in this thesis might be due to different parameters being used, such as batch size and choice of optimizer. The use of a larger batch size in ALBERT Base might have lead to increased performance, but with some loss of comparability with ALBERT Large and Xlarge, that had to be used with smaller batch sizes due to limitations in memory. The experiments using ERNIE do not match with the experiments of Sun et al. (2020), who showed that ERNIE performed consistently better with larger model sizes.

With paired, two tailed student's t-tests on the different models, the only models that score significantly different on the macro F1 score, (with a significance level of 0.05), are ERNIE Base and ALBERT Large, with ERNIE Base being better than ALBERT Large. The average of ERNIE models versus the average of ALBERT models also yields a significant difference between ERNIE and ALBERT. The hypothesis used for the tests was:

- H_0: There is no difference between the models.
- H_1: There is a difference between the models.

The tests were run with paired training data from the different datasets. Dataset **C** and **F*** were not used in the tests, because it would break the assumption of independence of data between the samples. The significant tests suffer from a very small sample size of only three. This yields very low statistical power for the tests and the probability of type II error is therefore large, meaning that there is a low probability of rejecting H_0 even when H_1 is true. The in-domain experiments can therefore give an indication of ERNIE being better than ALBERT, but one is not clearly better than the other in these experiments. A larger sample size is needed to answer this, which can be done through i.e. n-fold cross-validation.

For all experiments where the difference between the best ERNIE and ALBERT models is over 1 percentage point, this is mainly due to the 'Hateful' class. For **D** and **S** almost all differences are due to this class, while **F** and **C** have differences more evenly distributed among the labels. The greatest display of difference between model size configurations can be seen in table 7.8 on the 'Hateful' class. Here, ERNIE has a difference of 11 percentage points between the base and large versions, while ALBERT has a difference of 21 percentage points between the base and xlarge versions. While this might be because the 'Hateful' class is hard to predict, this could also be due to the models being quite unstable. This was also experienced by Devlin et al. (2019) for experiments with BERT Large, who used an average of 5 random restarts when fine-tuning due to this issue. The larger versions of ALBERT were also experienced to be quite unstable during the exploratory phase, but no random restarts were conducted. This fits with the observations of Mosbach et al. (2020), who found that BERT, ALBERT, and RoBERTa suffer from unstable training due to vanishing gradients. The experiments show that ERNIE always favors better precision on the ‘Hateful’ class, while ALBERT is more split in-between the different datasets. This might be due to oversampling being used with ALBERT, but not ERNIE. The higher amount of ‘Hateful’ examples has probably given ALBERT an incentive to predict more ‘Hateful’ labels. The only significant difference on the F1 score of the ‘Hateful’ class is, as with the macro F1 score, between ERNIE Base and ALBERT Large.

8.1.4. Model Comparison on Cross-Dataset Experiments

To make experiments easier and more credible to compare, all the presented cross-domain experiments were done with the base version of either ERNIE or ALBERT, and the same parameters were used, for example the learning rate was e^{-5} for all experiments. The largest decline in score when doing the pure cross-domain experiments with single datasets can be seen when training on **O** and testing on **D**. The lowest drop in performance is seen when training with **O** and testing on **F***, or when training with **F*** and testing with **S**. This drop is still pretty substantial, with around 9 percentage points decline. The similarity between Founta et al. and OLID was also noted by Swamy et al. (2019). Their experiments were conducted on binary labels, mapping all offensive labels, both hate speech and offensive language, to one positive label. They also found that a model trained on Davidson et al. performed well on Founta et al. and mentioned that this might have to do with Davidson et al. containing a majority of positive labels. This thesis works with three labels and thus complicates the notion of a positive label. The increased granularity shows that the generalization found when testing on Founta et al. decreases greatly when introducing the ‘Hateful’ label.

Swamy et al. (2019) found that a model trained on Davidson et al. performed well on Founta et al. with binary labels, with a small performance drop of two percentage points. Introducing the ‘Hateful’ label in the experiments done in this thesis increases this performance drop to around 11 percentage points, which is more than the performance drop found between **S** and **F***, which is around 9. This makes the assumption that the ‘GRP’ label in SOLID is similar to the ‘Hateful’ label plausible, but on the assumption that the difference in dataset content is similar between all datasets, all performance loss is due to incompatibilities in labels, and that the performance loss is similarly distributed between the labels for both **S** and **F***. Swamy et al. (2019) found OLID and Founta et al. to be similar in content by searching terms used when compiling the datasets. If OLID and $SOLID_{test}$ share this characteristic, **S** and **F*** would be similar in content, which could be the explanation for how the bluntly generated ‘Hateful’ label of **S** can be comparable with cross-dataset performance between **F*** and **D**.

Student’s t-tests were also done on the cross-validation experiments, showing a significant difference between macro average F1 scores of ERNIE Base and ALBERT Large, ERNIE Large and ALBERT Base, and ALBERT Large and ALBERT Xlarge, with ERNIE models being better than ALBERT models, and ALBERT Xlarge being better than ALBERT Large. This is in line with the findings of Lan et al. (2020), where the larger model configurations of ALBERT lead to better performance. The hypothesis tests also show that ERNIE is significantly better at predicting the 'Offensive' class on cross-dataset experiments, compared to ALBERT.

The performance of the models naturally increase when continuing to train the models on multiple datasets. Adding the in-domain training sets has the potential to bring the score back up to in-domain performance. Fine-tuning with **DF***, and then testing on **F*** results in a score slightly higher than when just training on **F*** for both ALBERT and ERNIE. The differences are quite low, only 0.5 percentage points on ERNIE and 0.9 percentage points on ALBERT, and can thus only be regarded as being comparable with the single dataset experiments. More interesting experiments are the ones where the training data is **DF***, and test data is **D**, or training data is **F*D** and test data is **F***. The experiments show that ERNIE loses less performance on the first combination, while ALBERT loses less performance on the second. **F*** is clearly larger than **D**, and it thus makes sense that the performance decreases more when training on **F***. The performance when testing on **S** declines when training on **D** after **F***, showing the larger incompatibility between **S** and **D**. The presence of an incompatibility between **S** and other datasets is expected, due to the bold assumptions made when mapping the labels into the same label-space.

8.1.5. Biases and misclassifications

As was identified by Malmasi and Zampieri (2017), Davidson et al. (2017), and Isaksen (2019) when conducting experiments on **D**, the biggest challenge when it comes to hate speech detection seems to be separating hateful and offensive language. In the confusion matrices of ERNIE Base on the different datasets, in table 8.1, 8.2, 8.3, and 8.4, we see that the only dataset having more true positives than false negatives on the hateful label is **S**. This is also the only dataset without an original hateful class, but a label-mapping that might have done the classification easier because the classification of offensive tweets targeting a group is a simpler task than classifying hate speech. For both **D** and **S**, the separation between 'Neither' and 'Hateful' seems to be clear, while for **F** and **F*** the true hateful examples are distributed more equally between 'Hateful', 'Offensive', and 'Neither', with 'Abusive' and 'Hateful' regarded as the same class. When we look at **S**, this clear distinction might come from the hierarchical annotation methodology of **S**, allowing the annotators to easily separate offensive and normal language as a first step, which is a less demanding task than keeping track of several labels at hand at all times.

Another option is that the separation of hateful and normal language should be easy, where both **S** and **D** have very few misclassifications of true hate speech as normal, and that the peculiarity lies in **F**. It is strange that the model is able to distinguish offensive and normal language when the true label is offensive, but not when it is hateful, looking at **F** and **F***, even though it is natural to assume that hateful language is also offensive. This indicates that **F** might contain more implicit hateful language than the other datasets. This would make the misinterpretation of hateful language as normal language more probable. This same misclassifications were also observed in the experiments of Isaksen (2019), who upon closer inspection of the misclassified tweets saw that some hateful tweets might have been wrongly annotated as normal language.

		Predicted label		
		Hateful	Offensive	Neither
True label	Hateful	107	137	14
	Offensive	111	3705	63
	Neither	11	70	739

Table 8.1.: Confusion matrix for **D** using Ernie Base

		Predicted label			
		Hateful	Abusive	Neither	Spam
True label	Hateful	220	190	205	8
	Abusive	104	2573	128	18
	Neither	110	201	7293	691
	Spam	6	47	826	1040

Table 8.2.: Confusion matrix for **F** using Ernie Base

		Predicted label		
		Hateful	Offensive	Neither
True label	Hateful	195	212	198
	Offensive	87	2569	136
	Neither	88	207	8094

Table 8.3.: Confusion matrix for **F*** using Ernie Base

The first cross-dataset confusion matrices getting evaluated are the ones with $\mathbf{D}_{\text{test}}$ as test set. The original confusion matrix for the in-domain tests can bee seen in table 8.1, while the matrix for the model trained on $\mathbf{F*}_{\text{train}}$ can be seen in table 8.5 and $\mathbf{O}_{\text{train}}$ in table 8.6.

In the following section, a model trained on a dataset, i.e. $\mathbf{F*}_{\text{train}}$, will be denoted as model$_{\mathbf{F*}}$ to make the language more understandable. The in-domain experiments already had a tendency of annotating hateful tweets as offensive. The cross-domain experiments continue to wrongly predict on hateful tweets, but model$_{\mathbf{F*}}$ mostly predicts hateful tweets as offensive, while model$_{\mathbf{O}}$ predicts the majority of hateful tweets as neither. One would expect that model$_{\mathbf{O}}$ had many hateful tweets annotated as offensive, but it actually mislabel less hateful tweets as offensive than the in-domain experiment on **D**. The 'Neither' class is not disputed in terms of any label conversions, and this indicates that there exists some deep incompatibility between the two datasets. Mode$_{\mathbf{O}}$ continues this tendency on the 'IND' label. A possibility is that model$_{\mathbf{O}}$ has not been able to capture implicit hateful or offensive speech. Caselli et al. (2020) found that a majority of **O** is explicit. Since the 'Hateful' class of **O** is based on the 'GRP' class, the number of

		Predicted label		
		Hateful	Abusive	Neither
True label	Hateful	101	76	2
	Abusive	40	1118	30
	Neither	25	236	2546

Table 8.4.: Confusion matrix for **S** using Ernie Base

		Predicted label		
		Hateful	Offensive	Neither
True label	Hateful	82	145	31
	Offensive	311	3100	468
	Neither	42	95	683

Table 8.5.: Confusion matrix for ERNIE Base, trained on **F** and tested on **D**.

implicit hate speech might be very small. If the model looks for explicit offense, more implicit offensive or hateful tweets might be predicted as neutral. Both model$_{\mathbf{F*}}$ and model$_{\mathbf{O}}$ predicts many offensive tweets to be neutral. This indicates that the annotators of Davidson et al. (2017) were more easily offended than the ones of Founta et al. (2018) and Zampieri et al. (2019b).

The second dataset evaluated is $\mathbf{F*}_{\text{test}}$. The tendency of predicting hateful tweets as offensive continues here. The confusion matrices for training sets $\mathbf{F}_{\text{train}}$, $\mathbf{D}_{\text{train}}$, and $\mathbf{O}_{\text{train}}$ are given in tables 8.2, 8.7, 8.8 respectively. All the models predict many of the offensive and hateful tweets as neutral. This might be due to a larger amount of implicit offensive language in the test set, which Caselli et al. (2020) show is very hard to predict. A possible reason for this might be that more explicit tweets have been easier to discover on Twitter, and they have therefore been more prone to be removed.

Model$_{\mathbf{O}}$ is a little better than model$_{\mathbf{D}}$ at predicting hateful tweets, but this might be due to a stronger bias towards hateful tweets, where lots of offensive tweets are wrongly predicted as hateful. The precision of the 'Hateful' label is strongly influenced by the relatively large amount of offensive test data. A small percentage of offensive tweets being wrongly annotated as 'Hateful' will heavily impact the precision of the 'Hateful' class. Likewise, any misclassifications of hateful tweets will probably not affect the precision of the 'IND' class to a large extent due to the imbalance.

The last cross-domain confusion matrices to be evaluated are the ones on $\mathbf{S}_{\text{test}}$. This dataset is closely related to the former discussion on label mapping in section 8.1.2. As with all other test sets, lots of hateful tweets are predicted as offensive, where model$_{\mathbf{D}}$ has the hardest time on the 'Hateful' class. It is hard to say whether this is because tweets have been annotated wrongly as hateful, or if this is due to the models being bad at predicting. For model$_{\mathbf{F*}}$, few offensive tweets are mislabeled as hateful relative to the amount of total hateful tweets. This indicates that tweets annotated as offensive in $\mathbf{S}_{\text{test}}$, actually are offensive. If many of them were supposed to be annotated as hateful, one

		Predicted label		
		Hateful	Offensive	Neither
True label	Hateful	33	95	130
	Offensive	517	2137	1225
	Neither	6	21	793

Table 8.6.: Confusion matrix for ERNIE Base, trained on **O** and tested on **D**.

		Predicted labels		
		Hateful	Offensive	Neither
True labels	Hateful	29	308	268
	Offensive	45	2482	265
	Neither	5	267	8117

Table 8.7.: Confusion matrix for ERNIE Base, trained on **D** and tested on **F**.

would expect to see more offensive tweets misclassified as hateful. Offensive and hateful tweets are rarely classified as neutral, which might be due to the large proportion of explicit offense in $\mathbf{S}_{\text{test}}$.

8.1.6. Comparison with state-of-the-art

While it is valuable to compare the models between each other using the current experiments, it is also valuable to compare with other work done on the same datasets. Some datasets used are not comparable to previous work done on hate speech detection, due to the converted label-space used. Luckily, dataset **F** and **D** have been used in several experiments. The performance on the dataset is not necessarily the main focus of the different publications, such as Swamy et al. (2019) where cross-domain performance is central, and Liu et al. (2020) where the use of data augmentation techniques to improve performance when down-sampling the dataset is the focus. Nevertheless, the results are still up for comparison.

Different results obtained on dataset **D** can be seen in table 8.11. When presenting results, some publications use weighted average F1, and some macro average F1 score. Both are therefore presented here. The results from both ERNIE Base and ALBERT Base are included in the comparison. Dataset **D** has the weakness that no clearly defined test-set is given. This means that the different experiments have different test data. Some have used cross-validation, which tackles this issue, but some have not. Nevertheless, all the models score very similarly with all results around 0.90 on weighted average F1. Kshirsagar et al. (2018) and ERNIE Base score equally on the weighted average, but Kshirsagar et al. has 2 absolute percentage points improvement over the next best model on macro averaged F1 score using a Transformed Word Embedding Model (TWEM). This is due to an F1 score on the 'Hateful' class of 0.49. Isaksen and Gambäck (2020) and ERNIE Base also report F1 on individual labels. They score 0.428 and 0.439 respectively. In the original experiments of Davidson et al. (2017) they achieved an F1 score of 0.51

		Predicted labels		
		Hateful	Offensive	Neither
True labels	Hateful	82	225	298
	Offensive	262	2116	414
	Neither	26	106	8257

Table 8.8.: Confusion matrix for ERNIE Base, trained on **O** and tested on **F**.

		Predicted		
		Hateful	Offensive	Neither
True labels	Hateful	16	134	29
	Offensive	66	970	152
	Neither	8	71	2728

Table 8.9.: Confusion matrix for ERNIE Base, trained on **D** and tested on **S**.

on the ‘Hateful’ class. All the different work show that the ‘Hateful’ label is the most difficult label to get right, but the simpler models show better results. This counters the notion that BERT based models can capture complex structures that simpler models are not able to capture. It might also be that the BERT based models more easily introduce bias when a class is hard to predict. This is again the opposite of what experiments of Swamy et al. indicated, where the LSVM and LSTM were worse than BERT, though not by a lot.

The issue of different test and training data is worse in dataset **F** than in **D**. Because different amounts of tweets were available for the different publications, the available tweets are presented next to the results for a fair comparison. The best results have access to the whole dataset and were achieved with BERT based models. Liu et al. (2020) have a suspiciously high score, considering that they only used vanilla BERT. There is no apparent reason why this should be much better than i.e. Swamy et al. (2019). They filtered out all tweets with more than 30 tokens but were still left with 99,603 tweets, so this cannot have impacted the score much. They also tested with a CNN, Bi-LSTM with attention, and a Transformer network, where all models were close to 10 absolute percentage points better than the best score of the other publications reported. This indicates that the results are not credible. The next highest macro average F1 score is therefore also highlighted in table 8.12. From the other models, Lee et al. (2018) is the only publication that did not use a BERT based model and has lowest macro average F1 score. They had access to less tweets than the top scoring models but had more than Isaksen and Gambäck (2020), ERNIE Base, and ALBERT Base. Isaksen and Gambäck (2020), Swamy et al. (2019), and Liu et al. (2020) all had baselines with recurrent and classical machine learning methods that were outperformed by BERT based models. The results on Founta et al. (2018) therefore point towards BERT based models being better than the classical. The number of available tweets leaves BERT from Isaksen and Gambäck (2020) and the GRU from Lee et al. (2018) as the models fit for comparison

		Predicted		
		Hateful	Offensive	Neither
True labels	Hateful	67	105	7
	Offensive	112	1005	71
	Neither	63	100	2644

Table 8.10.: Confusion matrix for ERNIE Base, trained on **F** and tested on **S**.

Model	Weighted Avg	Macro avg
TWEM (Kshirsagar et al., 2018)	**0.92**	**0.79**
BERT (Swamy et al., 2019)	-	0.77
GRU + Metadata (Founta et al., 2019)	0.89	-
BERT (Isaksen and Gambäck, 2020)	0.90	0.76
LR (Davidson et al., 2017)	0.90	-
ERNIE Base	**0.92**	0.77
ALBERT Base	0.91	0.75

Table 8.11.: Results of different publications on dataset **D** given as weighted and macro average F1.

with ALBERT and ERNIE Base. Both ALBERT Base and ERNIE Base score similarly to BERT and better than the GRU. Since ERNIE is technically the same model as BERT, this is not very surprising. The comparable score of ALBERT indicate that the size reduction does not limit ALBERT to a great extent. This is in line with the observations of Lan et al. (2020), where ALBERT Base scored slightly worse than BERT.

The instability of BERT and ALBERT (Mosbach et al., 2020) also has to be taken into account when evaluating these observations. Cross-validation was not used by Isaksen and Gambäck (2020) or the experiments on ALBERT and ERNIE, and the results might therefore be sensitive to statistical fluctuations.

Model	Weighted avg.	Macro avg.	Tweets
GRU-LTC (Lee et al., 2018)	0.805	0.653	70,904
BERT (Swamy et al., 2019)	-	0.696	99,996
BERT (Isaksen and Gambäck, 2020)	0.806	0.672	68,299
CRAB (Zahiri and Ahmadvand, 2020)	-	**0.702**	99,996
BERT (Liu et al., 2020)	-	**0.814**	99,603
ERNIE Base	**0.810**	0.683	68,299
ALBERT Base	0.787	0.667	68,299

Table 8.12.: Results of different publications on dataset **F** given as weighted and macro average F1.

8.2. Discussion

Section 8.1 discussed several topics related to the experiments but focused on one section of the experiments at a time. This section will discuss the work of this thesis up against other work in the field and will be guided by the research questions presented in section 1.2.

Research question 1 *How much do irreconcilable properties of different offensive language datasets impact performance of deep learning models?*

Some properties of datasets that might affect how models train on them are definitions, annotation, topics, language, and format. Some of the categories are closely related, such as definitions with annotation, and language with format. The definitions and annotation deal with how the dataset was annotated, how the different classes were defined, and important steps in the methodology used to annotate the dataset. This also includes an inherent impossibility of testing inter-annotator agreement on cross-domain annotations. Topic is related to language, where topic is closely related to the words used in the dataset.

Zampieri et al. (2019b) used a list of terms to search for offensive language, and this list is biased towards political terms. Swamy et al. (2019) found that **O** gets 12,200 hits when searching with these terms, while **F** gets 6,600. Searching the datasets used in this thesis, results in **O**, **F**, and **D** having 11,450, 1,470, and 290 hits respectively on these terms. If only the political terms "gun control", "conservatives", "antifa", "maga", and "liberals" are counted, **D** and **F** have 23 and 244 hits each, while **S** has 5,611. Whether the low overlap on these terms is due to changes in the political landscape, Twitter usage, or data collection methodologies is uncertain, but nevertheless, there is a difference between the datasets. The terms *"ni**a"*, *"ni**er"*, *"dyke"*, and *"jihadi"* are used much more in **D** than in the two other datasets, indicating that topics related to these terms are more prevalent in **D**. This is in line with that the worst cross-domain experiments are on $\mathbf{D}_{\text{test}}$, where one would expect to find many words not used in the other datasets. This is also in line with Nobata et al. (2016)'s findings, that recent tweets have to be used in training to reliably predict on other recent tweets.

The lack of words in datasets is not an irreconcilable property, and one would expect the total performance to stay stable when combining the datasets. The experiments on dataset **C** show that this is the case, where the results are comparable to experiments done on the single dataset experiments. The largest performance drop is found when testing $\text{model}_{\mathbf{C}}$ on $\mathbf{S}_{\text{test}}$, purely due to a performance drop on the 'Hateful' label. This is natural, since the performance on 'Hateful' was much larger for **S** than the two other datasets on in-domain experiments.

A possibility of how the combined dataset performs so closely to the in-domain experiments is by learning the format of the inputs. Even with preprocessing and data cleaning, practical differences in formatting can still be present in the datasets, such as the number of user mentions, hashtags, and general cleanliness of tweets. This can make the network learn which tweet belongs to which dataset and make prediction based on this. It would not directly make the datasets incompatible, but would be a form of overfitting. After all, the goal is not to make the model make decisions based on what dataset the data comes from. With this in mind, the larger loss of performance on **S** might be a good thing. Contrary to **D** and **F**, in **S** all users are anonymous. This is done by replacing user mentions with @USER. This tag appears on a majority of tweets, and would therefore be a strong marker for dataset **S**. If the model was always able to recognise the dataset based on the user mention, one would not expect the performance of **S** to drop from the in-domain experiments to the combined-domain experiments. The model could still be exploiting this feature, but it is clear that it does not do it in all cases. The experiments on **C** show that a model does not lose much performance by combining datasets, which at least means that the models are able to efficiently utilise all datasets.

The definitions used for annotation, and the possible errors these can introduce are thoroughly discussed in section 8.1.2. The discussion brings forth great uncertainties. The experiments done on **C** indicate that a model can learn to perform well on all datasets, as long as all training is done with shuffled datasets, and thus avoids the problem of *catastrophic forgetting* (Mosbach et al., 2020). The term catastrophic forgetting describes the phenomenon where a model trained on a task A forgets this task once it is trained on task B, because the weights of the model are changed to favor task B. In a sense, this formulates the different datasets as being different tasks. For compatibility between datasets, one could explain catastrophic forgetting by the model disregarding certain topics. For incompatibility between datasets, catastrophic forgetting could be explained with i.e. identification of a dataset becoming unimportant and thus forgotten, leading to loss in performance on the forgotten task. The features with high predictive power on in-domain training seem to be generalizable. This does not mean that generalizable features do not exist, so the question that remains is whether combining the datasets yields good results because of predicting dataset formatting or because of well generalized hate speech features being learned.

Research question 2 *Which models can best capture the continued training of several offensive language datasets?*

The experimental results show that ERNIE consistently outperforms the ALBERT based models on in-domain and cross-domain experiments. A closer statistical analysis reveals that ERNIE Base only significantly outperforms ALBERT Large on in-domain experiments, while ERNIE Base outperforms all ALBERT models on cross-domain experiments, and ERNIE Large outperforms ALBERT Base and Large. This indicates that ERNIE is better at generalizing between datasets than ALBERT, but it could be due to the model just being better at hate speech detection in general. The reason the significant differences can only be seen on the cross-domain experiments might be that model differences become enlarged because cross-domain prediction is harder and requires more generalization.

The continued training of several offensive language datasets seems to suffer from catastrophic forgetting, which is why Sun et al. (2020) used continual multi-task training to avoid this when pre-training ERNIE. The combined training of several datasets is therefore a more strategic approach that helps the models to learn. While this might introduce forms of overfitting to cope with the different datasets, it also enables the models to learn representations for all data simultaneously. ERNIE Base and Large have significantly better results than the ALBERT models on dataset **C**, showing that ERNIE deals with the combined data better than ALBERT. Karan and Šnajder (2018) used the FEDA (Daumé III, 2007) framework to combine several datasets with good results, but this framework enables a model to learn features from several domains at the same time by duplicating features. This would not be applicable in the experiments, because having data-specific features would clearly enable the model to overfit on the different datasets instead of learning a shared representation of hate speech.

When exploring the effects of cross-dataset evaluation, previous work has mostly looked at the data, not the model in use. Swamy et al. (2019) compared several models on in-domain experiments, but only BERT was used in cross-domain experiments. The same applies for Nejadgholi and Kiritchenko (2020), who also used BERT as their only model during cross-domain experiments. It is understandable that only one model is utilised, as Gröndahl et al. (2018) argued that data is more important than model. The experiments of this thesis therefore stand somewhat alone, and show that ERNIE performs similar to or better than ALBERT.

Clearly stating what parameters work best for these models is hard. The experimental phase showed that ALBERT was very sensitive to random re-initialisation, similarly to experiments done by Mosbach et al. (2020) on BERT. Good parameters could therefore be discarded during the extended hyperparameter optimization, due to vanishing gradients. Larger batch sizes should probably be used when using the LAMB optimizer, but as long as the batch size is 32 or above, the results are reasonable. Any lower than this will probably result in unstable training. A learning rate of around 1×10^{-5} seems to be working well, and AdamW or LAMB seems to work better than Adam. The optimal hyperparameters remain very task dependent, and a hyperparameter optimization is therefore necessary. A more rigid hyperparameter optimization could be conducted by i.e. averaging over several runs with the same parameters.

Research question 3 *Can the use of knowledge training objectives during pre-training of a transformer improve performance on hate speech and offensive language identification?*

This research question is directly linked with the performance of ERNIE versus ALBERT. The discussion and evaluation in previous sections have mostly argued ERNIE to be better than ALBERT, with certain caveats. The question that remains is whether ERNIE's good performance can be attributed to inclusion of knowledge-incorporating learning goals in the pre-training stages. Deep learning models are know as black box models, and it is therefore hard to infer what information the models use to predict. Explaining the black box of ERNIE through a comparison with ALBERT in order to answer the current research question is partly flawed, due to their many differences. ALBERT has been scaled down from BERT, and while ALBERT has done well on benchmarks like

GLUE, this does not necessarily extend to separating hate speech and offensive language. Wang et al. (2020a) got the best performance on task B and C in OffensEval 2020, using ERNIE. They also used ALBERT XXLarge during validation, and even though ERNIE was better than ALBERT here, it was only by a small margin. The experiments of this thesis show a quite large difference between ALBERT and ERNIE. A reason for this might be that the smaller configurations of ALBERT do not perform as well as the larger models (Lan et al., 2020), which is in line with the experiments showing that ALBERT XLarge is significantly better than ALBERT Large in cross-dataset experiments.

When comparing results with previous work in section 8.1.6 and tables 8.11 and 8.12, BERT and ERNIE score very similarly. This indicates that the performance gap between ERNIE and ALBERT in the experiments is a result of ALBERT being worse than BERT, not ERNIE being better. Sun et al. (2020) showed that ERNIE performed better than BERT on several GLUE benchmarks, but has later been surpassed by ALBERT and BERT based ensembles.[1]. The small margins, low interpretability of deep models, instability of BERT based models, and lack of proper hypothesis tests with larger sample sizes between ERNIE and BERT leaves this research question unanswered.

While the experiments of this thesis fail to clearly show the effects of language integration, a possibility of improvement could come from further language modelling on Twitter hate speech data. Isaksen and Gambäck (2020) tried extended pre-training of BERT on in-domain data, but found that this performed worse than normal fine-tuning. ERNIE still might benefit from further in-domain pre-training, enabling the model to retain knowledge on in-domain topics such as i.e. politics and racism. ERNIE is already trained on data from Wikipedia, BookCorpus, Reddit, CommonCrawl[2] in English. It is also trained on news and other data in the Chinese corpus. Due to the type and large amount of data ERNIE has been trained on, one could expect that just a small amount of it is related to the domain of hate speech detection. The knowledge related to hate speech and offensive language would therefore also be small. For knowledge-integration to help improve performance, it might be necessary to continue pre-training with the framework proposed by Sun et al. (2020). This could be interesting to explore in future work.

[1]https://gluebenchmark.com/leaderboard
[2]http://commoncrawl.org/

9. Conclusion and Future Work

This chapter will present a conclusion to the thesis, how the thesis contributes to the field, and suggestions for future work. The conclusion will include central findings of experiments and insights from discussions and the literature review. Several interesting ways to explore the exciting field of hate speech detection were encountered during the work on this thesis, and these will be presented in the final section. Some further paths are divagations, while some are based on the need for further exploration and validation of results presented earlier.

9.1. Conclusion

The field of automatic hate speech detection on social media is continuously growing. Moderating incoming tweets manually using people is intractable, and automatic systems that can discover unwanted content is therefore important. Research is focused both on improving models and creating new datasets, the first being driven by a general need for better models in NLP for a series of tasks. Recent work have been focused on the successful use of transfer learning in the form of language models, especially through the use of large Transformer based architectures. The work done in this thesis focused on how reduction of parameters and inclusion of factual knowledge impact separation of hateful, offensive, and normal language.

A literature review of existing work was presented in chapter 4, where existing solutions in hate speech detection, generalizability, and transfer learning in NLP were surveyed. The review found that word embeddings like GloVe and word2vec are still popular as future extractors for both recurrent and classical machine learning methods, but large pre-trained language models often perform better. Word and character n-grams are both used in the literature, where character n-grams seem to perform better. This is probably due to the large number of ways words are spelled and misspelled online. A character n-gram approach is more immune to this and can navigate around similar looking words. WordPiece and SentencePiece, mostly used in transformer based language models, capture the best aspects of both approaches.

Kshirsagar et al. (2018) is an example of how word embeddings can still be used to perform on the same level as transformer based models on hate speech detection. The use of a knowledge integrating language model was first introduced in ERNIE, and has therefore seen very little use in hate speech detection. When being used in OffensEval 2020, it outperformed all other models on the hardest tasks and therefore shows great promise.

Because of the novelty of ERNIE 2.0 and transfer learning, few experiments have been conducted using knowledge integrating networks for hate speech classification. Experiments were therefore conducted to gain insights on ERNIE's ability to separate hateful, offensive, and normal language. Also memory saving techniques using transformers have been sparsely applied in hate speech detection, and five systems based on ALBERT and ERNIE were therefore implemented. The lack of any baseline transformer model like BERT or RoBERTa makes it difficult to clearly assess how much the results are due to knowledge integration or parameter reduction. ERNIE scored consistently better than ALBERT on macro averaged F1, but only a small or no improvement was observed compared with other state-of-the-art systems with the same number of available tweets. While ERNIE Base scored better than all ALBERT models on all datasets, the scores were only significantly different between ERNIE Base and ALBERT Large, and between the averaged results of the ERNIE and ALBERT models. The sample size of the significance tests were only three in order to not break the assumption of independence between experiments. Such a small sample size yields very low statistical power, and the probability of type II error is therefore large, meaning there is a large probability that ERNIE is still significantly better than all ALBERT models, but experiments with larger sample sizes from i.e. cross-validation is needed to answer this.

A model's usability is limited to how good the dataset is. Annotating coherent datasets is difficult, and keeping annotations synchronized between datasets has proven challenging. Nobata et al. (2016) argue that timely relevant data is important for performance, which showcases the difficulty of generalizing when a limited scope of data is included during training. Therefore, the work in this thesis also focused on the possibility of combining several datasets and the effect this had on generalization. This was done with experiments, and an extensive discussion was carried out in chapter 8 regarding the use of several datasets in conjunction. In order to have systems that are able to work well in practice, training models iteratively over time and combining datasets with different domains is important to handle the changing nature of online content. Most previous work on cross-dataset performance have been conducted on binary labels, and experiments were therefore conducted to gain insights on the compatibility of three datasets converted to a common, three class annotation scheme. While this was done with the aim of testing generalizability, it also raised questions about the compatibility of the different label schemes. A thorough analysis of how wrong assumptions in label mappings can impact results was therefore conducted. It showed that wrong assumptions on the relationship between insult target and hateful language would most probably punish the model. Results of the cross-domain experiments showed that ERNIE was able to capture structures of all three datasets on a level comparable with training on the datasets separately, while ALBERT lost more performance when datasets were combined. This suggests that the datasets are compatible, but ERNIE could also have used markers to identify the dataset of a tweet and leveraged this to apply different definitions of hate speech when predicting. Further research is needed to answer this. The same significance tests as for in-domain experiments were conducted, and ERNIE scored significantly better than ALBERT between several model configurations. This could probably be attributed to cross-domain experiments being more difficult which increase the performance gap between the models.

9.2. Contributions

This thesis contributes to the field of hate speech detection by introducing cross dataset experiments on separating hateful and offensive language. This has been not been addressed earlier because of possible incompatibilities between the labels, but this thesis intends to encourage further exploration of the actual incompatibilities, so that inconsistencies can be identified.

Zampieri et al. (2019b) argued that their hierarchical label structure can be used to match class definitions commonly used in offensive language detection. This thesis puts this into practice. Section 8.1.2 presents an analysis on how wrong assumptions when trying to match the hierarchical structure of OLID most probably will affect the macro F1 score. This is valuable to identify the usefulness of combining datasets. The thesis also contributes with a discussion of the relationship between definitions of hateful, abusive, and offensive language used in Zampieri et al. (2019b), Rosenthal et al. (2020), Davidson et al. (2017), and Founta et al. (2018), as well as a discussion on the quality of some datasets in chapter 3.

Even though the results of ERNIE were similar to previous work, the easily achieved good performance suggests that the model may have room for improvement. The identification of unstable training of ALBERT, especially on unbalanced data, might encourage the use of averaging results in future work, or even better, the use of cross-validation. This might lead to more stable and interpretable results.

9.3. Future Work

The experiments gave insights on how some of the experiments can be improved, and what further research is needed to remove sources of error. Several ideas also came up during the work on how to extend the research. This section will propose some further paths that might answer the questions that arose, and it will also propose improvements on the work done in this thesis, in order to find insights that this work might have missed.

9.3.1. Stripped Down Comparison

When discussing the results on cross-dataset performance and trying to deduce if the annotations were compatible or not, a possible loophole for the models became apparent. A model could "cheat" when predicting on a combined dataset by identifying what dataset the tweet came from. For the OLID dataset, this could be done by noticing the "@USER" tag often present in OLID. A suggestion for future work is to try to remove as much syntactical identification from the datasets as possible, making it harder for the models to discover what dataset a tweet belongs to. A model could still identify what dataset a tweet belongs to through identification of topic. As an example, a token in

OLID that would have a lot of predictive power is the word “liberals”. The word appears 1,445 times in the converted training set of OLID used in this thesis, 35 times in the full dataset from Founta et al. (2018), and 10 times in Davidson et al. (2017). Note however, that exploiting topics in order to predict better is not overfitting, and it could be regarded as a good thing that the model has knowledge about the topic. Nevertheless, it is valuable to have a sense of the degree to which this happens. In order to test how easily a model can identify a dataset, an experiment could be conducted with a combined dataset, where the labels represent the original dataset of a tweet. The metrics would then represent how separable the datasets are with the respective model.

9.3.2. Thorough Model Comparison

Even though doing experiments with ALBERT and ERNIE gave valuable insights, a more systematic approach exists for answering whether knowledge-integration in pre-training yields benefits. Firstly, comparing ERNIE to a more similar model is a better approach to identify the benefits of knowledge-inducing pre-training. While the work in this thesis compared with previous works using BERT, a direct comparison where one can be sure that all parameters are the same is probably beneficial. RoBERTa may be a better baseline model than BERT, since Liu et al. (2019b) discovered that BERT was sub-optimally pre-trained. This would reduce the chance of finding a difference between the models due to sub-optimal training, which would increase the probability of the effects coming from the knowledge-integration. Since the structures are completely similar, models could also be compared with the exact same hyperparameters and optimizers, instead of the strategy of the work in this thesis which was to use an extensive hyperparameter optimization for ALBERT and a very limited for ERNIE. The work in this thesis observed unstable training of ALBERT, which was also observed by Mosbach et al. (2020). Cross-validation is not always used, but the observations suggest that this is crucial to get a credible result. The lack of significant differences in the models used in this thesis might be due to the high variance in the results of ALBERT. If cross-validation is not used, at least random restarts as was done by Devlin et al. (2019) are suggested.

9.3.3. Further Knowledge-Integration by Pre-Training on Relevant Data

ERNIE is trained on Chinese encyclopedic, news, and dialogue data on the token level and on English Wikipedia, BookCorpus, and Reddit. This suggests that ERNIE has some relevant English knowledge, but there is a potential for gaining more relevant data through Twitter or English news. OLID has a lot of political data, and retaining relevant knowledge from i.e. recent American news articles may prove helpful for the model. Isaksen and Gambäck (2020) tried continued pre-training on other offensive datasets, but this performed worse than the original models. The use of ERNIE’s continual multi-task learning framework might yield better results by avoiding previously learned tasks to be forgotten.

9.3.4. Avoiding Catastrophic Forgetting

Merging and shuffling datasets yielded considerably better results than training with each dataset iteratively. This is not an issue when only using three datasets and the project utilizing them only spans a short time period, but it becomes problematic if a real live system is supposed to keep up to date with the vast amounts of data and expand while new data becomes available. It is therefore important to use strategies that do not result in useful parameter structures being overwritten in order to optimize for new data. Kirkpatrick et al. (2017) proposed Elastic Weight Consolidation (EWC), which mimics synaptic consolidation in the brain. The method penalises a model for moving important parameters further away from previously learned tasks. By using this, one could train on newer data iteratively and choose how much to weigh individual datasets.

9.3.5. Using Other Dataset Resources

Caselli et al. (2020) did a thorough review of the OLID dataset and expanded it with ‘explicit’ and ‘implicit’ annotations in the AbuseEval v1.0 dataset. This could be utilized, in conjunction with other techniques and strategies, to analyze how explicitness relates to model performance on cross-dataset experiments. The other datasets could also be analyzed in terms of explicitness by being tested with a model trained on AbuseEval v1.0, similarly to the way Swamy et al. (2019) categorized tweets into targeted offense or untargeted offense, and targeting groups, individuals, or other by training on the different levels of OLID hierarchy. There are also several other annotation schemes and datasets that can be explored. Price et al. (2020) published a dataset categorizing healthy and unhealthy content, as well as using binary labels for six attributes of unhealthy conversations. Resources like this could be utilized for further insight into what attributes are present across different hateful language categories.

Many multilingual datasets are readily available[1] and participation on multilingual tasks was high in OffensEval 2020, where five languages were included. Several models are now pre-trained on large multi-lingual corpora, and it could be interesting to expand the experiments of this thesis to the multilingual domain. Use of multilingual corpora increases the number of available datasets considerably and challenges the models further. The experiments could also include other English datasets.

[1]https://hatespeechdata.com/

Bibliography

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 759–760, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349147. doi: 10.1145/3041021.3054223.

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 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1023.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, March 2003. ISSN 1532-4435.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, Oct 2018. doi: 10.1016/j.neunet.2018.07.011.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *CoRR*, abs/1309.0238, 2013. URL `http://arxiv.org/abs/1309.0238`.

Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017. ISSN 0036-8075. doi: 10.1126/science.aal4230.

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`.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal Sentence Encoder. *CoRR*, abs/1803.11175, 2018. URL `http://arxiv.org/abs/1803.11175`.

N V Chawla, K W Bowyer, L O Hall, and W P Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun 2002. doi: 10.1613/jair.953.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390177. URL `https://doi.org/10.1145/1390156.1390177`.

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`.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Language*, 20 (3):273–297, September 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL `https://doi.org/10.1023/A:1022627411411`.

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P07-1033`.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal,* *Québec, Canada, May 15-18, 2017*, pages 512–515. AAAI Press, 2017. URL https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665.

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.

Facebook. Facebook reports fourth quarter and full year 2019 results. https://s21.q4cdn.com/399680738/files/doc_financials/2019/q4/FB-12.31.2019-Exhibit-99.1-r61_final.pdf, January 2020.

Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys*, 51(4), July 2018. ISSN 0360-0300. doi: 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. In *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM 2018, Stanford, California, USA, June 25-28, 2018*, pages 491–500. AAAI Press, 2018. URL https://aaai.org/ocs/index.php/ICWSM/ICWSM18/paper/view/17909.

Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, page 105–114, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362023. doi: 10.1145/3292522.3326028.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994. ISSN 0898-9788.

Björn Gambäck and Utpal Kumar Sikdar. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3013. URL https://www.aclweb.org/anthology/W17-3013.

Lei Gao and Ruihong Huang. Detecting online hate speech using context aware models. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 260–266, Varna, Bulgaria, September 2017. INCOMA Ltd. doi: 10.26615/978-954-452-049-6_036. URL https://doi.org/10.26615/978-954-452-049-6_036.

Lei Gao, Alexis Kuppersmith, and Ruihong Huang. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 774–782, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL `https://www.aclweb.org/anthology/I17-1078`.

Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. Detecting hate speech and offensive language on Twitter using machine learning: An n-gram and tfidf based approach. *arXiv preprint arXiv:1809.08651*, 2018.

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`.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*, 2015. URL `http://arxiv.org/abs/1503.03535`.

Xinjian Guo, Yilong Yin, Cailing Dong, Gongping Yang, and Guangtong Zhou. *On the class imbalance problem*, page 192–201. IEEE, Oct 2008. ISBN 978-0-7695-3304-9. doi: 10.1109/ICNC.2008.871.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. *FewRel: A Large-Scale Supervised Few-Shot Relation Classification Dataset with State-of-the-Art Evaluation*, page 4803–4809. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1514.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 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`.

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, 2019.

Vebjørn Isaksen and Björn Gambäck. Using transfer-based language models to detect hateful and offensive language online. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 16–27, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.alw-1.3. URL `https://www.aclweb.org/anthology/2020.alw-1.3`.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1162. URL `https://www.aclweb.org/anthology/P15-1162`.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-2068`.

Mladen Karan and Jan Šnajder. Cross-domain detection of abusive language online. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 132–137, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5117. URL `https://www.aclweb.org/anthology/W18-5117`.

Robert A Khan. Why do Europeans ban hate speech? A debate between Karl Loewenstein and Robert Post. *Hofstra Law Review*, 41(3):2, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/content/114/13/3521`.

Jason Koebler and Joseph Cox. Content moderator sues facebook, says job gave her ptsd. *Vice*, Sep 2018. URL `https://www.vice.com/en/article/zm5mw5/facebook-content-moderation-lawsuit-ptsd`.

Rohan Kshirsagar, Tyrus Cukuvac, Kathy McKeown, and Susan McGregor. Predictive embeddings for hate speech detection on Twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 26–32, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5104. URL `https://www.aclweb.org/anthology/W18-5104`.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL `https://www.aclweb.org/anthology/P18-1007`.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL `https://www.aclweb.org/anthology/D18-2012`.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=H1eA7AEtvS`.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-540-46805-9. doi: 10.1007/3-540-46805-6_19. URL `https://doi.org/10.1007/3-540-46805-6_19`.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. Comparative studies of detecting abusive language on Twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 101–106, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5113. URL `https://www.aclweb.org/anthology/W18-5113`.

Ping Liu, Wen Li, and Liang Zou. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/S19-2011. URL `https://www.aclweb.org/anthology/S19-2011`.

Ruibo Liu, Guangxuan Xu, and Soroush Vosoughi. Enhanced offensive language detection through data augmentation. *CoRR*, abs/2012.02954, 2020. URL `https://arxiv.org/abs/2012.02954`.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b. URL `http://arxiv.org/abs/1907.11692`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Shervin Malmasi and Marcos Zampieri. Detecting hate speech in social media. In *Proceedings of the International Conference Recent Advances in Natural Language*

Processing, RANLP 2017, pages 467–472, Varna, Bulgaria, September 2017. INCOMA Ltd. doi: 10.26615/978-954-452-049-6_062. URL https://doi.org/10.26615/978-954-452-049-6_062.

Johannes Skjeggestad Meyer and Björn Gambäck. A platform agnostic dual-strand hate speech detector. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 146–156, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3516. URL https://www.aclweb.org/anthology/W19-3516.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukas Burget, and Jan Honza Cernocky. Rnnlm - recurrent neural network language modeling toolkit. In *IEEE Automatic Speech Recognition and Understanding Workshop*, pages 196–201, December 2011. URL https://www.microsoft.com/en-us/research/publication/rnnlm-recurrent-neural-network-language-modeling-toolkit/. Presented as demo session in ASRU 2011.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013b. URL http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis, and Ekaterina Shutova. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1093.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884, 2020. URL https://arxiv.org/abs/2006.04884.

Karsten Müller and Carlo Schwarz. From hashtag to hate crime: Twitter and anti-minority sentiment. *SSRN*, October 2019. doi: 10.2139/ssrn.3149103. URL https://ssrn.com/abstract=3149103.

Isar Nejadgholi and Svetlana Kiritchenko. On cross-dataset generalization in automatic detection of online abuse. *CoRR*, abs/2010.07414, 2020. URL https://arxiv.org/abs/2010.07414.

Casey Newton. The trauma floor: The secret lives of facebook moderators in america. *The Verge*, Feb 2019. URL https://www.theverge.com/2019/2/25/18229714/cognizant-facebook-content-moderator-interviews-trauma-working-conditions-arizona.

Alex Nikolov and Victor Radivchev. Nikolov-radivchev at SemEval-2019 task 6: Offensive tweet classification with BERT and ensembles. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 691–695, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2123. URL https://www.aclweb.org/anthology/S19-2123.

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, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883062. URL https://doi.org/10.1145/2872427.2883062.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35, Vancouver, BC, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3004. URL https://www.aclweb.org/anthology/W17-3004.

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, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Ilan Price, Jordan Gifford-Moore, Jory Flemming, Saul Musker, Maayan Roichman, Guillaume Sylvain, Nithum Thain, Lucas Dixon, and Jeffrey Sorensen. Six attributes of unhealthy conversation. *CoRR*, abs/2010.07410, 2020. URL https://arxiv.org/abs/2010.07410.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Sarah T. Roberts. *"I Call Myself a Sin-Eater"*, page 158. Yale University Press, 2019. ISBN 9780300235883. URL http://www.jstor.org/stable/j.ctvhrcz0v.7.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi: 10.1037/h0042519.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. A large-scale semi-supervised dataset for offensive language identification. *CoRR*, abs/2004.14454, 2020. URL `https://arxiv.org/abs/2004.14454`.

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`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL `https://www.aclweb.org/anthology/P16-1162`.

Mention Solutions. Mention's Twitter Report, 2018. URL `https://info.mention.com/twitter-report#rec257652194`.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223, 2019. URL `http://arxiv.org/abs/1904.09223`.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975, Apr 2020. doi: 10.1609/aaai.v34i05.6428.

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, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1088. URL `https://www.aclweb.org/anthology/K19-1088`.

Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, October 2003. ISSN 1046-8188. doi: 10.1145/944012.944013. URL `https://doi.org/10.1145/944012.944013`.

Twitter. The 2014 #YearOnTwitter, 2014. URL `https://blog.twitter.com/en_us/a/2014/the-2014-yearontwitter.html`.

Twitter. Productionizing ML with workflows at Twitter, 2018. URL https://blog.twitter.com/engineering/en_us/topics/insights/2018/ml-workflows.html.

Twitter. Twitter rules enforcement, 2019. URL https://transparency.twitter.com/en/twitter-rules-enforcement.html.

Twitter. Developer agreement and policy, 2020a. URL https://developer.twitter.com/en/developer-terms/agreement-and-policy.

Twitter. Hateful conduct policy, 2020b. URL https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy.

Twitter. Fourth quarter selected company metrics and financials. https://s22.q4cdn.com/826641620/files/doc_financials/2019/q4/Q4-2019-Selected-Financials-and-Metrics.pdf, February 2020c.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Subhashini Venugopalan, Lisa Anne Hendricks, Raymond Mooney, and Kate Saenko. Improving LSTM-based video description with linguistic knowledge mined from text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1204. URL https://www.aclweb.org/anthology/D16-1204.

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 2020a. International Committee for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.semeval-1.189.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. StructBERT: Incorporating language structures into pre-training for deep language understanding. In *8th International Conference on Learning Representations,* *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=BJgQ4lSFPH.

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, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-5618. URL https://www.aclweb.org/anthology/W16-5618.

Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-2013. URL https://www.aclweb.org/anthology/N16-2013.

Gregor Wiedemann, Seid Muhie Yimam, 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 Fourteenth Workshop on Semantic Evaluation*, pages 1638–1644, Barcelona (online), December 2020. International Committee for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.semeval-1.213.

Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. Inducing a lexicon of abusive words – a feature-based approach. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1046–1056, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1095. URL https://www.aclweb.org/anthology/N18-1095.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4148–4158. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7003-the-marginal-value-of-adaptive-gradient-methods-in-machine-learning.pdf.

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, Lukasz 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.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1391–1399, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052591. URL `https://doi.org/10.1145/3038912.3052591`.

Yang You, Jing Li, Sashank J. 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. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=Syx4wnEtvH`.

Sayyed M. Zahiri and Ali Ahmadvand. CRAB: class representation attentive BERT for hate speech identification in social media. *CoRR*, abs/2010.13028, 2020. URL `https://arxiv.org/abs/2010.13028`.

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, June 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, June 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 Çagri Çöltekin. Semeval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In Aurélie Herbelot, Xiaodan Zhu, Alexis Palmer, Nathan Schneider, Jonathan May, and Ekaterina Shutova, editors, *Proceedings of the Fourteenth Workshop on Semantic Evaluation, SemEval@COLING 2020, Barcelona (online), December 12-13, 2020*, pages 1425–1447. International Committee for Computational Linguistics, 2020. URL `https://www.aclweb.org/anthology/2020.semeval-1.188/`.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J. Reddi, Sanjiv Kumar, and Suvrit Sra. Why ADAM beats SGD for attention models. *CoRR*, abs/1912.03194, 2019a. URL `http://arxiv.org/abs/1912.03194`.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1004. URL `https://www.aclweb.org/anthology/D17-1004`.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. *ERNIE: Enhanced Language Representation with Informative Entities*, page 1441–1451. Association for Computational Linguistics, 2019b. doi: 10.18653/v1/P19-1139.

Ziqi Zhang, David Robinson, and Jonathan Tepper. Detecting hate speech on Twitter using a Convolution-GRU based deep neural network. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web*, pages 745–760, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93417-4.

A. Appendices

A.1. Macro Average F1 Given as Conditional Probabilities

This section presents how the F1 macro average score was calculated in section 8.1.2. The scenario consists of a model being trained on an original dataset **D**, with “Hateful” and “Offensive” as original labels. The model is then tested on a dataset with original labels found in the label-space of OLID, and then mapped to the label space of **D**. *Grp* and *Ind* means that a tweet is targeted towards a group or an individual. *Hate* and *Off* means that the tweet is hateful or offensive in the original dataset **D**. It does not mean that the tweet was labeled “Hateful” or “Offensive” in the dataset mapped from the OLID hierarchy of section 7.1. *Correct* and *Wrong* means that the model trained on dataset **D** was correct or wrong in its prediction on the mapped dataset. The macro averaged F1 score when testing on the mapped dataset is given as:

$$MacroF1 = \frac{PrecHate \times RecHate}{PrecHate + RecHate} + \frac{PrecOff \times RecOff}{PrecOff + RecOff}$$

Here the precision and recall for hateful and offensive are given as:

$$PrecHate = \frac{TP}{TP + FP}$$

$$RecHate = \frac{TP}{TP + FN}$$

$$PrecOff = \frac{TN}{TN + FN}$$

$$RecOff = \frac{TN}{TN + FP}$$

TP, FP, TN, and FN can be given as the combination between a tweet being targeted against a group or individual, really being hateful or offensive, and if the model was correct in its prediction. The probabilities of TP, FP, TN, and FN are thus:

$$P(TP) = P(Grp \cap Hate \cap Correct) + P(Grp \cap Off \cap Wrong)$$

$$P(FP) = P(Ind \cap Hate \cap Correct) + P(Ind \cap Off \cap Wrong)$$

$$P(TN) = P(Ind \cap Off \cap Correct) + P(Ind \cap Hate \cap Wrong)$$

$$P(FN) = P(Grp \cap Off \cap Correct) + P(Grp \cap Hate \cap Wrong)$$

All these intersections can be given as the products of conditional probabilities:

$$P(Grp \cap Hate \cap Correct) = P(Grp)P(Hate|Grp)P(Correct|Hate \cap Grp)$$

By assuming that that the model's ability to be correct is independent of the target of the tweet, this reduces to:

$$P(Grp \cap Hate \cap Correct) = P(Grp)P(Hate|Grp)P(Correct|Hate)$$

By the law of total probability:

$$\begin{aligned} P(Ind) &= 1 - P(Grp) \\ P(Off|Grp) &= 1 - P(Hate|Grp) \\ P(Off|Ind) &= 1 - P(Hate|Ind) \\ P(Wrong|Hate) &= 1 - P(Correct|Hate) \\ P(Wrong|Off) &= 1 - P(Correct|Off) \end{aligned}$$

These steps result in the macro averaged F1 score being given by 5 probabilities:

$$\begin{aligned} &P(Grp) \\ &P(Hate|Grp) \\ &P(Hate|Ind) \\ &P(Correct|Hate) \\ &P(Correct|Off) \end{aligned}$$

By calculating metrics from these probabilities, an intuition of how the probabilities affect the metrics can be obtained. The widget used for exploring these probabilities can be seen in figure A.1. When evaluating these probabilities, it can be noted that:

- $P(Grp)$ can be approximated through analysis of development data.
- $P(Hate|Grp)$ and $P(Hate|Grp)$ are the assumptions made.
- $P(Correct|Hate)$ and $P(Correct|Off)$ can be approximated from previous experiments.

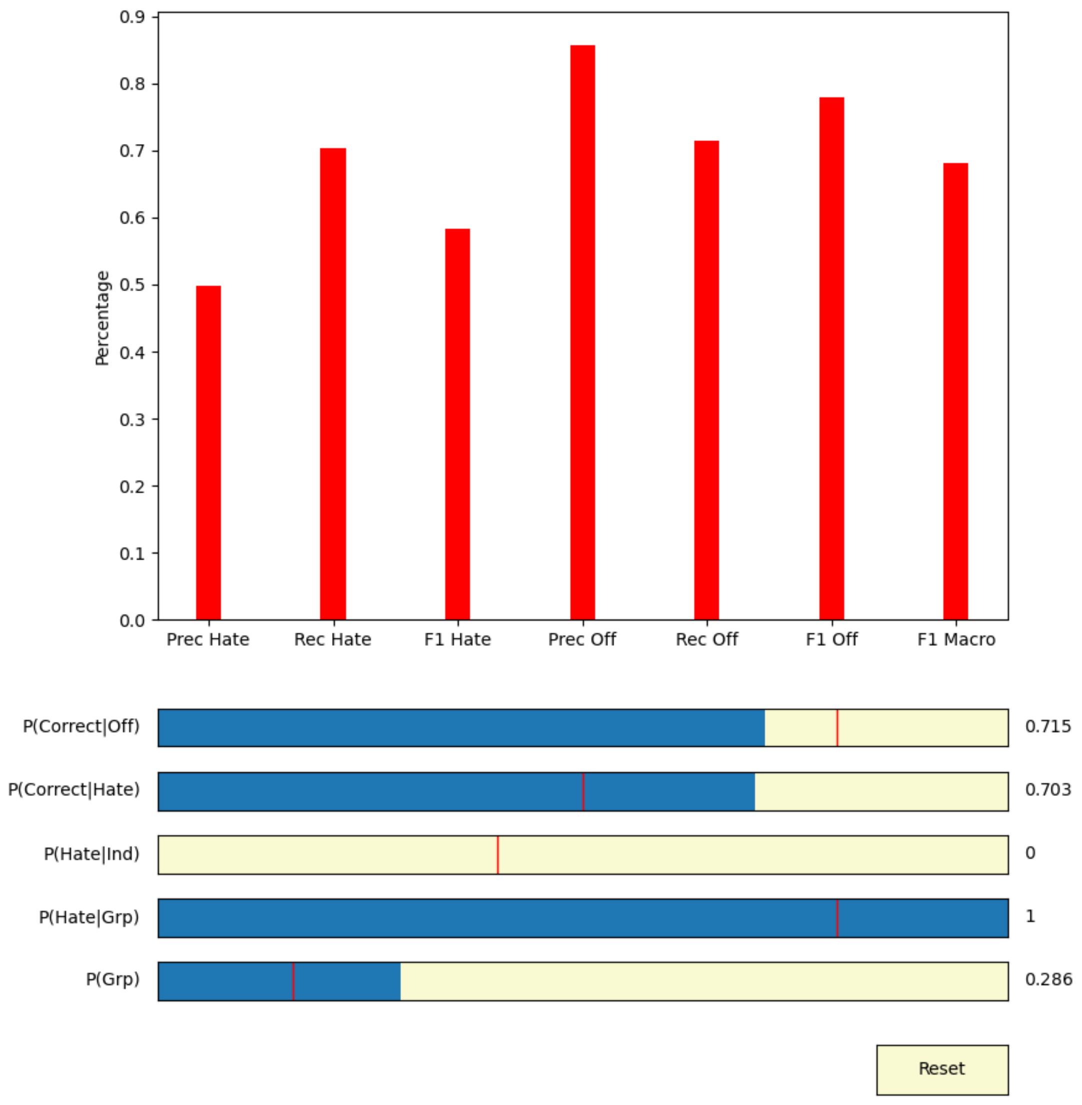

Figure A.1.: The widget used for exploring the effects of violating the assumptions used when mapping labels.

A.2. Results on Cross-Dataset Evaluation

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.189	**0.199**	0.141	0.149	0.178
	0.318	0.384	**0.403**	0.240	0.229
	0.237	**0.262**	0.209	0.184	0.200
Off	0.928	0.923	**0.957**	0.931	0.922
	0.799	0.791	0.694	0.733	0.753
	0.859	0.852	0.805	0.820	0.829
Neither	**0.578**	0.570	0.467	0.454	0.456
	0.833	0.789	0.801	0.823	0.811
	0.682	0.662	0.590	0.585	0.584
Weighted Average	0.832	0.827	**0.834**	0.811	0.807
	0.780	0.770	0.697	0.722	0.736
	0.797	0.790	0.738	0.748	0.756
Macro Average	**0.565**	0.564	0.522	0.511	0.519
	0.650	**0.655**	0.633	0.599	0.598
	0.593	0.592	0.535	0.530	0.538
Accuracy	**0.780**	0.770	0.697	0.722	0.736

Table A.1.: Results on $\mathbf{D}_{\text{test}}$ using $\mathbf{F*}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.209	**0.222**	0.184	0.186	0.154
	0.399	0.446	**0.624**	0.465	0.543
	0.274	**0.297**	0.284	0.265	0.240
Off	0.945	0.938	**0.969**	0.961	0.961
	0.817	0.807	0.717	0.743	0.672
	0.876	0.868	0.824	0.838	0.791
Neither	**0.665**	0.633	0.576	0.556	0.532
	0.901	0.849	0.852	0.890	0.867
	0.765	0.725	0.687	0.684	0.659
Weighted Average	0.860	0.850	**0.863**	0.854	0.848
	0.809	0.795	0.734	0.753	0.698
	0.827	0.814	0.773	0.783	0.741
Macro Average	**0.606**	0.598	0.576	0.568	0.549
	0.706	0.701	**0.731**	0.699	0.694
	0.639	0.630	0.599	0.596	0.564
Accuracy	**0.809**	0.795	0.734	0.753	0.698

Table A.2.: Results on $\mathbf{D}_{\text{test}}$ using $\mathbf{DF*}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.425**	0.199	0.309	0.416	0.371
	0.372	0.384	**0.531**	0.353	0.380
	0.397	0.262	0.390	0.382	0.375
Off	0.948	0.923	**0.963**	0.951	0.950
	0.955	0.791	0.902	0.947	0.939
	0.952	0.852	0.931	0.949	0.945
Neither	**0.899**	0.570	0.859	0.864	0.872
	0.905	0.789	**0.922**	0.918	0.915
	0.902	0.662	0.889	0.890	0.893
Weighted Average	**0.913**	0.827	0.911	0.908	0.907
	0.916	0.770	0.886	0.912	0.906
	0.915	0.790	0.896	0.910	0.907
Macro Average	**0.757**	0.564	0.710	0.743	0.731
	0.744	0.655	**0.785**	0.739	0.745
	0.750	0.592	0.737	0.740	0.738
Accuracy	**0.916**	0.770	0.886	0.912	0.906

Table A.3.: Results on $\mathbf{D}_{\text{test}}$ using $\mathbf{F*D}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.084	0.085	0.076	0.078	**0.086**
	0.260	0.225	0.182	0.264	**0.287**
	0.127	0.123	0.107	0.120	**0.132**
Off	0.891	0.887	**0.906**	0.901	0.897
	0.701	**0.755**	0.675	0.630	0.638
	0.785	**0.816**	0.773	0.742	0.746
Neither	0.563	**0.613**	0.450	0.475	0.474
	0.761	0.723	**0.796**	0.793	0.773
	0.647	**0.663**	0.575	0.594	0.588
Weighted Average	0.795	**0.800**	0.787	0.788	0.785
	0.688	**0.722**	0.669	0.638	0.642
	0.728	**0.755**	0.706	0.685	0.688
Macro Average	0.513	**0.528**	0.477	0.484	0.486
	0.574	0.568	0.551	0.562	0.566
	0.520	**0.534**	0.485	0.485	0.489
Accuracy	0.688	**0.722**	0.669	0.638	0.642

Table A.4.: Results on $\mathbf{D}_{\text{test}}$ using $\mathbf{O}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.417**	0.415	0.195	0.225	0.241
	0.388	0.426	**0.640**	0.535	0.496
	0.402	**0.421**	0.298	0.317	0.324
Off	0.943	0.946	**0.971**	0.961	0.948
	0.956	0.952	0.814	0.859	0.872
	0.950	0.949	0.886	0.907	0.909
Neither	**0.921**	0.910	0.809	0.834	0.824
	0.880	0.876	0.846	**0.891**	0.861
	0.900	0.892	0.827	0.862	0.842
Weighted Average	0.912	**0.913**	0.904	0.902	0.891
	0.914	0.912	0.810	0.847	0.851
	0.913	0.912	0.846	0.869	0.867
Macro Average	**0.760**	0.757	0.658	0.673	0.671
	0.742	0.751	**0.767**	0.762	0.743
	0.751	**0.754**	0.670	0.695	0.692
Accuracy	**0.914**	0.912	0.810	0.847	0.851

Table A.5.: Results on $\mathbf{D}_{\text{test}}$ using $\mathbf{C}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.396**	0.393	0.274	0.303	0.225
	0.063	0.091	0.144	0.073	**0.179**
	0.108	0.148	0.189	0.117	**0.199**
Off	0.824	0.820	0.774	0.785	**0.829**
	0.885	0.879	0.857	0.881	0.801
	0.853	0.849	0.813	0.830	0.815
Neither	**0.937**	0.936	0.933	0.934	0.936
	0.970	0.966	0.932	0.947	0.961
	0.953	0.951	0.933	0.940	0.949
Weighted Average	**0.882**	0.881	0.862	0.866	0.874
	0.904	0.900	0.874	0.887	0.883
	0.886	0.886	0.866	0.872	0.878
Macro Average	**0.719**	0.717	0.660	0.674	0.663
	0.639	0.645	0.644	0.634	**0.647**
	0.638	0.649	0.645	0.629	**0.654**
Accuracy	**0.904**	0.900	0.874	0.887	0.883

Table A.6.: Results on $\mathbf{F*}_{\text{test}}$ using $\mathbf{D}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.552**	0.505	0.340	0.333	0.323
	0.340	0.352	**0.577**	0.441	0.464
	0.421	0.415	**0.428**	0.379	0.381
Off	0.868	0.867	**0.897**	0.881	0.887
	0.917	0.916	0.856	0.871	0.856
	0.892	0.891	0.876	0.876	0.871
Neither	0.960	0.961	**0.969**	0.960	0.963
	0.969	0.964	0.935	0.941	0.944
	0.964	0.962	0.951	0.951	0.953
Weighted Average	0.917	0.915	**0.919**	0.909	0.912
	0.924	0.921	0.898	0.899	0.899
	0.919	0.917	0.907	0.904	0.905
Macro Average	**0.793**	0.777	0.735	0.725	0.725
	0.742	0.744	**0.789**	0.751	0.755
	0.759	0.756	0.752	0.735	0.735
Accuracy	**0.924**	0.921	0.898	0.899	0.899

Table A.7.: Results on $\mathbf{F*}_{\text{test}}$ using $\mathbf{DF*}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.400	**0.451**	0.330	0.287	0.303
	0.096	**0.372**	0.306	0.157	0.119
	0.155	**0.408**	0.317	0.203	0.171
Off	0.818	0.858	**0.861**	0.814	0.812
	0.918	0.910	0.815	0.852	0.883
	0.865	**0.883**	0.837	0.833	0.846
Neither	0.953	**0.962**	0.943	0.942	0.945
	0.967	0.955	0.965	0.958	0.959
	0.960	0.959	0.954	0.949	0.952
Weighted Average	0.893	**0.911**	0.892	0.878	0.880
	0.910	**0.914**	0.896	0.892	0.898
	0.896	**0.912**	0.894	0.884	0.887
Macro Average	0.724	**0.757**	0.711	0.681	0.686
	0.660	**0.746**	0.695	0.656	0.653
	0.660	**0.750**	0.703	0.662	0.656
Accuracy	0.910	**0.914**	0.896	0.892	0.898

Table A.8.: Results on $\mathbf{F*}_{\text{test}}$ using $\mathbf{F*D}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.273**	0.242	0.224	0.241	0.271
	0.205	0.182	0.187	**0.258**	0.253
	0.234	0.208	0.204	0.249	**0.262**
Off	0.749	0.732	**0.762**	0.750	0.741
	0.844	0.825	0.826	0.819	**0.849**
	0.794	0.776	0.792	0.783	0.791
Neither	0.958	0.954	0.953	0.964	**0.968**
	0.935	0.931	**0.938**	0.930	0.926
	0.946	0.942	0.946	**0.947**	0.946
Weighted Average	0.873	0.865	0.870	0.876	**0.878**
	0.876	0.867	0.873	0.869	0.873
	0.874	0.865	0.871	0.872	**0.875**
Macro Average	**0.660**	0.643	0.646	0.652	0.660
	0.661	0.646	0.650	0.669	**0.676**
	0.658	0.642	0.647	0.660	**0.666**
Accuracy	**0.876**	0.867	0.873	0.869	0.873

Table A.9.: Results on $\mathbf{F*}_{\text{test}}$ using $\mathbf{O}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.546**	0.435	0.297	0.357	0.347
	0.363	0.302	**0.582**	0.445	0.362
	0.436	0.356	0.393	0.396	0.354
Off	0.811	0.805	**0.892**	0.865	0.819
	0.943	0.930	0.838	0.879	0.904
	0.872	0.863	0.864	0.872	0.859
Neither	**0.982**	0.979	0.969	0.964	0.969
	0.936	0.933	0.921	**0.941**	0.932
	0.959	0.956	0.945	0.952	0.950
Weighted Average	0.915	0.906	**0.916**	0.909	0.901
	0.913	0.905	0.884	0.901	0.896
	0.912	0.903	0.897	0.905	0.898
Macro Average	**0.780**	0.740	0.719	0.728	0.711
	0.747	0.722	**0.781**	0.755	0.733
	0.756	0.725	0.734	0.740	0.721
Accuracy	**0.913**	0.905	0.884	0.901	0.896

Table A.10.: Results on $\mathbf{F^*}_{\text{test}}$ using $\mathbf{C}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.194**	0.188	0.134	0.171	0.175
	0.101	0.084	0.140	0.112	**0.196**
	0.132	0.116	0.137	0.135	**0.185**
Off	0.831	0.823	0.717	0.765	**0.832**
	0.815	**0.822**	0.811	0.819	0.765
	0.823	0.822	0.761	0.791	0.797
Neither	0.936	0.936	**0.945**	0.938	0.931
	0.973	0.969	0.890	0.930	0.956
	0.954	0.952	0.917	0.934	0.944
Weighted Average	**0.874**	0.872	0.845	0.855	0.870
	0.890	0.889	0.835	0.863	0.869
	0.881	0.879	0.839	0.859	0.869
Macro Average	**0.654**	0.649	0.599	0.625	0.646
	0.629	0.625	0.614	0.620	**0.639**
	0.636	0.630	0.605	0.620	**0.642**
Accuracy	**0.890**	0.889	0.835	0.863	0.869

Table A.11.: Results on $\mathbf{S}_{\text{test}}$ using $\mathbf{D}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.318**	0.280	0.201	0.230	0.223
	0.436	0.374	**0.559**	0.425	0.536
	0.368	0.321	0.296	0.298	0.315
Off	0.847	0.847	**0.890**	0.838	0.871
	0.851	0.850	0.783	0.806	0.785
	0.849	0.849	0.833	0.821	0.826
Neither	0.972	0.973	**0.984**	0.968	0.971
	0.948	**0.951**	0.922	0.931	0.925
	0.960	**0.962**	0.952	0.949	0.948
Weighted Average	0.909	0.908	**0.924**	0.899	0.911
	0.898	0.897	0.867	0.873	0.868
	0.903	0.902	0.890	0.885	0.886
Macro Average	**0.713**	0.700	0.692	0.678	0.689
	0.745	0.725	**0.755**	0.720	0.749
	0.726	0.710	0.694	0.689	0.696
Accuracy	**0.898**	0.897	0.867	0.873	0.868

Table A.12.: Results on $\mathbf{S}_{\text{test}}$ using $\mathbf{DF*}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.277	0.254	0.217	0.242	**0.305**
	0.374	0.408	**0.492**	0.318	0.285
	0.318	0.313	0.301	0.275	0.295
Off	0.831	0.829	**0.838**	0.799	0.783
	0.846	0.835	0.803	0.788	0.844
	0.838	0.832	0.820	0.794	0.812
Neither	0.971	0.974	**0.979**	0.943	0.955
	0.942	0.933	0.917	0.929	0.928
	0.956	0.953	0.947	0.936	0.941
Weighted Average	0.901	0.902	**0.906**	0.872	0.878
	0.890	0.883	0.866	0.863	0.876
	0.895	0.891	0.883	0.867	0.877
Macro Average	**0.693**	0.686	0.678	0.661	0.681
	0.721	0.725	**0.737**	0.679	0.686
	0.704	0.699	0.689	0.668	0.683
Accuracy	**0.890**	0.883	0.866	0.863	0.876

Table A.13.: Results on $\mathbf{S}_{\text{test}}$ using $\mathbf{F*}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	0.229	**0.254**	0.242	0.234	0.144
	0.134	**0.408**	0.341	0.190	0.089
	0.169	**0.313**	0.283	0.210	0.110
Off	0.823	0.829	**0.847**	0.824	0.813
	0.840	0.835	0.766	0.802	0.837
	0.831	**0.832**	0.804	0.813	0.825
Neither	0.949	**0.974**	0.944	0.934	0.939
	0.965	0.933	0.957	0.956	0.950
	0.957	0.953	0.951	0.945	0.944
Weighted Average	0.882	**0.902**	0.886	0.873	0.869
	0.894	0.883	0.876	0.879	0.881
	0.887	**0.891**	0.880	0.876	0.874
Macro Average	0.667	**0.686**	0.678	0.664	0.632
	0.646	**0.725**	0.688	0.649	0.625
	0.652	**0.699**	0.679	0.656	0.626
Accuracy	**0.894**	0.883	0.876	0.879	0.881

Table A.14.: Results on $\mathbf{S}_{\text{test}}$ using $\mathbf{F^*D}_{\text{train}}$ as training set.

	Ernie		Albert		
	Base	Large	Base	Large	Xlarge
Hate	**0.515**	0.472	0.209	0.240	0.263
	0.293	0.336	**0.598**	0.486	0.374
	0.373	**0.392**	0.310	0.322	0.309
Off	0.851	0.852	**0.868**	0.850	0.792
	0.915	0.905	0.762	0.821	0.864
	0.882	0.878	0.811	0.835	0.827
Neither	0.959	0.959	**0.984**	0.979	0.982
	0.965	0.959	0.918	0.929	0.917
	0.962	0.959	0.950	0.953	0.949
Weighted Average	0.911	0.909	**0.917**	0.910	0.897
	0.919	0.914	0.860	0.879	0.879
	0.913	0.911	0.883	0.893	0.886
Macro Average	**0.775**	0.761	0.687	0.690	0.679
	0.724	0.733	**0.759**	0.745	0.719
	0.739	**0.743**	0.690	0.703	0.695
Accuracy	**0.919**	0.914	0.860	0.879	0.879

Table A.15.: Results on $\mathbf{S}_{\text{test}}$ using $\mathbf{C}_{\text{train}}$ as training set.

Peter Cook Bulukin

Classifying Hateful and Offensive Language Across Datasets and Domains

NTNU
Norwegian University of
Science and Technology