Spring 6-1-2021

Fine-Grained Detection of Hate Speech Using BERToxic

Yakoob Khan
Dartmouth College, yakoob.khan.21@dartmouth.edu

FINE-GRAINED DETECTION OF HATE SPEECH USING BERTOXIC

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Computer Science

by

Yakoob Khan

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 2021

Advised By

Professor Soroush Vosoughi

Department of Computer Science

Preface

A version of the work presented in this thesis has been published in the Proceedings of the 15$^{\text{th}}$ International Workshop on Semantic Evaluation (SemEval) co-located at the Joint Conference of the 59$^{\text{th}}$ Annual Meeting of the Association of Computational Linguists and the 11$^{\text{th}}$ International Joint Conference on Natural Language Processing (ACL – IJCNLP 2021). I am the first author of the workshop paper [12] in collaboration with Weicheng Ma and Soroush Vosoughi. Given the nature of this work, we caution readers that the examples included in this thesis contain explicit language to illustrate the severity and challenges of hate speech detection.

Abstract

This thesis describes our approach towards the fine-grained detection of hate speech using deep learning. We leverage the transformer encoder architecture to propose **BERToxic**, a system that fine-tunes a pre-trained BERT model to locate toxic text spans in a given text and utilizes additional post-processing steps to refine the prediction boundaries. The post-processing steps involve (1) labeling character offsets between consecutive toxic tokens as toxic and (2) assigning a toxic label to words that have at least one token labeled as toxic. Through experiments, we show that these two post-processing steps improve the performance of our model by 4.16% on the test set. We further examined the effect of ensemble models for hate speech detection. The ensemble neural architectures we studied include late fusion where predictions from token and sequence classification models are aggregated in the prediction phase and multi-task learning where the two aforementioned models are trained jointly. Finally, given the scarcity and costs of obtaining labeled data, we explored data augmentation strategies such as appending hate speech-related external datasets and token modification techniques to generate synthetic training examples. Our system significantly outperformed the baseline models and achieved an F1-score of 0.683, placing our model in the 17^{th} place out of 91 teams in a hate speech detection competition. Our code is made available at `https://github.com/Yakoob-Khan/Toxic-Spans-Detection`

Acknowledgments

This thesis is the culmination of my four years of undergraduate study at Dartmouth College and I owe a debt of gratitude to many people who helped me along the way.

First and foremost, I would like to express my deepest appreciation to my advisor Professor Soroush Vosoughi. I first met him during my junior year of college when I took his Machine Learning course. I was immediately impressed by his easygoing nature and sought the opportunity to enroll in every course he taught. Despite his busy schedule, he was willing to supervise my senior thesis and introduced me to the world of research. I thank him for pointing me towards the problem that this thesis addresses, which resulted in my first peer-reviewed publication. I am grateful for his mentorship and will miss our weekly Friday afternoon meetings.

Another person I am deeply indebted towards is Weicheng Ma, a graduate student pursuing his Ph.D. under Professor Vosoughi. He offered valuable guidance and was instrumental in the success of this project. Thank you for providing me with coding pointers to natural language processing libraries and responding to my emails literally within minutes with helpful suggestions.

Special thanks to Professor Prasad Jayanti and Tim Pierson for serving on my thesis committee.

I am also extremely grateful to numerous professors who have taught me at Dartmouth. I am lucky to have had many great teachers, but I can say without a doubt that Professor Prasad Jayanti was the best teacher I ever had. He ignited my passion

for Computer Science and encouraged me to keep at it when I questioned my ability in the subject. Thank you for inspiring my love of learning with your infinite wisdom, enthusiasm and humor. I am indebted to Professor Deeparnab Chakrabarty for his excellent course that exposed me to the beauty of algorithms. His raw intellect serves as a humbling reminder of my limited abilities. I am grateful to have known Professor Thomas H. Cormen before he retired from Dartmouth. He gave me chance to work with him as a Teaching Assistant for his Introduction to Computer Science course and advised me during his time as the undergraduate program director. Many thanks to Tim Tregubov for being a supportive mentor throughout my time in college. He taught me Full Stack Web Development and offered me the opportunity to work as a developer in the DALI Lab building web and mobile applications. Thank you for indulging my interest to learn beyond the curriculum and supervising my independent study courses despite your hectic schedule. Finally, I thank Professor Lorenzo Torresani for his superb course on Deep Learning, a subject whose ideas I extensively applied in this work.

Besides my professors, I wish to thank the Stamps Scholars Program at Dartmouth and the Strive Foundation for the financial support that culminated in this project. I am indebted to the Department of Computer Science for providing a supportive learning environment and welcoming students with no prior background.

Last, but certainly not least, I want to thank all my friends that have made my time at Dartmouth so memorable. Special thanks to my freshman year roommate Rik Abels for listening to me rant about this thesis and being my workout buddy. Many thanks to Aadil Islam for being a wonderful friend and giving me a place to stay in Boston with his family when I needed a change in scenery. Finally, I thank my parents for their eternal love and encouragement in supporting my dreams despite living halfway across the world.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Section 1.1

Motivation

The promotion of respectful discourse has always been a core tenet of civilized societies. Unfortunately, the FBI reports that hate incidents are on the rise, with 57.6% of physical incidents motivated by race and ethnicity [9]. This problem is worsened in the online space, where the cloak of anonymity enables malicious actors to surreptitiously post toxic comments. The wide adoption of social media platforms further amplifies the spread of offensive content, such as the recent rise of anti-Asian rhetoric linked to the COVID-19 pandemic [1]. The United Nations Secretary-General Antonio Guterres eloquently remarks that "hate speech is in itself an attack on tolerance, inclusion, diversity and the very essence of our human rights norms and principles." He continues and mentions that "it undermines social cohesion, erodes shared values, and can lay the foundation for violence, setting back the cause of peace, stability, sustainable development and the fulfillment of human rights for all."[20] Given the negative consequences of hate speech, online platforms have attempted to combat this problem by building content moderation systems that enable users to flag offensive

content for review by human moderators. While this is a step in the right direction, human moderators are unable to keep pace with the large volume of user-generated content today and manually verify whether each flagged post violates community standards. Furthermore, the offensive content might have already spread and caused considerable damage before it is addressed by a platform. These issues motivate the research and development of natural language processing systems to automatically detect hate speech to ensure that online platforms remain healthy and inclusive for all. Before considering the development of hate speech models, it is crucial that we first have a nuanced understanding of what constitutes hate speech.

Section 1.2

What is Hate Speech?

Determining whether a piece of text is considered hate speech is by no means a simple task even by humans, let alone machines. Hate speech is highly subjective in nature and what construes as offensive to one individual may not be the case for another depending on one's background, context, socio-cultural factors and language nuances. To gain a visceral understanding of hate speech in all its subtleties, it is prudent to learn how multiple sources chose to define it (Table 1.1).

While the specific definitions of hate speech may vary depending on the source, they share a number of commonalities, as pointed out by Fortuna and Nunes [10]:

1. Hate speech has **specific targets** and is based upon specific characteristics of groups like race, religion, ethnicity etc.
2. Hate speech **incites** violence and retaliation against such groups.
3. Hate speech **attacks or diminishes** such groups.
4. Hate speech is **complicated** by the use of humor and sarcasm.

Source	Definition
Cambridge Dictionary	"Public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation." [6]
Facebook	"We define hate speech as a direct attack against people on the basis of what we call protected characteristics: race, ethnicity, national origin, disability, religious affiliation, caste, sexual orientation, sex, gender identity and serious disease." [8]
YouTube	"We consider content hate speech when it incites hatred or violence against groups based on protected attributes such as age, gender, race, caste, religion, sexual orientation, or veteran status. This policy also includes common forms of online hate such as dehumanizing members of these groups; characterizing them as inherently inferior or ill; promoting hateful ideology like Nazism; promoting conspiracy theories about these groups; or denying that well-documented violent events took place, like a school shooting." [32]
Twitter	"You may not promote violence against or directly attack or threaten other people on the basis of race, ethnicity, national origin, caste, sexual orientation, gender, gender identity, religious affiliation, age, disability, or serious disease. We also do not allow accounts whose primary purpose is inciting harm towards others on the basis of these categories." [27]

Table 1.1: Selected definitions of hate speech from various sources.

A number of related concepts that often accompany hate speech include: discrimination, flaming, abusive language, profanity, extremism, radicalization [10]. While by no means an exhaustive list, we are now equipped with a better understanding of the various kinds of hate speech that could be present on online platforms.

Section 1.3

Problem Formulation

Before technically formulating the hate speech problem that the remainder of this thesis seeks to address, it is helpful to define some key terminologies that will allow us to frame the problem (Table 1.2).

Term	**Definition**
Document	This refers to any piece of distinct text like an online post, comment, article etc.
Span	An ordered sequence of words extracted from a document, represented using a list of character offsets (zero-indexed).

Table 1.2: Definition of key terms used for formulating the problem.

A natural question one might ask is whether a given document contains hate speech? While relevant, this binary classification task does not explain why a given document is considered toxic by the model. Another question one might be prompted to ask is the degree of toxicity contained in a given document? Again, merely providing an unexplained toxicity score in such a regression task does not sufficiently assist human moderators to address questionable content. The inability of these course-grained questions to identify the spans that ascribe the offensive sections of a document motivated the formulation of the **Toxic Spans Detection** problem by

Pavlopoulos et al. [21].

Formally, given a document D consisting of the sequence of character indexed $[0, 1, \cdots, n-1, n]$, a system S is tasked to extract the list of character offsets S_d that maps to the toxic spans contained within D, if present. Consider the following example:

Document	Because he's a **moron** and **bigot**. It's not any more complicated than that.
Span	[15, 16, 17, 18, 19, 25, 26, 27, 28, 29]

Table 1.3: A sample example from the dataset to illustrate the problem.

As there are two toxic spans in the above text, systems are asked to extract the character offsets (zero-indexed) corresponding to the sequence of toxic words. This is a challenging task as classification at the word-level is inherently more difficult than at the document-level. The intentional obfuscation of toxic words, use of sarcasm and the subjective nature of hate speech further adds complexity to the problem. Being able to solve this difficult problem will assist human moderators to efficiently locate offensive content in long posts and elucidate further insight into hate speech explainability.

Section 1.4

Contributions

Our contributions to the hate speech problem are threefold:

1. We propose **BERToxic**, an empirically powerful system that fine-tunes a pre-trained BERT model with additional post-processing steps to achieve an F1-score of 0.683, placing our model in the 17$^{\text{th}}$ place out of 91 teams in a hate

speech competition.

2. We examine late fusion and multi-task learning neural architectures and conclude that they under-perform compared to the standalone BERT model for this task.

3. We study the effects of simple data augmentation strategies on our system and find that they yield no improvement in classification performance.

Section 1.5

Organization of Chapters

The following list provides an overview of how the remainder of this thesis is organized.

- Chapter 2 reviews related work on the automatic detection of hate speech.
- Chapter 3 describes the models we develop for the fine-grained detection of hate speech.
- Chapter 4 describes the experimental set-up to evaluate the performance of our models.
- Chapter 5 analyzes the results of our experiments.
- Chapter 6 concludes this thesis by summarizing our findings, highlighting limitations and providing future avenues of work.

Chapter 2

Related Work

There has been extensive research on hate speech detection and the literature on this subject is vast. Rather than providing a systematic literature review that many excellent survey papers already provide [25, 10], we will highlight the main approaches used for the automatic detection of hate speech. Our review of the related work seeks to contextualize readers to some previous approaches to address the hate speech detection problem and is by no means exhaustive. We summarize the main techniques from the comprehensive survey conducted by Fortuna and Nunes [10].

We start the exploration by reviewing some feature extraction techniques to capture salient features related to hate speech.

Dictionaries. One of the simplest strategies to detect offensive language is the use of dictionaries that consists of a collection of words. The dictionaries could be constructed using websites like `https://www.noswearing.com` that contain pre-compiled lists of offensive words like profanity, insults, slurs, etc. Hence, this rudimentary dictionary technique essentially creates a hand-crafted "blacklist" of toxic words that can be used to filter texts that are likely to contain offensive language. Given a piece of text, the document is tokenized into words and looked up in the dictionary. The

matched words and their frequencies can either directly be used as features or computed into scores. This is one of the techniques used by Dinakar et al. [7] to detect cyberbullying on social media. However, the obfuscation of offensive words (such as, $ass \mapsto a\$\$$) can easily evade the detection of this simple key-word spotting approach.

Bag-of-words. This is another approach that is similar to dictionaries where words are used as surface-level features. Unlike dictionaries that use a pre-defined list of offensive words, bag-of-words techniques utilize a training corpus to collect the list of words and their associated frequencies. This information is then used as features to train a classifier for hate speech detection. For instance, Greevy and Smeaton [11] use this approach to train a Support Vector Machine (SVM) to classify racist texts. The disadvantage of this technique is that it completely ignores word sequences and examines words in isolation. This results in a loss of the semantic content of the training examples. Unable to capture the words in context, misclassifications are likely to occur as offensive words can be used in neutral contexts.

N-grams. This model builds upon the earlier approaches by collecting sequences of words, instead of single words. More technically, this technique combines sequential words into lists with the goal of enumerating all the expressions of size N and computing their associated frequencies. Note that bag-of-words is a special case of this approach where $N = 1$ (unigrams). Other common values of N used are 2 (bigrams) and 3 (trigrams). Using N-grams as features typically improves the classification performance of hate speech detection as it incorporates some degree of context for each word. It is also possible to consider character or syllable N-grams as features as this finer level of granularity is not as susceptible to spelling variations compared to word-level N-grams. Indeed, one study by Mehdad and Tetreault [17] found that character N-gram features proved to be more predictive than word N-gram features

for abusive language detection. Nevertheless, N-gram techniques suffer drawbacks when contextual words are distanced further apart from each other. While increasing the value of N may be a way to overcome this problem, this solution increases computational time.

TF-IDF. The **T**erm **F**requency-**I**nverse **D**ocument **F**requency is a statistical measure of the relevance of a word in a document within a corpus by proportionally increasing the number of times a word appears in the document. This approach is distinct from bag-of-words and N-grams as the TF-IDF value is offset by the number of documents in the corpus that contain the word. This adjusts for the fact that some words appear more frequently than others (e.g stopwords). TF-IDF can be used to extract features to train a classifier. In the same study, Dinakar et al. [7] applies the TF-IDF technique to detect cyberbullying on social media. However, TF-IDF also has limitations as it makes no use of semantic similarities between words and computes document similarity directly in the word-count space, which may be slow for large vocabularies.

Word Embeddings. This is a class of methods that learn a real-valued vector representation of words for a pre-defined fixed-sized vocabulary from a corpus of text. Words that have similar meanings have similar representations in the vector space, thereby capturing the semantic similarity between words. Popular word embedding algorithms include word2vec developed by Mikolov et al. [18] and GloVe by Pennington et al. [22]. Given a document, such pre-trained word embeddings can be used to extract semantic features for training a hate speech classifier. For example, Badjatiya et al. [2] reported that word embeddings used in deep learning models for hate speech detection in tweets improved the F1-score by 18% compared to the character and word N-gram methods.

Classical Machine Learning. Having explored a number of feature extraction methods, we will briefly mention the classical models that the features could be fed into for hate speech prediction. These include Logistic Regression, Support Vector Machines (SVM) and Random Forest Decision Tree, among others. Many authors report that combining the models into ensembles and aggregating the predictions often improve the classification performance. Most machine learning solutions on hate speech detection rely on manually labeled training examples in a supervised learning setting.

Deep Learning. Rather than relying on feature extraction methods selected manually, deep learning uses feature learning to automatically learn hidden patterns end-to-end. For instance, Saksesi et al. [23] used recurrent neural networks (RNN) to process text data for hate speech detection. However, RNNs are known to suffer from the vanishing gradient problem, making it difficult to learn from long-term dependencies. Long Short-Term Memory (LSTM) / Gated Recurrent Units (GRU) attempt to overcome this problem by having multiple gates to improve gradient flow. However, LSTMs are easy to overfit and take longer to train due to the inherent recurrence structure that prevents parallelization. These issues were overcome with the invention of the transformer architecture by Vaswani et al. [28], subsequently inspiring the development of the BERT model by Devlin et al. [5] that have achieved state-of-the-art performance in numerous NLP benchmark tasks. The high performance achieved by BERT-based language models has made it the most popular approach for hate speech detection in recent times. Typical solutions leverage transfer learning by fine-tuning a pre-trained deep learning model on a hate speech training dataset. The success of this approach and capability for end-to-end learning motivated us to leverage deep learning for the fine-grained detection of hate speech.

Datasets. To the best of our knowledge, there is currently no standardized hate speech benchmark dataset for the fair comparison of various approaches. Many authors have collected their own datasets to study specific aspects of hate speech detection. Nevertheless, it is worth noting that progress has been made in open-sourcing hate speech datasets by some researchers. For example, Wulczyn et al. [30] released a corpus of approximately $100,000$ human-annotated English Wikipedia comments. This data formed the basis of the Toxic Comments Classification Challenge on Kaggle and resulted in the release of the Perspective API by Jigsaw/Google that categorizes hate speech into six categories (severe toxicity, insult, profanity, identity attack, threat and sexually explicit). Competitions on hate speech detection have further resulted in the curation of some hate speech datasets. For instance, Zampieri et al. created the Offensive Language Identification dataset [33, 34] that consists of $14,200$ tweets that were labeled using a hierarchical three-level annotation model.

Problem Formulation. Prior work has hitherto focused on classification at the document-level based on various taxonomies, such as whether a given text contains offensive language, if it is targeted towards an individual or group, and categorizing the text into a number of pre-defined labels. While these are good questions to gain a multi-faceted understanding of hate speech, such problem formulations do not identify the words and phrases that attribute to the text's toxicity. In other words, hate speech detection has not yet been formulated as a sequence labeling problem, which this thesis addresses.

Chapter 3

Models

In this chapter, we develop various models for solving the toxic spans detection problem.

Section 3.1

Baselines

To have a better sense of our final system's performance, we initially examined two baseline models to establish lower bounds for classification performance.

3.1.1. Random

First, we created a trivial model that randomly predicts each character offset of a text as toxic if its probability is greater than half (i.e $\rho > 0.5$), drawn from a continuous uniform probability distribution. This dummy model relies purely on randomization and makes no use of the semantic content of the underlying text.

3.1.2. SpaCy

To have a stronger baseline model, we fine-tuned the off-the-shelf spaCy NER model. This model consists of a multi-hash embedding layer (feed-forward sub-network) that

uses sub-word features and an encoding layer consisting of a CNN and a layer-normalized max-out activation function. The model uses a transition-based algorithm that assumes that the "most decisive information" regarding the entities "will be close to their initial tokens", with a loss function that optimizes for whole-entity accuracy.

Section 3.2

BERToxic

Having established baseline models for comparison, we are ready to describe our proposed BERToxic system. Our model builds upon the Transformer architecture [28], which we describe in the following section.

3.2.1. Transformer Architecture

In the seminal paper "Attention Is All You Need" [28], Vaswani et al. introduced the Transformer architecture that revolutionized the field of NLP. This architecture forms the basis of numerous language models today and understanding it is crucial to appreciate our proposed BERToxic system. To benefit readers who are unfamiliar with transformers, we will provide a brief overview of the architecture in this section based on the aforementioned paper.

The transformer architecture is a sequence transduction model that was initially proposed for the task of machine translation. Its novelty relies on the use of attention mechanisms, enabling the neural network to effectively learn relationships between the input and output sequences. Furthermore, the transformer eliminates the use of recurrence and convolutional structures that used to be prevalent in the past. Removing the sequential dependency enables the effective use of parallelism, thereby reducing the time taken to train models.

Having briefly introduced the transformer architecture, we will now describe its

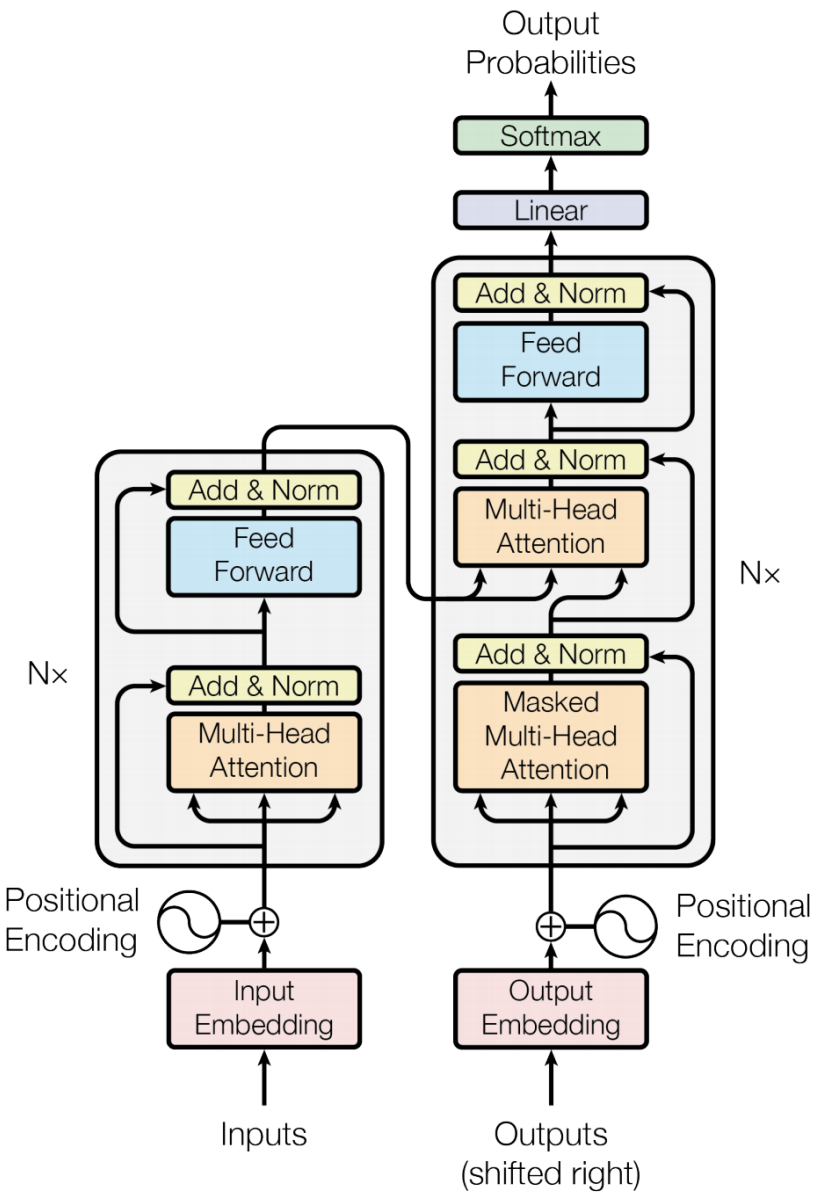

Figure 3.1: The transformer architecture. Image credits: Vaswani et al. [28]

main components, referencing the visualization provided in Figure 3.1. The transformer uses an encoder-decoder architecture, with the encoder and decoder shown on the left and right sides of Figure 3.1 respectively. The encoder maps a sequence of inputs $(x_1, x_2, \cdots, x_n)$ to a sequence of continuous representations $z = (z_1, z_2, \cdots, z_n)$. Given z, the decoder then auto-regressively generates the output sequence $(y_1, y_2, \cdots, y_m)$ one token at a time. The overall transformer consists of multiple encoders and decoders stacked on top of each other, represented as N_x in Figure 3.1. Let us now delve deeper to study the constituents of the encoder and decoder blocks.

The encoder consists of a stack of $N = 6$ identical layers. Each layer consists of two sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network. Residual connections around each of the two sub-layers exist to improve gradient flow and layer normalization follows subsequently. All the encoding blocks produce an output of dimension 512, which is the maximum sequence length allowed by the architecture.

The decoder is similarly composed of a stack of $N = 6$ identical layers. Besides the two sub-layers, each decoder block also includes a third multi-head attention sub-layer. Again, residual connections are featured here and the self-attention sub-layer in the decoder is modified to ensure that predictions at position i are only attended by previous positions less than i.

The power of the transformer lies upon its novel self-attention mechanism (Figure 3.2). Based on the intuition that humans pay attention to certain words in a sentence more so than others, the attention mechanism looks at an input sequence and learns which parts of the input to attend towards. Technically, the attention function maps a query (Q) and a set of key (K) value (V) pairs to an output, where the query, key, value and output are all vectors. Multi-head attention allows the model to jointly

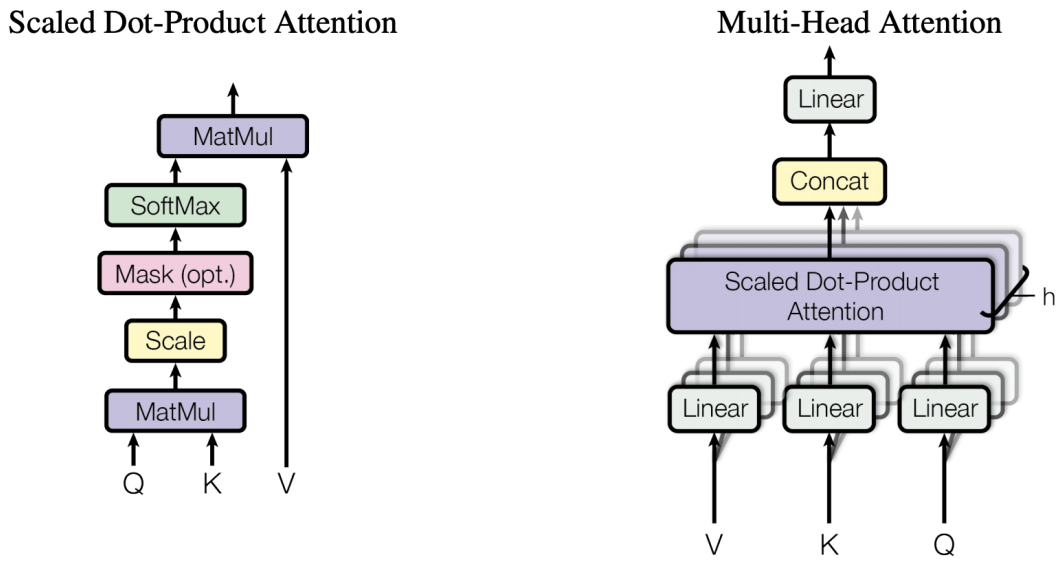

Figure 3.2: The transformer's attention mechanism. Image credits: Vaswani et al. [28]

attend to information from different representation sub-spaces at different positions, which is not possible using a single attention head. Unfortunately, this self-attention mechanism introduces the main efficiency bottleneck in transformers. Each token's representation is updated by attending to *all* other tokens in the previous layer, incurring a computation complexity of $O(n^2)$ time. As such, this quadratic time complexity increases the training time of transformer-based models.

3.2.2. BERT

Having provided an overview of the transformer architecture, we will now describe the BERT model on which our namesake BERToxic system is based upon. The **B**idirectional **E**ncoder **R**epresentation from **T**ransformers (BERT) model was proposed by Devlin et al. [5] as a general language representation model. As the name suggests, it utilizes the encoder stack from the transformer architecture and innovates

by pre-training deep bidirectional representations from unlabeled text by jointly conditioning on both the left and right context in all layers. The pre-trained BERT model can be used to approach a wide variety of downstream tasks by having just one additional task-specific output layer to achieve state-of-the-art results. Given its high performance and uniform architecture, the BERT model has been a popular model of choice in recent years. Based on the excellent BERT paper [5], we provide an overview for readers who are unfamiliar with this model.

A major limitation of language models prior to BERT was the uni-directionality constraint during pre-training. The inability to incorporate context from both directions limits the language model's effectiveness when fine-tuned for downstream tasks. BERT overcomes the uni-directionality constraint by using a Masked Language Model (MLM) pre-training objective. The key idea is to randomly mask some of the input tokens, with the goal of predicting the masked tokens based only on the context. Besides the MLM objective, BERT also utilizes the Next Sentence Prediction (NSP) objective that pre-trains text-pair representations. Both these tasks are unsupervised and do not require the use of labeled data. Figure 3.3 visualizes the two pre-training tasks used by BERT, which we elaborate on below.

It makes intuitive sense that a deep bidirectional model obtains richer contextual feature representations than a unidirectional model. In the Masked Language Modeling (MLM) task, 15% of the tokens in each sequence is masked with the `[MASK]` token at random. The fill-in-the-blanks exercise enables the model to incorporate the context in both the left and right sides to predict the missing tokens. Note that since the input tokens are masked at random and standard cross-entropy loss is used to train the MLM objective, the task is unsupervised in nature and does not require the use of labeled data.

Some downstream tasks such as question-answering rely on understanding the

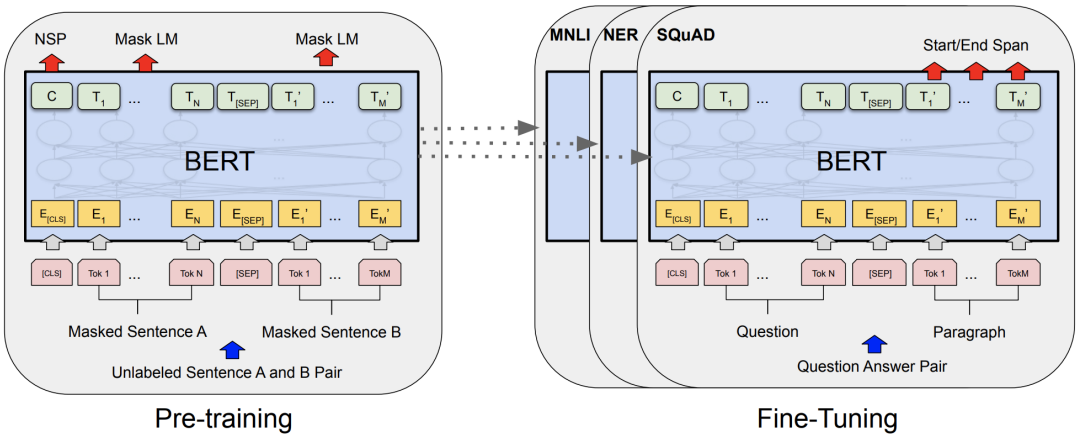

Figure 3.3: The pre-training and fine-tuning procedures for BERT. Image credits: Devlin et al. [5].

relationship between two sentences. Since MLM does not directly capture this relationship, BERT also incorporates the Next Sentence Prediction (NSP) task. The NSP task is set up by first creating pairs of sentences from the original corpus. For any given sentence pair A and B, 50% of the time, sentence B directly follows sentence A and 50% of the time, it is a random sentence. The `[SEP]` token is used in BERT's input representation for this purpose while the `[CLS]` token is used for sequence classification. Again, note that the NSP task is unsupervised in nature and does not require the use of any labeled data.

The unsupervised nature of the two pre-training tasks enables BERT to learn from a large corpus of text. BERT is pre-trained with the BookCorpus (800M words) and the English Wikipedia (2500M words), enabling it to learn rich language feature representations. The parameters of the pre-trained BERT model can then be fine-tuned with just one task-specific output layer for a wide variety of downstream tasks.

BERT traditionally comes in two model sizes - $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$. The former model consists of 12 layers, 768 hidden size and 12 self-attention heads with

$110M$ parameters. The latter model consists of 24 layers, 1024 hidden size and 24 self-attention heads with $340M$ parameters. It is notable that the original BERT model inspired a whole family of BERT-based model variants such as DistilBERT [24], RoBERTa [15], ALBERT [13] etc.

3.2.3. Toxic Spans Detection

Having described the preliminary background literature, we are now ready to introduce our proposed BERToxic system (Figure 3.4). We framed the toxic spans detection task as a sequence labeling problem and leverage the BERT model to extract rich feature representations from the input texts.

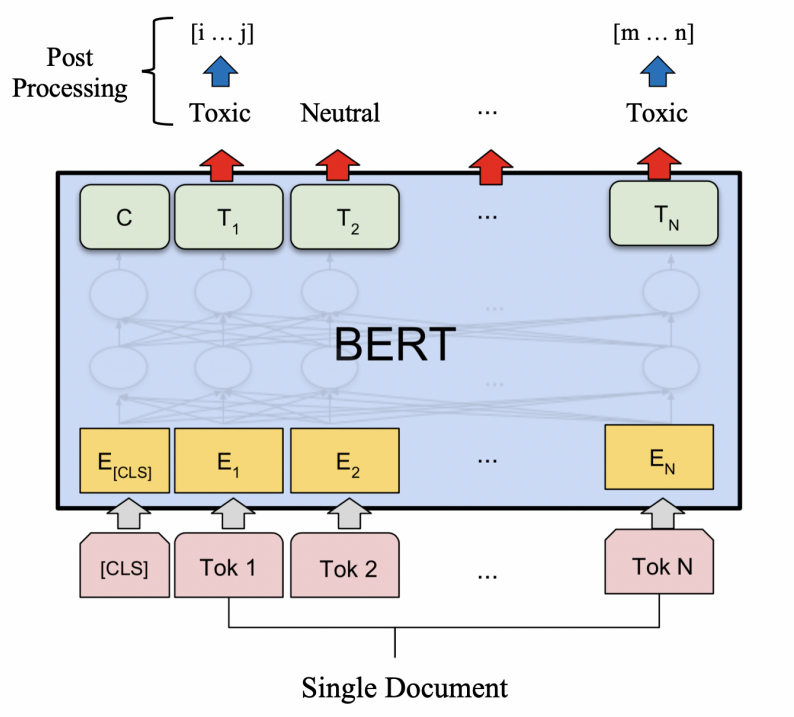

Figure 3.4: The BERToxic model architecture. Image modified from Devlin et al. [5].

The first step in our system's pipeline was to tokenize the text inputs and generate the word embeddings using BERT's WordPiece tokenizer. This sub-word tokenization algorithm by Schuster and Nakajima [26] tokenizes a word like `"moron"` into

`["mo","##ron"]` and we ensured that the ground truth labels were preserved across all tokens of a word. As BERT uses absolute position embeddings, we padded shorter sequences with `[PAD]` tokens on the right side such that all tensor inputs are set to equal the maximum sequence length observed for batched parallelized training. Long sequences were truncated to 512 tokens, the maximum sequence length allowed by BERT. As the data was obtained from online comments that are generally shorter in nature, the truncation procedure was not needed in this task but nevertheless served to handle long sequences if present.

We also stored the mapping

$$\mathcal{M} : t_i \mapsto (start_i, end_i)$$

of each token to its relative character offsets in the original string, used for outputting the toxic span predictions at the post-processing stage.

We performed all of our experiments using the $\text{BERT}_{\text{BASE}}$ model architecture that consisted of 12 layers, 768 hidden size, 12 self-attention heads and $109M$ parameters. The $\text{BERT}_{\text{LARGE}}$ model was not explored in this work due to its compute-intensive nature. Our intuition suggested that letter casing could be helpful for this task as proper nouns (e.g *Muslim*) can be used offensively, so we selected the cased model for our experiments. A token classification head containing a linear layer was applied on top of the final hidden-states output, with a label prediction of 1 denoting a toxic token, 0 otherwise. For each token t_i labeled as toxic, we utilized $\mathcal{M}$ to output all character indices in the range of $(start_i, end_i)$ inclusive as the toxic span of this token.

Additionally, our system performed two post-processing steps to refine the boundary predictions. Consider the following tokenized sequence:

$$t_1, \cdots, t_i, t_{i+1}, t_{i+2}, \cdots, t_n$$

First, for any two consecutive tokens t_i and t_{i+1} whose prediction labels are toxic, we output the character indices in the range of $(end_i + 1, start_{i+1} - 1)$ inclusive as toxic as well. This had the effect of including the delimiter characters between consecutive toxic words, thereby detecting toxic phrases. Second, recall that BERT's WordPiece tokenizer could split a word into multiple tokens, say t_i, t_{i+1} and t_{i+2}. If at least one token was predicted toxic by the model, our system assigned a toxic label to all constituent tokens of this word. This achieved coherence in the prediction of toxic words and phrases, thus avoiding incomplete word piece issues.

We also attempted to vary the thresholds of the confidence scores before SoftMax for toxic token predictions but observed no improvement in performance.

Section 3.3

Ensemble Modeling

Ensemble modeling is an approach where multiple different models are trained and their predictions are aggregated. By adding bias to counter the variance of a single model, this line of work has been shown to improve the predictive performance of a system [14]. While numerous ensemble modeling techniques like boosting, bagging, etc. exist, we investigated two techniques of interest: late fusion and multi-task learning.

3.3.1. Late Fusion

We reframed the problem as a binary classification task and trained a sequence classifier to predict whether a given sentence is toxic. In the late fusion approach, we utilized NLTK's tokenizer to split each document into sentences. If a sentence contained a ground truth toxic span, we assigned the toxic class label 1, 0 otherwise. In this way, a binary classification dataset was created to separately fine-tune a pre-trained

BERT sequence classifier. We hypothesized that token labels should be predicted toxic only if the corresponding sentence was classified as toxic as well. Late fusion was performed at the prediction phase, where both the sequence and token classifiers voted in the predictions by having the former model filter toxic sentences on which the latter model made final toxic span predictions. Figure 3.5 below visualizes the late-fusion neural architecture.

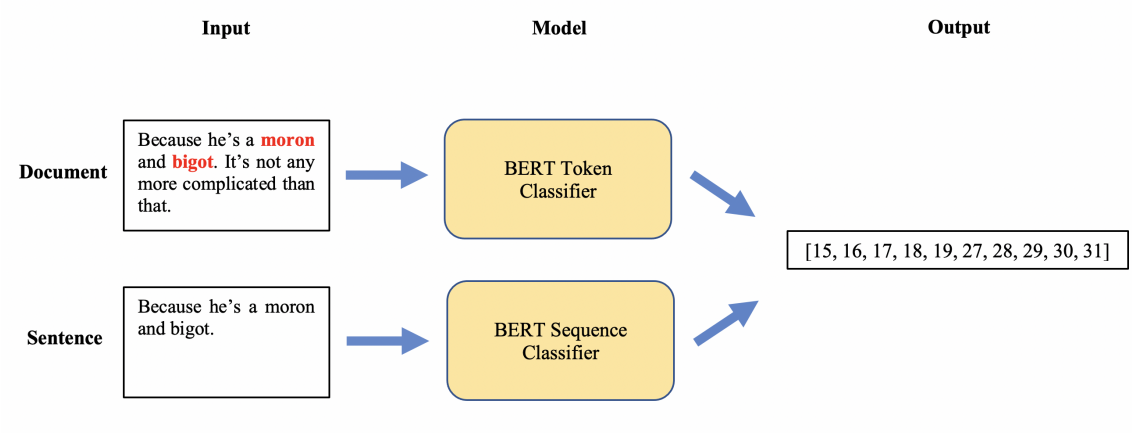

Figure 3.5: The BERT late-fusion model architecture.

3.3.2. Multi-task Learning

Rather than fine-tuning the two models separately, we also investigated if multi-task learning (MTL) improved the predictive performance of the ensemble model. We hypothesized that a training regime where the two classifiers were learned jointly could be useful as the knowledge gained in learning one task could benefit the other. To perform MTL, we fine-tuned the Multi-task Deep Neural Network (MT-DNN) model proposed by Liu et al. [14]. In the MT-DNN model, the text encoding lower BERT layers are shared across the two tasks while the top layers are task-specific. During fine-tuning, a mini-batch b_t is selected and the model is updated according to the task-specific objective for the task t. This approximately optimizes the sum of

all multi-task objectives. Figure 3.6 below visualizes the multi-task learning neural architecture.

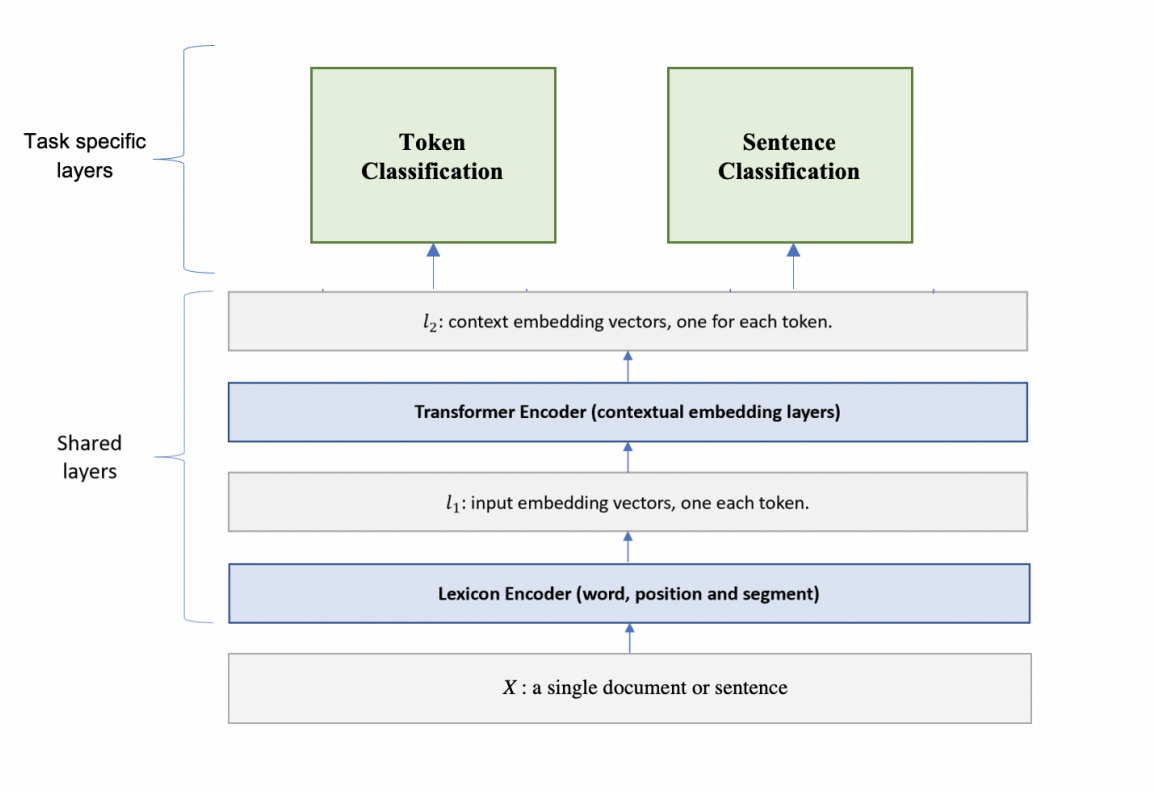

Figure 3.6: The BERT multi-task model architecture. Image modified from Liu et al. [14].

Section 3.4

Data Augmentation

Data augmentation is widely used to improve the generalization of models by acting as a regularizer to reduce overfitting. While various sophisticated techniques exist to artificially enhance the size and quality of the training set without collecting additional manually labeled examples, we chose to investigate two data augmentation techniques of interest: Easy Data Augmentation and using an external dataset.

3.4.1. Easy Data Augmentation

We chose to apply the set of **E**asy **D**ata **A**ugmentation (EDA) techniques by Wei and Zou [29] to generate synthetic training data for this task. As outlined in their paper, the four operations in EDA are the following:

1. **Synonym Replacement (SR)**: Randomly pick n words from the document that are not stop words. Replace each of these words with a random synonym obtained from WordNet [19].

2. **Random Insertion (RI)**: Randomly pick a synonym of a random word in the document that is not a stop word. Insert this synonym into a random position in the document. Repeat this procedure n times.

3. **Random Swap (RS)**: Randomly pick two words from the document and swap their positions. Repeat this procedure n times.

4. **Random Deletion (RD):**: Randomly remove each word in a document with probability ρ.

Shorter documents are disproportionately more affected by these operations if a fixed number of words are modified per document. To ensure that all documents experienced the augmentation strength proportionately, the number of words n modified was varied based on the document length l using the formula

$$n = \alpha \cdot l$$

where α is a hyper-parameter that indicates the percentage of words changed per document. Each operation was applied once per document and care was taken to ensure that the ground truth labels were preserved.

Our experiments revealed that the recommended value of $\alpha = 0.1$ was too low for this task and we observed small but consistent improvements as α increases. Furthermore, we noted that the SR technique alone leads to better performance than using all four operations to create the augmented training set for this task.

3.4.2. External Dataset

We also attempted data augmentation using the external HateXplain dataset by Matthew et al. [16] that contains $20,148$ documents with word-level annotations that we processed to conform to the toxic span's detection data format. Each document consisted of $2 - 3$ annotations and we used their intersection to maximize the inter-annotator agreement in constructing the ground truth labels. HateXplain's annotation strategy appeared to be different and included labeling pronouns, conjunctions and stop words as toxic when located between offensive words. We removed such toxic labels so that the external dataset annotation was more similar to this task. When our task dataset was augmented with the full external dataset, the model experienced underfitting, while removing all the non-toxic labeled documents from the external dataset alleviated the issue to some extent.

Chapter 4

Experiments

In the following sections, we describe the experimental setup of our work.

Section 4.1

Dataset

The task data was sourced from the Civil Comments dataset by Borkan et al. [3], which contains public comments made between 2015 – 2017 that appeared on approximately 50 English-language news sites across the world. As the original dataset contained only document-level class labels, the task organizers selected a subset of the data for crowd-sourced toxic spans annotation. For the data split, we chose to fine-tune our models using the entire provided training dataset ($N = 7939$) to maximize performance, validate using the trial dataset ($N = 690$), and evaluate our model using the test data ($N = 2000$). The test labels were withheld during the evaluation phase of the competition and were only released afterward.

It is useful to perform exploratory data analysis to get a nuanced understanding of the data. Let us first examine the ground truth annotations. Figure 4.1 visualizes the distribution of the span lengths in the three data splits. We plot the histograms using bins of size 150 and only consider span lengths less than 100 as there is a negligible

number of spans beyond this length. As seen from the histogram plots, the span lengths of all three data sets follow a right-skewed distribution, with the majority of documents having toxic span annotations that are less than 20 character offsets long.

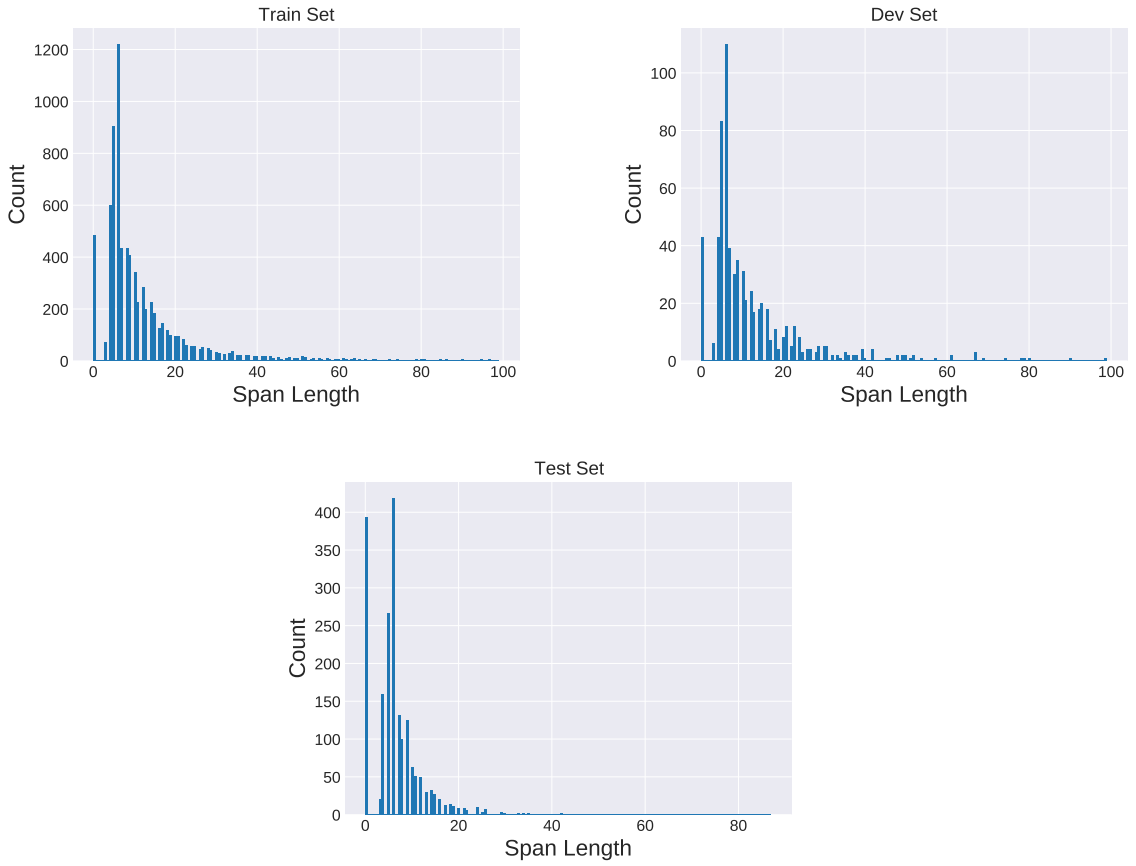

Figure 4.1: Histograms of Span Length in the data splits.

Table 4.1 provides further insights by showing the summary statistics observed in the annotations. The statistics reveal that there is significant variation in the distribution of toxic annotations in various data splits. In particular, note that the mean span length in the test set is much lower than the other two sets. Furthermore, the standard deviation in the three data splits varies considerably. We note that these differences can impact the performance metrics of the models in the data splits.

Finally, we visualized the 100 most frequent offensive words that are not stop

Set	Mean	Median	Mode	Standard Deviation
Train	17.5	8.0	6.0	45.7
Dev	14.7	8.0	6.0	25.5
Test	7.2	6.0	6.0	17.7

Table 4.1: Summary statistics of the span lengths in the data splits.

words in Figure 4.2. The word clouds reveal that some of the most common toxic words found in the data include “idiot”, “stupid” and “moron”.

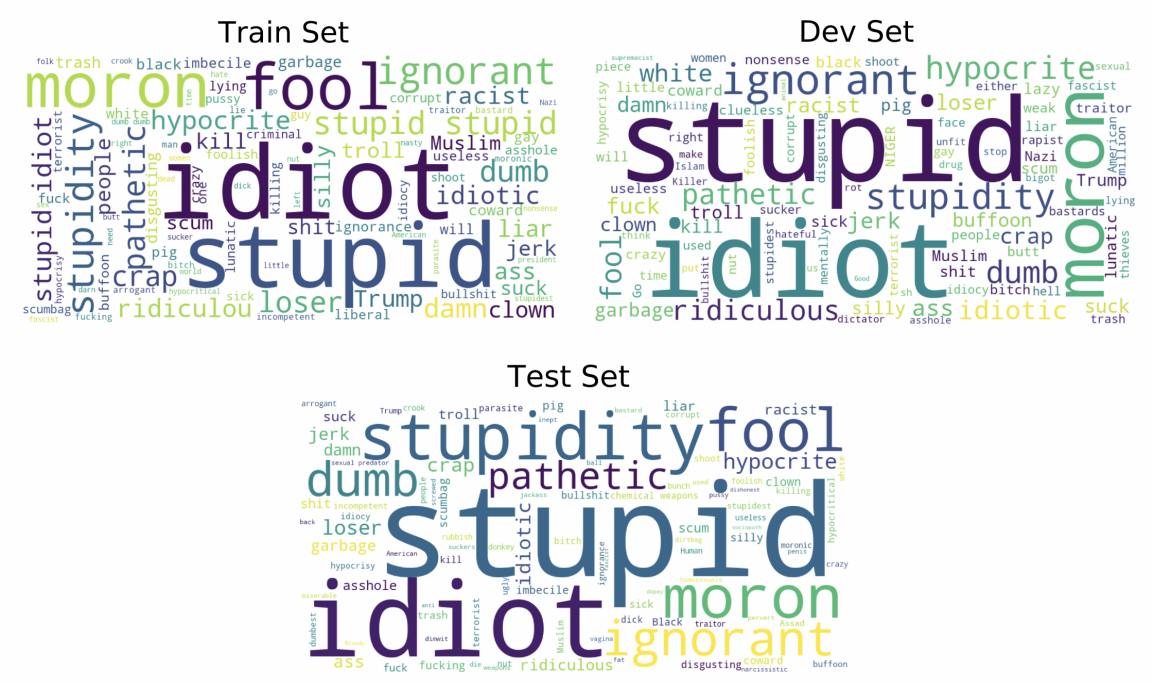

Figure 4.2: Word Clouds generated from the data splits.

Section 4.2

Evaluation Metric

To evaluate the performance of the models, the task organizers employed a variant of the F1-score proposed by Da San Martino et al. [4]. For a document d, define S_d as the set of toxic character offsets predicted by a system and G_d as the set of ground truth annotations. Then the F1-score of the system with respect to ground truth G for d is defined as

$$F_1{}^d(G) = \frac{2 \cdot P^d(G) \cdot R^d(G)}{P^d(G) + R^d(G)}$$

where

$$P^d(G) = \frac{|S_d \cap G_d|}{|S_d|}$$

$$R^d(G) = \frac{|S_d \cap G_d|}{|G_d|}$$

If a document has no ground truth annotation ($G_d = \emptyset$), or the system outputs no character offset prediction ($S_d = \emptyset$), we set

$$F_1{}^d(G) = \begin{cases} 1 & G_d = S_d = \emptyset \\ 0 & otherwise \end{cases}$$

We finally take the arithmetic mean of $F_1{}^d(G)$ over all the documents of an evaluation dataset to obtain a single F1-score for the system.

Section 4.3

Implementation Details

We utilized the PyTorch framework for the development of our system, HuggingFace's transformers library for the BERT-based models and Microsoft's implementation of the MT-DNN model. All models were trained on Google Colab Pro's High-RAM environment using a single NVIDIA P100 GPU. The training policy used the following hyper-parameters: batch size of 16, sequence length of 512, weight decay of 0.01. For optimization, we used Adam with a learning rate of 5e-5 and a linear warm-up schedule over 500 steps. All our models were fine-tuned for approximately 2 epochs and we practiced early stopping by monitoring the dev F1-score to reduce overfitting. The MT-DNN model was fined-tuned for 3 epochs with a batch size of 8. The EDA experiment was performed with $\alpha = 0.8$ using only the SR technique. All other hyper-parameters were set to their default values according to HuggingFace's implementation. We set a random seed for all our experiments and open-sourced the code for reproducibility.

Chapter 5

Results

In this section, we present the results of our experiments and analyze our findings.

Section 5.1

Model Performances

On the following page, Table 5.1 summarizes the performance metrics of all our models. The BERToxic model outperformed the strong spaCy baseline by 4.16% on the test set, placing our system in the 17^{th} place out of 91 teams in the SemEval Toxic Spans Detection competition. In comparison, the top-ranked submission achieved an F1-score of 0.708. The experiments also revealed that our data augmentation and ensemble modeling strategies did not outperform the standalone BERT model.

An interesting observation we noted from Table 5.1 was that the F1 scores for the test set were higher than the dev set for many of the models. We hypothesize that this is because the models have an inductive bias to predict shorter toxic spans, evidenced by the average ground truth span length of 7.2 in the test set and 14.7 in the dev set (Table 4.1).

Figure 5.1 shows the confusion matrix of the BERToxic system at the token level on the test set, revealing insights about the classification performance in each category

Model	**Dev**			**Test**		
	Precision	Recall	F1	Precision	Recall	F1
Random	0.143	0.463	0.175	0.089	0.413	0.122
SpaCy	0.692	0.588	0.595	0.664	0.686	0.656
BERToxic	0.781	0.678	0.681	0.683	0.732	**0.683**
+ EDA	0.787	0.683	0.684	0.681	0.725	0.678
+ HateXplain	0.792	0.674	0.681	0.683	0.721	0.678
BERT late fusion	0.733	0.636	0.639	0.675	0.709	0.669
BERT multi-task	0.744	0.629	0.634	0.665	0.694	0.656

Table 5.1: Summary of the performance of all our models, reporting the precision, recall and F1 scores on the dev and test sets.

and highlighting the imbalance of the class labels.

Figure 5.2 displays the learning curves of the BERToxic model during the training process. It can be seen that the model converges after 2 epochs, with the training loss curve fluctuating throughout the fine-tuning process while the dev loss curve steadily plateaus.

Figure 5.3 visualizes the performance metrics of the BERToxic model on the dev set during the training process. It can be seen that the precision curve is above the recall and F1-score curves. Along with the loss curve in Figure 5.2, these visualizations ensure that the model convergence has occurred and overfitting was prevented.

Figure 5.4 compares the precision-recall curves of all the models at the token-level on the test set. The area under the curve is enclosed within parentheses in the figure. We note that the curves for the spaCy and BERT multi-task model are less detailed due to the ambiguity in obtaining the probability scores from their respective implementations, necessitating the use of their predicted labels instead.

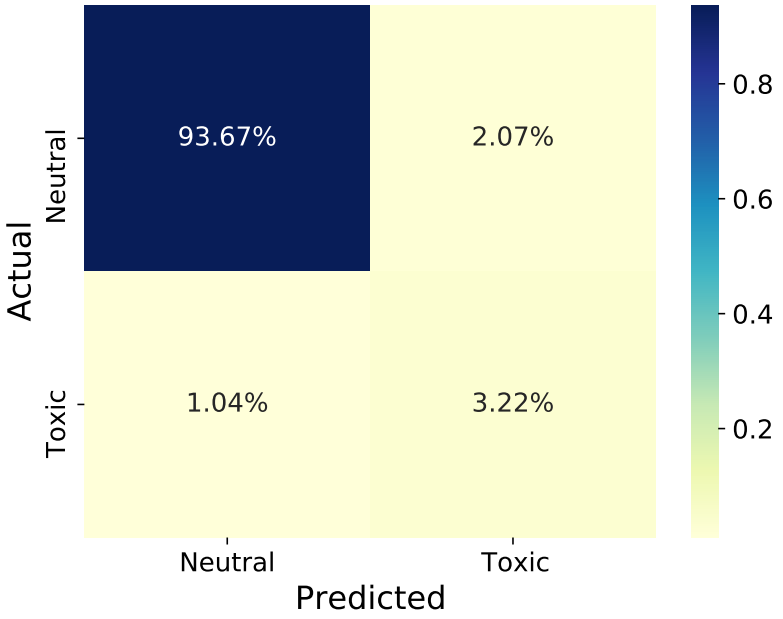

Figure 5.1: Confusion matrix of the BERToxic model.

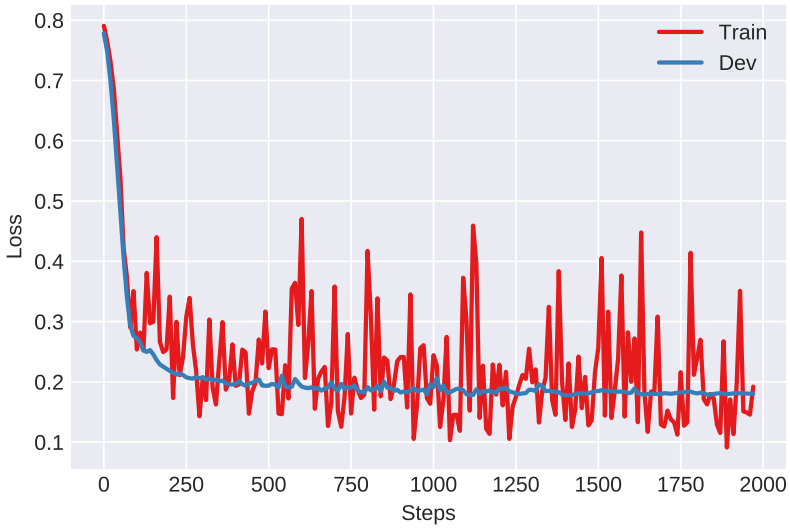

Figure 5.2: Learning curves when training the BERToxic model.

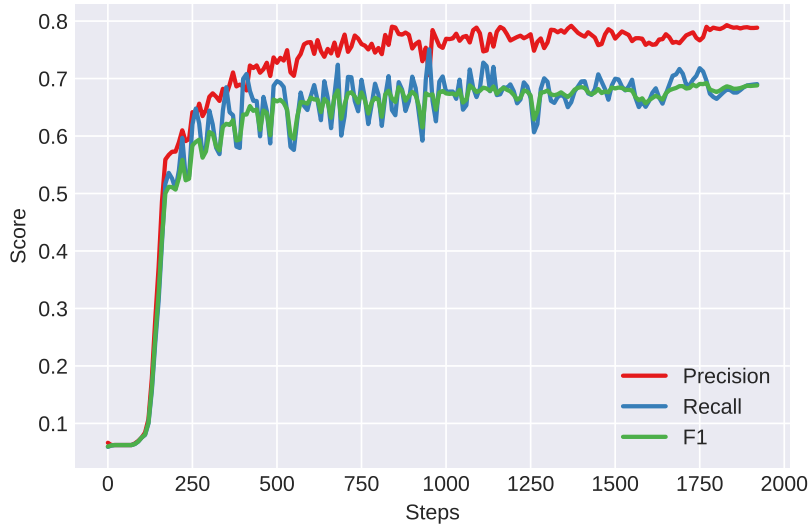

Figure 5.3: Performance curves when training the BERToxic model.

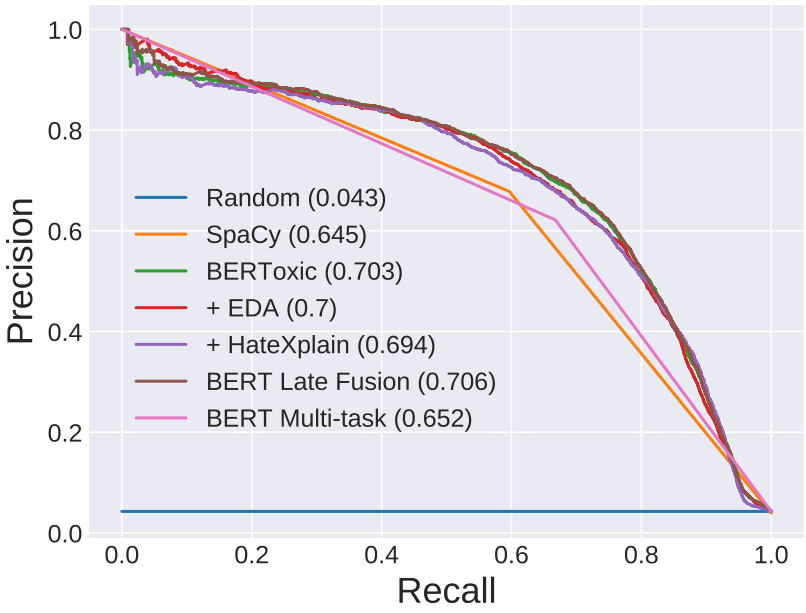

Figure 5.4: Comparison of the precision-recall curves of all the models.

Section 5.2

Error Analysis

In this section, we analyze the performance of our best performing BERToxic model. Table 5.2 on the following page highlights selected predictions that our model made on the test set. Our proposed system performed well at the toxic spans detection task, showing strength in identifying profanity and common toxic words like "idiot" and "stupid". The model identified the obfuscation of offensive words and successfully detected hate speech from such adversarial cases (Example 1).

The error analysis revealed that the system lacked nuance as it would sometimes classify toxic words used in neutral contexts (Example 2). It is also worth mentioning that there was considerable noise in the ground truth annotations. Our manual inspections concurred with the model's predictions that some words and phrases were used in offensive contexts but the annotators thought they were neutral (Example 3 and 4). Furthermore, we observed some inconsistencies in the labeling scheme as some annotations spanned entire sentences (Example 5) while others only highlighted a few words in the sentence. These issues point to the subjective nature of hate speech and the challenges involved in its fine-grained classification.

We found through our ablation studies of data augmentation that generating synthetic data using the EDA techniques did not improve the performance of the system. This suggested that the dataset size does not appear to be the limiting factor affecting the performance of BERT in this task. Using HateXplain's external dataset, we learned that different data sources and annotation guidelines can introduce noise that hurts the performance of models.

Finally, the ensemble modeling strategies we explored did not outperform the standalone BERT model. The late fusion technique performed slightly better than

the spaCy baseline, but it seemed that the sequence classifier made errors on similar parts of the input space as the token classifier. The multi-task learning approach underperformed compared to late fusion, suggesting that the sequence labeling and classification tasks are not closely related enough to benefit their joint training.

1.	Kill this *F'n W*ore* on site.
2.	.. how I am an ignorant fool ..
3.	Nazi boneheads deserve being punched.
4.	@ remoore Shut up, *racist*.
5.	*Cruz is a piece of garbage a globalist fraud*

Table 5.2: Selected examples obtained from the test set. BERToxic's predictions are shown in red while ground truth annotations are italicized.

Chapter 6

Conclusion

Section 6.1

Summary

In this work, we have proposed BERToxic, an empirically powerful system that performed fine-grained detection of hate speech to address the Toxic Spans Detection problem. We showed how fine-tuning a pre-trained BERT model with additional post-processing steps can create a high-performing hate speech classifier that outperforms strong baseline models. We found that our exploration of ensembled BERT models using the techniques of late fusion and multi-task learning did not boost performance. Similarly, we learned that our strategies for data augmentation through the EDA techniques and using an external dataset did not show any performance gain. The thorough error analysis we conducted on the predictions made by BERToxic on the test set revealed that BERT lacked nuance in understanding the use of offensive words in neutral contexts and encountered boundary detection issues when faced with noisy ground truth annotations.

Section 6.2

Limitations

There are a number of limitations that we would like to highlight in this section. First, our approach solely leverages the BERT model for hate speech detection. While BERT remains a popular transformer model, there are many other transformer-based models that could be applied to this problem. These include the BERT inspired model variants like RoBERTa [15], ALBERT [13], DistilBERT [24] etc. Furthermore, due to computational restraints, this work did not explore the use of more powerful models like XLNET [31] that have outperformed the BERT model.

Besides the choice of models, we are unable to conclusively point towards the specific ensemble modeling and data augmentation techniques that could yield performance gain over the standalone BERT model. It is likely that these techniques could yield small gains in performance, but one is then led to question if such gains are significant enough to be of practical importance.

Finally, we recognize that our specific formulation of hate speech classification as a sequence labeling task does not fully encompass the hate speech problem. While it is useful to identify the particular words and phrases in a document that is offensive, this identification does not inform us on the nature or severity of the hate and which individual or groups the hate might be directed towards. It is also worth mentioning that this work only focused on detecting hate speech for the English language in the textual domain. In reality, hate speech is multi-lingual and multi-modal in nature, thereby requiring a larger-scoped solution to fully address hate speech on online platforms.

Section 6.3

Future Work

Future avenues of work could address the limitations we highlighted above and explore other transformer-based models to compare their relative performance to BERT for the Toxic Spans Detection problem. This specific formulation of the problem could also be further expanded to categorize the specific forms of hate identified in the offensive words and phrases as well as identify if the hate is targeted towards a particular individual or group. Future work could also explore other paradigms of deep learning such as unsupervised learning, self-supervised learning and reinforcement learning for hate speech detection. These efforts will make further progress to develop more robust hate speech detectors. We hope that our findings and suggestions inspire more creative approaches towards the fine-grained detection of hate speech so that online discourse can remain healthy and inclusive for all.

Section 6.4

Broader Impact

We recognize that deep learning models exhibit bias from the data they are pre-trained and fine-tuned on. The lack of careful use of hate speech detection technologies runs the risk of reinforcing social biases. The ideas we have developed towards fine-grained detection of hate speech in this thesis serve only to assist online platforms in quickly identifying user comments that may be potentially hateful. Discussion of ethics and fairness in content moderation is essential in civic society as online platforms attempt to strike a delicate balance between the freedom of expression and the restriction of hate speech.

Bibliography

[1] Davey Alba, *How Anti-Asian Activity Online Set the Stage for Real-World Violence*, New York Times, March 2021, Retrieved from https://www.nytimes.com/2021/03/19/technology/how-anti-asian-activity-online-set-the-stage-for-real-world-violence.html.

[2] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma, *Deep Learning for Hate Speech Detection in Tweets*, Proceedings of the 26th International Conference on World Wide Web Companion (Republic and Canton of Geneva, CHE), WWW '17 Companion, International World Wide Web Conferences Steering Committee, 2017, p. 759–760.

[3] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman, *Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification*, (2019).

[4] Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov, *Fine-Grained Analysis of Propaganda in News Article*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (Hong Kong, China), Association for Computational Linguistics, November 2019, pp. 5636–5646.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 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) (Minneapolis, Minnesota), Association for Computational Linguistics, June 2019, pp. 4171–4186.

[6] Cambridge Dictionary, *Definition of Hate Speech*, Retrieved from https://dictionary.cambridge.org/us/dictionary/english/hate-speech.

[7] Karthik Dinakar, Roi Reichart, and H. Lieberman, *Modeling the Detection of Textual Cyberbullying*, The Social Mobile Web, 2011.

[8] Facebook, *Community Standards: Hate Speech*, Retrieved from https://www.facebook.com/communitystandards/hate_speech.

[9] FBI, *2019 Hate Crime Statistics*, November 2020, Retrieved from https://ucr.fbi.gov/hate-crime/2019.

[10] Paula Fortuna and Sérgio Nunes, *A Survey on Automatic Detection of Hate Speech in Text*, ACM Computing Surveys (2018).

[11] Edel Greevy and Alan F. Smeaton, *Classifying Racist Texts Using a Support Vector Machine*, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (New York, NY, USA), SIGIR '04, Association for Computing Machinery, 2004, p. 468–469.

[12] Yakoob Khan, Weicheng Ma, and Soroush Vosoughi, *Lone Pine at SemEval-2021 Task 5: Fine-Grained Detection of Hate Speech Using BERToxic*, Proceedings of the 15$^{\text{th}}$ International Workshop on Semantic Evaluation, 2021.

[13] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut, *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*, 2020.

[14] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao, *Multi-Task Deep Neural Networks for Natural Language Understanding*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Florence, Italy), Association for Computational Linguistics, July 2019, pp. 4487–4496.

[15] 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*, 2019.

[16] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee, *HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection*, arXiv preprint arXiv:2012.10289 (2020).

[17] Yashar Mehdad and Joel Tetreault, *Do Characters Abuse More Than Words?*, Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (Los Angeles), Association for Computational Linguistics, 2016, pp. 299–303.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, *Efficient Estimation of Word Representations in Vector Space*, 2013.

[19] George A. Miller, *WordNet: A Lexical Database for English*, Commun. ACM **38** (1995), no. 11, 39–41.

[20] United Nations, *United Nations Strategy and Plan of Action on Hate Speech*, May 2019, Retrieved from https://www.un.org/en/genocideprevention/hate-speech-strategy.shtml.

[21] John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos, *SemEval-2021 Task 5: Toxic Spans Detection*, Proceedings of the 15th International Workshop on Semantic Evaluation, 2021.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning, *GloVe: Global Vectors for Word Representation*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (Doha, Qatar), Association for Computational Linguistics, October 2014, pp. 1532–1543.

[23] A. S. Saksesi, M. Nasrun, and C. Setianingsih, *Analysis Text of Hate Speech Detection Using Recurrent Neural Network*, 2018 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2018, pp. 242–248.

[24] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, 2020.

[25] Anna Schmidt and Michael Wiegand, *A Survey on Hate Speech Detection using Natural Language Processing*, Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media (Valencia, Spain), Association for Computational Linguistics, 2017, pp. 1–10.

[26] M. Schuster and K. Nakajima, *Japanese and Korean Voice Search*, 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012, pp. 5149–5152.

[27] Twitter, *Hateful Conduct Policy*, Retrieved from https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is All You*

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

[29] Jason Wei and Kai Zou, *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (Hong Kong, China), Association for Computational Linguistics, November 2019, pp. 6383–6389.

[30] Ellery Wulczyn, Nithum Thain, and Lucas Dixon, *Ex Machina: Personal Attacks Seen at Scale*, Proceedings of the 26th International Conference on World Wide Web (Republic and Canton of Geneva, CHE), WWW '17, International World Wide Web Conferences Steering Committee, 2017, p. 1391–1399.

[31] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le, *XLNet: Generalized Autoregressive Pretraining for Language Understanding*, Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019.

[32] Youtube, *How does YouTube protect the community from hate and harassment?*, Retrieved from https://www.youtube.com/howyoutubeworks/our-commitments/standing-up-to-hate.

[33] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar, *Predicting the Type and Target of Offensive Posts in Social Media*, 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) (Minneapolis, Minnesota), Association for Computational Linguistics, June 2019, pp. 1415–1420.

[34] Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin, *SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020)*, Proceedings of the Fourteenth Workshop on Semantic Evaluation (Barcelona (online)), International Committee for Computational Linguistics, December 2020, pp. 1425–1447.