5-2022

Using a BERT-based Ensemble Network for Abusive Language Detection

Noah Ballinger

University of Arkansas, Fayetteville

Citation

Ballinger, N. (2022). Using a BERT-based Ensemble Network for Abusive Language Detection. *Computer Science and Computer Engineering Undergraduate Honors Theses* Retrieved from https://scholarworks.uark.edu/csceuht/108

Using a BERT-based Ensemble Network for Abusive Language Detection

A thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor's Degree in Computer Science with Honors

By

Noah Ballinger
University of Arkansas

April 2022
University of Arkansas

Abstract

Over the past two decades, online discussion has skyrocketed in scope and scale. However, so has the amount of toxicity and offensive posts on social media and other discussion sites. Despite this rise in prevalence, the ability to automatically moderate online discussion platforms has seen minimal development. Recently, though, as the capabilities of artificial intelligence (AI) continue to improve, the potential of AI-based detection of harmful internet content has become a real possibility. In the past couple years, there has been a surge in performance on tasks in the field of natural language processing, mainly due to the development of the Transformer architecture. One Google-developed Transformer-based model known as BERT has been used as a core part of many current research in the field of detecting toxic language. The methods presented in this paper propose to ensemble multiple BERT models trained on three classification tasks in order to improve capabilities of detecting abusive language in particular. This model uses sub-models using the BERT architecture trained on datasets labeled for hate speech, offensive language and abusive language. The approach presented in this paper is able to outperform the standard BERT model, and HateBERT, a re-trained variation of the BERT model used for detecting abusive language and other similar tasks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

As platforms of online discussion have risen in number, so has the amount of toxicity and profanity. Additionally, many concerns have been raised in recent years about the effect of the growing stream of negative and toxic content being presented on online discussion and social media platforms. While the percentage of toxic content varies widely from platform to platform and there aren't widespread statistics collected on major platforms, some measurements have found profanity to occur in between 10.7% and 5.2% of posts, insults to occur in between 26.8% and 14.7% of posts and directed insults to occur in between 14.3% and 3.8% of posts [5]. Despite this rise in prevalence, there has been limited growth in ability to automatically moderate online discussion platforms. Many platforms are still heavily reliant on manual moderation. When compared to the scale of internet discussions, manual moderation only can cover a small percentage of online interactions and requires a substantial time cost. However, as machine learning models continue to improve and the substantial increase of performance in modern models trained for language tasks, the potential of AI-based detection of harmful internet content has become a real possibility. Particularly challenging tasks for computers, such as detecting abusive language are finally beginning to see results. The field of abusive language detection has seen significant growth in the past couple years and is continuing to develop as new approaches are being proposed. Methods such as these offer the possibility of a healthier internet environment where users can have an overwhelming amount beneficial interactions instead of negative ones.

2 Related Work

Some attempts have already been made to challenge toxic and offensive content online. It has become increasingly likely to find simple offensive language checks, especially on kid-friendly platforms. These checks usually take the form of going through lists of prohibited words (usually consisting of common swear words or other inappropriate content that may be platform-specific). Once one of these words/phrases is detected, the post is removed or the prohibited content is censored. While this simple vetting process can act as an important filter, this method has many ways it can be bypassed and only eliminates a portion of the hateful content that users might encounter. First of all, this system isn't usually flexible when detecting variations on prohibited words. Most alterations of these words can pass undetected but retain the same meaning through methods such as misspelling, adding characters, or replacing vowels with numbers. Since there are so many ways to manipulate the spelling of any word or phrase, it becomes nearly impossible to account for all the permutations with a profanity list. As developers have been searching for ways to support safer online discussion, some have proposed to detect other variations of online toxicity instead of only offensive language. Instead of focusing on the specific inappropriate words being said, it can be even more impactful to focus on the meaning, targeting posts focused on hateful or abusive content directed toward a specific user or group of people. While being able to detect these types of harmful interactions over the internet can have even more profound effects than simply detecting offensive words, it is a much more challenging task to implement. Even with modern computing technologies, this task has seemed far from achievable. In recent years, with the rapid growth of computing capabilities that can imitate human abilities, the field of artificial intelligence, and specifically machine learning, the task of processing and working with human languages has rapidly developed as a field. Referred to as natural language processing (NLP), this field has made huge progress and has led to widely used applications such as auto complete, spell check, language translation and chat bots. (CITE) Some topics in the area of natural language processing have found success much faster than others. Unlike the applications previously listed, abusive language detection still hasn't reached the levels of accuracy in detection needed for a trustworthy content moderation method to be implemented without additional human interaction. Despite this, there has been a growing need for this type of technology alongside a giant jump in performance over the last couple years with new machine learning methods and models and this topic area shows promise to reach new levels in the near future. Many other NLP tasks can benefit from the detection of abusive language. For example, several chat bots, such as Microsoft's twitter bot Tay, became failed experiments after turning into sources of extreme toxicity when trained on unfiltered user data [6]. This is because AI models produce results directly based on the data they are trained on. This means that in order to create chat bots or other forms of text generation without offensive or abusive language, either those categories of language can't exist in the dataset or they need to be filtered through additional models that can detect them. Although the field holds a lot of potential for success, there are still many challenges, both in the specific task of detecting abuse and in the current level collaboration and coordination between research in the field.

Although offensive language and abusive language represent different categories, overlap between these classifications is common since offensive words are often used in hateful messages, so many of the issues faced with the detection of offensive language are also issues in abusive language detection (I will discuss such language categories and their relationships in detail later). By purposefully obscuring the spelling of profanities, it can be much more challenging to detect whether content is offensive, abusive or incoherent. There are also reasons that cause abusive language detection to be even more challenging than offensive language detection. This is mainly due to the fact that offensive language detection revolves mainly around detecting specific words, while abusive language detection focuses on detecting sentences or groups of sentences that are insulting messages targeted at a specific person or group of people. By needing to understand a wider scope, artificial intelligence models need to detect complex human language techniques such as sarcasm, track context between sentences and determine if a set of non-offensive words can combine to create insulting messages.

The other major challenge with abusive language detection has to do with the current state of research in the field and the lack of large, accurate, standardized datasets. Very few of the published papers in this field train or even test on the same datasets, as much of the research is separated based on language or the type of online platform data is taken from. Since abusive language detection is a relatively new field in terms of the amount of published research compared to other NLP or machine learning tasks, there is still limited comparison since it is hard to determine which models are the state-of-the-art since datasets, and the application of models, can vary widely. Implementations have been published across many different languages. Some research has focused on presents new datasets such as TOCAB, which provides the largest abusive language dataset in Chinese [7]. Others attempt to implement machine learning models that have been designed and tested on other types of machine learning tasks. There have been multiple notable publications from Indonesia including ones detecting abuse from online news comments [8] and twitter posts [9] that have attempted to test common classification methods. Both of these papers test out a variety of simple classification models including Naive Bayes and Support Vector Machines. While it is possible to compare methods such as these across languages, there have been few independent research publications that have done so. Additionally, some methods' unique approach requires more information than standard datasets and can make comparisons even more difficult. For example, French paper ”Abusive Language Detection in Online Conversations by Combining Content- and Graph-Based Features” uses the messages around the specific message being analyzed for abusive language and requires the sequence of messages to be given as part of the data when training [10]. Since comparisons between models in the field can be surprisingly challenging, it can be difficult to determine which approaches are the best. Several approaches that extend other natural language machine learning state-of-the-art models show the most potential for producing the highest results currently. In particular, models which implement the Transformer architecture, such as BERT [3]and GPT-3 [11] have found significant success in similar fields over the past few years. BERT tends to be more commonly used as a base for abusive language detection projects, mainly due to its ease of accessibility and bidirectional capabilities. Two notable implementations of the BERT model for abusive language detection are CyberBERT [12] and HateBERT [13]. This paper decides to focus more on HateBERT, as it is directly detecting abusive language whereas CyberBERT detects cyberbullying which, although may have mostly similar characteristics, may differ in definition more so than variations of abusive language detection. I will go into further detail about the architecture of transformers, BERT and HateBERT in the following sections alongside other important background information.

3 Background

3.1 Natural Language Processing

Languages that have developed naturally by humans are known as natural languages. This includes all of the most common languages spoken by humans including English, Mandarin and Hindi. When discussing computing capabilities the term natural language can aid to differentiate between the types of languages that are easily understood and commonly used by computing programs and languages that are used for communication between humans. Although the idea that computing devices could be used in order to process natural language was original proposed around the 1950's [14], it has taken a while become a prominent field. Initially the first implementations of natural language processing (NLP) originated from the need to create an automatic translation system from Russian to English following World War II [15]. However, computers were very primitive at the time, and with a 1966 ALPAC Report concluding that machine translation was nowhere near achievement, research in the area of natural language processing faded out [14]. Even as NLP research developed in the late 1980's and 1990's, it was understood that developing tools for NLP tasks was a much harder challenge than was initially theorized. Human languages are surprisingly complex and don't necessarily match logical or mathematical structures that can be easily coded using traditional methods. There are sometimes different interpretations of the same phrases, idioms without logically apparent meanings and region-specific colloquialisms. In order to for a computer to understand the linguistic structure as a human would using traditional coding techniques would require a massive amount of case statements since context is such an important attribute in the overall understanding. This is because human's don't solely learn languages through the rules that make up a language but through years of interactions to develop an understanding through experience. In recent years the field of artificial intelligence (AI) has skyrocketed. AI attempts to simulate human intelligence in machines. This directly attempts to meets the need of automating certain tasks that are generally easy for humans and hard for computers such as natural language processing and image classification. While AI is still far from achieving overall human intelligence, many NLP tasks have seen giant improvements in results with some tasks achieving human-level accuracy. There are many methods of implementing AI and I will further discuss supervised learning, neural networks, and the improved methods arising in the field later.

3.2 Abusive Language

One specific task in the field of natural language processing is abusive language to detection. In order to be able to detect and flag posts that contain abusive content, we need to first define what is considered abusive language in the first place. There are several similar terms used to refer to various types of negative language in discussions including offensive language, aggressive language, hateful language, abusive language and toxic language. Many

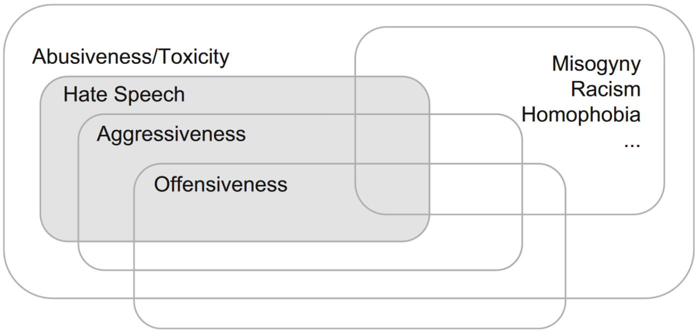


Figure 3.1: Relation between categories of abusive and offensive language [1]

of these descriptors can be confused with each other and differ in definition from person to person and paper to paper. For this discussion I will define abusive language as strongly negative or insulting messages targeted at a specific person or group of people. I will also be using the comparison between the different categories of toxicity as represented in Figure 3.1 as described in "Resources and benchmark corpora for hate speech detection: a systematic review" [1]. This representation helps to show one way of defining these categories in relation to each other. Hate speech is considered to be the most specific form of toxicity/abusiveness, with aggressive language covering a wider category and offensive language an even broader definition with the inclusion of inappropriate words or phrases not aggressively directed toward a specific person or group. One important takeaway from this diagram is the idea that offensive language is not necessarily abusive or toxic. By using this idea of abusive language, our task becomes separated from tasks that only focus on detecting profanities, meaning it is required to focus on the important context and meaning of a post instead of the individual words used. According to this definition of abusive language, the specific categories of prejudice such as racism, misogyny, homophobia and others are included. Although the amount of cyberbullying, toxicity and abusive interactions has continued to increase with the growth of total interactions that are being held on the internet [16], current methods have found limited success. The field of abusive language detection has grown rapidly with modern machine learning language models that are being developed and is expected to continue to grow until a high enough accuracy is achieved in order for implementations in online discourse platforms to be reliably helpful.

3.3 Neural Networks

One major sub-field of artificial intelligence is machine learning, which is the study of creating algorithmic models that can improve with experience through interactions with data. Due to this, having large, useful and accurate datasets is an integral aspect of most research. As we reach even more specific subset of AI and machine learning, we arrive at the field of deep learning. Deep learning focuses on machine learning algorithms that follow structures that attempt to mimic brain-like logical structures (usually indirectly). The core structure that is usually implemented in deep learning models is the neural network. The base concept is that neural networks are a set of layered nodes connected by weights in order to turn a set of inputs into a desired output. A representation of a simple neural network is displayed in Figure 3.2. While this terminology may seem confusing to those

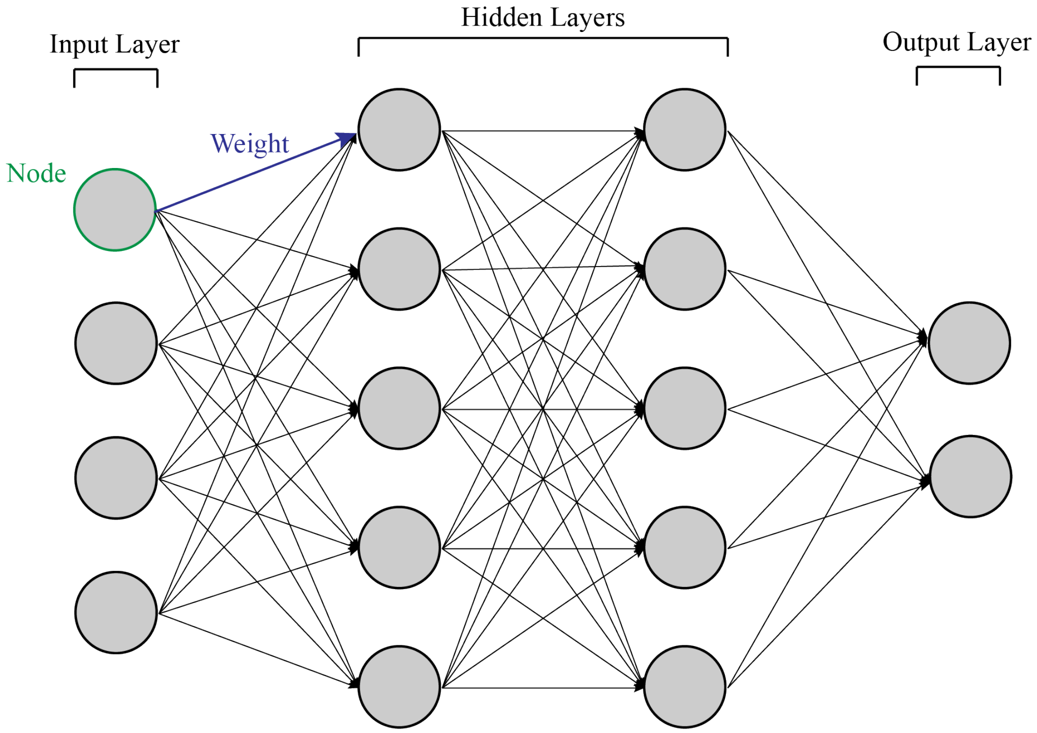


Figure 3.2: Example of a simple neural network

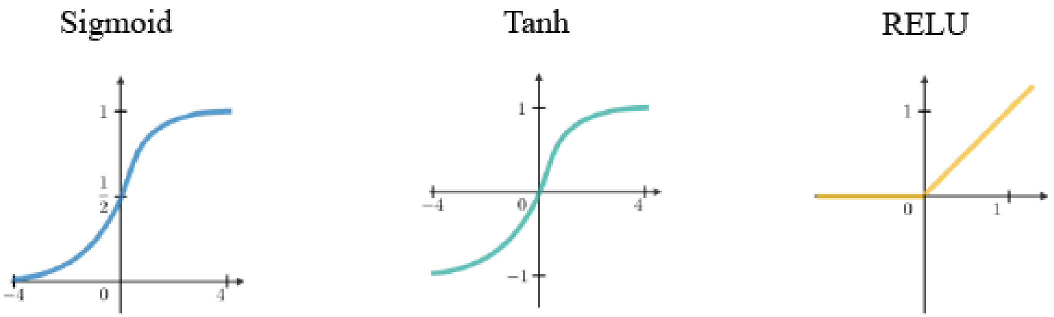
new to the field at first, the base concepts become simple when understanding that each node represents a single number and each weight (or connection between nodes) is also a single number. The network starts with an input layer (set of data stored as one number per node). Each of these nodes is connected to each of the nodes in the next layer by a weight that numerically represents the strength of a connection between nodes. Sometimes an additional parameter known as a bias is used and This second layer and following layers before reaching the output layer are considered hidden layers. In order to calculate a value for a node in the second layer, all the previous nodes are multiplied with their weights and summarized to generate a numerical value. This is continued for however many hidden layers the network is composed of until the output layer is reached. The reason this neural network structure is able to achieve good results is through a training process where the network attempts to learn what weights achieve the optimal output predictions based on the input data. This is done by having a training dataset, where inputs are loaded into the network and the predicted results are compared with the actual results. Based on if the prediction was correct or not, the network will adjust the weights in order to slowly achieve more accurate results. The output is represents the resulting label or final output object. One common task implemented by is image classification, where an image is is used as the input to a network and the output is a prediction of what is displayed in the image [17]. In order for this to be achievable, the picture and label have to represented by a layer of numbers. One way for this to be done is to use the color value of each pixel in the image. This is sometimes done by using three nodes per pixel, each representing one of the three red, green or blue values that numerically represent a specific color. One potential way to set up an output layer can be to establish a set of nodes with each node representing a potential classification label. The node with the highest output value can be considered to be the resulting label. This is a simplistic representation of how non-numerical data can be implemented in a neural network and there are many different approaches and additions the neural networks that I will discuss in further sections.

3.4 RNNs

One type of neural network architecture that is relevant to the area of abusive language detection is the recurrent neural network (RNN). The development of RNNs was a fundamental step for many more advanced architectures and is important for several key attributes. RNNs are able to handle sequence data, have a form of "memory" that allows them to keep track of context longer than traditional NNs and can handle different length inputs. All three of these attributes are important for many NLP tasks including abusive language detection. The meaning of sentences are heavily dependent on the order of the words in the sentence. This meaning is also dependent on the context established throughout the sentence and in previous sentences. Additionally, natural language isn't based on having a limited set of a certain number of words, so being able to have text of varying length as an input is necessary. Recurrent neural networks implement the concept of feeding the output of a layer back to that as an input. This allows the network to learn through a feedback loop instead of the traditional feed-forward neural networks that only use outputs as the input to the next successive layer. As one of the first architectures capable of all of these tasks, future models proved to have found ways to improve upon this base concept.

3.5 LSTMs

First published in 1997, the Long Short-Term Memory (LSTM) network improved upon the idea of the RNN with a much more developed approach to storing information and maintaining it over a longer period of time [18]. LSTMs are considerably more complicated than RNNs, with many gates interacting with a cell state that acts as the “memory” of the network. These gates allow the architecture to optionally add or remove information in the cell state.


Sigmoid

Tanh

RELU

Figure 3.3: Common activation functions and their graphical representations

In order to discuss the specific gates in detail it is important to first discuss activation functions, another important component of many neural network architectures. Activation functions define the output given a set of inputs and are commonly used to map values in a range to an altered value range. Several common activation functions are sigmoid, tanh and ReLU. Sigmoid turns an input into a value between 0 and 1 and uses the function:

$$g(z) = \frac{1}{1+e^{-z}} \tag{3.1}$$

Tanh turns an input into a value between -1 and 1 and uses the function:

$$g(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \tag{3.2}$$

ReLU uses the function:

$$g(z) = \max(0, z) \tag{3.3}$$

and turns an input into a value between 0 and positive infinity [19]. An example of the way these functions distribute the given outputs is given in Figure 3.5. The LSTM model first uses a sigmoid gate for a forget layer. This layer decides how much information to keep with a 0 output value meaning completely remove and a 1 output value mean keep completely. Next, an sigmoid input gate layer and a tanh layer decide what values to change and list a vector of new candidate values to add to the cell state, respectively. Finally, the network decides what to output with a sigmoid function connected to a tanh function.

3.6 Transformers

A new neural network architecture known as a Transformer has been developed and risen to prominence, particularly in the field of NLP, in the past few years. Transformers extend the idea of sequence altering models like LSTMs by additionally including the concepts of encoders, decoders and attention. The original Transformer structure consists of two parts: an encoder and a decoder. These parts can be broken up and for some tasks and models, it may be useful to only use one of them. The approach described in this paper only implements the encoder half of the Transformer architecture, so that will be the focus of this discussion. The encoder takes in an input sequence (this sequence is a sentence or larger group of text in NLP tasks) and maps the input into a higher dimensional space, tracking what what keywords are important for the context and meaning of the sentence. By encoding the sequence into a higher dimension, it is possible to support a more adaptable and effective way of tracking memory. However, this is very reliant on establishing a good method for determining what keywords based on the full sequence are important in order to interpret a sentence as accurately as possible. Highlighted by the title of the initial research paper proposing the transformer architecture, "Attention is all you need", the Transformer

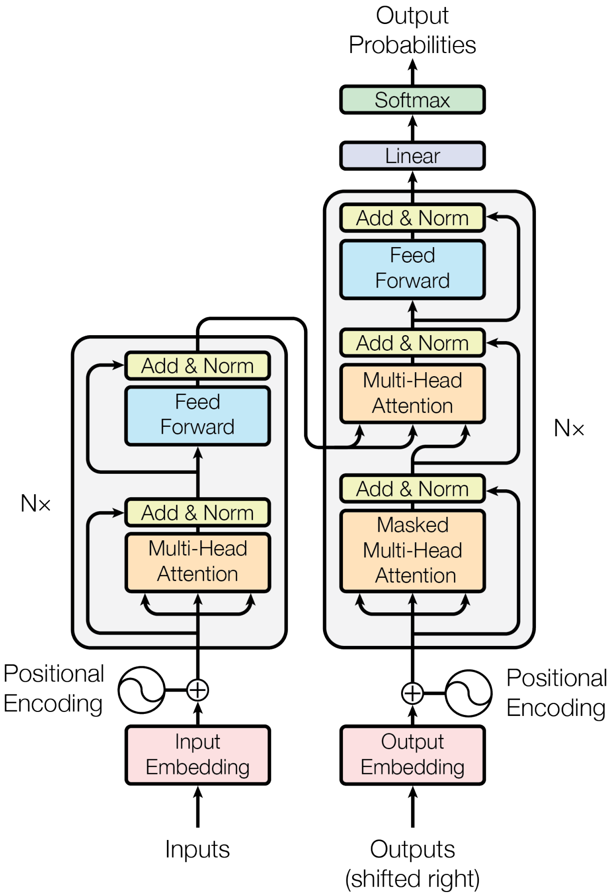


Figure 3.4: Structure of a Transformer [2]

uses multi-head attention modules [2]. As shown in Figure 3.4, these attention modules are key parts of both the encoder (the group of modules on the left side of the figure) and the decoder (the group of modules on the right side of the figure). These attention modules try to mimic the human ability to focus on the current word you are reading and all of the important previous keywords whose context dictate the interpretation up to that point in the sentence. These modules find connections between each word in the sentence with each of the other words in the sentence. By being able to figure out what previous words in the sequence are the most important for context, the Transformer model can provide a simplistic imitation of human attention.

Transformers have brought a substantial increase in performance in most NLP areas. Transformer models have been able to create the most realistic generations so far and have increased the accuracy of many NLP tasks such as sentiment analysis, language translation and question answering. Some fields such as abusive language detection have started to become feasible with this technology. However, natural languages are complex and in order for transformer models to fully learn the context of a language it needs to be trained on a massive amount of data. This requires not only a large amount of time, but also enormous computation power and the largest datasets possible. This has led to only a few major companies being able to achieve the best trained models. Google developed BERT (Bidirectional Encoder Representations from Transformers) [3], which is a core aspect of the approach in this paper and will be further discussed below, and Open-AI developed GPT-3 (Generative

Pre-trained Transformer 3) [11], which implements and pre-trains its model differently. Most research in this field that implements these methods revolves around either new applications for a model or finding new ways fine-tuning these models to improve results on specific NLP tasks.

3.7 BERT

As one of the major innovations that extend the Transformer architecture, BERT, which stands for Bidirectional Encoder Representations from Transformers, has achieved some of the best results in the field of natural language processing and is a core aspect of many models in modern research in the area [3]. The main advancement that sets BERT apart form other Transformer models is its use of bidirectional training. This means that the network can learn a word's context by examining the words on both sides of it in the sequence. This is a particularly important addition for NLP tasks since natural language is not usually unidirectional in meaning. Sometimes the interpretations of words can change based on words that appear later in a sentence. BERT is able to achieve this by using masked-language modeling (MLM) and next sentence prediction (NSP) when pre-training. As a pre-processing step during training, with MLM, 15% of the words are hidden and replaced by the masked token, [MASK]. Then, the model uses all of the non-masked words to predict the masked ones [3]. This provides an advantage over the standard method of only looking at words up to the current word in the sequence. In order to learn connections between sentences instead of only between words in a single sentence, NSP is implemented. In this training process, two sentences are provided as inputs and the model is tasked with determining if the second sentence appears directly following the first sentence in the training data or if it is at another random position. This method uses the [SEP] token to mark the end of each sentence and a [CLS] token to mark the beginning of the first sentence. By combining both of these methods, BERT is able to innovate upon the base Transformer model.

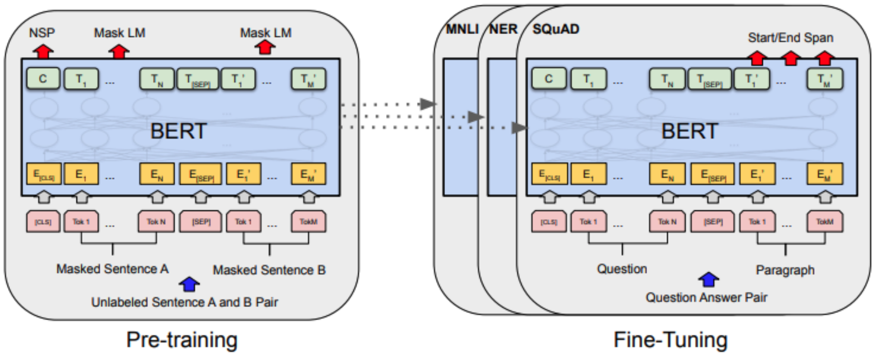


Figure 3.5: Model structure for pre-training and fine-tuning BERT models [3]

Upon BERT's initial release, it was able to achieve state of the art results for question answering on the Stanford question and answering dataset (SQuAD) and natural language inference on the Multi-Genre Natural Language Inference dataset (MultiNLI) [3]. One interesting aspect to note is that both of these datasets consist of large amounts of data with SQuAD having over 100,000 questions [20] and MultiNLI having 433,000 examples [21]. As with other Transformer-based methods, BERT is able to achieve a lot of success with the ability to train off of large datasets. BERT can also benefit from an increased number of parameters. The standard model for BERT, BERT_base consists of 110 million parameters. This is a substantial increase from most models, but pales in comparison with BERT_large, which consists of 345 million parameters [3]. Although BERT_large can generally achieve higher accuracy results, BERT_base is still the common choice since training time starts to become an issue with BERT models. Bidirectional training, the need for large datasets, and a massive amount of parameters can lead to expensive pre-training.

3.8 HateBERT

While standard pre-trained BERT models have found significant success at more general language understanding tasks, retraining BERT models on data specific to a particular domain can produce a significant advantage over base BERT models. HateBERT is one specific version of BERT model and focuses on detecting specific forms of online toxicity including offensive language, hate speech and, in particular, abusive language [13]. HateBERT doesn't modify any aspect of the BERT architecture. Instead HateBERT focuses on retraining BERT on data that is more tailored to social media platforms where most online discussion occurs. HateBERT uses the RAL-E (Reddit Abusive Language English) dataset as the basis for retraining. This dataset collected 1,478,348 messages from Reddit communities that were banned for having overtly offensive, hateful and/or abusive content. After pre-training on this large dataset, HateBERT was fine-tuned and tested for a variety of tasks for SemEval, a series of international workshops focused on various forms of semantic evaluation. Specifically, HateBERT is tested on the dataset for OffensEval 2019 [22] for offensive language detection, and two unofficial extensions of this dataset that extend the tasks by providing labels for abusive language (AbusEval) [23], and hate speech (HatEval) [23]. HateBERT outperforms BERT on all three classification tasks and achieves state-of-the-art results on the AbusEval dataset.

4 Approach

In order to detect abusive language, it can be helpful to track not only the classification labels directly related to the task, but also the data from complementary detection tasks. As discussed previously, many categories of negative or toxic language have correlations with each other. By combining models trained on other tasks related to abusive language detection such as hate speech detection (which can be considered a direct subset) or offensive language detection (which tends to be broader, but not necessarily overlap), it is possible to provide the main abusive language model with more context and information. The approach presented in this paper builds upon each of the concepts described in the background and combines multiple models. In particular, the model in this paper extends the BERT_base model structure and develops an ensemble method that combines the outputs of multiple models fine-tuned There are four key parts to the BERT architecture that are implemented in the BERT-based ensemble network presented in this paper, including the embedding layers, the encoding layers, the pooling layers and the classification layer. An overarching representation of the model implemented is illustrated in Figure 4.1. This shows how three separate models are utilized in tandem and combined in the classification layer. The full model is fine-tuned with the classifier set to receive all of the hidden layers from each of the three BERT models.

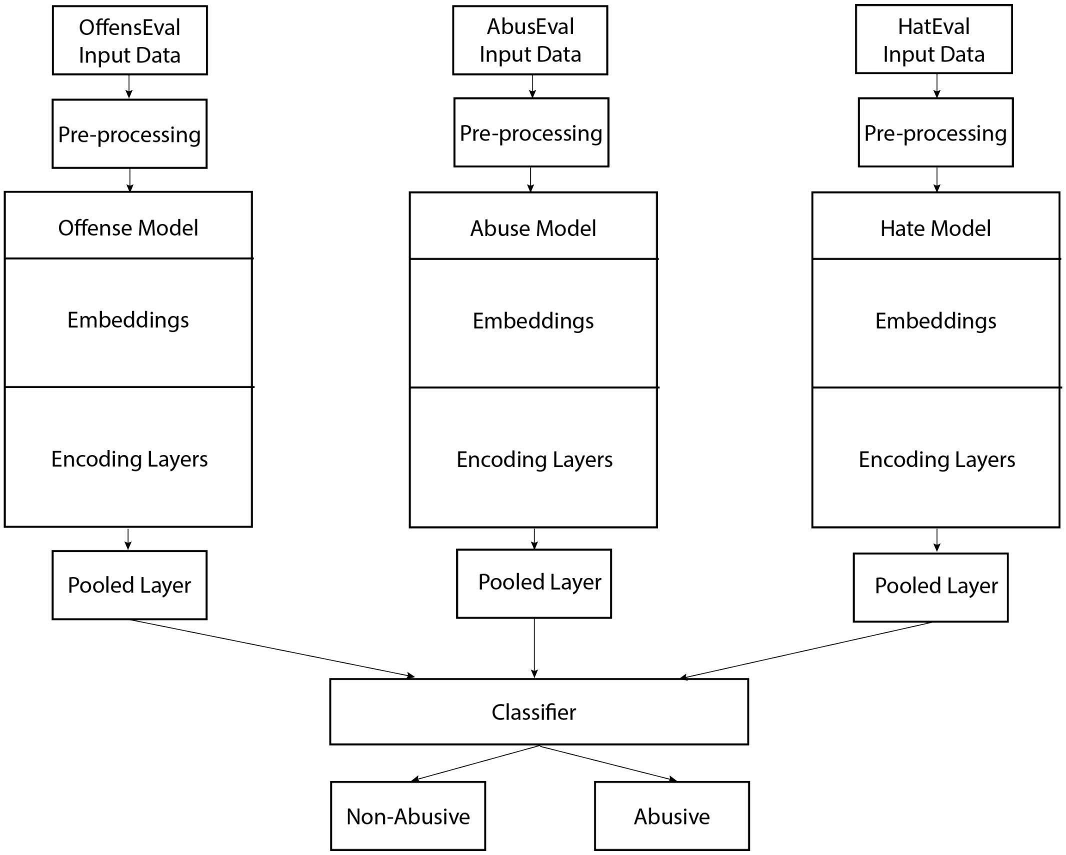


Figure 4.1: Structure of proposed BERT-based ensemble model

First of all, the input data from each of the three datasets is adjusted through a pre-processing step. Since the datasets used all are based on the same selection of tweets, this process can be replicated for each of the three model inputs instead of having to be customized for each one. The post processing methods in this paper also follow the ones implemented in HateBERT fine-tuning. Emojis are turned from their icons to a text-based tag. Any specific calls to a particular user (common on Twitter) are generalized to "@USER". The hashtag symbol (#) is removed so that the connected word doesn't need to be interpreted as a different token. URLs are also altered from being specific to a generalized tag "URL". Finally all redundant blank spaces and new lines have been removed. These steps allow the model to learn from slightly more generalized data and provides an overall cleaner dataset to work with.

Next, each of the datasets is inputted into each corresponding model and sentences are split into tokens (words or sub-words). The first group of parameters make up the embedding layer. First the tokens are used in a word_embeddings weight. This parameter consists of a (30522, 768) matrix, where 30522 is the number of words in the vocabulary dimension and 768 is the number of features used by the model when representing a word. Additionally a position_embeddings matrix is used with a dimension of (512, 768). 512 marks the maximum length of a sentence and 768 consists of the same number of features used in the word_embeddings layer. A token_type_embeddings parameter tracks similarity between two sentences. These three parameters combine as a full sentence embedding. After these three important parameters a LayerNorm step which contains a weight and a bias is implemented in order to increase the model's ability to generalize and also helps make the training process faster. After this a common machine learning method known as dropout is implemented. During a dropout function, some values are set to zero based on a dropout probability in order to reduce overfitting.

Following an embedding section comes the main bulk of the BERT model: the encoder. The encoder implemented consists of twelve BERT-layers. These layers follow the structure of a Transformer model as shown in Figure 3.4 in the background. The output of each layer is used as part of the input of the next. In addition to this, calculations are made in order to compute an attention score. Using this attention score and a value matrix, new context-aware features can be established. After this, multiple layers are used to normalize the data including a GELU activation function is used. GELU stands for Gaussian Error Linear Unit and is similar to RELU, but without being fully linear and rigid [4]. A diagram of GELU and a comparison with other Error Linear Units is shown in Figure 4.2.

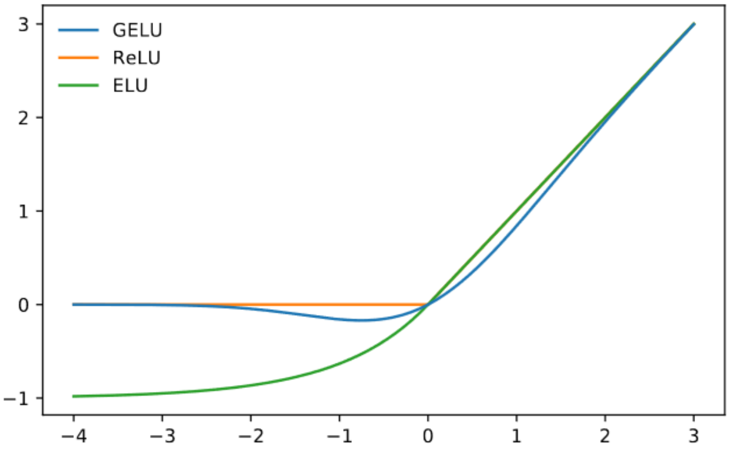


Figure 4.2: A graphical representation of common Error Linear Unit activation functions [4]

During the following pooler layer, The first token of the output of the final BERT-layer is taken and passed into a tanh activation function to get a pooled output value. This first token of the output of the BERT-layer is always the [CLS] token. This is used for the pooling since the hidden state for this particular token is used for classification tasks as the aggregate sequence representation.

Finally, the network reaches the classifier layer where the three are combined to a single classifier and fine-tuned for detecting abuse. This classifier is a basic Linear layer and takes an input size of 2304, the addition of the three 768-sized feature layers established in the embedding layer. Since abusive language detection is a binary classification task, the input of 2034 is mapped to a two value output, where 0 denotes non-abusive and 1 denotes abusive language. Two additional aspects of the training to note are that cross entropy loss is used as the loss function and AdamW is used as the optimizer.

This approach was coded in Pytorch (using TenserFlow as the backend since BERT is a google-based model) and relies on many of the BERT functions from the transformers package. Pre-trained models are extended from HateBERT and its training on the RAL-E dataset [13] to be used when fine-tuning this new model.

5 Experimentation

5.1 Datasets

As discussed in the approach a core component of the BERT-based ensemble method presented is its training on multiple related datasets. Most models in this field tend to be less robust, partially due to the wide variety of platforms, demographics, and even languages that the range of abusive language datasets cover. Due to this, the one core dataset that the ensemble model was tested on is AbusEval. AbusEval contains 14,100 tweets and contains the same tweets used as part of the OffensEval 2019 dataset, but with labels marking abuse instead [23]. Although the methodology presented in this paper was not tested on OffensEval or AbusEval, which contain the same 14,100 tweets with labels for offensive language [22] and hate speech [24] respectively, these datasets were integral parts of the ensemble model during the training process. The datasets were split into training and testing sets, with 13,240 tweets being used for training and 860 in testing. Out of the training set 2,749 out of the 13,240 tweets were labeled abusive and for the testing set 178 out of the 860 tweets were labeled abusive.

Macro-Averages			
Model	F1 Score	Precision	Recall
BERT	0.509	0.509	0.510
HateBERT	0.751	0.817	0.719
BERT Ensemble	0.779	0.880	0.737

Table 5.1: Results for the Macro-Average of each model's F1 score, precision and recall values

5.2 Results

The testing results below give a comparison between the BERT-based ensemble approach presented in this paper, a BERT for sequence classification model, and HateBERT. These models are tested on the AbusEval dataset. The multiple tests were run and the average of the results are given below.

The macro-averages take the average of each measurement for abusive and non-abusive predictions. The macro-average does not take account the proportion of each label in the dataset. The weighted averages on the other hand take into account the proportion of each label in the dataset.

The results from all three of the aggregations presented demonstrate how the approach described in this paper provides an increase in accuracy over current leading models on the AbusEval dataset. As seen from the results the change of training data from BERT to HateBERT creates a massive increase across all of the experiment values. While the

Weighted Average Results			
Model	F1 Score	Precision	Recall
BERT	0.668	0.678	0.658
HateBERT	0.848	0.851	0.861
BERT Ensemble	0.875	0.889	0.885

Table 5.2: Results for the Weighted Average of each model's F1 score, precision and recall values

Overall Accuracy Results	
Model	Accuracy
BERT	65.8%
HateBERT	86.1%
BERT Ensemble	88.5%

Table 5.3: The percentage of labels predicted correctly for each model

change in dataset makes a more dramatic difference than the approximately 2% increase in performance between HateBERT and the BERT Ensemble method, the results demonstrate that additional classification models related to, but not directly trained for abusive language classification, can potentially give an additional boost to the performance of BERT-based abuse detection models.

6 Conclusion

With the performance improvement displayed in the results comparing BERT, HateBERT and the BERT-based ensemble method proposed in this paper, several important conclusions can be made. While the performance of BERT models on abusive language is most heavily reliant on the quality of data used during the pre-training and fine-tuning portions of training, it is possible that additional related datasets that don't directly contain abusive language labels could offer additional improvements to performance. The main issue plaguing the field of abusive language detection seems to be the lack of quality and comparable datasets. Because of this, it could be challenging to replicate the same method of using datasets consisting of hate speech and offensive language labels on the same selection of tweets used in the abusive language dataset. Although similar methods could prove challenging due to this dataset limitation, there are still multiple ways the ideas presented in this paper can be extended. Also, because of the limited selection of datasets the possibility of using other similar fields such as offensive language detection that are more developed can be particularly valuable if the right complementary datasets are found given the current state of research. While this paper focused on using hate speech and offensive language models to increase the performance of abusive language detection, it is potentially possible to alter which task is implemented as the central goal. This could be examined further in additional research. The largest application for this approach is likely to extend the architecture implemented to other datasets and/or other categories of related classification tasks such as cyberbullying detection and detection of specific forms of prejudice.

Bibliography

[1] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, and V. Patti, “Resources and benchmark corpora for hate speech detection: a systematic review,” *Language Resources and Evaluation*, vol. 55, no. 2, pp. 477–523, Jun 2021. [Online]. Available: https://doi.org/10.1007/s10579-020-09502-8

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[4] D. Hendrycks and K. Gimpel, “Bridging nonlinearities and stochastic regularizers with gaussian error linear units,” *CoRR*, vol. abs/1606.08415, 2016. [Online]. Available: http://arxiv.org/abs/1606.08415

[5] S. Sood, J. Antin, and E. Churchill, “Profanity use in online communities,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 1481–1490. [Online]. Available: https://doi.org/10.1145/2207676.2208610

[6] M. J. Wolf, K. W. Miller, and F. S. Grodzinsky, “Why we should have seen that coming: comments on microsoft’s tay “experiment,” and wider implications,” *The ORBIT Journal*, vol. 1, no. 2, pp. 1–12, 2017.

[7] I. Chung and C.-J. Lin, “Tocab: A dataset for chinese abusive language processing,” in *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, 2021, pp. 445–452.

[8] D. R. Kiasati Desrul and A. Romadhony, “Abusive language detection on indonesian online news comments,” in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2019, pp. 320–325.

[9] M. O. Ibrohim and I. Budi, “A dataset and preliminaries study for abusive language detection in indonesian social media,” *Procedia Computer Science*, vol. 135, pp. 222–229, 2018, the 3rd International Conference on

Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918314583

[10] N. Cécillon, V. Labatut, R. Dufour, and G. Linarès, "Abusive language detection in online conversations by combining content- and graph-based features," *Frontiers in Big Data*, vol. 2, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdata.2019.00008

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[12] S. Paul and S. Saha, "Cyberbert: Bert for cyberbullying identification," *Multimedia Systems*, Nov 2020. [Online]. Available: https://doi.org/10.1007/s00530-020-00710-4

[13] T. Caselli, V. Basile, J. Mitrovic, and M. Granitzer, "Hatebert: Retraining BERT for abusive language detection in english," *CoRR*, vol. abs/2010.12472, 2020. [Online]. Available: https://arxiv.org/abs/2010.12472

[14] K. S. Jones, "Natural language processing: a historical review," in *Current Issues in Computational Linguistics: in Honour of Don Walker (Ed Zampolli, Calzolari and.* Kluwer, 1994.

[15] W. J. Hutchins, "The georgetown-ibm experiment demonstrated in january 1954," in *Machine Translation: From Real Users to Research*, R. E. Frederking and K. B. Taylor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 102–114.

[16] C. Nobata, J. R. Tetreault, A. O. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[17] Lorente, I. Riera, and A. Rana, "Image classification with classic and deep learning techniques," 2021. [Online]. Available: https://arxiv.org/abs/2105.04895

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 807–814.

[20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016. [Online]. Available: https://arxiv.org/abs/1606.05250

[21] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," 2017. [Online]. Available: https://arxiv.org/abs/1704.05426

[22] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval)," in *Proceedings of the 13th International Workshop on Semantic Evaluation.* Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 75–86. [Online]. Available: https://aclanthology.org/S19-2010

[23] T. Caselli, V. Basile, J. Mitrović, I. Kartoziya, and M. 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.* Marseille, France: European Language Resources Association, May 2020, pp. 6193–6202. [Online]. Available: https://aclanthology.org/2020.lrec-1.760

[24] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, "SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter," in *Proceedings of the 13th International Workshop on Semantic Evaluation.* Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 54–63. [Online]. Available: https://aclanthology.org/S19-2007