Leverage Fusion of Sentiment Features and Bert-based Approach to Improve Hate Speech Detection

Kai-Hsiang Cheng

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Chang Tien Lu, Chair

Ing Ray Chen

Jin-Hee Cho

May 5, 2022

Falls Church, Virginia

Keywords: hate speech detection, sentiment features, BERT

Leverage Fusion of Sentiment Features and Bert-based Approach to Improve Hate Speech Detection

Kai-Hsiang Cheng

(ABSTRACT)

Social media has become an important place for modern people to conveniently share and exchange their ideas and opinions. However, not all content on the social media have positive impact. Hate speech is one kind of harmful content that people use abusive speech attacking or promoting hate towards a specific group or an individual. With online hate speech on the rise these day, people have explored ways to automatically recognize the hate speech, and among the ways people have studied, the Bert-based approach is promising and thus dominates SemEval-2019 Task 6, a hate speech detection competition. In this work, the method of fusion of sentiment features and Bert-based approach is proposed. The classic Bert architecture for hate speech detection is modified to fuse with additional sentiment features, provided by an extractor pre-trained on Sentiment140. The proposed model is compared with top-3 models in SemEval-2019 Task 6 Subtask A and achieves 83.1% F1 score that better than the models in the competition. Also, to see if additional sentiment features benefit the detection of hate speech, the features are fused with three kind of deep learning architectures respectively. The results show that the models with sentiment features perform better than those models without sentiment features.

Leverage Fusion of Sentiment Features and Bert-based Approach to Improve Hate Speech Detection

Kai-Hsiang Cheng

(GENERAL AUDIENCE ABSTRACT)

Social media has become an important place for modern people to conveniently share and exchange their ideas and opinions. However, not all content on the social media have positive impact. Hate speech is one kind of harmful content that people use abusive speech attacking or promoting hate towards a specific group or an individual. With online hate speech on the rise these day, people have explored ways to automatically recognize the hate speech, and among the ways people have studied, Bert is one of promising approach for automatic hate speech recognition. Bert is a kind of deep learning model for natural language processing (NLP) that originated from Transformer developed by Google in 2017. The Bert has applied to many NLP tasks and achieved astonished results such as text classification, semantic similarity between pairs of sentences, question answering with given paragraph, and text summarization. So in this study, Bert will be adopted to learn the meaning of given text and distinguish the hate speech from tons of tweets automatically. In order to let Bert better capture hate speech, the approach in this work modifies Bert to take additional source of sentiment-related features for learning the pattern of hate speech, given that the emotion will be negative when people trying to put out abusive speech. For evaluation of the approach, our model is compared against those in SemEval-2019 Task 6, a famous hate speech detection competition, and the results show that the proposed model achieves 83.1% F1 score better than the models in the competition. Also, to see if additional sentiment features benefit the detection of hate speech, the features are fused with three different kinds of deep learning architectures respectively, and the results show that the models with sentiment features perform better than those without sentiment features.

Dedication

Dedicated to my parents.

Acknowledgments

First, I would like to thank Dr. Chang Tien Lu for his guidance and caring of my academic performance and career path.

I would also like to thank Jimmy Chen, Stephen Sun, and William Liao for their support through out the time in Virginia Tech.

I would also like to thank Tao Ran Ji for giving me advice on my thesis.

I would also like to thank Jian Feng He for helping me fix the GPU issues.

I would also like to thank Yi-ting Lee for supporting and accompanying by me no matter I am happy or struggling.

Finally, I would like to thank my family for their support. They always get my back.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Hate speech is a kind of speech or expression that denigrate a people or a specific group of people on the basis of identified attributes such as color, race, gender, ethnicity, nationality, sexual preference, religion and age etc. Typical hate speech contains defamation, statement that promote fixed and oversimplified malicious idea toward specific people, and speech deliberately stirring up hatred against a group. Hate speech can also be symbols and gestural illustration. For instance, the rising sun flag of Imperial Japan that representing human rights abuses and war crimes during World War II.

The escalation of social media such as Facebook and forums has dramatically accelerated the communication and the change of ideas between people by revolutionizing content publishing. However, it also increasingly promotes the propagation of hate speech and the hate-based events [4]. The anonymity and mobility offered by such social media also promotes the spread of hate speech that people could hide their identities and attack others with hate speech. Surveys focusing on young people in the EU show the rising of hate speech and hate crimes based on gender, sexual preference, and religious beliefs [2]. The hate speech and crime in the US is also rising. There is a hate speech story in the US that led to Ryan's death in the end. In 2003, because of the history of academic and physical struggles, Ryan's friends bullied him by making him think he had struck a friendship with popular girl in school and share more his embarrassing past. When Ryan realizing the truth, he hanged himself in the family bathroom [1]. So, recognizing hateful speech is urgent for analyzing public opinions of a group of users towards another group, and for discouraging associated hateful activities. In fact, in May 2016, the European Commission made an agreement with Big techs such as Microsoft, Facebook, YouTube, and Twitter a 'Code of conduct on countering illegal hate speech online'. During 2018, Instagram, Snapchat, and Dailymotion joined to conduct the Code. Jeuxvideo.com joined in January 2019, and TikTok claimed to be participating in the Code in 2020.

In recent years, there are more and more researchers in the social media field involving in developing automatic methods for detecting hate speech. The task of hate speech detection, it typically involves recognizing whether textual content contains hateful messages and the types of the hate speech, and the researchers usually apply Natural Language Processing (NLP) and Machine Learning (ML) techniques in the development of detection methods and achieve promising results. In addition to hate speech detection, sentiment analysis is also an important topic when people analyze the behavior and intention of users' published content. In the context of sentiment analysis, polarity and emotion recognition are two highly related tasks that researchers have been studied for a while. Polarity recognition task determines the polarity of a word, sentence, or document and measures the level of polarity in the document. Otherwise, emotion recognition focuses on classifying texts into different emotional categories in a fine-grained fashion. Messages with negative sentiment can indicate negative emotions, and positive messages contain positive emotions [27]. As well, offensive language and behavior are also related to the emotional and psychological status of the speaker.

To find a better solution for hate speech detection and accelerate the study of the field, there are a couple of competitions out there, and one of the renowned events is SemEval-2019 [45]. One could see the trend of technologies employed in the contest. Overall, 115 teams participated in the contests, and the evaluation results have shown that the outstanding systems are deep learning based models, such as CNN, RNN, CRNN, and BERT. Though deep learning models and traditional machine learning methods were adopted in the contest, many results of sub-tasks showed that deep learning based models generally perform better than traditional machine learning classifiers like SVM. BERT models that based on Transformers, which achieve state-of-the-art in many NLP tasks, dominated in most sub-tasks.

However, thought the performance of automatic hate speech detection is improving and seems promising, there is still space for the improvement. Also, many teams in the competition considered only textual information and focused on architecture design. By considering the close relation between hate speech detection and sentiment analysis, sentiment feature which could provide classifiers with additional clues for effectively recognizing hate speech is considered in this study, and a fusion network is proposed that the classic Bert architecture for hate speech detection is modified to fuse with additional sentiment features, provided by an extractor pre-trained on Sentiment140. To evaluate the performance of the proposed model, it is compared with top-3 models in SemEval-2019 Task 6 Subtask A.

The key contribution of the work is proposing the fusion network and suggesting that the fusion of sentiment features boosts the performance of the model on hate speech detection.

Chapter 2

Literature Review

This chapter is going to review the literature regarding automatic hate speech detection using deep learning and non deep learning approach. Those literature involving hate speech detection with sentiment feature is also reviewed.

2.1 Hate Speech Detection

2.1.1 Non Deep Learning Approach

Traditional methods require people to manually design the functionality of data instances and encode them into feature vectors. Then it is used directly by the classifier.

Schmidt et al. [34] sums up specific kinds of features applied in conventional techniques. Simple features like bag of words, n-grams are used as detection of hate speech [6, 7, 11, 20, 22, 36, 38, 39, 40], and as other related tasks such as detection of offensive and abusive content [8, 25, 29], discrimination [43], cyberbullying [46]. Other features include hashtags, punctuation, length of word and document [8, 11, 29]. In order to make words more generalized, low dimensional high density vector word representation that learned by clustering [38], topic modeling [42, 46] and word embedding [14, 29, 36, 43] are used. Such representations are used to represent sentences. Vocabulary resources are often used to find certain negative words in sentences, and language features take advantage of syntax information such as Part of speech

(PoS) and specific dependencies as features [6, 8, 11, 17, 46]. Meta-information pertains the data of messages, such as user gender [39, 40] or high frequency of offensive words in a user's posts [10, 42]. Besides, knowledge-based features such as messaging Mapping to knowledge-based stereotyped concepts [13] and multimodal information such as image captions and pixel features [46] are used to detect cyberbullying, but only in very limited contexts [34]. For classifiers, existing methods are primarily monitored. Among them, Support Vector Machine (SVM) is the most popular algorithm [6, 8, 11, 20, 25, 38, 42, 43], while other algorithms like Naive Bays [8, 11, 22, 25, 43], Logistic regression [11, 14, 25, 39, 40] and Random Forest [11, 42] are used too.

2.1.2 Deep Learning Approach

There is a trend that the literature regarding hate speech detection with deep learning based approach raised rapidly since 2017, and studies have also shown superior performance than classic methods on hate speech detection [21].

Different from classic ML approach, deep artificial neural networks do not directly use input features for hate speech classification, and the input of the neural networks can be various from raw text to different form of feature encoding. The neural networks learn abstract feature representations through multi-layer structure that prove to be more effective in learning than classic approach. Therefore, people adopting deep learning based approach focus more on carefully designing network architecture that automatically extracts useful and semantic features from a simple input than manually engineering feature.

Among neural networks, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term memory network (LSTM), and Bidirectional Encoder Representations from Transformers (BERT) are the most popular ones. CNN is well known for its effective feature extracting, whereas RNN is proficient in modelling orderly sequence learning tasks [30]. BERT is a powerful attention model that combine the merits of CNN and RNN. In previous studies, CNN extracts feature from the combinations of word or character, e.g. phrases, n-grams [4, 15, 31], RNN learns dependencies between word or character [4, 36]. BERT demonstrates the importance of this model by achieving state-of-the-art method in many NLP tasks [21] and dominating NLP-related competitions, e.g. SemEval-2019 [45].

2.2 Hate Speech Detection + Sentiment Analysis

Previous works have shown that sentiment features can play an big role in detecting hate speech. Malmasi and Zampieri [24] shows that combination of n-gram features and sentiment features can be effectively applied to hate speech detection by providing empirical evidence. Rodríguez et al. [33] curates a hate speech dataset from Facebook, and came up with a set of sentiment features, including negative sentiment words and sentiment symbols for hate speech detection. Vigna et al. [37] determines whether a sentence contains hate speech by adopting the sentimental value of words as the main feature. Gitari et al. [18] carries out experiments on several sentiment features and get promising results.

Chapter 3

Methodology

In the study, the proposed model, shown in figure 3.1, is evaluated on the hate speech detection task in SemEval-2019 Task 6 Subtask A [45]. The model are consist of three parts in this study.

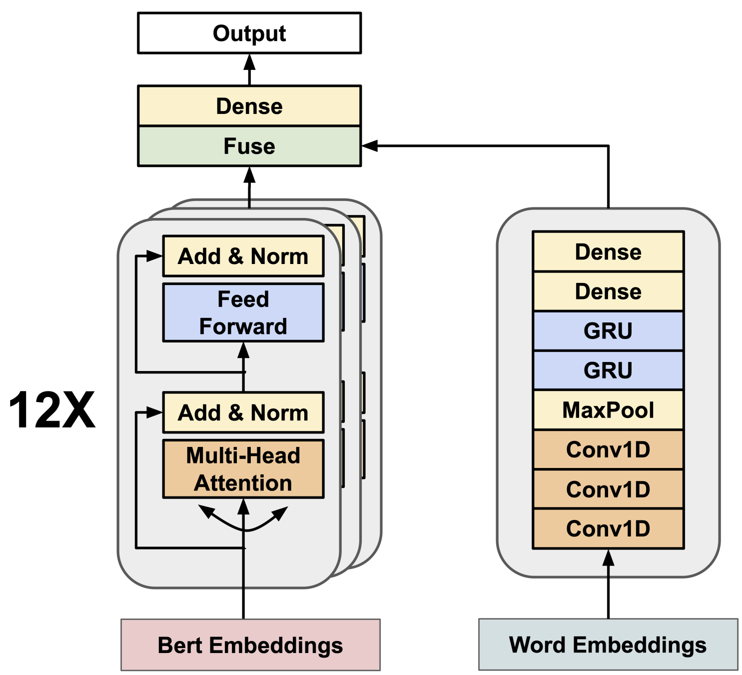

Figure 3.1: Model architecture doing hate speech detection.

The first part, on the left of figure 3.1, is BERT (Bidirectional Transformers) [12], which is a powerful language representation model, and its architecture is the encoder part of the transformer [35]. BERT is design to pre-train a model on large unlabeled-text corpuses and benefit downstream tasks with the learned semantic representations. The characteristics of BERT are the multi-head attention, a self attention mechanism that each token of an input sentence looks at each other and pays attention to those are important to it, and its ability to consider both left and right context of a token from the input sentence. Because of the BERT's architecture design, an input sentence should go through some transformations, like adding special characters, doing tokenization, adding segmentation embeddings, and adding position embeddings, before feeding it into the model. In this study, a pre-trained BERT model, which is trained on wikipedia and bookcorpus datasets, is adopted and fine-tuned on the hate speech dataset.

The second part of the model, on the right of figure 3.1, is sentiment features extractor, which uses CRNN (a combination of convolutional neural network and recurrent neural network) as the model architecture. As a sentiment features extractor, the model is first trained to do sentiment classification task on the sentiment analysis dataset. In order to feed input text into the model to do the sentiment classification, fastText[5] generates the word embeddings of the input text. When done training, the model is able to extract sentiment-related features from the input text, in addition to the ability of classifying the input text as positive or negative sentiment. In other word, the model learns to represent the input with d-dimensional vector in latent space. The trained model, therefore, can serve as the sentiment features extractor, and the sentiment features extracted by the model can provide the BERT model with additional information that boost the performance of hate speech detection.

The third part is a feature fusion layers that fuses the sentiment features and the features from the BERT model. In the layers, there is one attention layer to capture important connections between both features and pay attention to them.

The workflow of the hate speech detection in the study is that, for each input sentence, it goes through some text pre-processing steps for cleaning the input text, and the cleaned input text will be embedded into the bert embeddings and the word embeddings. With the two embeddings, they will be fed into the bert model and sentiment extracting model respectively, then be fused in the fusion layer, and output the binary classifying result indicating whether the input text is hate speech or not.

The following sections will detail the components of BERT, sentiment features extractor, corresponding input embeddings, and the fusion layers.

3.1 Bidirectional Transformers

3.1.1 Transformer Components

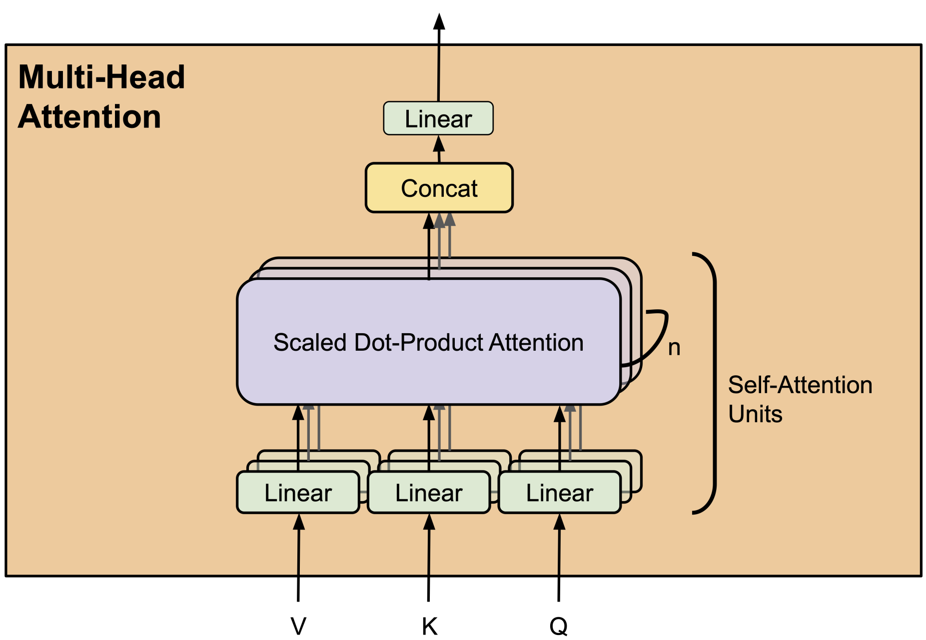

Figure 3.2: Multi-head attention and self-attention units

Self-Attention Unit

The transformer considers representation of the input as a set of key-value-query pairs. It adopts the scaled dot-product attention as figure 3.2 shown to key and query and know which part of value it should focus on. The key-value-query concept can be analogized to retrieval systems. For instance, when one search for websites on the internet, the search engine will map a query, the keywords that he/she provide with the search engine, against a set of keys, such as webpages's contents and title, in the database, then the search engine will present he/she the best matched results. To illustrate the mechanism in a formal way, let's say, for an input x, the words in x are embed into vector A as self-attention input, and the key, value, query of A^i, which is the ith element of the vector A can be calculated as follows.

$$k^i = W^k A^i$$

$$v^i = W^v A^i$$

$$q^i = W^q A^i$$

where W^k, W^v, and W^q are parameter matrices.

Therefore, the scaled dot-product attention can be obtained by:

$$Attention(Q, K, V) = softmax(QK^T/\sqrt{n})V$$

where Q, K, and V are query, key, value on the input embedding A. Because Q and K are the same inputs, this kind of attention is called self-attention.

Multi-Head Attention

Rather than only computing the attention once, the multi-head mechanism combines multiple outputs of self-attention units computed in parallel. The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions. According to the paper, "multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this."

$$MultiHead(Q, K, V) = [head_1; ...; head_h]W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, W_i^V, and W^O are parameter matrices to be learned.

Feed Forward

The feed forward layer in the BERT model is two linear layers applying position-wisely on the token embeddings. Since feed forward is applied without any communication with or inference by other token positions, it is a highly parallelizable part of the model.

According to Geva et al. [16] study, the role feed forward layers play in transformer is key-value memories. In fact, the feed-forward layers can act as persistent memory cells for self-attention layers. In their study, the key vectors in feed-forward layers can be considered as pattern detectors over the input sequence, the values supplement the keys' input patterns by inducing output distributions that concentrate on tokens likely to appear immediately after each pattern, and the output of a feed-forward layer is a composition of the previous layer's memories, which is refined via residual connections by the following Add and Norm layer. In addition, the study shows that the patterns learned by the feed forward layers are human-interpretable. The lower layers of the feed-forward tend to capture shallow patterns, while upper layers learn more semantic patterns.

Add and Norm

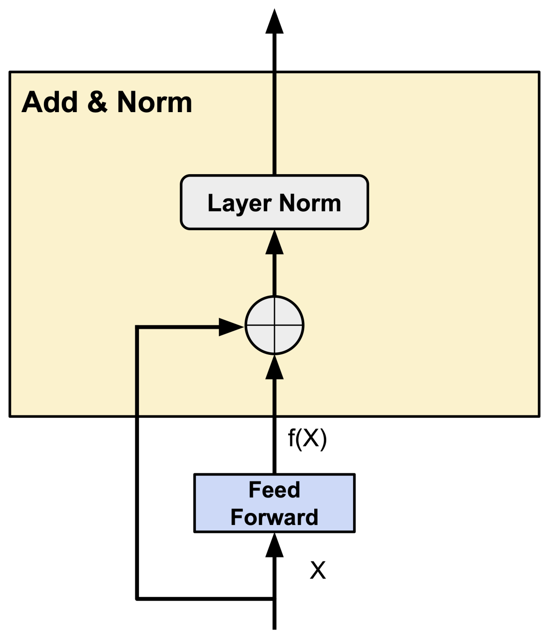

Figure 3.3: Add and norm layer of the BERT model

The Add and Norm layer in figure 3.1 is composite of two steps of computation. The first is a residual connection immediately followed by the second step, layer normalization. The residual connection sums together the previous layer's input with the previous layer's output. For example, in figure 3.3, the input of the previous layer, which is feed forward layer, is x, and the output of the feed forward layer is $f(x)$. Then, the residual connection in figure 3.3 is $x + f(x)$. After the residual connection, it is normalized with layer normalization layer. Layer normalization [3] is the same as batch normalization except that the former normalizes across the feature dimension.

3.1.2 BERT Embeddings

The input of the model is a sentence that consists of multiple words, like the rectangles in black background in figure 3.4. Note that, an input sentence always has a [CLS] token at the beginning of the sentence. Because the input sequence ultimately needs to be reduced to a single vector for classification tasks, Devlin et al. [12] choose a simple way that take the hidden state corresponding to the [CLS] token. Furthermore, the model uses [SEP] token to separate two sentences in an input, and the intention of using [SEP] token is that some of tasks, such as question answering, takes two sentences as a pair in training phase. With the input sequence prepared, the input sequence can then be represented by summing token, segment, and position embeddings that will be detailed below.

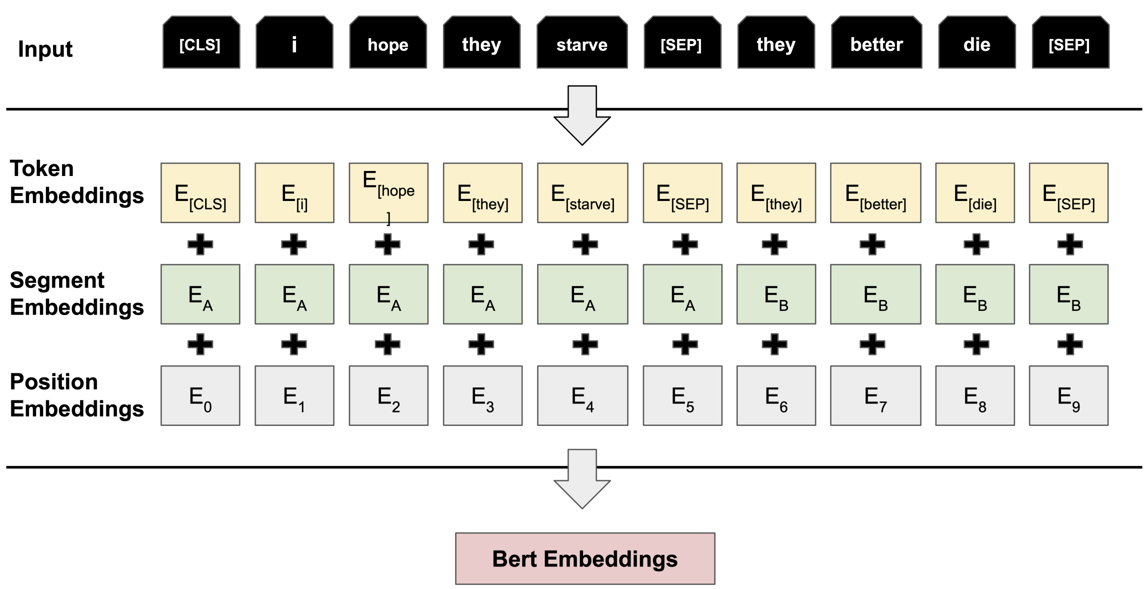

Figure 3.4: BERT embeddings

Token Embeddings

Token embeddings are the embeddings for different tokens, which is words, and it is pre-trained. In [12], it uses WordPiece [41] to tokenize words. Specifically, WordPiece will splits tokens like ”speaking” to ”speak” and “##ing” in order to cover Out-Of-Vocabulary (OOV) words and convert them into vocabulary IDs, in other word, each word in the input sentence will be represented by an positive integer. After that, WordPiece will futher project each token’s ID into d-dimensional vector representation, in the study, d is equal to 768.

Segment Embeddings

Segment embeddings are applied to distinguish sentence A and sentence B from a input sequence. Take figure 3.4 as an example, the input sequence has two sentences, which are ”i hope they starve” and ”they better die,” in one input, and in segment embeddings, each token of the sentence A will be marked as sentence A, and that of the sentence B will be marked as sentence B.

Position Embeddings

Position embeddings is a d-dimensional vector that possesses positional information regarding each token of a input sentence. It is crucial mechanism for transformer because, unlike traditional recurrent neural networks that naturally consider the order of a sequence and processes the sequence one by one, transformer processes the whole sequence in a parallel manner resulting in significant speed-up in the training time. That is to say, the transformer models itself have no sense about position/order for each token. Therefore, they need positional information to encode the sense of order of the input sequence into the models.

One solution for this may be assigning a number in range [0, 1] to each token; 0 means the first token and 1 represents the last. Nonetheless, it is hard to figure out how many tokens would be. In other words, this approach does not have consistent meaning across different input sequences.

Another solution might be linearly assigning a number to each token. In detail, the first token is assigned ”1”, the second is given ”2”, and so on. But, the issues with this method are that the largest value representing the last token could be very large when the number of the tokens in an input sequence are large, and that the model may not see a fixed sequence length which the model would not be generalized.

For an ideal approach to positional embeddings, there are some criteria:

- Distance between any two positions should be consistent across sequences with different lengths.
- It should generate an unique encoding for each token
- Models should generalize to longer input sequences with few efforts.
- Its values should be bounded.
- It must be deterministic.

By considering the criteria, Vaswani et al. [35] proposes a simple yet brilliant approach. First, the position embeddings generated by the approach are not a single number but a d-dimensional vector having information about a specific position in an input sequence. Secondly, the embeddings are not integrated into transformer models. Instead, they are used to add additional information about corresponding position in the input sequence. In other words, the input is enhanced by injecting the positional information.

The following is going to formally define how it works. Let t be the position in an input sequence, $\vec{p_t} \in \mathbb{R}^d$ be the corresponding embedding, and d be the embedding dimension. Then $f : \mathbb{N} \to \mathbb{R}^d$ is the function producing $\vec{p_t}$. The function is defined as follows:

$$\vec{p_t}^{\,i} = f(t)^i := \begin{cases} sin(\omega_k \cdot t), & \text{if } i = 2k \\ cos(\omega_k \cdot t), & \text{if } i = 2k+1 \end{cases}$$

where

$$\omega_k = \frac{1}{10000^{2k/d}}$$
$$k = \text{the dimension in the embeddings}$$

The frequencies are decreasing along the vector dimension, and it is a geometric sequence from 2π to $10000 \cdot 2\pi$ on the wavelengths

3.2 Sentiment Features Extractor

3.2.1 CRNN Architecture

In the study, a trained CRNN model is used to extract sentiment features from input text. The CRNN is composed of three one-dimensional convolution layers, max-pooling, two GRU (Gated Recurrent Unit), and two feed forward layers. For three convolutional layers, they are placed sequentially, and each is followed by batch normalization, ELU activation, max-pooling and dropout. For GRU, it is bidirectional instead of the unidirectional RNN layers because the bidirectional layers consider not only the future steps but also the future representations as well, which optimize the performance of the model.

Convolutional Layer

The convolutional layer is typically the core of a CNN (Convolutional Neural Network), and it is majority source of computations occurring in the CNN. In the convolutional layer, there is feature detector, also called filter, moving across the receptive fields of the input and finding if the feature is present or not. This kind of operation is called convolution. Take an image input as an example, the filter is a two-dimensional array of weights and can be vary in size. The filter is then applied to an area of the image and calculates the similarity between the input pixels and itself by dot product to determine the existence of the feature in this area. Afterwards, the filter moves by a stride, doing the process over and over again until the filter has gone across the whole image. The final result of the series dot products is known as a feature map. Note that, the convolutional layer is partially connected which means it is not necessary for the output value in the feature map to connect to every pixel in the input image, instead, it only needs to link to the receptive field, where the filter is being applied. In the end, the convolutional layer of a CNN model learns to extract meaningful features through out the training phase.

The design of the convolutional layer is inspired by human eyes processing visual data, so it was originally for computer vision tasks. But, in recent days, people also apply it on text recognition tasks for extracting textual features. Though convolutional layer is not naturally for textual data, it still can extract semantic features from it. For example, when doing sentiment classification task, considering the phrase below, some words containing informative sentimental information (e.g. **good**, **slow**), and others which are not important. In such phrase, convolutional layer is able to learn and match important features. Different from applying two-dimensional convolutional layer on visual data, one-dimensional convolutional layer is applied on textual data.

The pizza is really **good** here, but the service was **slow**

Recurrent Layer

A recurrent neural network (RNN) can use sequential data and achieve memorization by taking information from previous inputs to affect the current input and output. To put it another way, while traditional neural networkds assume the independence property between inputs and outputs, RNN's output depend on the prior elements among the input sequence.

Sharing parameters across layers is another characterirstic of RNNs. While feed-forward networks possess different weights for nodes, RNNs share the same parameter within layers.

In order to learn, RNNs determine gradients by leveraging backpropagation through time (BPTT) algorithm, which is specific for sequence data and is different from traditional backpropagation. The core of BPTT adjusting weights is the same as traditional backpropagation, where the models are trained by computing errors from their output and input. The difference between traditional backpropagation and BPTT is that BPTT will sum errors at each time step.

Short-term memory had been a problem to RNNs. Specifically, they have a hard time carrying information from earlier time to later time if a sequence is too long because during backpropagation, the gradients of RNNs may become extremely small and thus not contribute much learning information for the models. So, if one tries to process a long paragraph of text, RNN's may drop out important information from the beginning.

GRU, introduced by Cho et al. [9] in 2014 and used as recurrent layer in the CRNN architecture in the study, is proposed to tackle the short-term memory problem. In fact, it has internal mechanism called gates that are able to control the flow of information. The gates of GRU can learn to decide if the data in a sequence is important or not and throw those unimportant away. By doing so, it can keep relevant information all the way down to the long chain of sequences.

3.2.2 Word Embeddings

Before feeding input text into CRNN model for sentiment feature extracting, the input text will go through text pre-processing steps, which are tokenization using regular expression and lemmatization, and finally, the pre-processed text will be represented by word vectors using fastText [5].

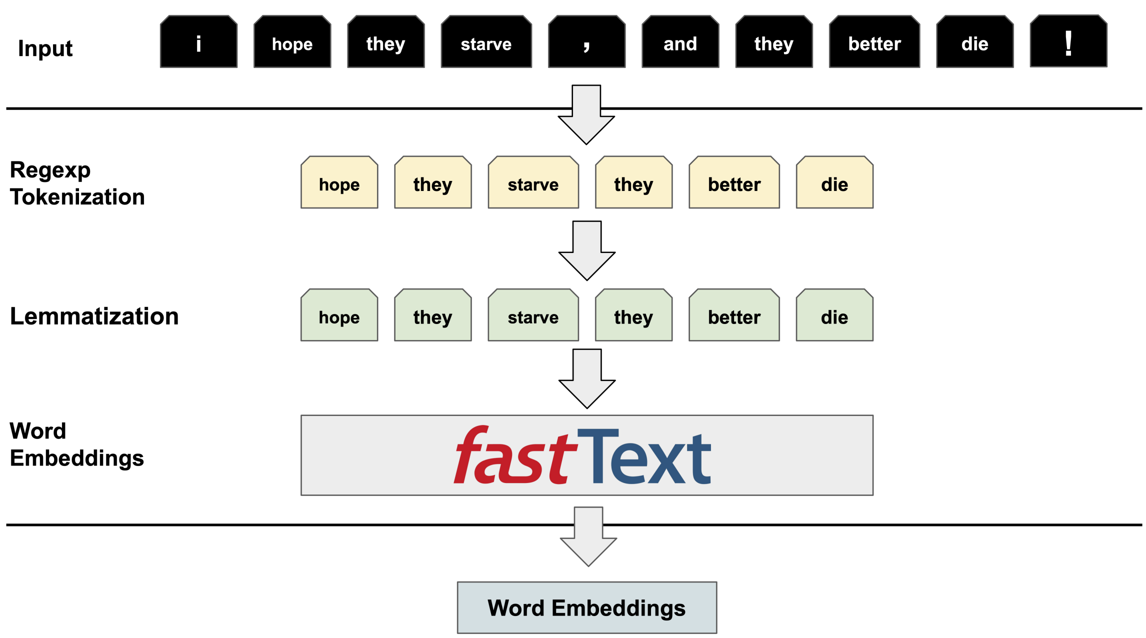

Figure 3.5: Word embeddings

Regexp Tokenization

Regexp Tokenization is a pre-processing step that use regular expression to get rid of characters except alphabets and numbers. In the study, **"[a-zA-Z0-9]\w+"** is the regular expression pattern for cleaning the input text. For example, the input in figure 3.5 is **"i hope they starve, and they better die !"**, and after the input processed by the tokenizer, the comma and exclamation are removed and the input becomes **"hope they starve they better die"**.

Lemmatization

Lemmatization is a process of grouping together the different inflected forms of a word such that they can be analyzed as a single item. Lemmatization is similar to stemming, but Lemmatization consider context of the words. So it links words with close meanings to one word. Lemmatization algorithms usually also use positional arguments as inputs, such as whether the word is an adjective, noun, or verb.

Whenever we do text preprocessing for NLP, we need both Stemming as well as Lemmatization. Sometimes one will even find articles or discussions where both words are used as synonyms even though they aren't. Usually, Lemmatization is preferred over Stemming because it is a contextual analysis of words instead of using a hard-coded rule to chop off suffixes. However, if the text documents are very long, then Lemmatization takes considerably more time which is a severe disadvantage.

Word Embeddings

Word embeddings are vectors that represent text data with numerical values, which can be operated by computers. In the study, fastText [5], which is a variation of Word2Vec [26] and an open-source library, is used to convert input text into word embeddings.

Word2Vec is a unsupervised model that creates distributed numerical representations of word features for words such that we can use those features to represent words, and the words also can be manipulated by computers doing some comparisons. To put it another way, we can use word embeddings to establish the relation of a word with another similar meaning word.

In order to create word embeddings, Word2Vec utilizes two techniques, which are Continuous Bag of Words (CBOW) and Skip-Gram. To train the model with CBOW, a word of a sentence is selected as target word, and the rest of words are served as context words, then, the model predicts the target word with context words as input. On the contrary, the mechanism of Skip-Gram is kind of the inverse version of CBOW. That is to say, the original target word is fed into the model to predict all the other words in the sentence.

Different from Word2Vec using words to predict words, fastText operates at a more granular level with character n-grams, where words are represented by the sum of the character n-gram vectors.

3.3 Feature Fusion Layers

When the input text going through BERT and CRNN respectively in figure 3.1, the end of BERT is textual feature containing hate speech information, and the end of CRNN is sentiment feature. Two features will be fused in the feature fusion layers. In the fusion layers, attentions are applied to attend useful information in both of two features.

Chapter 4

Experiments and Results

This chapter discusses the experiments, settings, and results.

4.1 Dataset

Two datasets are adopted in this study. The first one is *Offensive Language Identification Dataset*, curated by Zampieri et al. [44] for the SemEval-2019 Task 6 [45], an offensive language identifying and categorizing competition. Because SemEval-2019 is a large event that many researchers participate in and develop top-notch algorithms based on the dataset for the competition, we choose OLID to evaluate our proposed model on hate speech detection task to compare to those models in the competition. The second dataset is *Sentiment140* that is used to pre-train a sentiment extracting model. Sentiment140 is a well-curated dataset and used in many works, it is able to compare the performance of the model with other works by adopting such dataset.

4.1.1 Offensive Language Identification Dataset (OLID)

Because twitter has 330 million monthly active users, and tremendous tweet posts from diverse spectrum of users are generated everyday including hate speech, it is reasonable that Zampieri et al. [44] collected offensive text from the platform.

In order to crawl tweets containing offensive language from the twitter, the author prepared a list of keywords that the posts with those keywords often have offensive message, e.g. gun. The keywords were used to retrieve relevant posts via twitter api.

With text been collected, the author annotated the data by a fine-grained three-layer annotation scheme as table 4.1. For each instance, Level A label means the tweet contains offense words or not: **OFF** is offensive and **NOT** is not offensive; Level B stands for the category of offensive language: **TIN** means the post contains insult to an individual, a group, or others while **UNT** means that the offensive words not target any one; Level C recognizes the offended target: **IND** means the target is individual, **GRP** is group, and **OTH** is other. In addition, to ensure the labeling quality, the author only hired experienced annotators and assessed the annotators' labeling quality during their annotation.

The dataset contains 14k tweets, and 30% of them contain hate speech.

Tweet	**Level A**	**Level B**	**Level C**
gun control kids asking	OFF	TIN	OTH
kind like conservatives wanna associate everyone left communist antifa members	OFF	TIN	GRP
feel like better chasing title	NOT	-	-

Table 4.1: Sample data of OLID

4.1.2 Sentiment140

The dataset is curated by Go et al. [19], and the data source is also from twitter posts. Go et al. [19] used twitter api for retrieving tweets and took positive and negative emoticons, e.g. **":)"** or **":("**, as keywords to search emotion-relevant tweets containing emoticons. With numerous tweets collecting by twitter api, the author did the following post-processes to further clean the text and filter out unwanted characters or tweets.

1. The emoticons are removed from the text such that sentiment classifiers will not classify the tweet based on the emoticons.

2. The tweets containing both positive and negative emotions are removed in order to not mix positive and negative features together.

3. Retweets are removed because they are copying of original tweet.

4. Duplicated tweet are removed.

After post-processing, there are 1,600,000 tweets in the dataset, half are positive and the other half are negative. Some sample data of the dataset is shown in table 4.2. 0 is negative and 4 is positive.

Tweet	**Label**
Sad, sad, sad. I don't know why but I hate this feeling I wanna sleep and I still can't!	0
Damn... I don't have any chalk! MY CHALKBOARD IS USELESS	0
shenagians! my team is awesome	4

Table 4.2: Sample data of Sentiment140

4.2 Evaluation Metric

To compare our models with top three models in SemEval-2019 Task 6, we chose F1-score that primarily used to compare the performance of two classifiers. The F1-score is composed of the precision and recall of a classifier by taking their hrmonic mean.

$$F1 = \frac{2(P * R)}{P + R}$$

$$P = \text{the precision}$$

$$R = \text{the recall of the classification model}$$

4.3 Sentiment Extractor Pre-training

The proposed model is two-step-training, and the sentiment extracting model is first pre-trained withSentiment140 dataset. In the training time of classifying hate speech, the model does not update its parameters but does inference only.

4.4 Results

In order to evaluate the performance of proposed model, fusing the sentiment features with the text features, on the hate speech detection task, two experiments are conducted in the study. The first experiment is comparing the model with top-3 models, which all are BERT-based, in SemEval-2019 Task 6 Subtask A to learn how well the proposed model with additional sentimental information performs comparing to the same kind of architecture. The second experiment evaluates the adding of sentiment features across deep learning architectures.

In the first experiment, three baselines are top-3 models in SemEval-2019 Task 6 Subtask A. They are all adopting BERT architecture but with variations in the parameters and in the pre-processing steps. As the first place team, Liu et al. [23] used BERT-base-uncased model, which is pretrained on English language using a masked language modeling (MLM) objective, with default-parameters. As the second place team, Nikolov and Radivchev [28] used BERT-large-uncased model, which is double the encoder layers than the BERT-base-uncased model.

Also, the team adopted the pre-trained word vectors[32] on Twitter to encode words to a vector space. As the third place team, Zhu et al. [47] also used BERT-base-uncased model as the first place team did. Although the first and the third team utilized the same models, the difference of the model performance comes from the difference in the hyper parameters of the two models. From the table 4.3, the 83.1% F1 score of the proposed model is 0.2 point better than the top-1 BERT model in the contest. The results of the experiment shows that sentimental information could provide additional clues for the BERT model and boost the performance of the model.

Table 4.3: F1 for BERT-based models on SemEval-2019 Task 6 Subtask A

Author	**Model**	**F1**
This work	BERT + Sentiment	**0.831**
Liu et al. [23]	BERT	0.829
Nikolov and Radivchev [28]	BERT	0.815
Zhu et al. [47]	BERT	0.814

In the second experiment, three kind of deep learning architectures, which are BERT, CNN, and BiLSTM, are evaluated by comparing them with their sentiment features version. The CNN and BiLSTM are the baseline models in the SemEval-2019 Task 6 Subtask A competition, and the implementation of the models for the experiment follows Zampieri et al. [44] settings. In addition, the BERT-base-uncased model is used for BERT architecture. As a result, the table 4.4 shows that in all architectures, the models with sentiment features perform better than those without sentiment features.

In conclusion, the two experiments show that it is able to transfer the knowledge of capturing sentimental features learning from Sentiment140 dataset to the hate speech detection task and boost the performance of hate speech detection on various deep learning architectures.

Table 4.4: F1 for different architectures on SemEval-2019 Task 6 Subtask A

Model	F1
BERT + Sentiment	**0.831**
BERT	0.829
CNN + Sentiment	**0.802**
CNN	0.800
BiLSTM + Sentiment	**0.758**
BiLSTM	0.750

Chapter 5

Conclusions and Future Works

In this study, a sentiment fusion model is proposed for hate speech detection. The model consists of two branches, one is BERT for capturing hate speech contents, and the other is CRNN as a sentiment feature extractor. The model is trained with two-step fashion. The CRNN is first trained with Sentiment140 dataset to learn how to extract sentimental information. Then, the BERT is trained with OLID dataset, while the CRNN extracting sentiment features from the dataset without updating its parameters. The proposed model is comparing with top three models in SemEval-2019 Task 6 Subtask A. The results show that our model performs better than the models in the contest. In addition, the sentiment feature is added to three kind of deep learning architectures, and wee can see that the sentimental feature helps the models get better performance in hate speech detection task. Therefore, we suggest that sentiment feature can provide useful signal for the models and achieve better results.

In the future, we can further dig into the model to see how it learns the features for classification and how sentiment features benefit the model. For example, we could analyze the attention of the BERT model and the fusion layers and learn what the model is focusing on. Besides, the model is currently performed on English dataset, and we could further extend the dataset to non-English dataset, e.g. Chinese, Arabic, and Spanish. A qualitatively evaluation could also be conducted by inviting some domain experts.

Bibliography

[1] The truth behind 6 disturbing cyberbullying cases that turned into suicide stories.... URL https://www.wtps.org/cms/lib8/NJ01912980/Centricity/Domain/745/The%20truth%20behind%206%20disturbing%20cyberbullying%20cases%20that%20turned%20into%20suicide.pdf.

[2] Countering hate speech online. URL http://eeagrants.org/News/2012/.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 7 2016. doi: 10.48550/arxiv.1607.06450. URL http://arxiv.org/abs/1607.06450.

[4] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. ACM Press, 2017. doi: 10.1145/3041021.3054223. URL https://doi.org/10.1145%2F3041021.3054223.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[6] Pete Burnap and Matthew L. Williams. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242, 2015. doi: https://doi.org/10.1002/poi3.85. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/poi3.85.

[7] Peter Burnap and Matthew Leighton Williams. Us and them: identifying cyber hate on twitter across multiple protected characteristics. *Epj Data Science*, 5, 2016.

[8] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. Detecting offensive language in social media to protect adolescent online safety. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 71–80, 2012. doi: 10.1109/SocialCom-PASSAT.2012.55.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. 6 2014. doi: 10.48550/arxiv.1406.1078. URL http://arxiv.org/abs/1406.1078.

[10] Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. Improving cyberbullying detection with user context. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, page 693–696, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 9783642369728. doi: 10.1007/978-3-642-36973-5_62. URL https://doi.org/10.1007/978-3-642-36973-5_62.

[11] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language, 2017. URL https://arxiv.org/abs/1703.04009.

[12] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, 10 2018. doi: 10.48550/arxiv.1810.04805. URL https://arxiv.org/abs/1810.04805v2.

[13] Karthik Dinakar, Birago Jones, Catherine Havasi, Henry Lieberman, and Rosalind Picard. Common sense reasoning for detection, prevention, and mitigation of cyber-

bullying. *ACM Trans. Interact. Intell. Syst.*, 2(3), sep 2012. ISSN 2160-6455. doi: 10.1145/2362394.2362400. URL https://doi.org/10.1145/2362394.2362400.

[14] Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 29–30, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334730. doi: 10.1145/2740908.2742760. URL https://doi.org/10.1145/2740908.2742760.

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

[16] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. 12 2020. doi: 10.48550/arxiv.2012.14913. URL https://arxiv.org/abs/2012.14913.

[17] Njagi Dennis Gitari, Zhang Zuping, Zuping Zhang, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. In *MUE 2015*, 2015.

[18] Njagi Dennis Gitari, Zhang Zuping, Zuping Zhang, Hanyurwimfura Damien, and Jun Long. A lexicon-based approach for hate speech detection. In *MUE 2015*, 2015.

[19] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009. URL http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

[20] Edel Greevy and Alan F. Smeaton. Classifying racist texts using a support vector

machine. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, page 468–469, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138814. doi: 10.1145/1008992.1009074. URL https://doi.org/10.1145/1008992.1009074.

[21] Md Saroar Jahan and Mourad Oussalah. A systematic review of hate speech automatic detection using natural language processing, 2021. URL https://arxiv.org/abs/2106.00742.

[22] Irene Kwok and Yuzhou Wang. Locate the hate: Detecting tweets against blacks. In *AAAI*, 2013.

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

[24] Shervin Malmasi and Marcos Zampieri. Challenges in discriminating profanity from hate speech. 2018. doi: 10.48550/ARXIV.1803.05495. URL https://arxiv.org/abs/1803.05495.

[25] Yashar Mehdad and Joel Tetreault. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles, September 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-3638. URL https://aclanthology.org/W16-3638.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781.

[27] John Nezlek and Peter Kuppens. Regulating positive and negative emotions in daily life. *Journal of personality*, 76:561–80, 07 2008. doi: 10.1111/j.1467-6494.2008.00496.x.

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

[29] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 145–153, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883062. URL https://doi.org/10.1145/2872427.2883062.

[30] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016. ISSN 1424-8220. doi: 10.3390/s16010115. URL https://www.mdpi.com/1424-8220/16/1/115.

[31] Ji Ho Park and Pascale Fung. One-step and two-step classification for abusive language detection on twitter, 2017. URL https://arxiv.org/abs/1706.01206.

[32] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[33] Axel Rodríguez, Carlos Argueta, and Yi-Ling Chen. Automatic detection of hate speech on facebook using sentiment and emotion analysis. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pages 169–174, 2019. doi: 10.1109/ICAIIC.2019.8669073.

[34] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-1101. URL https://aclanthology.org/W17-1101.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[36] Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. In *ITASEC*, 2017.

[37] Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. In *ITASEC*, 2017.

[38] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://aclanthology.org/W12-2103.

[39] Zeerak Waseem. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *NLP+CSS@EMNLP*, 2016.

[40] Zeerak Waseem and Dirk Hovy. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *NAACL*, 2016.

[41] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. 9 2016. doi: 10.48550/arxiv.1609.08144. URL https://arxiv.org/abs/1609.08144.

[42] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 1980–1984, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311564. doi: 10.1145/2396761.2398556. URL https://doi.org/10.1145/2396761.2398556.

[43] Shuhan Yuan, Xintao Wu, and Yang Xiang. A two phase deep learning model for identifying discrimination from tweets. In *EDBT*, 2016.

[44] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Predicting the type and target of offensive posts in social media. 2 2019. doi: 10.48550/arxiv.1902.09666. URL http://arxiv.org/abs/1902.09666.

[45] Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). 3 2019. doi: 10.48550/arxiv.1903.08983. URL http://arxiv.org/abs/1903.08983.

[46] Haoti Zhong, Hao Li, Anna Squicciarini, Sarah Rajtmajer, Christopher Griffin, David Miller, and Cornelia Caragea. Content-driven detection of cyberbullying on the instagram social network. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 3952–3958. AAAI Press, 2016. ISBN 9781577357704.

[47] Jian Zhu, Zuoyu Tian, and Sandra Kübler. UM-IU@LING at SemEval-2019 task 6: Identifying offensive tweets using BERT and SVMs. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 788–795, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2138. URL https://aclanthology.org/S19-2138.