Hate Word Detection with Contribution Measure

CS376 Final Project Report
Team 23
권용빈, 김동근, 이재희, 최우진

1. Introduction

There are many projects and models already available online which deals with text classification. However, it is hard to find a project which detects a specific hate/swear word in a speech with the application of machine learning. Of course, simple detection of hate words is possible with pre-built dictionaries containing hate words, which checks whether the text includes any vocabularies in the dictionary. However this is not a smart approach which doesn't generalize well. Therefore we thought it would be an interesting topic for us to explore.
In this work, we propose a model that performs binary hate speech classification of a text segment, and the detection of hate words that takes into account the contribution of each word, by obtaining a quantitative measure of said contribution.

2. Related Works

a. Hate Speech Detection

Hate Speech Detection is the task of identifying words or strings that can be considered as hate speech from a given set of structured or raw text data. Hate speech is defined as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation". Hate speech detection is a type of text classification natural language processing (NLP) task. Various algorithmic and machine learning approaches can be, and has been applied to this task, ranging from simple pattern matching, support-vector machines, to transformer-based deep learning methods.

b. Language Models

Language model is the task of assigning some probability distribution over sequences of words. Previous works have shown that a pre-trained language model, trained on large corpora can be fine-tuned for most down-stream NLP tasks with some modifications. This is usually done by adding task-specific layers fine-tuned on the target dataset. These models can be applied to various text classification tasks, including hate speech detection.

c. Glove (Global Vectors for Word Representation)

GloVe. GloVe (Pennington et al., 2014) is an unsupervised learning algorithm for obtaining vector representations for words. It is a word embedding methodology that is both count-based and predictive-based.

3. Method

a. Datasets

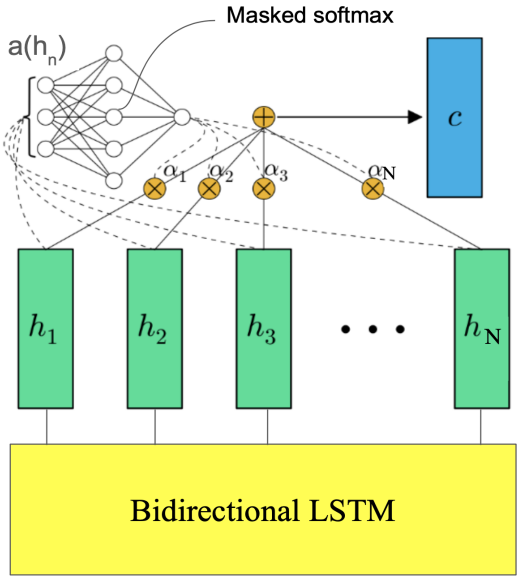
The main dataset used for this work is the Dynamically Generated Hate-Speech Dataset (retrieved from Kaggle.com), which is a dataset composed of text segments,

each labeled as "hate" or "nothate" to indicate whether they are classified as hate speech or not.
In order to fine-tune the dataset for this work, we first extracted the dataset entries that contain profanities, using the English Profanity dataset (retrieved from Github). Also, another dataset was created from the "Reddit : WallStreetBets Posts" dataset, by extracting text segments (post titles) that contain profanities.

b. Hate Speech Detection Model

In simple text classification, usually only the hidden state of the last time-step from LSTM,GRU,RNN etc is passed over to a fully connected layer. Since it is a matrix which contains condensed information about input text, it can be sufficient enough to perform simple binary classification. However, in order to achieve what we intended, we needed a representation of each word. Since the hidden state of each time-step of LSTM contains long/short term information and relationship between words, we thought they would be a reasonable representation of each word.

It is always important to build a model where the wanted value becomes a learnable parameter of the model. In the case of our project, the wanted parameter is a contribution value. In order to make the contribution value as a learnable parameter of our model, we've concatenated bidirectional LSTM & MLP. Input texts are embedded into numerical values using pre-built GLOVE embedding and embedding layers. During the embedding process, texts are padded with a 'pad' token in respect to the text with maximum length and valid length. Then embedded values and valid lengths of each text are given as input of the bidirectional LSTM. As mentioned above, the hidden-states of every time-steps are fed to the MLP layer. The process is illustrated in the diagram above.



<Fig 1. the structure of our model>

For our convenience, we will refer to the contribution value of a word as alpha, a_n. For example

Input text: 'I hate you'
Output of MLP layer: [0.2, 0.5, 0.3] (alpha of I=02,
alpha of hate = 0.5
alpha of you = 0.3)

Masked softmax function is used in order to mask the alpha values for invalid time steps which contain padding tokens. Then our MLP layer becomes a function which takes hidden states as input and outputs alpha values. Since the sum of those alpha values are 1, they are used as the weights of each word and a weighted sum represented as C in the diagram is obtained. Finally, weighted sum is fed to our final

classification layer in order to conduct binary classification. Since such alpha values are weights of each hidden state which is a representation of each word, alpha could be interpreted as a quantitative measure of each word's contribution in resulting classification, which is a similar concept as attention. We thought if we construct a model with such structure, the model will be trained in a way of outputting a higher alpha value for a hate word in a hate speech. Since there's a possibility that few words have similarly high alpha values, we've decided to output words with top 3 alpha values.

c. web service

To implement a single page web application, we used React with Typescript, which is one of the most popular web framework. After making a website, Vercel is used for hosting an application.

Flask is a python library to build a web server. We make a simple GET method REST API with Flask and attach a model code to return hate speech detection responses. Response object contains two keys, result and index, that website can show the detection result and highlight the request string's hate-detected words.

4. Experiments / Results

a. Hate Speech Classification

The binary classification performance of our hate-speech detection model is illustrated in the table below. The baseline for comparison was the Pre-Trained BERT model for classification, provided by Google. The training was done on a dataset of size 10,424. Testing was done on a dataset of size 400. From the results, we can identify that our model performs well for binary classification of hate-speech compared to the pre-trained BERT model.

Performance	**Our Model**	**Pre-Trained BERT**
Accuracy	0.7600	0.7775
F1-Score (Micro)	0.7600	0.5125

<Table1. Binary Classification Performance for Hate Speech>

b. Hate Word Detection

First we've trained the model with a non-selective entire dataset that we've got from kaggle and github. Despite the high accuracy of hate speech classification, contribution values of hate words weren't significantly as high as we expected. In fact most of the words had similar alpha values.

For example:

tetxt1: Gays shxts have no rights [0.212, 0.205, 0.204] ($a_{top} \approx ⅕ = 0.2$)

text: 2Black people should be fxxcked [0.216, 0.211, 0.209] ($a_{top} \approx ⅕ = 0.2$)

If the detection was performed well, for text1, alpha value of gay& shxts, and for text 2, fxxked should be considerable higher than others. However as shown above, each

alpha value is almost equally distributed. This means that the model is not capable of calculating alpha values in a manner that we wanted to. After thinking about the cause of the result, we came to the conclusion that the high ratio of non-hate speech containing swear words was leading our model to output such contribution values.

For example,

Non-hate: Damn, this shxx tastes so fxxking good

Hate: You are fxxking lame piece of shxx

Even though the text is classified as non-hate speech, it contains swear words. Despite the existence of swear words, the context of the sentence is not offensive but rather a more aggressive way of expressing satisfaction. Of course these kinds of data would be meaningful data for a model which focuses on classifying tricky speeches by understanding the context of the sentence deeply, but it didn't match with our model's purpose.

Therefore, in order to provide quality data that could lead our model to be trained in the intended direction, we've fine-selected data from the original set. We've reduced the ratio of non-hate speech containing swear words while increasing the ratio of hate speech containing swear words. But the total data size was shrunk to 1/10th. As we expected, the model has successfully outputted the high alpha for swear words in hate speech. Here's some example
Text1: No one likes fxxking rude black chicks [0.190, 0.187, 0.183]

(a_{top} > 1/7 =0.143)

Text2: black ladies don't understand shxt [0.330, 0.199, 0.172]

(a_{top} > 1/5 =0.2)

Text3: No one likes fxxking rude whxres [0.225, 0.217, 0.215]

(a_{top} > 1/6 =0.167)

However the performance on classification was done extremely poorly. Due to lack of data, generalization wasn't achieved.

Training data	**Large, non-selective**	**Small, selective**
Classification accuracy	0.76	0.38
Most frequently detected words (if hate speech)	['are', women', 'be']	['fxxcking', 'shxt', 'retarded']

<Table 2. Comparison of Hate Word Detection Performance>

The table shows the comparison of results between two data sets. For a non-selective large dataset, despite the high accuracy, the most frequently detected words weren't the hate words we expected. However for the model trained with selective data, despite the low classification rate, swear words were most frequently detected.

c. web-service side

There are two trouble shootings in implementing applications. We tried to host the inference server and website so that peer students can use our service. However, to host an inference server, cloud needs at least 1.8GB memory and many free hosting services do not provide enough memory. (usually 500MB for free plan) Also, the building process of the hosting server depends on the specific python virtual environment. Therefore, we need several experiments to run the inference server. As a result, we run the website and server in a local environment. We used two local ports, web and server, for sending a request and receiving a detection response.

5. Discussion / Further improvements

Replacing bidirectional LSTM with BERT could result in better performance. Since BERT is a model which is built by stacking multiple encoder-parts of transformers which involves self-attention, it would be able to output a better representation of each word in a text by considering the context of the text more deeply and preserving the relationship between every word. Also we should train the model with larger dataset in a more selective manner. Despite the same structure of model, the way it performs is significantly dependent on the quality and quantity of data and it was proven with our results. Finally, we could possibly introduce the concept of stop word to the model. Stop words are words like I, are and the, which have relatively less importance in the context. If we come up with a way to train the model in a manner that it considers stop words less in calculating contribution measures, the model will be capable of performing better detection of hate words.

6. Contribution

a. Yongbin Kwon, Donggeun Kim, Woojin Choi
Data Gathering, Model Design and Implementation, Coding for classification model and baseline (Pre-Trained BERT), Experiments and Evaluation
https://github.com/joseph1723/CS376_Final_Project

b. Jaehee Lee
Website & Server Framework Implementation
: Implementing website wireframe, UI design, server API connection and hosting application service.
https://github.com/jaehuiui/cs376_web https://github.com/jaehuiui/cs376-server (public until 6/30)

7. Reference

[1] Malik, Jitendra Singh, Guansong Pang, and Anton van den Hengel. "Deep Learning for Hate Speech Detection: A Comparative Study." arXiv preprint arXiv:2202.09517 (2022).

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota. Association for Computational Linguistics.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. 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. Association for Computational Linguistics.

[4] surge-ai. (2021). profanity
Retrieved from https://github.com/surge-ai/profanity/blob/main/profanity_en.csv

[5] TENSOR GIRL. (2021). Dynamically Generated Hate Speech Dataset
Retrieved from
https://www.kaggle.com/datasets/usharengaraju/dynamically-generated-hate-speech-dataset

[6] Gabriel Preda. (2021). Reddit WallStreetBets Posts Dataset Retrieved from
https://www.kaggle.com/datasets/gpreda/reddit-wallstreetsbets-posts

[7] Jurafsky, Dan; Martin, James H. (2021). "N-gram Language Models". Speech and Language Processing (3rd ed.). Retrieved 24 May 2022.

[8] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.