Final Project Milestone Report

Team 23
20160502 : 이재희 **/ 20160791 :** 권용빈
20180679 : 최우진 **/ 20190074 :** 김동근

1. Motivation

Hate speech detection is one of the major research topics in Natural Language Processing. It is important now more than ever, due to the enormous influx of content on the internet. In this project, we aim to propose a deep learning model for a specific case of hate speech detection, which is intentionally obscured text. By intentionally obscured, we mean cases such as spelling errors or random characters inserted in order to disguise the hateful words or sentences. We also propose a simple web-interface as a proof-of-concept for this model. With this project, we want to contribute to online content moderation for social media and other platforms.

2. Method

2-1. Dataset

Our Dataset will be texts which contain profanities, or can be labeled as hate speech. As this project's goal is to find hate speech text with spelling errors or other alterations, we will be using datasets which consist mostly of those texts.

2-2. Model

In order to classify whether the source text is hate speech or not, we are planning to divide the structure of our model into 2 main layers. Since people can intentionally make alterations to swear words in order to bypass the hate speech filtering, we will first detect any possible (if it exists) in a given source sentence and create a recovered version of it. The restored sentence will then be classified. Here is a detailed explanation of the structure of our model below.

Layer 1: Typo correction
In order to perform typo correction, each data will be changed into word units from sentence units. Then each word with a typo will be labeled as the correct word.
Ex) Original text = I fucking hate korea
Augumented text = I fxcking hate Korea, I fu1cking hate Korea
(Input, label) = ([I, fxcking, hate, Korea], [I, fucking, hate , Korea])
([I, fu1ckng, hate, Korea], [I, fucking, hate , Korea])
CNN will be used to perform typo correction of such inputs. Network will be trained in order to classify augmented words to the original word. Corrected version of the augmented data will be obtained by reconstructing a sentence based on the output of CNN.

Layer 2: Hate-speech classification
Reconstructed sentences will be then given as an input of the BERT model in order to perform binary classification(hate-speech or not). Each data will be labeled as hate if it is a hate-speech and nothate otherwise.
Ex) (Input, label) = ([I fucking hate Korea], [hate])
([I am enjoying the final project], [nothate])

2-3. Data Augmentation

Data augmentation is the method of altering existing data in order to provide more data for training. For this task, data augmentation is important as there are no large-scale datasets that are labeled and include text with obscuration such as spelling errors. We will be using data augmentation to alter existing text data in various ways. These include inserting, removing, replacing some characters, adding whitespaces or symbols, and more.

2-4. Web Application

We will implement a single page web application with Typescript and React on client side, and backend API server with Flask framework with Python.

3. Preliminary Experiments

3-1. Pytorch Pre-trained BERT

BERT (Bidirectional Encoder Representations from Transformers), is a transformer-based machine learning technique for natural language processing pre-training developed by Google.

3-2. Train Data

For data, we used a hate speech-labeled text dataset, which is composed of more than 40,000 english text including social media posts, each labeled as “hate” or “not hate” as indication of hate speech. As this dataset is crawled randomly and not specifically containing obscured text, we decided to use data augmentation to provide the training data.

We used “nlpaug”, which is a python package that offers various tools for text data augmentation. By using the provided spelling-error augmenter, we augmented the text adding random spelling errors. For the preliminary experiments, we used a batch of 2,000 (1,000 of which are augmented) for training, and 200 (100 augmented) for test data.

3-3. Model

For the preliminary experiment, we’ve only implemented a mild version of Layer 2 of our final model. We used Pytorch pre-trained BERT for sequence classification, which is an implementation of BERT provided in Pytorch. Our preliminary model consists of 2 layers. First, we used BERT tokenizer to tokenize our text data. Then, we feed the data into the classification layer to get the output label. Adam is used as the optimizer.

3-4. Evaluation

For the measure of accuracy, we are using the value of
{(total correct labels) / (total labels)}.

We ran the experiment twice, for both the original dataset and the augmented dataset. This is to compare the current performance of classification between normal and obscured text.

3-5. Analysis

The accuracy we got was 0.78 for the original dataset, and 0.70 for the augmented data. We can see that by adding obscuration such as spelling errors, the performance of classification is lowered.

3-6 Application

The web application implementation is based on React framework, typescript. Fundamental configurations for building the project, basic web structure and styling components are in progress. We are also trying to host the web server with github

pages modules, and researching backend server with python to execute AI model and transfer data with REST API to client side .

4. Next Steps

4-1. Data gathering & augmentation

Instead of merely inserting random characters, to make a more meaningful dataset, we will use other methods such as replace, delete, replace, whitespace, split, etc. Also if possible we will try to gather what people actually use in a chat that has filtering, like game chats.

4-2. Model

In the preliminary experiment, we used pre-trained BERT. However, in the final experiment, we may use an untrained model and train it ourselves if we think it will give better accuracy and performance. In this case, we will find a way to train our own model.
We will add some layers or architecture to find spelling-errors and correct them.

4-3. Evaluation

We will quantify the quality of our model by calculating the accuracy of the model's prediction of the test set and applying F1 score on it. In order to improve the model's performance and obtain better results, we will perform multiple training involving hyperparameter tuning (learning rate, batch-size, hidden dimension , etc..), optimizer changes and data. Ratio of augmented text and original text will be altered in order to see the effect on accuracyA model with the highest F1 score will be used.

4-4. Application

We are planning to implement a website interface and GUI, so that users can play with our hate speech classifying model by typing their own words and checking whether our model classifies it correctly or not. API server will be implemented with Flask framework. Further research on cloud services such as AWS EC2 for running servers should be conducted and GPU will be checked whether it is suitable to run our model.

5. Contributions

Woojin Choi, Yongbin Kwon, Donggeun Kim : Data gathering. Implementation of the classification model for preliminary experiments using Python / Pytorch. Data augmentation using nlpaug-library. Running and evaluation of preliminary experiments.

Jaehee Lee : Working on Front-end & Back-end side. Implement a web application for executing trained models. Also Implement server based on python, transfer data with API to client side.

6. Reference

Mozafari, Marzieh, Reza Farahbakhsh, and Noel Crespi. "A BERT-based transfer learning approach for hate speech detection in online social media." *International Conference on Complex Networks and Their Applications*. Springer, Cham, 2019.

Hu, Yifei, et al. "Misspelling correction with pre-trained contextual language model." *2020 IEEE 19th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 2020.

surge-ai. (2021). *profanity*
Retreived from https://github.com/surge-ai/profanity/blob/main/profanity_en.csv

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