CSC413 Final Project

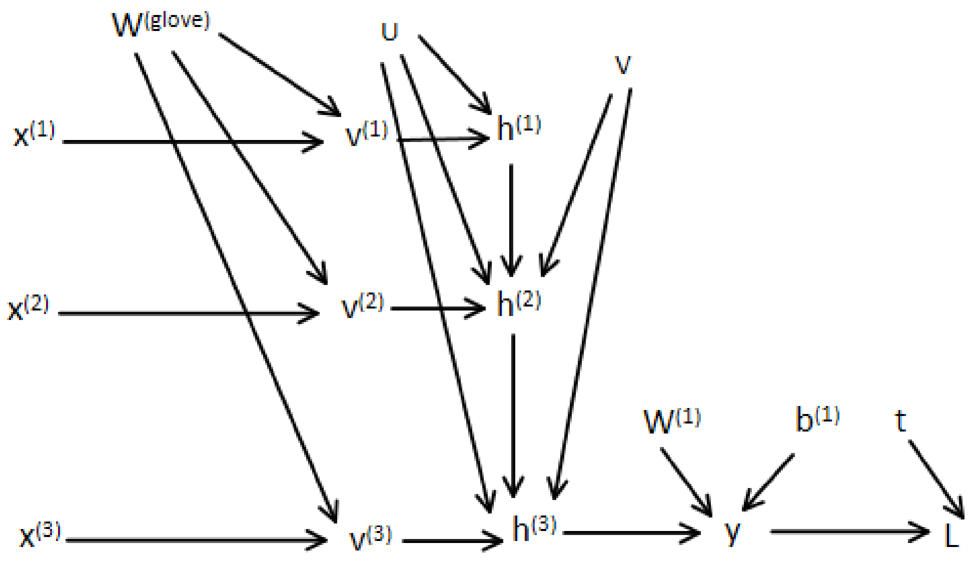
Group Members: Dhiren Tulsiani, Fareeha Fatima, Ruopeng Liu

Introduction

Our main task for this project is to perform binary classification on social media comments to classify them as hate speech. The model takes as input a text sequence that is represented as indices based on a vocabulary derived from the GloVe embeddings. The model uses this sequence to generate a class label of 0 or 1 where 0 classifies the sequence as not hate speech and 1 classifies the sequence as hate speech.

Since the task is sequence classification, we used an RNN based model. The model is analyzing text, so GloVe embeddings are used so our model can benefit from the pre-trained clustering of semantic meanings. The embedding vectors were also trained to fine-tune them to the dataset. After converting the sequence into its embedding the sequence is fed to the RNN layer, specifically a GRU. Finally, the last hidden unit output from the RNN is put through a tanh activation and inputted to a Linear layer for the binary classification.

Model Description



To demonstrate the model architecture we can use the above diagram that shows how a sequence of three tokens is processed. First the tokens are converted to embeddings using the glove vectors of size 50. Then, the embeddings are inputted one by one to the

GRU unit. Each token embedding is multiplied with the same weight matrix, U of size 50x32 and the hidden states feed into each other by multiplying with the weight matrix V of size 32x32. The final hidden state is fed to the linear layer where it is multiplied by another weight matrix $W^{(1)}$ of size 32x2 and summed to a bias vector $b^{(1)}$ of size 2x1 to produce the output classification. Note that since the GRU has multiple hidden units, the portion of the diagram representing the RNN is repeated for each hidden unit and the output of all the hidden units is what is given to the linear layer.

Running the model on the test dataset below we have an example sequence that was correctly classified as hate speech:

```
Cowards Saudis knows very well that Iran is way too powerful and will
roasts them like a chicken. They're playing under the USA dog's agenda and
will get the entire middle East region under fire. Illitertates Saudis,
```

An example that was not correctly classified as hate speech by our model is:

```
FGM a vile third world problem, brought to our shores by foreigners, is
practised by minorities This cannot be taught to all primary school
children. This would be defiling their education and childhood. It's
unacceptable  Target only those at risk.They're easily identifiable URL
```

Data

The data set used is from UC Berkley's data lab. It is named: "Measuring Hate Speech", and is freely available on Hugging Face, to be used with citation. It consists of 39565 unique comments from a selection of social media websites. Words in non-english languages do show up occasionally, though the data set is almost completely in English. For each comment, different annotators assign values to labels like: ‘violence’, and ‘insult’. Then the values of the labels from all the individual annotators will be used to provide a single ‘hate speech score’ for each unique comment. This score is centered at 0, if the value is greater than 0 then the comment is considered as hate speech, and not considered as hate speech otherwise. The greater the magnitude of the hate speech score, the greater the harshness or benevolence of the comment. Our model takes as the input a transformed version of a unique comment, and our target is whether the comment is classified as hate speech or not; 1 for hate speech, 0 for not hate speech.

We transformed the original comment in order to provide an input our model can plug into a glove embedding. The embedding we used was imported from the TorchText library, specifically: torchtext.vocab.GloVe. The glove embedding only considers words written in lower case. Every punctuation mark is treated as a separate word in the glove embedding. For any English word with a punctuation mark inside it, e.g “they're”, the glove embedding treats it as 3 unique words: “they”, “'”, “re”. Each comment in our dataset must be split into words that we can input into the glove embedding. Next we built a custom vocabulary from the vocabulary of the GloVe embedding. The custom vocabulary contains two additional tokens, “<unk>” and “<pad>”. The “<unk>” token is used as a default for any words in the sequences that do not exist in the GloVe vocabulary. The “<pad>” token is used to pad the variable length sequences so that the model can accept batches to process. Using the custom vocabulary, the tokenized sequences are converted to indices. Finally we input the list of indexes and whether the original comment was labelled as hate speech into our model.

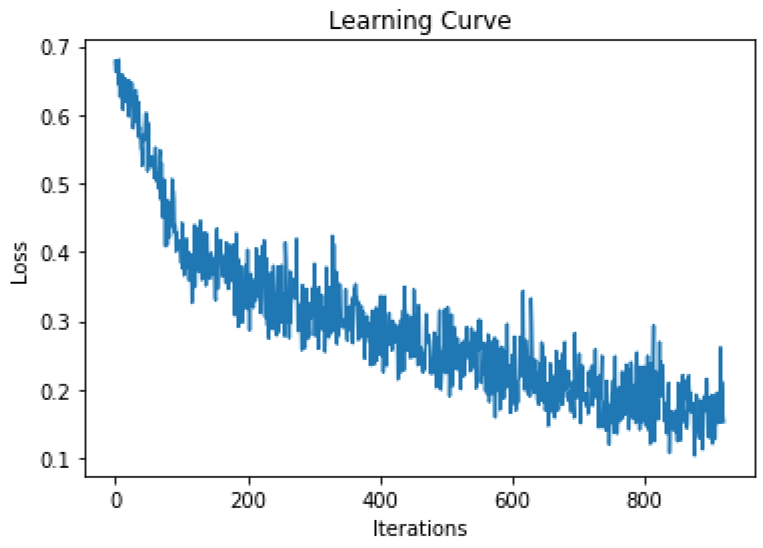
Let's look at some summary statistics about our dataset. Our dataset consists of 39565 comments, 14321 are labelled as hate speech. Which is approximately 36%. In real life, the percentage of comments that are hate speech may be less than 36%, our dataset needs to include more examples of hate speech so that our model has enough examples to learn from.

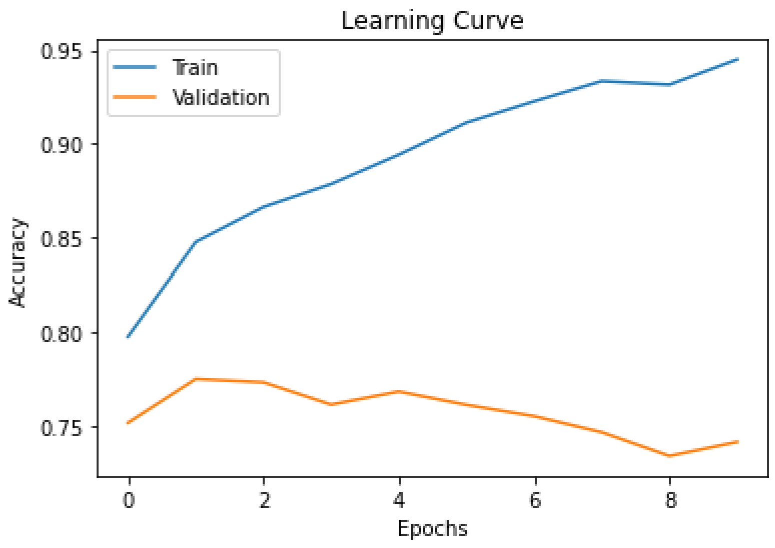
We talked about how each comment was split into words earlier. Based on this style of splitting, there are a total of 45779 unique words in the data set. 28583 of these words are actually represented in the glove embedding, which is about 62.4%. The majority of the discrepancy is due to spelling errors, or other minor human errors, like not adding a space between two words.

The average length of a comment in our dataset was 30.36 words, with the shortest comment having 1 word, the longest having 136 words. The most common words are common English words like “the” or some punctuation marks like a comma. The least common words are ones with spelling mistakes or other minor errors.

Our training set contains 60% of the unique comments in our dataset, and the validation and test sets both contain 20%. We made sure that each of the training, validation, and test sets contain the original ratio of comments labelled as hate speech versus not hate speech. This is to ensure that the training or testing of our model is not skewed towards one class. The data set did not provide the exact date of when each comment was posted, so it is not considered in our split.

Training Curves





Hyperparameters

When tuning the hyperparameters, we quickly noticed that once our model achieved a training accuracy of over 85%, the validation accuracy started to reduce. Models with large hidden layer sizes, and multiple hidden layers achieved a high training accuracy (over 85%) often in just the first epoch. As a result, we lowered the number of parameters in our model. We finally chose a model with 1 hidden layer, having a size of 32, as it led to the highest validation accuracy with a relatively quick training time.

Similarly, we chose a learning rate of 0.001 because when it was higher, the model overfit very quickly, leading to a lower validation accuracy. In the same vein, 3 epochs was how long it took before the training accuracy became too high, and the validation accuracy started reducing. Thus we used early stopping to use the model weights generated at epoch 3 for our final model.

Another parameter we needed to tune was the size of each batch we used in training. We achieved the highest validation accuracy when our batch size was either 256 or 32. The validation accuracy was often significantly lower when the batch size was in between the two numbers. We finally decided to choose a batch size of 256. This is because, when we had a batch size of 32, though the validation accuracy was consistent, the loss at each iteration had a very high variance. Whereas it had a low variance when the batch size was 256.

Results

The model achieved a test accuracy of around 76.8%, which is our quantitative result.

The test accuracy is not too bad when comparing the error rate to benchmarks on the Dynabench site (https://dynabench.org/tasks/hs). However, looking at the training curve we can see that the model did not significantly improve its validation accuracy over the course of training. It was quite difficult to train and tended to overfit easily, despite lowering the hidden units in the layer or adding dropout. This is likely due to there not being enough data to train with. The model may have needed more examples to be able to better generalize and improve its accuracy.

Another problem could be that the data was not processed well enough before training. Misspellings and other out of vocabulary words would all have been mapped to the same embedding vector which results in the loss of information during training.

Hate speech detection is also quite inherently difficult. Sometimes a statement could be rude, or offensive, or contain harsh language, but not be considered hate speech. A good example of these kinds of sentences are given in the ethical considerations portion below. If you recall, when talking about the data, we mentioned that each comment was given to multiple different humans to give labels to the comment and also classify it as hate-speech. Sometimes, the humans classified a comment (into hate speech or not hate speech) differently. So even humans disagree on whether statements are hate speech or not. This further supports our proposition that hate speech detection is by nature hard to detect.

Given that so many words were not in the glove embedding (37.6%), and that we believe detecting hate speech is a difficult task, our model performs quite well. This indicates that the glove embedding is well made, and that we chose a good model for the task. As mentioned earlier, in order to improve our model so that it could be effectively used in a real life situation, we would need more data, and a better way to deal with words not in the glove embedding (especially the spelling mistakes).

Limitations and Ethical Considerations

Due to issues like sexism, racism, and such social problems that exist in modern society, our hate speech detection model could also be affected due to biases in the data. Our training data is annotated and classified by humans, so there is no guarantee that there exists no personal opinion within the data. What is more, in a multicultural society, modern languages are constantly changing and updating. Due to such reasons, our model could produce incorrect classifications that miss out on potential hate speech or misclassify non-hate speech as hate.

Example One:

“Bitches be salty as hell when a nigga treat they Ma dukes like a queen but not them 🤣”

Our model classified this sentence as hate speech, most likely due to the use of AAVE(African-American Vernacular English). However, this sentence is only representing a social situation, which would not be classified as hate speech by a human.

Example Two:

“Americans forgot get that they are the invaders! This wasn't they're land ! You guys took it by raping and murdering innocent people !”

Although the language associated with this sentence is toxic or hateful, it represents a call for help from the native people in America, and our model classifies this sentence as hate speech as well.

Both of these examples demonstrate that our model presents some biases towards already vulnerable minority groups. Due to the nature of our model and dataset, if words that are often associated with swearing, and insults are present in a tweet, the model would be more likely to classify it as hate speech regardless of the tone or the purpose of the speech. In real life, the emotions and tones of a sentence could greatly affect the meaning of words.

The model also faces the potential harm of being interpreted in a wrong way that misleads people’s judgment of hate speech, thus the intended use of this model is for research purposes, fine-tuning on carefully constructed datasets that reflect real world demographics and/or to aid content moderators in flagging out harmful content quicker.

Authors

Ruopeng worked on the training functions and the ethical decisions of the readme. Dhiren worked on some of the data preparation, training the model and writing the data & model readme sections. Fareeha worked on data processing, built the model, worked on training it and wrote the model and results readme portions.