Project Report R3: Toxic Comment Classification

Devi Krishnan & Srujana Rachakonda

Project Overview:

Discussing things that you care about on a social platform can be difficult. The threat of abuse and harassment online either means people stop expressing themselves and give up on seeking different opinions, or lose sight of what they really care about and go down the rabbit hole of abusing each other. Especially in this day and age where freedom of speech is acknowledged to be a basic human right, everyone should be able to have a healthy online presence and share their opinions. Social media platforms struggle to facilitate such communication however, and most of the time just end up shutting down the comment section if things get out of hand.

This project aims at helping social media improve online conversation by identifying toxicity in comments. We plan on identifying which type of toxicity the text represents, so that users can pick and choose which ones they are fine with and which ones they want to avoid.

NLP Keywords: Convolutional neural network (CNN), word representations, text encoding, Bag-of-Words, text classification, BERT, FastText

Dataset:

We will be using the Wikipedia comments dataset provided by Jigsaw/ Conversational AI which is a combination of ~ 550,000 comments from Wikipedia articles. The toxicity is categorized into the following categories: Toxic, Severe_Toxic, Obscene, Threat, Insult, and Identity_hate.

The labelling for the comments in the dataset was done by human raters for toxic traits. On performing some exploratory data analysis we noticed that the dataset is highly imbalanced. The dataset also falls under the category of multi-label classification. i.e, Each instance can fall into one or more categories. The test dataset initially consisted of ~ 150000 instances but however, the creators of the dataset did not reveal the ground truth labels for them all. Therefore, the test dataset size is reduced to ~63,000 instances.

Hypotheses:

1. Classify Wikipedia comments into groups based on whether they are toxic or not, and to identify the level of toxicity in these comments into one of the following categories:

    a. Toxic
    b. Severe_Toxic
    c. Obscene
    d. Threat
    e. Insult
    f. Identity_Hate
    g. Non-toxic
2. CNNs are good at identifying spatial relationships in data and are extensively used for image data for this reason. Text data also has localized spatial correlations and CNNs will exploit this. We want to capture the word order and the context using this method and investigate whether it performs better than traditional word embeddings + classification techniques.
3. *Reusable knowledge*:
    a. A comparative analysis of 1D CNN for text classification against Text CNN model. The Text CNN model gained a lot of popularity and was tested for various natural language tasks such as question answering and sentiment analysis. However, they did not experiment with a text classification problem. We wanted to see how the model fares in this situation.
    b. We also wanted to compare the performance of CNN models against traditional machine learning approaches for classification. The paper presented in [2] uses a traditional bag-of-words model with a CNN. We want to explore the effect of using pre-trained word embeddings as weights to the CNN model to see if that improves the performance of the model or not.

NLP Techniques:

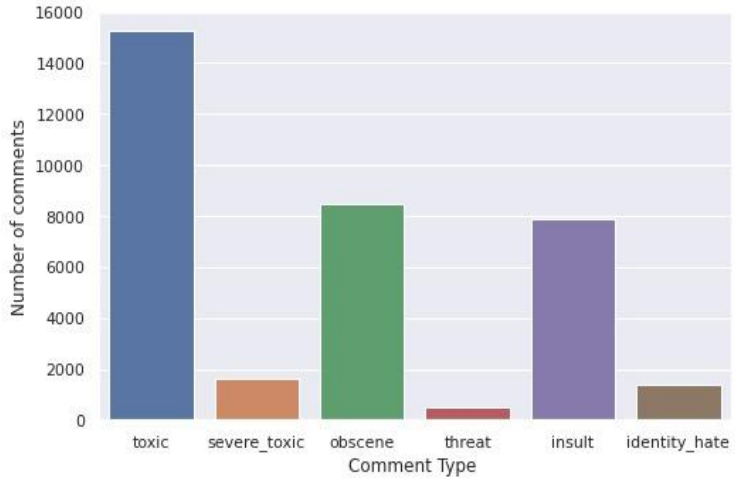
- Text classification is the process by which we assign words to different categories based on their content. It is a common task in natural language processing. Feature extraction is an important part of the process, and traditionally it is done using bag-of-word approaches, TF-IDF or word2vec. We evaluated all of these in order to find the best method.
- BERT is commonly used across different NLP tasks and has been known for its state-of-the-at performance for many of these tasks, which was our motivation behind using it for vectorization of our features. Its transformer encoder allows an entire sequence to be read at once, learning the surrounding context of a word, which is what makes its generated embeddings so efficient.
- Our proposed model makes use of FastText pre-trained word embeddings. FastText is a library created by the Facebook research team for efficient word representations and sentence classification. It is different from word2vec in the sense that while word2vec treats every word as the smallest unit, FastText considers the n-grams of every word

along with the word. We wanted to use it so that we could compare its performance to word2vec and TF-IDF.

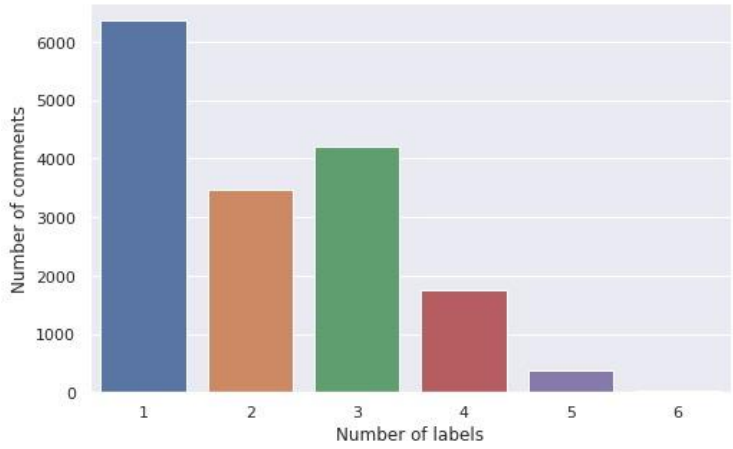
Implementation:

We implemented various different competing approaches in order to thoroughly investigate different models and identify the method with the highest accuracy. We have chosen logistic regression as our baseline model and tabulated results with various vectorization techniques such as term frequency - inverse document frequency (TF-IDF) , Word2Vec and BERT. We tried two different CNN models, a simple sequential model with 1D CNN layers and then a 2D CNN model proposed by Kim. We also explored the impact of having pre-trained embedding weights to fine-tune the CNN model. For this purpose, we leveraged FastText embeddings. We evaluated all the models on the entire training set except for BERT. We were not able to utilize the whole dataset for BERT since we did not have enough computational power. We considered about a 2000-instance subset of the dataset for BERT.

<u>Preprocessing & Exploratory Data Analysis:</u>



The dataset contains uncensored and unfiltered comments, many of which are strewn with noisy data, including random escape characters, hash symbols, extra spaces, IP addresses etc. The first step was to clean the data. The next step was to remove stopwords, which are structural words that do not add anything to the meaning of a sentence. We replaced any empty comments with sampled text to avoid errors during the training process. We also set a limit on the number of words for the vocabulary we constructed to 100,000 words to avoid words with typos. Exploratory Data Analysis also revealed a class imbalance problem, which is addressed in Technical Challenges.



Method:

The three baseline models we tested were as follows:

1. TF-IDF with Logistic Regression
2. Word2vec with Logistic Regression
3. BERT with Logistic Regression

For each of these models, we followed the general approach of cleaning and preprocessing the data, vectorizing the data using the specific vectorization method, and then classifying the data points on first the validation data set and then the test data set.

Proposed Model:

We used Keras, a framework built on top of tensorflow for this model. We converted the preprocessed text data into sequences using the text to sequences method. It maintains an index of all the words and replaces the words with their corresponding index here. We used FastText embeddings to add weights to the words during the training process. In order to facilitate multi-class classification we set the number of nodes in the output layer to 6 (expected number of classes) and chose binary cross entropy loss. Altogether we get k hot encoded labels as our output, where more than one label can be true.

Following that, we implemented different variations of our proposed model as follows:

1. Sequential Convolutional Neural Network model (1D CNN):

   Here we used 2 Convolutional layers with 64 filters, each interleaved with a max pooled layer and a global max pooling layer. We used a dropout value of 0.5 for regularization followed by a dense layer and an output layer. We incorporated early stopping based on validation loss to avoid overfitting of the

model and the model ran for about 5 epochs with a batch size of 256. This model gave us a significant improvement over our logistic regression model.

2. Parallel Convolutional Neural Network model (TextCNN):

   We tried both with and without pre-trained embedding weights for this model. The Model consists of similar pre-processing and vectorization as the previous model. We only changed the CNN architecture here. We used 3 different convolutional layers here all running simultaneously each with a filter size of 3, 4 and 5 respectively. We then use pooled layers followed by concatenating the results into a fully connected layer. We then use a dropout layer with 0.5 and a dense layer followed by an output layer as before.

For the former model, we first obtained an embedding matrix from pretrained FastText Embeddings, and then trained the Sequential CNN model on those weights. The output consisted of the predicted labels for the test data. This was a basic CNN model. For the latter model, in which the CNN model was based upon Yoon Kim's architecture [1], we tried both with and without the pre trained word embeddings. Here we tried with both word2vec and fasttext embeddings. We performed preprocessing and vectorization as in the previous model.

Results & Discussion:

Metrics:

We used Area Under the Receiver Operating Characteristic Curve (ROC-AUC) score as our metric since it is a multi-label classification problem. ROC curves work by plotting the True positives rate vs the True negatives rate. No overlap between the TP and TN curves indicates the best model with zero overlap and highest degree of separation. The higher the score, the better the performance of the model.

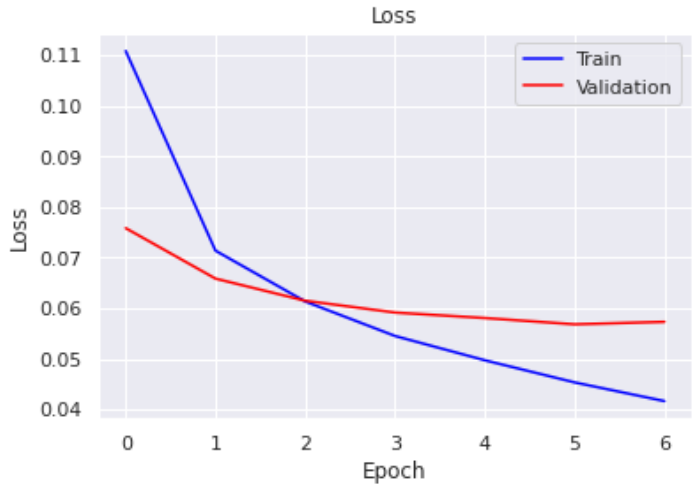
Results:

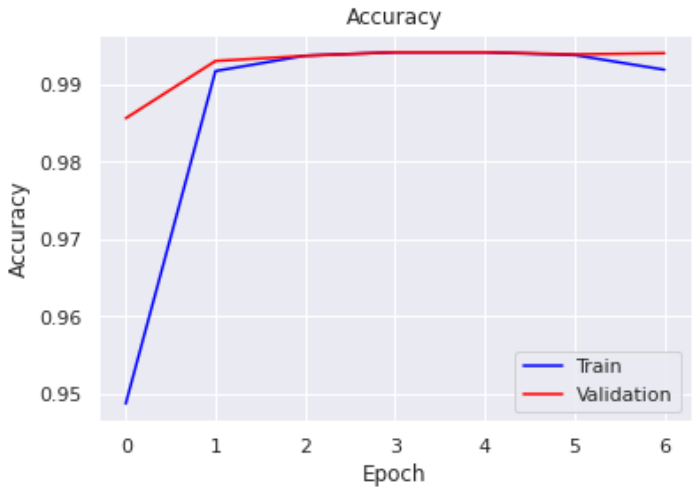
We split the training dataset into 80% training and 20% validation sets. We used the validation loss to tune our hyperparameters and finally predicted on the test set. The test set was initially around 150,000 instances, but the creators of the dataset annotated the ground truth labels only for around 63,000 rows, therefore our test set was of the same size.

The following is a table of all the models we tried and the corresponding ROC scores. KR refers to keras preprocessing in the following table.

Model	Vectorization Technique	ROC score
Logistic Regression	TF-IDF	0.763
Logistic Regression	Word2vec	0.549
Logistic Regression	BERT	0.61
1D (Sequential) CNN	FastText weights + KR	0.968
TextCNN	Word2vec + KR	0.971
TextCNN	**FastText weights + KR**	**0.978**
TextCNN	KR	0.958

The training and validation accuracy and loss plots for the best model (TextCNN) are as follows:





Discussion:

The results are significantly better for the CNN models when compared to the logistic regression models. One possible reason for this could be the imbalance of classes which logistic regression is unable to handle. We notice that with different vectorization techniques there is a slight improvement in FastText over word2vec, as well a slight improvement in word2vec over not using an embedding matrix at all. However, even without weight fine-tuning the CNN model was giving us about a 95% ROC score.

Importing the pre-trained embeddings from FastText or word2vec can still be very computationally intensive, therefore at the cost of slight dip in the performance, one can ignore the use of an embedding matrix for fine-tuning the weights and still achieve decent performance using Kim's TextCNN.

Technical Challenges:

- We noticed a class imbalance problem during our initial data analysis. In each of the 6 categories, there was a 1 if the comment was classified as that category, and 0 otherwise. But many comments were non-toxic, so they had no 1s in any category, which is what caused the imbalance. Our baseline models did not have a high ROC score because of this, and their high accuracy values were not meaningful. Our CNN models overcame this.
- Our BERT model was initially going to be in one of our proposed models, due to its high performance across NLP tasks, but we did not have the computational resources to generate word embeddings using BERT for more than a small subset of our data. Due to the class imbalance issue, even the small subset of data we worked on with BERT had a low ROC score even though its accuracy was high.
- We also realized that BERT is conventionally used for sentence encoding directly and it heavily utilizes the context of the word. Therefore it is unconventional to get the word vectors for it and utilize it as weights for the CNN models. We were unable to find many resources to move forward with this path.

Future Directions:

- A possible future direction this project could take is classifying the sentiment of the comments, so as to provide another aspect for classification. If we know the overall sentiment of a comment, we could filter out the most negative comments as a kind of classification technique, since those would intuitively have some level of toxicity.
- Another aspect could be named entity recognition, to find the topics of interest of each comment and find which topics are the most negatively (in a toxic way) commented about. Both of these aspects could be useful to social media platforms.

- We can compare the performance of our CNN model to a preferred RNN model such as Bi-LSTM. CNNs are far less computationally intensive than RNNs. This would show that these CNN models can be used as a good and viable alternative to RNNs.
- Subsampling the dataset to get a more balanced data set in order to compare the performance of the baseline model without the effects of class imbalance is another possible direction. We can even try other sampling techniques such as SMOTE, over-sampling the minority class or under sampling the majority class.

References:

[1] Yoon Kim. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[2] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. Convolutional neural networks for toxic comment classification. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence, SETN '18, New York, NY, USA, 2018. Association for Computing Machinery.

[3] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[6] Kartik Nooney. Deep dive into multi-label classification, Jun 2018.

[7] Francois Chollet. The Keras Blog

[8] Joshua Kim. How CNNs perform text classification with word embeddings.

[9] Jason Brownlee. Multi label classification with deep learning, Aug 2020.

[10] Using pre-trained word embeddings in a keras model.