OFFENSIVE TEXT ANALYSIS

CSD312 - Information Retrieval

Project Report

Ayush Nair 1610110085
Sargam Jalota 1610110336

1. Abstract

Freedom of speech and empowerment are the norms of the decade and we can see this trend on multiple fronts such as social media, newspapers, human interactions, etc. With this comes the need to protect individuals from other individuals who might pose a threat to them and take action preemptively.

With respect to digital media, companies seek to ascertain the level of threat in such interactions through their websites maybe chat websites, blog, forums, etc. to protect both themselves and the victims from criminal activity and avoid prosecution by law should they get entangled in it and avoid paying huge settlement bills, etc.

As such they aim to take action prior and maybe ban such vagrants from their websites or report such activity for future use. Therefore the need arises to categorise the level of threat or danger from his comments, posts and articles and take preemptive action accordingly.

2. Introduction

The aim of this project is to identify the level of threat/danger that a particular individual had poses through written communication through the form of the comments, articles posts. These “texts” need not be in any particular format just that they follow the lexicon of English.

The prior assumption is that the texts in question currently are offensive and the purpose is to classify/label them into categories listed below:

- Severe_Toxic
- Toxic
- Obscene
- Threat
- Insult
- Identity_Hate

The above labels should classify all offensive text appropriately and cover all areas of obscenity and vulgarity of the English lexicon with all its semantics and inferences.

The end result of this project would be the accuracy or error rate of our algorithm for the machine learning model which the companies might require to judge the usefulness and efficacy of our system should they wish to use this on their websites.

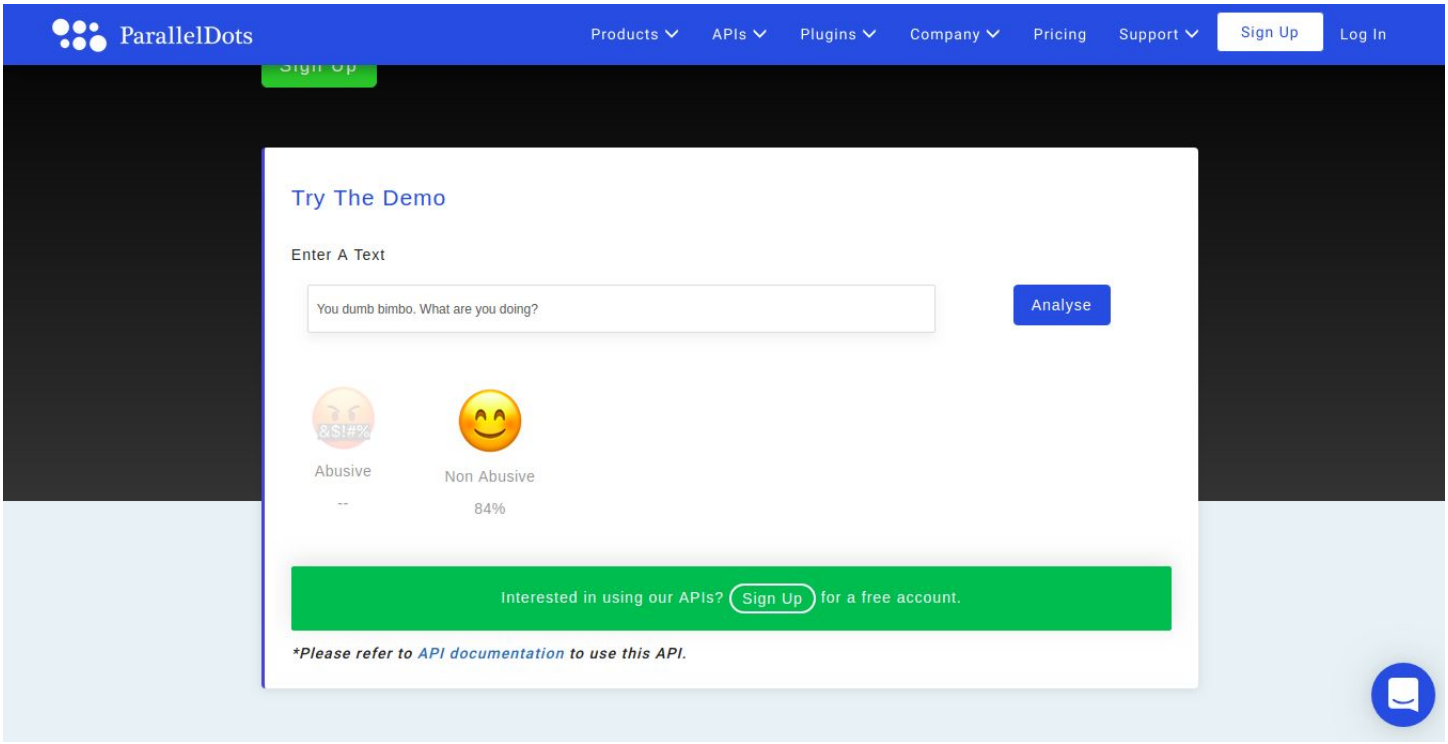
3. Related Work

Various machine learning approaches have been made in order to tackle the problem of toxic language. Majority of the approaches deal with feature extraction from the text. Lexical features such as dictionaries and bag-of-words [1] were used in some studies. It was observed that these features fail to understand the context

of the sentences. N-gram based approaches were also used which shows comparatively better results [2].

Although lexical features perform well in detecting offensive entities, without considering the syntactical structure of the whole sentence, they fail to distinguish sentences' offensiveness which contains the same words but in different orders. In the same study, the natural language process parser, proposed by Stanford Natural Language Processing Group, was used to capture the grammatical dependencies within a sentence

Companies employ various techniques which are undisclosed as of now to filter messages and comments effectively. ParallelDots already has an API to detect the level of abusiveness in a comment which it licences out to companies.



It's not perfect as you can clearly see in the above example. The input text is offensive at the very least and it classifies it as non-abusive with an 84% probability.

4. Proposed Method

We will attempt to tackle this classification problem using the Keras LSTM. The approach taken is to feed the comments into the LSTM as part of the neural network but the data can't be fed as is.

1. Tokenization - the sentence is broken down into words. For eg, "I love cats and love dogs" will become ["I","love","cats","and","dogs"]
2. Indexing - the words are put in a dictionary-like structure and given an index each. For eg, {1:"I",2:"love",3:"cats",4:"and",5:"dogs"}
3. Index Representation- the sequence of words in the comments is represented in the form of index, and feed this chain of indexes into our LSTM. For eg, [1,2,3,4,2,5]

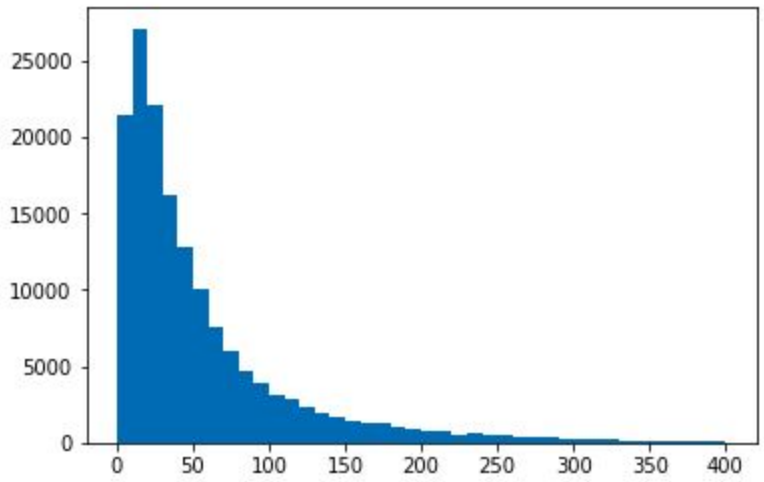
The next problem to tackle is the length of the comments. Some comments are quite long while some are short. We need to feed the LSTM an input of uniform length by normalising the length of all comments to fixed a length that is neither too short or long and can vividly showcase all the features it portrays.

Let's explore the data.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

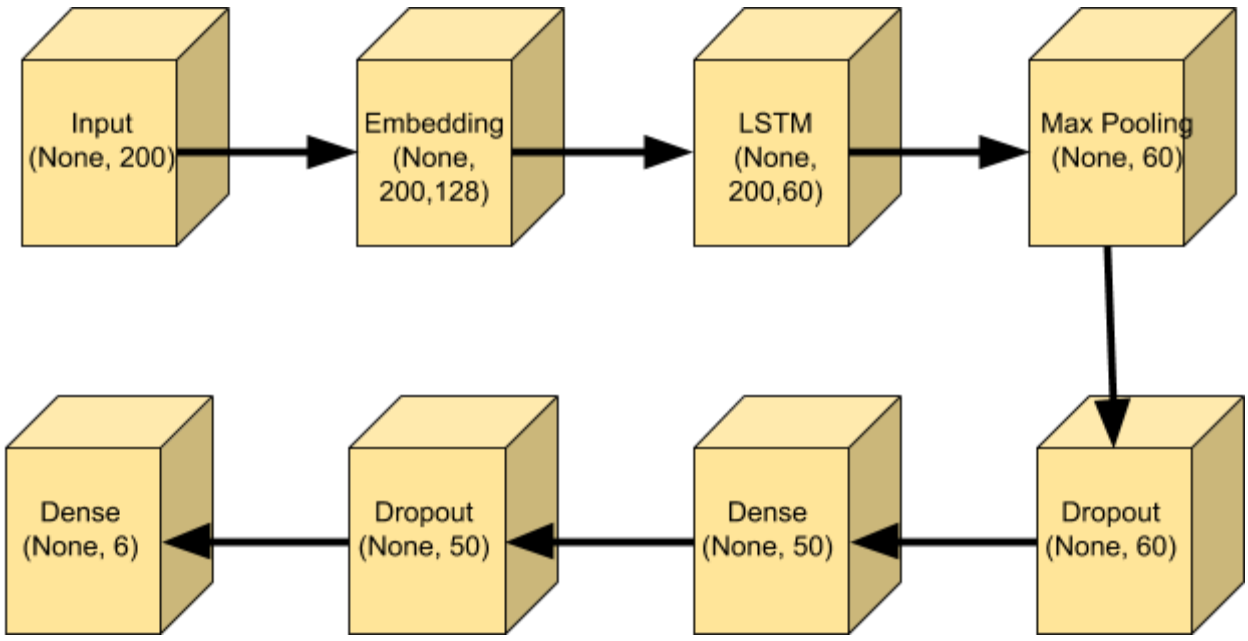
The training data has around 150k rows and 8 columns. The dataset is pretty clean with no null values.

After converting the texts into their token representations, we can plot the length of the comments.



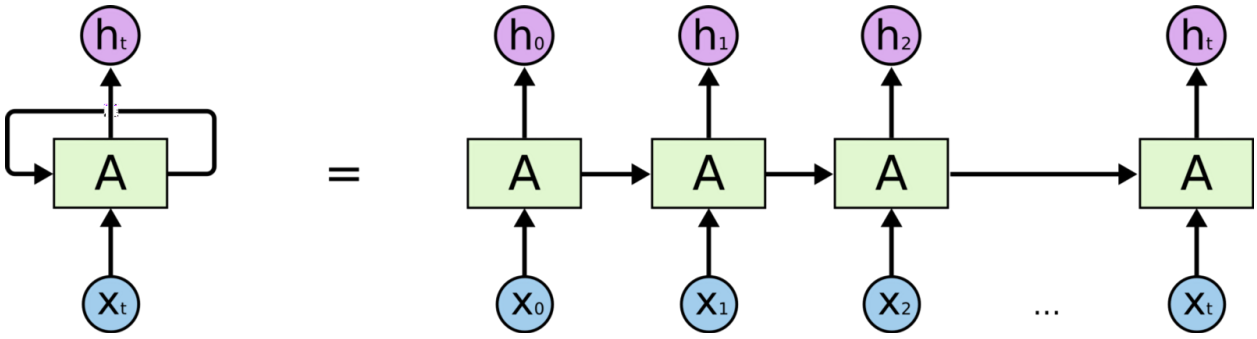
Most of the sentences are 30+. We could set the optimal length to 50 but to be on the safe side and capture all the features we go a little paranoid and set it to 200. Later we can find the perfect number that is required to match the expectations of accuracy.

The architecture of the model is as follows:



The inputs i.e. the tokenized representations are then passed to the embedding(one-hot encoding) layer whose output is then passed onto the LSTM layer.

LSTM's or RNN's work by recursively feeding the output of a previous network into the input of the current network, and we get the final output after 'x' number of recursions. But in this use case, we take the unrolled, or the outputs of each recursion as the result to pass to the next layer.



The result is then passed through a couple of densely connected layers with the end layer being passed through a Sigmoid activation function for each of the 6 labels.

Finally, the model will be compiled with the Adam optimiser and binary cross-entropy as the loss function.

5. Implementation and Results

The embedding size in the Embedding layer has been set to 128 after trial and error. It can be fine-tuned and experimented with. The output from the LSTM layer is a 3D tensor which needs to dimensionally reduced to a 2D tensor to pass

to the normal layers. Global Max Pooling is usually used in Convolutional Neural Networks (CNN) to reduce dimensionality but it serves the same purpose here.

10% dropout has been chosen to deal with the outliers and force the nodes in the next layer to handle the representation of any missing data thereby creating better generalization.

The "None" in the inputs of the layers mean that the input dimension is inferred from the previous layer.

During compilation, the Adam optimiser is used for it is the most robust and commonly used optimiser in today's day and age. The default learning rate is used which 0.001. It can be experimented with. The sigmoid function is used to force the outputs to be between 0 and 1 in the output layer.

Finally to test the model we run the model on the data with a batch size of 32 i.e. we send 32 comments/texts at a time for a total of 2 epochs. We split the data to a 1:9 ratio for validation testing which means around 140k rows are for training and around 15k rows for validation.

For the above parameters we get the following results for each epoch:

Epoch	Training loss	Training acc.	Validation Loss	Validation Acc.
1st	0.0707	0.9777	0.0489	0.9818
2nd	0.0651	0.9812	0.0409	0.9882

The above results are pretty good for the initial attempt and can obviously be improved upon which we look at in the next section.

6. Conclusion | Future Work

The results obtained are preliminary and can easily be improved by modifying the parameters of the model itself as well as the preprocessing of data.

As of right now, the texts have been converted to their indexed token representations from a dictionary. For a more semantics oriented analysis, TF-IDF vector model can be taken with n-grams to categorise the comments into these labels.

The dataset itself is very small and restrictive to the English syntax. Natural Language Processing(NLP) requires huge amounts of training to accurately make predictions for all cases.

Nowadays, the trend is to use emoticons and acronyms to substitute their emotions in a fast and easy manner. Variants of English based on location should also impact the analysis as there are subtle differences in Australian and American English through the “way of speaking”. The current model does not account for that as the data in its description is English(U.K).

The model itself is in its preliminary stage. Hyperparameter tuning of the model’s parameters will exponentially improve the output/accuracy of the model by preventing overfitting. The learning rate is at its default value whose change might affect the model’s output. The number of epochs can be increased to increase the accuracy along with the increase in batch size.

Upon increment of the number of epochs, it might be possible the validation loss starts to degrade then we can introduce early stopping to avoid that. Experimenting with existing architecture and trying out other pre-existing models can help by leaps and bounds.

The system so far is pretty accurate but by no means perfect and work still needs to be done to make it industry standard so that it can be deployed on websites and other places.

7. References

[1] Detecting Offensive Language in Social Media to Protect Adolescent Online Safety - Ying Chen, Yilu Zhou, Sencun Zhu, Heng Xu - Pennsylvania State University

[2] Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TF-IDF based Approach - Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, Laxmi Bhagwat - Maharashtra Institute of Technology

[3] Hate Speech and Offensive Language Repository - @t-davidson

[4] Toxic Comment Classification Challenge - Kaggle