Toxic Comment Classification using Natural Language Processing

A. Akshith Sagar, J. Sai Kiran

Abstract - *Text classification has become one of the most useful applications of Deep Learning, this process includes techniques like Tokenizing, Stemming, and Embedding. This paper uses these techniques along with few algorithms, that are used to classify online comments based on their level of toxicity. We proposed a neural network model to classify the comments and compared the model's accuracy with some other models like Long Short Term Memory (LSTM), Naive Bayes Support Vector Machine, Fasttext and Convolutional Neural Network .The comments are first passed to a tokenizer or vectorizer to create a dictionary of words, then an embedding matrix is created after which it is passed to a model to classify. The proposed model achieved an accuracy of 98.15%.*

Key Words: **Toxic comment detection, Long short term Memory, Convolutional Neural Networks, Naive Bayes, Support Vector Machine, Fasttext**.

1. INTRODUCTION

Social media is a place where a lot of discussions happen, being anonymous while doing so has given the freedom to many people to express their opinions freely. But people who disagree with a point of view extremely can misuse this freedom sometimes. Sharing things that you care about will become a difficult task with this constant threat of harassment or toxic comments online. This will eventually lead to people not sharing their ideas online and stop asking for other people's opinion on them. Unfortunately, the social media platforms face these issues all the time and find it difficult to identify and stop these toxic remarks before it leads to the abrupt end of conversations.

In this paper, we will be using Natural Language Processing with Deep neural networks to solve this problem of identifying the toxicity of online comments. Word embeddings will be used in conjunction with recurrent neural networks with Long Short Term Memory (LSTM), Convolutional Neural Networks (CNN), and Naive Bayes (NB)-Support Vector Machine (SVM) and Fasttext separately and see which model fits and works best.

2. PRE-PROCESSING TEXT

Pre-processing of the text is the first step that is performed on the dataset. The dataset is cleaned and prepared for the classification tasks by removing punctuation, imputing missing values, normalisation, etc. Besides these common preprocessing functions there are other techniques that are used specifically for deep learning classification.

2.1 Tokenization

Tokenization is the process of converting a text corpus to a set of distinct tokens of any size. These tokens are usually numbers which are assigned to the words present in the text. As a computer cannot understand a language, this method helps us to map all the words to distinct numbers which makes it easier for the computer to understand. So the result of this process is a dictionary of fixed size that contains a mapping from words to numbers.

2.2 Vectorization

Vectorization is a technique in which words are converted to feature vectors. This paper uses the Term Frequency Inverse Document Frequency Vectorization (TFIDF). TFIDF Vectorization converts the words in the document to a vector that can be used as input to the estimator. It can be used to learn how important a word is to a document. This is done by assigning a score to each word in the document.
Term Frequency can be defined as follows: -

TF(w) = (Number of times word w appears in a document) / (Total number of words in the document)

Inverse Document Frequency is defined as follows: -
IDF(w) = log (Total number of documents / Number of documents with word 'w 'in it).

The TF-IDF score is simply the product of these two frequencies. i.e.

TF-IDF(w) = TF(w) * IDF(w)

2.3 Word Embeddings

Every word in the dataset is embedded into feature vectors, this is done by creating an embedding matrix. An embedding matrix is a list of words and their corresponding embeddings. Embeddings usually refer to n-dimensional dense vectors. The embedding matrix is of shape (vocab_size, embed_size). Here vocab_size is the number of words in the dictionary that are obtained from the tokenization method and embed_size is the number of features into which the words will be embedded. There are a lot of pre-trained word embeddings available with different embedding sizes like the GloVe (Global Vectors for Word Representation), word2vec, Fasttext-crawl, etc. This paper uses fasttext-crawl-300d-2m for the embedding matrix. This embedding matrix is then passed to different algorithms.

3. ALGORITHMS

3.1 Long Short Term Memory (LSTM)

Recurrent Neural Networks (RNN) are neural networks that contain cyclic connections. The output of a given hidden layer is fed back to itself in an RNN to remember some information as the memory from previous computations. This makes RNNs a powerful tool for sequential data like text, video, and speech.

Long Short Term Memory (LSTM) is an RNN that can learn long term dependencies which is something a traditional RNN finds difficult to do. An LSTM model, like RNN, has a chain-like architecture where each unit of this repeating structure is called an LSTM cell.

An LSTM cell contains an input gate, an output gate and a forget gate that regulates the data which is flown into and outside of the cell. The forget gate decides what information it's going to discard from the current cell. The input gate then decides what new information is going to be added to modify the current state of the memory and finally, the output gate decides what information leaves the cell.

3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) is a deep neural network that is usually applied to images. CNNs were inspired by the human brain. Like the human brain, CNN consists of interconnected neurons in different layers. Each neuron in a layer is a perceptron that performs some computation to the weights that are passed to it. Although CNNs are mostly used for image classification, they can also be used for text classification by passing the feature vectors of input text to CNN. The CNN then computes weights for different neurons which are used to determine a function that maps the feature vectors to the output. A CNN usually consists of the following layers: -

***a) Convolutional Layer**:* - The purpose of a Convolutional layer is to extract and learn features from the input vectors. A convolutional layer computes outputs of neurons by performing dot product operations to the weights and passes this output to an activation function.

b) Activation Function: - The output of a convolutional layer is passed to an activation function. An activation function is used to add non-linearity to the output of the Convolutional Layer. The most common activation function is the Rectified Linear Unit (ReLu) function. A ReLu function can be defined as follows:

$$f(x) = \max(0, x)$$

***c)Pooling Layer**:* - A pooling layer is used to reduce the dimensions of the input by preserving the important features. A Convolutional layer is often succeeded by a pooling layer to reduce the size and number of parameters from the previous layer.

3.3 *SVM with NB features (NBSVM)*

Naive Bayes classifier algorithm is based on Bayes theorem which determines the posterior probability of an event occurring based on prior knowledge or evidence. Multinomial Naive Bayes an instance of NB classifier which uses a multinomial distribution for each feature of data.

Support Vector Machine (SVM): SVM works on the principle of finding hyperplanes that distinctly classify the data units when you plot them onto an n-dimensional graph(here the n refers to the number of features of the data).

In Naive Bayes -Support Vector Machine the probabilities calculated in MNB are then fed to SVM to classify. NBSVM is observed to give better results than a simple NB classifier or SVM classifier when used separately.

3.4 Fasttext

Fasttext is a text classification library developed by Facebook. Fasttext can be used to learn word embeddings, create supervised or unsupervised classification models from these word embeddings. Fasttext has its word embeddings called Fasttext crawl which is trained on around 600 Billion tokens. These word embeddings are open and can be downloaded by anyone for their use.
Fasttext has multiple pre-trained models to choose from depending on the nature of the problem. In this paper we use the default supervised classifier model.

4. IMPLEMENTATION

Keras framework was used to implement the LSTM and CNN models. Keras is an open-sourced neural network library built on python which provides user-friendly, high-level APIs which enable easy implementations of different deep neural network algorithms.

The dataset used for classification in this paper is given as part of the competition hosted by Jigsaw and it contains 159571 comments taken from Wikipedia. Each comment is classified into one of 6 labels based on their level of toxicity. The results are then validated against a test data set of 153164 new examples.

4.1 Naive Bayes-Support Vector Machines

In this approach, we first compute TF-IDF scores for the words in the train data using the TF-IDF Vectorizer. This generates a matrix that contains an array of scores for each training example.
Then we use the Multinomial Naive Bayes theorem on each column of the labels to generate the NB probabilities from the TF-IDF matrix. This is then fed to SVM to predict the probabilities of each label.
After predicting the test set NB-SVM achieved an accuracy of 97.61 %.

4.2 Fasttext

The fasttext library takes the input in a text format, so all the comments from the train data are converted to a text document with each training example starting with ' __label__' followed by the respective label of the comment and then the comment itself. This text file is fed into the fasttext model and after fine-tuning the hyper parameters of the number of epochs and learning rate to 5 and 0.1 respectively, the model achieved an accuracy of 95.4%.

4.3 Long Short Term Memory

The first step of this algorithm is tokenization. This generates a sequence of numbers for each comment. As each comment may vary in their lengths, the output of tokenization is padded to a fixed length of 200. This is then passed to a Keras Embedding Layer which learns embeddings. The embedding size used was 300. The output of this embedding layer is then fed to an LSTM of 60 units, which then returns sequences.

This is then passed to a Pooling Layer, Dense Layer of 60 units, Dropout Layer, and finally to a Dense layer of 6 units with a sigmoid activation function. This predicts the probabilities of 6 classes.

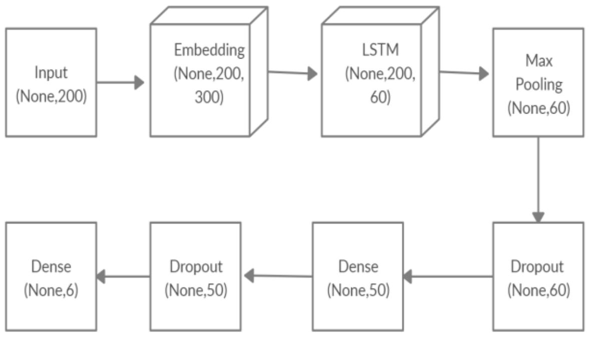

fig 1 LSTM Architecture

This model of LSTM achieved an accuracy of 96.92%.

4.4 Convolutional Neural Networks

The first few steps of this algorithm are the same as that of LSTM. But here, instead of allowing the embedding layer to learn the weights by itself, we provide the Embedding Layer with a matrix extracted from the fasttext-crawl file. After the embedding layer, we then use a series of convolutional layers in conjunction with pooling layers. This paper uses 4 convolutional layers and 4 max-pooling layers. The output of these layers is concatenated and then Flattened to an array, which is finally fed to a Dense layer of 6 units with a sigmoid activation function, which predicts the probabilities of each label.

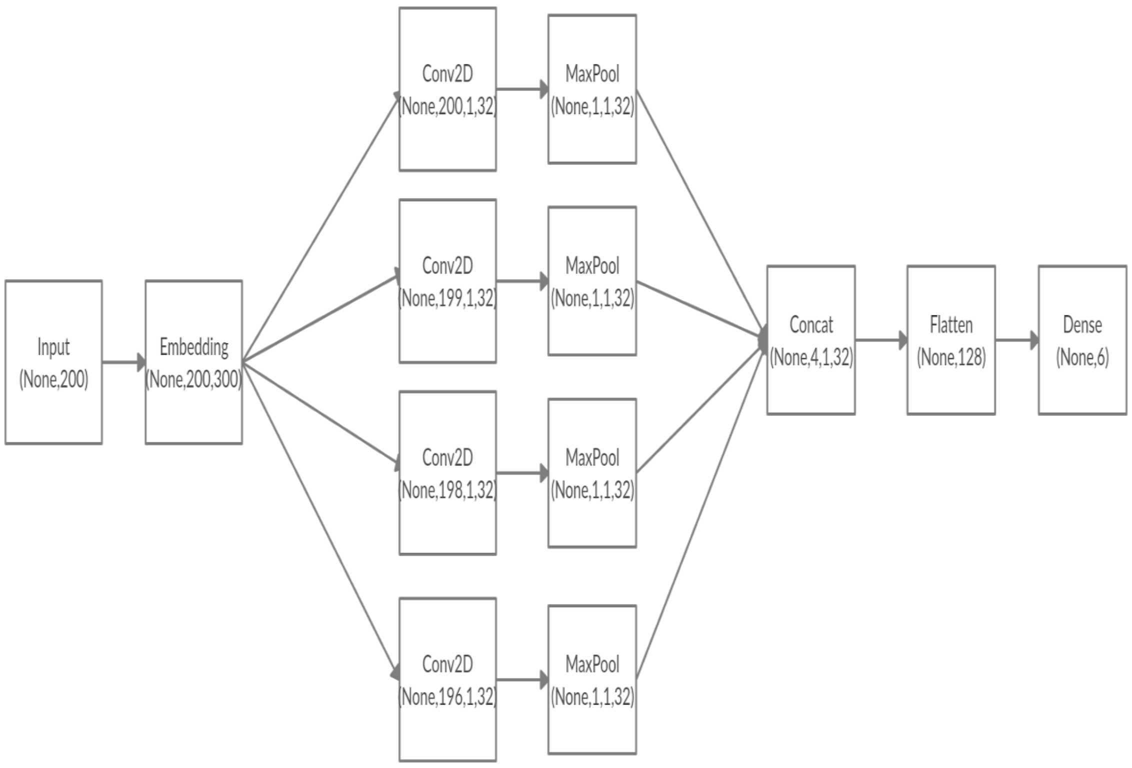

fig 2 CNN Architecture

This CNN model achieved an accuracy of 98.13%.

5. CONCLUSION

With the Internet being a platform accessible to everyone, it is important to make sure that people with different ideas are heard without the fear of any toxic and hateful remarks. And after analyzing various approaches to solve this problem of classification of toxic comments online, it is found that CNN model works slightly better than LSTM and NB-SVM with the accuracy of 98.13%. Future scope for this analysis would be integrating such classification algorithms into social media platforms to automatically classify and censor or toxic comments.

REFERENCES

[1] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao, " Recurrent Convolutional Neural Networks for Text Classification" Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, Texas, 2015.

[2] Sida Wang and Christopher D. Manning, "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification", Stanford, CA,

[3] Mujahed A. Saif, Alexander N. Medvedev, Maxim A. Medvedev, and Todorka Atanasova, "Classification of online toxic comments using the logistic regression and neural networks models", AIP Conference Proceedings 2048, 060011 (2018)

[4] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] Sepp Hochreiter, Jurgen Schmidhuber, "LONG SHORT-TERM MEMORY", Neural Computation 9(8):1735-1780, 1997

[6] Navaney, P., Dubey, G., &Rana, A. (2018). "SMS Spam Filtering Using Supervised Machine Learning Algorithms." 2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence).