EST. 1785 UNB
UNIVERSITY OF NEW BRUNSWICK

Computer Science

Sentiment Analysis from Text Using Convolutional vs Recurrent Neural Networks

Report for CS6735 – Machine Learning and Data Mining

Prepared by

Daiyan Khan 3745340

Mohammed Ghayasuddin 3765433

Rohith Kumar Saravanan 3756386

Shidur Sharma Durba 3757255

Introduction

In the ever-expanding digital age, one of the most basic and simple methods of communication between 2 parties is the exchange of texts – could be an email, a review, a status post, etc. At any point in time, the emotion or sentiment behind any piece of text can be interpreted by a human from context. However, if someone wanted to parse through such data at a large scale, for example a business trying to gauge customer response from thousands of reviews, or a social media page trying to understand the overall emotional engagement of their viewers' comments, it can be difficult at any point in time to understand the overall range of emotion or sentiment in available texts given the massive volume at which they are being passed around the internet. This issue can be addressed using Sentiment Analysis (SA).

Sentiment Analysis (SA) is the process of using text analysis methods, natural language processing (NLP) and computational linguistics to be able to extract, identify, and quantify the sentiment and connotation of any given text accordingly – that is, to be able to determine whether a string of words are conveying a positive or negative emotion, based on the context. The applications of SA include detecting the feeling of a text, testing interest in text, or even to test text polarity (whether a text is positive or not).

For our project, we have categorized possible sentiment values inferred from texts into 5 categories: joy, anger, fear, sadness, neutral. With the help of a pre-trained data set, we have trained 2 different Machine Learning (ML) models to be able to predict the sentiment from input text – a Convolutional Neural Network (CNN) model using Keras Sequential API, and a Recurrent Neural Network (RNN) model using Long short-term memory (LSTM) network. A model was also implemented using both the CNN and LSTM layers. An SVM, and Simple Naïve-Bayes (NB) model were implemented for comparison purposes. Below is a diagram summarizing the workflow of our proposed method. The purpose of our project is to compare the effectiveness of Sentiment Analysis using CNN (Keras) vs RNN (LSTM).

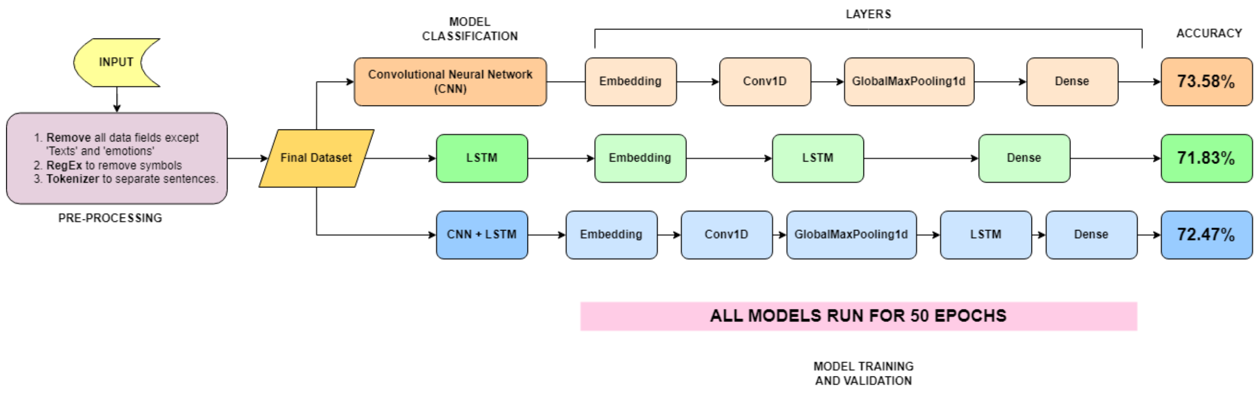

INPUT
1. Remove all data fields except 'Texts' and 'emotions'
2. RegEx to remove symbols
3. Tokenizer to separate sentences.
PRE-PROCESSING
Final Dataset
MODEL CLASSIFICATION
LAYERS
ACCURACY
Convolutional Neural Network (CNN)
Embedding
Conv1D
GlobalMaxPooling1d
Dense
73.58%
LSTM
Embedding
LSTM
Dense
71.83%
CNN + LSTM
Embedding
Conv1D
GlobalMaxPooling1d
LSTM
Dense
72.47%
ALL MODELS RUN FOR 50 EPOCHS
MODEL TRAINING AND VALIDATION

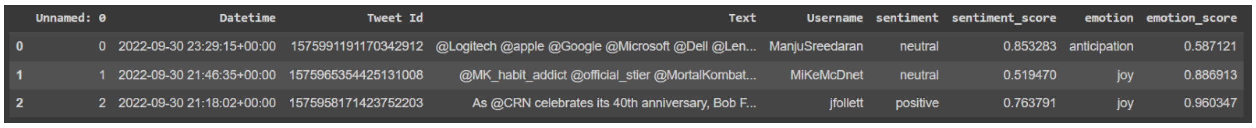
Fig 1: Workflow of our proposed method

Experiment Setup

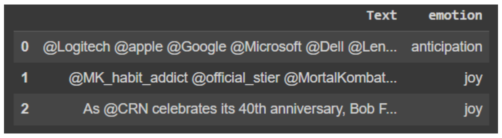
Data Collection and Pre-processing:

Using a publicly available dataset titled Sentiment & Emotions Labeled Tweets, from Kaggle, which contains 24,970 different Twitter posts regarding mentions of the company Dell between January till September 2022. This is a highly detailed dataset containing many details of each post including the pre-trained data labeling the sentiment (positive or negative value) and the emotion (11 different types) of each post, as well as scores (weights for each value).

For the purposes of pre-processing, we have extracted the data fields which include the posts (Text), and emotion values of all the posts. All other fields of the original dataset were dropped.



	Unnamed: 0	Datetime	Tweet Id	Text	Username	sentiment	sentiment_score	emotion	emotion_score
0	0	2022-09-30 23:29:15+00:00	1575991191170342912	@Logitech @apple @Google @Microsoft @Dell @Len...	ManjuSreedaran	neutral	0.853283	anticipation	0.587121
1	1	2022-09-30 21:46:35+00:00	1575965354425131008	@MK_habit_addict @official_stier @MortalKombat...	MiKeMcDnet	neutral	0.519470	joy	0.886913
2	2	2022-09-30 21:18:02+00:00	1575958171423752203	As @CRN celebrates its 40th anniversary, Bob F...	jfollett	positive	0.763791	joy	0.960347



	Text	emotion
0	@Logitech @apple @Google @Microsoft @Dell @Len...	anticipation
1	@MK_habit_addict @official_stier @MortalKombat...	joy
2	As @CRN celebrates its 40th anniversary, Bob F...	joy

Fig 2: Comparison of dataset layout before and after removal of required columns.

The dataset contained 11 labeled emotion values for each post, and we decided to work with these 5 - joy, anger, fear, and sadness. Other than these 5, all other emotions were labeled as neutral values.

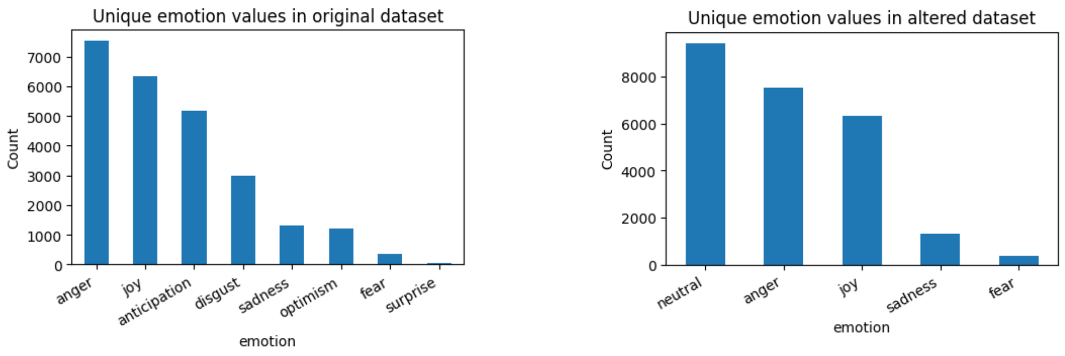


Fig 3: Comparison of available emotions in dataset before and after alteration.

After the dataset was prepared for our model implementation purposes, we proceeded to clean the dataset using Regular Expressions (RegEx) to remove all symbols from text except punctuation, and Tokenization to separate sentences and large bodies of text to smaller units for ease of creating arrays for the ML model to work on. The number of unique words in the dataset was recorded to be 34,970. For our model implementation, we chose to use the Text (post) values as the feature for our model, and the Emotion values to be the target for our model. The cleaned dataset was then split and labeled into a 70/30 split for training and testing (validation) purposes.

Importing pre-trained vectors:
Pre-trained vectors capture semantic relationships and contextual information. We used the Word2Vec library to aid us, as it can distinguish between words that may change meaning according to context.

Each unique word in our dataset is assigned an ID value so that they can be assigned weights. An embedding matrix is created with pre-trained word vectors (our project used the Word2Vec library). A maximum dimension of 300 was used for word embedding purposes, and a max sequence length of 500 (Twitter posts are limited to 250 characters, so we chose to double the value).

With the use of a pre-trained word vector file called "wiki-news-300d-1M", a dataset with 1-million-word vectors trained on Wikipedia circa, 2017. This matrix iterates through each line in the file and checks if the word in the current line exists in our input dataset. If the word was found in our dataset, the ID of the word was used to convert the associated word into a NumPy array for further processing.

Model Implementation:
Initially, we chose to work with a CNN model as mentioned. 5 classes were encoded for our training purposes, labelled the same as the 5 sentiment categories mentioned above. The dataset was fitted using Keras Sequential API. Models with this API are built by adding layers upon layers, and our model included these 4 layers:

- **Embedding**: this layer primarily helps in sequencing text data, transforming discrete data, such as words, into continuous vectors so that the neural network can work using the input data.
- **Conv1D**: this is a 1-dimensional convolutional layer used to pre-process sequential data. With the help of known weights, this layer helps to detect patterns and features in data.
- **GlobalMaxPooling1D**: this is another 1-dimensional layer that extracts the most valid features extracted from the Conv1D layer and performs a global max-pooling operation.
- **Dense**: this is the most important layer in a neural network, also known as a fully connected layer. This layer performs a weighted sum of its inputs, applies an activation function, and produces the output, helping with the classification of data by finding the connection between all features fed to it without any input parameters beyond convoluted layers.

After implementation of the CNN model, we also ran a model using an LSTM filter only, for sentiment prediction. Finally, we also ran a model where we added an LSTM filter just before the Dense layer of the first CNN model. SVM and Naïve Bayes models were also run on the dataset for comparison purposes.

The libraries used include Python's Pandas and NumPy, matplotlib for data visualization, NLTK tokenizer's punkt package, and Keras preprocessing module, among others.

Training the Model:
All models were run for 50 epochs, with a batch size of 256.

Results

After having run all our models, these are the accuracies achieved for each model implemented:

MODEL	ACCURACY
CNN	73.58%
LSTM	71.83%
CNN + LSTM	72.47%
SVM	70.52%
Naïve Bayes	68.38%

Fig 2: Accuracy comparison of all tested models.

We can see that the CNN model achieved the highest accuracy among all models, with the implementation of LSTM achieving a slightly lower accuracy value. Adding an LSTM filter at the final stages of the CNN model achieved an accuracy averaging between the separate CNN and LSTM model accuracy values. Classical models such as SVM and Naïve Bayes achieved comparatively lower accuracy percentages compared to all other models. We assume that the confusion matrices for all models showed consistently higher false positive values for the values 'fear' and 'sadness' because the dataset contained the lowest data count for these values.

Conclusion

It can be argued that an LSTM model would be better suited for the purpose of our testing, due to the fact that text or sentences used in SA are usually based on sentences of longer dependencies (sentences which are usually quite long, where a model may lose context the further it works on the sentence to analyze the sentiment or emotion behind it). Though CNN models are mostly used for image classification purposes, the reason we have chosen to use a CNN model is mainly due to the dataset we have chosen to use – a dataset of Twitter posts. Twitter posts are limited to 250 characters per post, and we believe that in the context of shorter sentences a CNN model would be more effective at SA.

Though LSTM models are better at Sequential Data Handling and working with Variable Sequence Lengths, CNN models are more effective at capturing local patterns in data and can make use of Pre-trained Embeddings (such as the Word2Vec model we used). According to our testing, this hypothesis holds true for the dataset used, as we achieved a higher accuracy using our CNN model than we did with any other model. What we can interpret from this data is the notion that CNN models are moderately viable to use for Sentiment Analysis purposes in cases where the dataset may contain sentences or text which is not individually too long or dense, compared to where an LSTM should in-theory perform better than CNN on larger scale text.

<u>REFERENCES</u>

1. Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016, August). Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 225-230).
2. Rehman, A. U., Malik, A. K., Raza, B., & Ali, W. (2019). A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis. *Multimedia Tools and Applications*, *78*, 26597-26613.
3. Yenter, A., & Verma, A. (2017, October). Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In *2017 IEEE 8th annual ubiquitous computing, electronics and mobile communication conference (UEMCON)* (pp. 540-546). IEEE.
4. Luan, Y., & Lin, S. (2019, March). Research on text classification based on CNN and LSTM. In *2019 IEEE international conference on artificial intelligence and computer applications (ICAICA)* (pp. 352-355). IEEE.
5. Gandhi, U. D., Malarvizhi Kumar, P., Chandra Babu, G., & Karthick, G. (2021). Sentiment analysis on twitter data by using convolutional neural network (CNN) and long short term memory (LSTM). *Wireless Personal Communications*, 1-10.