Sentimental Analysis of Text to Emojis

Srilekha Divyakolu
Panther ID: 002430959
College of Arts and Sciences
Georgia State University, Atlanta, GA-30302
sdivyakolu1@student.gsu.edu

Bhavya Induri
Panther ID: 002473152
College of Arts and Sciences
Georgia State University, Atlanta, GA-30302
binduri1@student.gsu.edu

Abstract

Emotions can be expressed in many ways that can be seen such as facial expression and gestures, speech or by written text. Each emotion is involved with sentiments like anger, happiness, excitement, sadness, anxiousness, hatred, irritation, content etc. Through this project we will analyze the text given as the input and classify the text into one such sentiment and then map them to related emoji's which we use on a daily basis. This system can be used in recommendation of emojis for the text typed in chatting apps. Our system uses deep learning models to learn and which is proved to have better accuracy when compared with existing machine learning systems.

1 INTRODUCTION

A variety of NLP (Neuro Linguistic Programming) tasks are limited by scarcity of manually annotated data. Therefore, co-occurring emotional expressions have been used for distant supervision in social media sentiment analysis and related tasks to make the models learn useful text representations before modeling these tasks directly. Distant supervision on noisy labels often enables a model to obtain better performance on the target task. We show that extending the distant supervision to a more diverse set of noisy labels enables the models to learn richer representations of emotional content in text, thereby obtaining better performance on benchmarks for detecting sentiment, emotions and sarcasm. We show that the learned representation of some pretrained models generalizes across 3 to 4 domains. Emojis are not always a direct labeling of emotional content. For instance, a positive emoji may serve to disambiguate an ambiguous sentence or to complement an otherwise relatively negative text. A similar duality in the use of emotional hashtags such as #nice and #lame. Our project tries to show that emojis can be used to classify the emotional content of texts accurately in many cases. Also emotion detection can be used in human computer interaction and recommender systems to produce interactions or recommendations based on the emotional state of the user

2 Problem and Goal

We have some limitations in this analysis like: a)Using emotion keywords is a

straightforward way to detect associated emotions, the meanings of keywords could be multiple and vague, as most words could change their meanings according to different usages and contexts. Moreover, even the minimum set of emotion labels (without all their synonyms) could have different emotions in some extreme cases such as ironic or cynical sentences. b) Without keywords :Keyword-based approach is totally based on the set of emotion keywords. Therefore, sentences without any keyword would imply that they do not contain any emotion at all, which is obviously wrong. c)Syntax structures and semantics also have influences on expressed emotions. For example, "I laughed at him" and "He laughed at me" would suggest different emotions from the first person‟s perspective. As a result, ignoring linguistic information also poses a problem to keyword-based methods. d)Learning-based methods can automatically determine the probabilities between features and emotions but the methods still need keywords, but in the form of features. The most intuitive features may be emoticons which can be seen as user‟s emotion annotations in the texts. The cascading problems would be the same as those in keyword-based methods. So, to overcome the problems we are trying to build best-models which analyze the text given and outputs the best emoji that fits to the text given.

3 Related Work

[1] implemented the text to emoji model to address the above problem using LSTM model but they only tried Bi-Directional layer of LSTM for which we tried our model in both bi-directional and unidirectional to know which is better and why [1] used only Bi-directional layer. [2] did the similar work but used machine learning models like SVM, Random Forest, Naive Bayesian, Decision tree, linear SVM models in which we have seen that Naive Bayesian and Decision Tree gave better accuracy for our data, so we included that into our system for comparison.

4 Data and Algorithm used

Here, we discuss the type of data used for training and testing and also the type of model used to achieve our goal.

4.1 Data used for this project

For this project, we use multiple datasets as there isn't one particular dataset which has all categories. So we will use data-world's sentimental analysis in text dataset of twitter which has 13 categories like sadness, empty, neutral, worry, enthusiasm, surprise, love, fun, hate, happiness, boredom, relief and anger (https://data.world/crowdflower/sentiment-analysis-in-text) and also we will kaggle dataset for twitter's sentimental analysis which has categories like positive, negative and neutral (https://www.kaggle.com/c/twitter-sentiment-analysis2/data) and also Dataturks's distress classification of new articles titles which has categories like distress or not (https://dataturks.com/projects/clint.albertyn/Fin%20Distress%20Articles) etc...

4.2 Algorithms used

As our system needs pre-labeled data to train, our model is said to be a supervised

learning model. We have used LSTM for deep learning models and also machine models like Naive Bayesian model and decision tree and compare the accuracy of both the models and give the resulting model having highest accuracy. In addition after getting output from the models we feed it as input to a map of emojis which are divided as per categories in the dataset, so the output of map will be emojis for that sentiment derived from the model.

The architecture we followed for machine learning models is as follows in Figure 1.

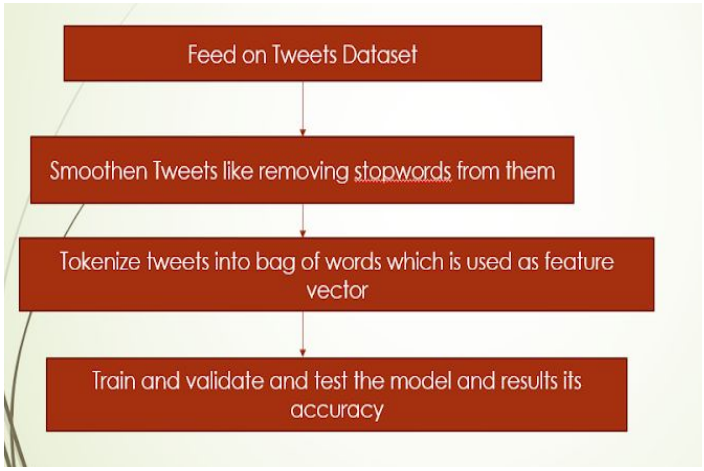


Figure 1

Our architecture for LSTM model is in the following Figure 2 which contains the stages of how we performed preprocessing to data and populated weights to train and test data based on them.

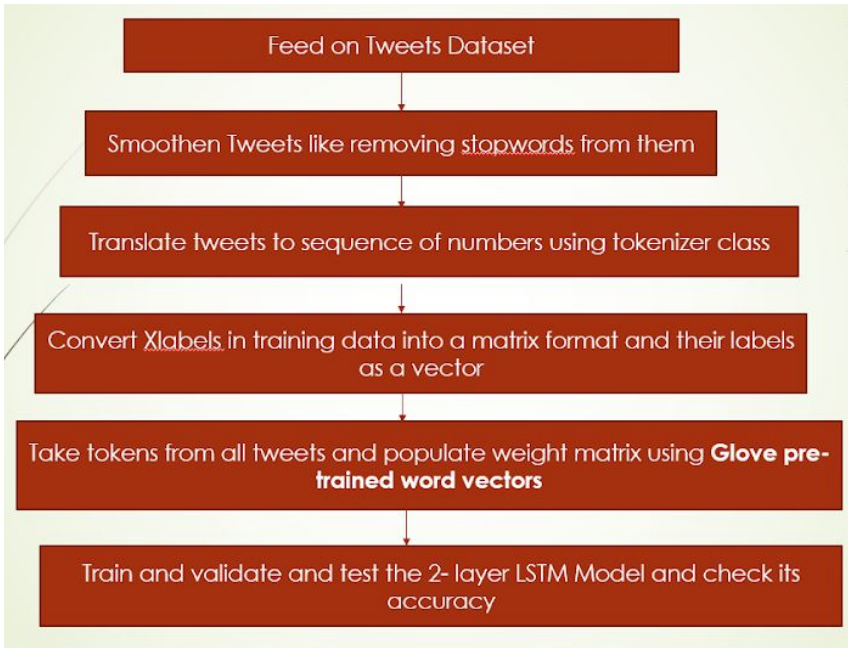


Figure 2

The last step in the above architecture which is Bi-layer LSTM, its individual architecture

looks like the below Figure 3.

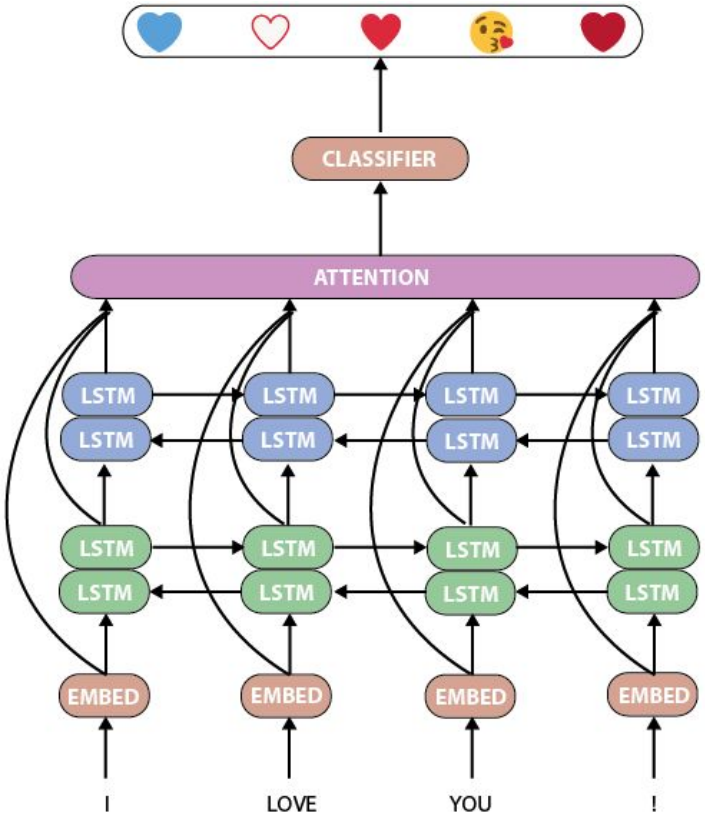


Figure 3.

5 Experiment Results

We have tested our model on both model-1 and model-2 i.e. both deep learning and machine learning models. We will show how our results folded out.

In the first phase of our experiment, we visualized the distribution of data to training and testing data and on which we observed the case of overfitting the data to the model which resulted in low accuracy and more error. The data distribution graph where overfitting occured is as below in Figure 4.

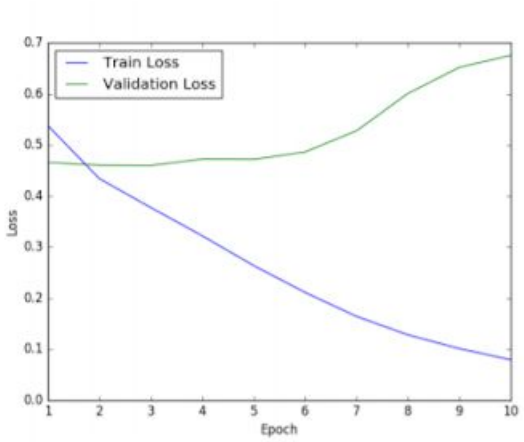


Figure 4.

The accuracy results of each models are shown as below:

1. Naive Bayesian

```
dsril@DESKTOP-3SAIIVF /cygdrive/d/fall_2019/project/emoji-prediction-master/src
$ python deploy_model.py nb
Importing Naive Bayes...
Successfully Imported Naive Bayes.
Running...
Execution Complete. Accuracy:28.310000000000002 %
```

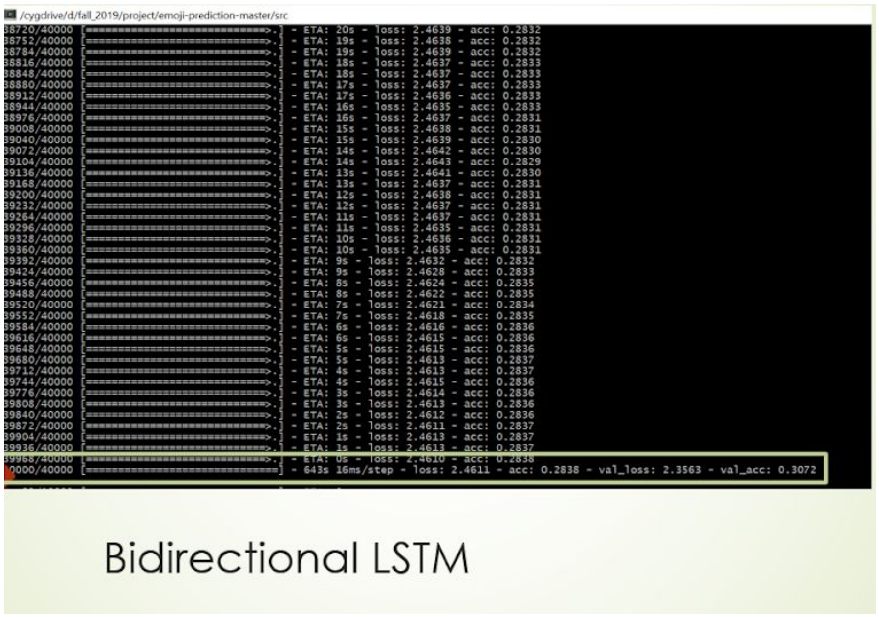
2. Decision Tree

```
dsril@DESKTOP-3SAIIVF /cygdrive/d/fall_2019/project/emoji-prediction-master/src
$ python decision_tree_classifier.py
Accuracy of Decision tree is: 20.0

dsril@DESKTOP-3SAIIVF /cygdrive/d/fall_2019/project/emoji-prediction-master/src
$
```

3. Unidirectional Bi-layer LSTM

4. Bidirectional Bi-layer LSTM



When comparing the results of all the models

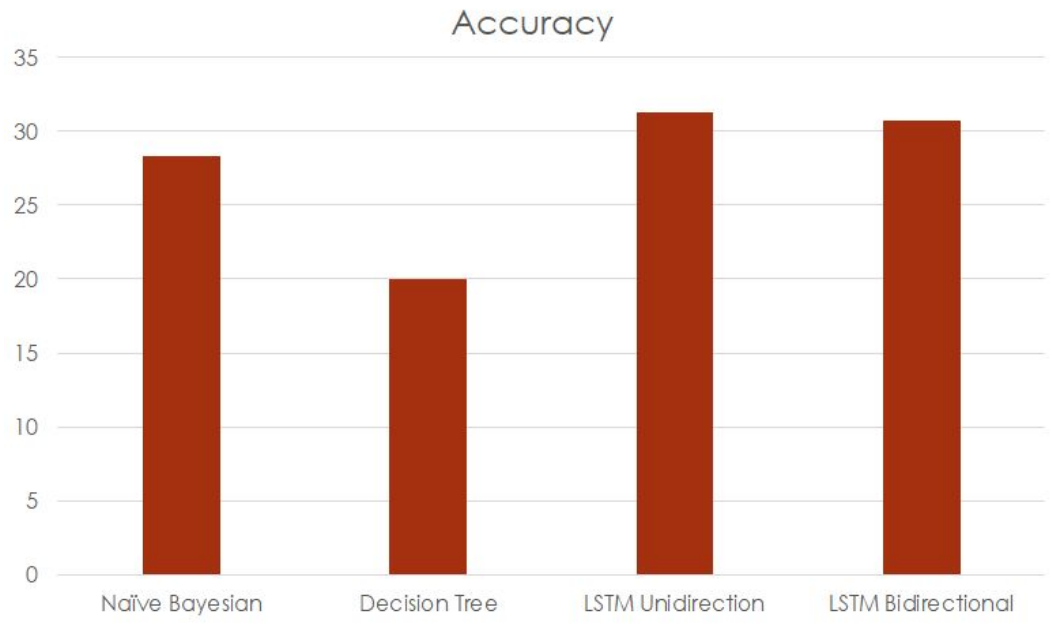


Figure 5

From the above figure, we found that the reason for the less accuracy is due to overfitting our models with data, so we expanded our model with data by removing the overfitting among them. Here are the results for the LSTM model after resolving the issue of overfitting and expansion of model with learning rate and normalization.

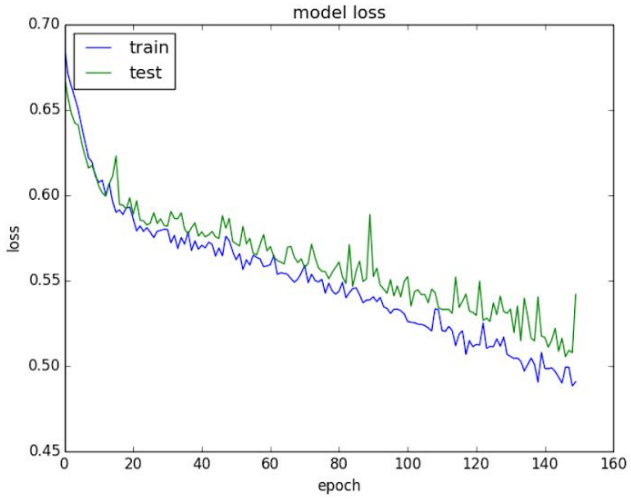


Figure 6

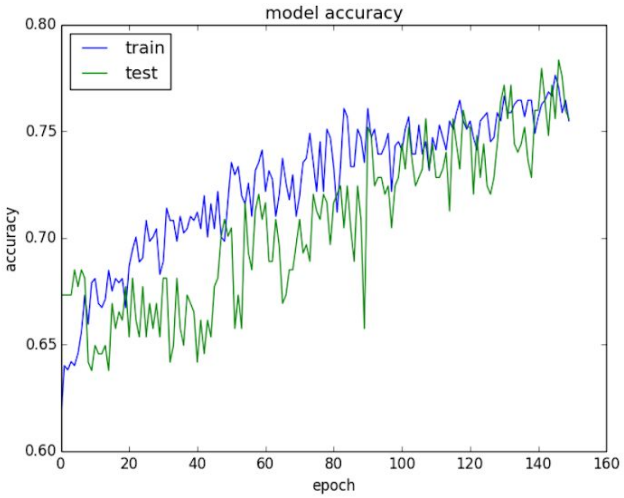


Figure 7

We can conclude from Figure 6 and 7 from this section that we have achieved better accuracy around **78%** after resolving our previous issues with LSTM. Both LSTM Uni and Bi-directional models are giving around the same accuracy. And for machine learning models , we achieved **51.43%** accuracy for Naive Bayesian and for Decision Tree we achieved **45.85%** accuracy.

The accuracy results of all models for comparison are as below in Figure 8.

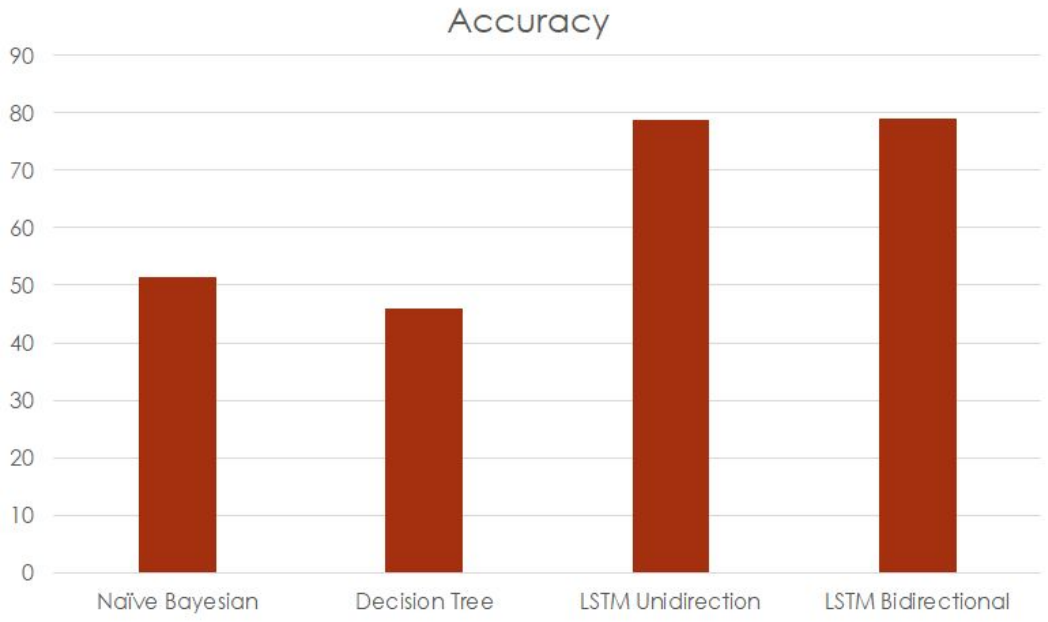


Figure 8.

These models can be better understood through UI as follows in Figure 9.

```
In [33]:
t1 = "I am happy"
t2 = "I don't like it"
t3 = "My car skidded on the wet street"
t4 = "The cat is so exciting"

texts = [t1, t2, t3, t4]
for text in texts:
    features = create_feature(text, nrange=(1, 4))
    features = vectorizer.transform(features)
    prediction = grid_obj.predict(features)[0]
    print("{} {}".format(emoji_dict[prediction], text))
```

```
😂 I am happy
😢 I don't like it
😱 My car skidded on the wet street
😒 The cat is so exciting
```

Figure 9.

In the above figure, we can see that for a given text, the output will be in the form of a map where text is linked to its corresponding emoji.

6 Conclusion

From Figure 8, we can conclude that though deep learning models take more time, memory and space for processing the training data, they showed **better** accuracy compared to Machine learning models.

7 Future Works

In Future, we can improve our model with cross-matching LSTM with other RNN layers like GRU, search based RNN to increase the model accuracy and its time and space

complexity issues. We can also embed this feature in chat application in the category of recommendation of emojis whenever the user types a text.

8 Division of Tasks

Tasks	**Time scheduled**
1. Data cleaning to clear of bias and preprocessing	Srilekha
2. Model-1 LSTM Bi-directional learning	Srilekha
3. Model-1 LSTM Uni-directional learning	Bhavya
4. Mode-2 Naive Bayesian	Bhavya
5. Mode-2 Decision Tree	Srilekha
6. calculate accuracy and error and compare both models	Srilekha, Bhavya
7. Slides, Demo, Presentation	Srilekha, Bhavya
8. Final report	Srilekha, Bhavya

9 References

[1] Bjarke Felbo1 , Alan Mislove2 , Anders Søgaard3 , Iyad Rahwan1 , Sune Lehmann4, *Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm.*

[2] Shiv Naresh Shivhare1 and Prof. Saritha Khethawat, *EMOTION DETECTION FROM TEXT.*

[3]https://www.microsoft.com/developerblog/2015/11/29/emotion-detection-and-recognition-from-text-using-deep-learning/

[4]http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[5]https://scikit-learn.org/stable/modules/naive_bayes.html

[6]https://www.liip.ch/en/blog/sentiment-detection-with-keras-word-embeddings-and-lstm-deep-learning-networks

[7]https://repositori.upf.edu/bitstream/handle/10230/32660/barbieri_MM16_emoj.pdf?sequence=1&isAllowed=y