SENTIMENT ANALYSIS: Multi-level Multi-Class Emotion recognition

Donia Ghazy donia.ghazy@aucegypt.edu
Marwan Eid marwanadel99@aucegypt.edu
Mohammed Abuelwafa mohammedabuelwafa@aucegypt.edu
CSCE 4604-02
December 17, 2020

I. Abstract

***Abstract*—Multi-class sentiment analysis is a topic of great interest. Sentiment Analysis with binary classification is performed with with high accuracies reaching an accuracy of 97%. However, the accuracy of multi-class emotion recognition is low varying from 55-60%. The selection of labels, number of classes and the features of the dataset regarding the balancing of the dataset and the frequency of each label are important factors regarding the determination of the accuracy of the multi-class classification process. Given the following facts, the authors of this paper hypothesized that multi-level classification could possibly lead to better results. The authors decided to adopt a mixture of experts approach where they train a classifier to classify positive emotions and a classifier for classifying negative emotions. Each of the classifier is trained separately given the corresponding output of a binary classifier. In this paper, the authors assumed that separating the positive and negative emotions in the dataset by parsing and adding some noise would be sufficient for testing the hypothesis. The hypothesis has been proved right by trying it on the following dataset that has shown an accuracy of 62.5% previously in the works of previous researchers shown in [2] since it had shown an overall accuracy of 72.45% using the proposed architecture. It has been proved that using a multi-level architecture is better than a single-level architecture in emotion recognition with multi-class classification.**

***Index Terms*—CNN, RNN, ANN, NLP, dataset, TF-IDF, tensorflow, keras, pandas, numpy**

II. Introduction

2.1 Problem definition:

Being in a data-driven world, it has become a necessity to analyze data in efficient and accurate ways. One important type of information to be analyzed is textual information. People find text all around the world in comments on social media, ratings, advertisements, captions... etc. Emotion detection and recognition is a recent field of natural language processing (NLP) which is closely related to sentiment analysis. Although sentiment analysis detects positive or negative feeling of an input textual data, emotion recognition classifies the sentiment of the input text into a category of given emotions such as happiness, fear, anger or sadness for example. Hence, one can say that emotion recognition is multi-class sentiment analysis. The goal of this paper is to implement multi-level multi-class emotion recognition. The process of emotion recognition works as following: Given an input English sentence, i.e., a short text sequence, the goal is to produce an output class label corresponding to the emotional state of the sentence from a small list of emotional classes.

2.2 Sentiment analysis applications:

Emotion recognition and sentiment analysis is a topic of great importance due to its various applications. These applications include but are not limited to enhancing user experience by analyzing customers reactions to different products (movie ratings for example). By knowing the user's needs, they can be satisfied. Another application is monitoring social media which is a goldmine of consumer stories and opinion data which can be of great use to advertisements and advertisement agencies. However, this can be a bit challenging since social media textual information are often complex containing abbreviation, acronyms and emojis. Another application is mental health analysis where being able to detect humans' emotions based on their writings aid in therapy and monitoring the mental health of the patients. Last, but not least, product analysis and market research where it has proven that sentiment analysis and emotion recognition is useful in business analysis and monitoring the effectiveness of products and marketing Campaigns in comparison to the competitors.

2.3 Narrowing the scope:

Generally, there are three variants of the emotional recognition problem: Binary classification where the emotions are generally classified to positive emotions and negative emotions figure [1], ternary classification where the emotions are classified to positive, neutral and negative emotions figure [2]. Multiclass classification where there is a small list of specific emotions figure [3].

The multiclass sentiment analysis is the most complex variant of the three variants. Some of the complexity added to this variant comes from the fact that the more classes used, the less the accuracy becomes. Moreover, the labelling of the emotions sometimes uses overlapping

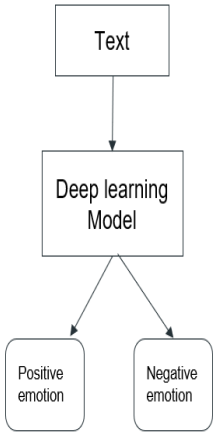

Figure 1. Binary sentiment Analysis

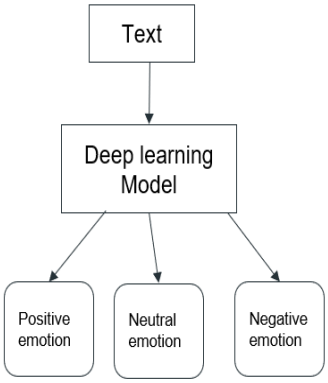

Figure 2. Ternary sentiment Analysis

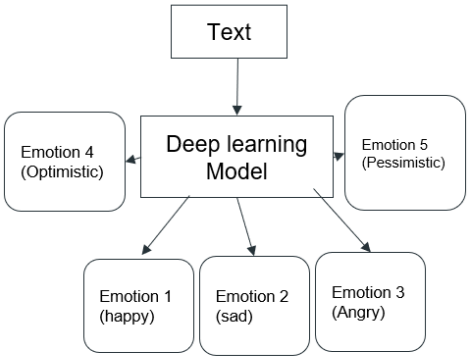

Figure3. Multi-class sentiment Analysis

emotions to an extent that the distinction between them is hard for humans like using depressed, sad, and disappointed for example. This fact has come to the attention of multiple researchers previously and this is clear in the following labelling from different papers figure [4].

Papers	Categories	Emotion Model	Approaches
(Strapparava and Mihalcea, 2008)	Anger, Disgust, Fear, Joy, Sadness, Surprise	Categorical	Lexical-based
(Gill et al., 2008)	Anger, Fear, Surprise, Joy, Anticipation, Acceptance, Sadness, Disgust	Categorical	Lexical-based
(Balahur et al., 2011)	Anger, Disgust, Fear, Guilt, Joy, Sadness, Shame	Categorical	Lexical-based
(Sykora et al., 2013)	Anger, Confusion, Disgust, Fear, Happiness, Sadness, Shame, Surprise	Categorical	Lexical-based
(Wang and Zheng, 2013)	Anger, Disgust, Fear, Guilt, Joy, Sadness, Shame	Categorical	Lexical-based
(Alm et al., 2005)	Anger, Disgust, Fear, Happiness, Sadness, Positively Surprise, Negatively Suprise	Categorical	Supervised Learning-based
(Strapparava and Mihalcea, 2008)	Anger, Disgust, Fear, Joy, Sadness, Surprise	Categorical	Supervised Learning-based
(Balabantaray et al., 2012)	Anger, Disgust, Fear, Happiness, Sadness, Surprise	Categorical	Supervised Learning-based
(Roberts et al., 2012)	Anger, Disgust, Fear, Joy, Sadness, Surprise, Love	Categorical	Supervised Learning-based
(Suttles and Ide, 2013)	Anger, Disgust, Fear, Happiness, Sadness, Surprise, Trust, Anticipation	Categorical	Supervised Learning-based
(Hasan et al., 2014b)	Happy-Active, Happy-Inactive, Unhappy-Active, Unhappy-Inactive	Dimensional	Supervised Learning-based
(Strapparava and Mihalcea, 2008)	Anger, Disgust, Fear, Joy, Sadness, Surprise	Categorical	Unsupervised Learning-based
(Agrawal and An, 2012)	Anger, Disgust, Fear, Happiness, Sadness, Surprise	Categorical	Unsupervised Learning-based
(Calvo and Kim, 2013)	Anger-Disgust, Fear, Joy, Sadness	Categorical	Unsupervised Learning-based
(Calvo and Kim, 2013)	Anger-Disgust, Fear, Joy, Sadness	Dimensional	Unsupervised Learning-based

Table 1: Emotion Detection approaches

The authors of this paper researched and found out that binary classification is done with high accuracies from 85-97%. However, fine-grained classification has an accuracy of 55-60%. Hence, it came to the authors' attention to propose a multi-level multiclass architecture where two separate classifiers are trained separately. One is trained on positive emotions whereas the other is trained on negative emotions. The input to each classifier is the classified corresponding input of the binary classifier. The general architecture is demonstrated in figure [4].

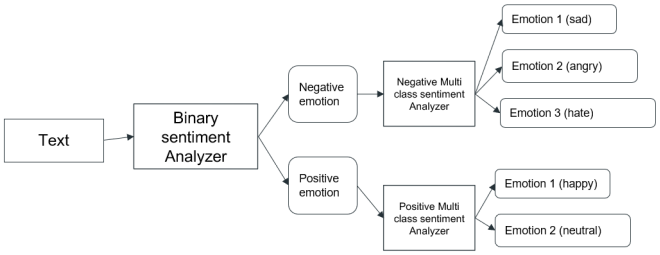

Figure 4.General architecture of binary classifier

To achieve the goal of this paper, it is divided into six main sections; existing solutions, architecture, experiment, results, conclusion, and future work. In the existing solutions section, some of the already existing models would be introduced in depth as we will present the accuracy of each of them. Furthermore, in the architecture section, the details of this paper's architecture would be analyzed based on the used parameters and the used layers. In addition, in the experiment part, all the preprocessing and training steps would be presented. In the results section, the testing accuracies of each model will be presented. Finally, in the conclusion section, a quick summary about the whole paper is introduced.

III. Existing solutions

There is much research that has been conducted on sentiment analysis to provide systems that are more accurate and reliable. The core task of sentiment analysis is the automatic identification of opinionated text in documents. The majority of the previous research used either rule based or statistical machine learning approaches for opinion mining and sentiment analysis. In this section, some techniques on sentiment analysis and their applications would be discussed.

Ibrahim et al. [12] introduced a detailed survey of different techniques used for opinion mining and sentiment analysis. The paper used the document-level sentiment classification which is a binary classification task of labeling a document as expressing either an overall positive or negative opinion. The task of this classification is to predict whether reviewer wrote a positive or negative review, based on an analysis of the text of the review. Othman et al. [12] discussed that there are two types of classification techniques that have been used in document-level sentiment classification, supervised method and unsupervised method. The following table (table 2.) shows some of the selected previous studies dealing with document-level sentiment classification.

Paper	Technique	Features	Dataset	Type
[1]	SVM, Naive Bayes	Unigram, Bi-grams, Trigrams	Restaurant Review	Supervised
[2]	SVM, Naive Bayes, Maximum Entropy	Unigrams, bi-grams, adjective, position of words	Movie Review	Supervised
[3]	SVM, Naive Bayes, character based N-gram model	Unigram Frequency	reviews to travel destination	Supervised
[4]	SVM , Rule-based Classifier	POS tag, N-grams	movie reviews, product reviews, MySpace comments	Supervised Learning and Rule-based classification
[5]	SVM	Unigram, bigram and extraction pattern feature	Movie Review, MPQA dataset	Supervised
[6]	SVM	Adjective word frequency, percentage of appraisal groups	Movie Review	Supervised
[7]	PMI-IR	adjectives and adverbs	Automobile, bank, movie, travel reviews	Unsupervised
[8]	Association Rule	adjectives and adverbs	Movie Review	Unsupervised
[9]	Dictionary based approach	Adjectives , Nouns, verbs , Adverbs, Intensifier , Negation	Movie Review Camera Review Epinions	Unsupervised

Table 2. previous studies on document-level sentiment analysis

Turney et al. [6] proposed an unsupervised algorithm that uses semantic orientation of the phrases for classification of reviews. The lexicon-based approach determines the polarity or sentiment using some function of opinion words in the document or sentence. The algorithm used has three main steps: (1) the clean text phase which phrases containing adjectives or adverbs were extracted, (2) the core phase which uses PMI-IR to estimate the semantic orientation of each phrase, and (3) the result phase which classifies the review based on the average semantic orientation of the phrases [3]. This algorithm attains an average accuracy of 74%.

Hana et al. [7] discussed that the sentiment analysis algorithm that uses RNN and LSTM. These are heavily used in NLP problems because of their ability to retain information from previous words and to learn the significance of the order of sequential words in the input sequence. In theory, classical RNN can keep track of arbitrary long-term dependencies in the input sequences. However, practically training an RNN through back-propagation could allow exploding the gradients or vanishing them, which is solved by LSTM units where a unit has an input gate, an output gate, and a forget gate to keep track of the dependencies between the elements in the input sequence. Gated feedback recurrent neural networks extend the existing approach of stacking multiple RNN layers by allowing and controlling signals to flow from upper recurrent layers to lower layers using a global gating unit for each pair of layers, where such signals exchanges are gated adaptively based on the previous hidden states and the current input [7].

The Recursive Neural Tensor Networks (RNTNs) have a tree structure with a neural net at each node; i.e. nodes are combined into parents using a weight matrix that is shared across the whole network. Then, word vectors are used as features and serve as the basis of sequential classification. They are grouped into sub-phrases, and the sub-phrases are combined into a sentence that can then be classified. This model outperformed its preceding methods on several metrics and pushed the state-of-the-art in single sentence binary classification from 80% up to 85.4% [8].

Furthermore, there is another promising algorithm which is called Attention-based Neural Networks (ANNs). It creates shortcuts between a single context vector and the source input and, thus, allowing the network to know where to focus while it is performing its task. Some proposed models incorporating such neural networks consist of two parts: bidirectional LSTM with self-attention mechanisms which is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence [8].

Bidirectional Encoder Representations from Transformers (BERT) is another solution that is used in sentiment analysis. It relies on bidirectional LSTM where bidirectional training is performed on sequences instead of unidirectional training. The latest model utilizing this approach pushed the state-of-the-art accuracy of fine-grained sentiment analysis to 55.5% [8].

Huang. et al. [9] introduced an a sequence of enhanced multi-task models for sentiment analysis, where each model is an enhanced version of the former and the last model is

the best. By combining a pooling layer and a bidirectional RNN, this model can take the input data and comprehensively extract the semantic text information. In addition, the attention mechanism linking the task-specific layer and the shared layer empowers the model to intelligently select effective features from the shared layer.

Lee et al. [11] presented an algorithm based on Convolutional Neural Network (CNN) for text sentiment analysis. This model takes advantages from deep learning, fuzzy modeling and neural networks and then propose a hybrid deep learning based fuzzy-neural model, Fuzzy Convolutional Neural Network, which integrates fuzzy logic and CNN, for text sentiment classification. FCNN can generate more reasonable features that achieve better classification accuracies on emotional data as compared to conventional approaches such as CNN. The proposed model addresses the problems of data ambiguities with linguistic labels that have relevance for emotion identification in sentiment analysis tasks.

IV. Architecture

Since high accuracy for the binary classification took place, the authors decided to separate the data set to positive data set (labels 1 for happy and 0 for neutral) and negative data set (labels 2 for sad, 3 for hate, 4 for anger). The statements had some human error in the labelling which was clear in the existence of duplicates and wrong labeling for some statements. the authors depended on this error as the binary classification error.

The positive and negative classifiers depended on Convolutional Neural Networks for feature extraction after performing TF-IDF on the text in order to get the word embedding vectors of the text, then three fully connected layers were used for classification. The architecture is designed by the authors of this paper in order to compare the results of the overall classification accuracy in contrast with multiclass classifiers with positive and negative emotions combined.

The architecture of the negative classifier followed this pattern:

1) **Conv1D**: 64 filters each of size 3 with RELU Activation function and input shape of $(Nx500)$ and padding was applied to preserve the dimensions.

2) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

3) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

4) **Maxpooling 1D**: with size 2 and padding was applied to preserve the dimensions.

5) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

6) **Maxpooling 1D**: with size 2 and padding was applied to preserve the dimensions.

7) **Flattening layer**

8) **Fully connected layer**: 64 nodes with RELU activation function and $L2$ regularization = $1e-2$

9) **Fully connected layer**: 64 nodes with RELU activation function and $L2$ regularization = $1e-2$

10) **Fully connected layer**: 3 nodes with SoftMax activation function and $L2$ regularization = $1e-2$

The architecture of the negative classifier followed this pattern:

1) **Conv1D**: 64 filters each of size 3 with RELU Activation function and input shape of $(Nx500)$ and padding was applied to preserve the dimensions.

2) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

3) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

4) **Maxpooling 1D**: with size 2 and padding was applied to preserve the dimensions.

5) **Conv1D**: 128 filters each of size 5 with RELU Activation function and padding was applied to preserve the dimensions.

6) **Maxpooling 1D**: with size 2 and padding was applied to preserve the dimensions.

7) **Flattening layer**

8) **Fully connected layer**: 64 nodes with RELU activation function and $L2$ regularization = $1e-2$

9) **Fully connected layer**: 64 nodes with RELU activation function and $L2$ regularization = $1e-2$

10) **Fully connected layer**: 2 nodes with SoftMax activation function and $L2$ regularization = $1e-2$

V. Dataset

Our training and validation testing set is originally composed of $47,288$ tweets from twitter with labelled emotions of five classes: neutral, happy, sad, anger and hate. Due to

the rareness of a labelled data set with positive emotions, the authors decided to treat the neutral category as a positive emotion for this specific problem.

VI. Experiment

Data preprocessing:

1) The data was filtered from duplicate and meaningless cells (aka empty statements with a label). Resulting in unique $43,924$ tweets for training and validation. The distribution of the labels in the following dataset is shown in figure 5.

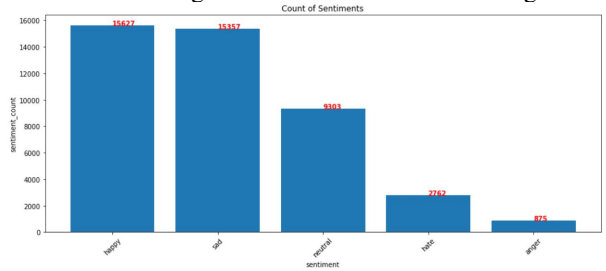

Figure 5. Distribution of the labels in the dataset

2) The data was divided to two separate data sets, one for positive and the other is negative:
   a) $24,930$ for positive emotions training and validation.
   b) $18,994$ for negative emotions training and validation.
3) The tweets were filtered from emojis, punctuation and special characters using a `clean_text`function that the authors tailored for this dataset.
4) The tweets were changed to lowercase.
5) 70 % of the data was used for training and 30% was used for validation.

After preprocessing the data properly, the authors needed to move forward and have some meaningful usage of this raw, text, data because statistical approaches since machine learning and deep learning work well with numerical data. However, natural language consists of words and sentences. Therefore, before the authors started to build a sentiment analysis model, they needed to convert text to numbers. There are different approaches that have been developed for converting text to numbers. Bag of Words, N-grams, and Word2Vec model are some of them. In this model, the authors chose to work with Bag of Words with TF-IDF scheme.

In the bag of words approach, the vocabulary of all the unique words in all the documents is formed. This vocabulary serves as a feature vector. Every word is given equal importance. However, in the TF-IDF, the words that occur more frequently in one document and less frequently in other documents should be given more importance as they are more useful for classification. TF-IDF is a product of two terms: TF and IDF.

Term Frequency (TF) is equal to the number of times a word occurs in a specific document. It is calculated as:

TF= Frequency of a word in the document / Total words in the document

Inverse Document Frequency (IDF) for a specific word is equal to the total number of documents, divided by the number of documents that contain that specific word. The log of the whole term is calculated to reduce the impact of the division. It is calculated as:

IDF = Log ((Total number of docs) / (Number of docs containing the word))

Fortunately, the authors did not have to get the values of TF and IDF analytically themselves, but instead, they used The **TfidfVectorizer** class from the **sklearn.featureextraction.text** module which can be used to create feature vectors containing TF−IDF values. This can be explained further in the code snippet as shown in figure 6.

```
max_feature = 500
vectorizer = TfidfVectorizer(max_features=max_feature,
                             min_df=4, max_df=0.8,
                             ngram_range=(1,1))

x_train = vectorizer.fit_transform(x_train).toarray()
x_test = vectorizer.transform(x_test).toarray()
```

Figure 6. Feature vectors creation

The attribute **maxfeatures** specifies the number of most occurring words for which you want to create feature vectors. Less frequently occurring words do not play a major role in classification. Therefore, the authors only retain 500 most frequently occurring words in the dataset. The **mindf** value of 4 specifies that the word must occur in at least 4 documents. Similarly, **maxdf** value of 0.8 percent specifies that the word must not occur in more than 80 percent of the documents. The rationale behind choosing 80 percent as the threshold is that words occurring in more than 80 percent of the documents are too common and are less likely to play any role in the classification of sentiment. Finally, to convert the dataset into corresponding TF-IDF feature vectors, the authors called the **fit_transform** method on **TfidfVectorizer** class and passed it to the preprocessed dataset.

The next step, the authors chose a suitable architecture as discussed in the previous section in detail and started to train the model and evaluate its performance on the testing dataset. The authors used Adam optimizer in this model with learning rate of $5e-4$. This value was based on fine and coarse search for the most optimal learning rate to produce the best weights that result in getting the minimum loss possible. This can be shown further in the code snippet as shown in figure 7.

```
optimizer=Adam(lr=5e-4)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy'
              ,metrics=['accuracy'])
```

Figure 7. Adam optimization

During training the model, the authors decided to stop the training at epoch number seven in the positive classifier and at epoch number eight in the negative classifier as the model starts to over-fit and the training loss started to increase instead of decreasing. The losses in the both cases are shown in figures 8,9.

Figure 8. Training stop point for positive classifier

Figure 9. Training stop point for negative classifier

VII. Results

After training the model with the proposed architecture, the authors ran the negative and the positive models for ten epochs and witnessed the following training and validation accuracies at each epoch.

```
history=model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(x_test,y_test))

Train on 17451 samples, validate on 7479 samples
Epoch 1/10
17451/17451 [==============================] - 50s 3ms/step - loss: 1.1935 - accuracy: 0.6369 - val_loss: 0.8972 - val_accurac
y: 0.6541
Epoch 2/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.7899 - accuracy: 0.6740 - val_loss: 0.7186 - val_accurac
y: 0.6776
Epoch 3/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.6690 - accuracy: 0.6832 - val_loss: 0.6465 - val_accurac
y: 0.6721
Epoch 4/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.6199 - accuracy: 0.6858 - val_loss: 0.6159 - val_accurac
y: 0.6819
Epoch 5/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.6012 - accuracy: 0.6910 - val_loss: 0.6057 - val_accurac
y: 0.6877
Epoch 6/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.5931 - accuracy: 0.6949 - val_loss: 0.6050 - val_accurac
y: 0.6879
Epoch 7/10
17451/17451 [==============================] - 52s 3ms/step - loss: 0.5898 - accuracy: 0.6983 - val_loss: 0.6023 - val_accurac
y: 0.6889
Epoch 8/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.5883 - accuracy: 0.6978 - val_loss: 0.6031 - val_accurac
y: 0.6847
Epoch 9/10
17451/17451 [==============================] - 50s 3ms/step - loss: 0.5877 - accuracy: 0.7033 - val_loss: 0.6021 - val_accurac
y: 0.6853
Epoch 10/10
17451/17451 [==============================] - 51s 3ms/step - loss: 0.5806 - accuracy: 0.7060 - val_loss: 0.5978 - val_accurac
y: 0.6853
```

Figure 10. Training process for positive classifier

```
history=model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=1, validation_data=(x_test,y_test))

Train on 17094 samples, validate on 1900 samples
Epoch 1/10
17094/17094 [==============================] - 61s 4ms/step - loss: 1.2706 - accuracy: 0.8253 - val_loss: 0.9793 - val_accurac
y: 0.8447
Epoch 2/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.8455 - accuracy: 0.8680 - val_loss: 0.7804 - val_accurac
y: 0.8726
Epoch 3/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.6956 - accuracy: 0.8857 - val_loss: 0.6759 - val_accurac
y: 0.8768
Epoch 4/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.6003 - accuracy: 0.8897 - val_loss: 0.6193 - val_accurac
y: 0.8768
Epoch 5/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.5393 - accuracy: 0.8916 - val_loss: 0.5618 - val_accurac
y: 0.8779
Epoch 6/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.4942 - accuracy: 0.8940 - val_loss: 0.5360 - val_accurac
y: 0.8774
Epoch 7/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.4662 - accuracy: 0.8941 - val_loss: 0.5360 - val_accurac
y: 0.8737
Epoch 8/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.4435 - accuracy: 0.8937 - val_loss: 0.5068 - val_accurac
y: 0.8737
Epoch 9/10
17094/17094 [==============================] - 80s 5ms/step - loss: 0.4269 - accuracy: 0.8941 - val_loss: 0.4958 - val_accurac
y: 0.8779
Epoch 10/10
17094/17094 [==============================] - 79s 5ms/step - loss: 0.4147 - accuracy: 0.8943 - val_loss: 0.4873 - val_accurac
y: 0.8789
```

Figure 11. Training process for negative classifier

In addition, these models reached testing accuracy of 87.8% on the negative classifier and testing accuracy of 68.5% on the positive classifier as shown in the following screenshots. This results in a weighted average of 72.45 % calculated as follows:

weighted accuracy= (0.6852 x 7479 + 0.8789 x 1900)/ 9379 = 72.45%

```
[49]:
#Evaluate the scores calculated
evaluate = model.evaluate(x_test, y_test, batch_size=128)
print('The testing accuracy is : ',evaluate[1])

1900/1900 [==============================] - 3s 1ms/step
The testing accuracy is :  0.878947377204895
```

```
[49]:
#Evaluate the scores calculated
evaluate = model.evaluate(x_test, y_test, batch_size=128)
print('The testing accuracy is : ',evaluate[1])

7479/7479 [==============================] - 6s 740us/step
The testing accuracy is :  0.6852520108222961
```

Figure 12. Testing accuracies

The following diagrams shows the comparison between training and testing accuracies and the losses for the both models.

for the negative classifier model:

Figure 13.Training and Testing Accuracy for negative classifier

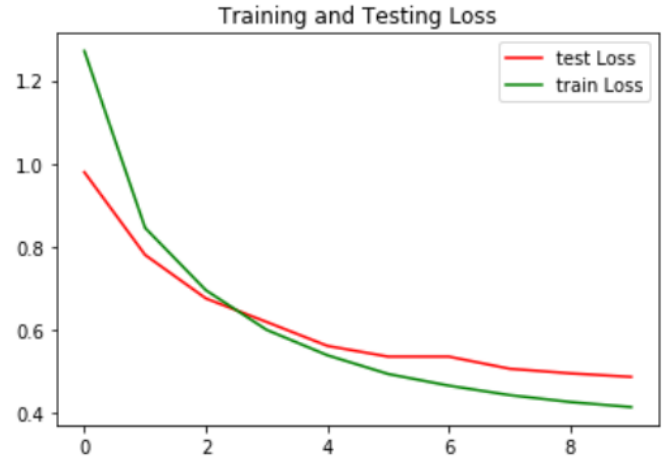

Figure 14. Training and Testing Loss for negative classifier

For the positive classifier model:

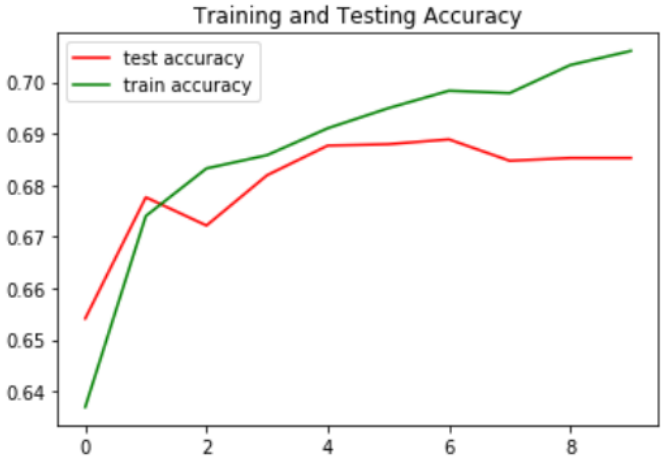

Figure 15. Training and Testing Accuracy for positive classifier

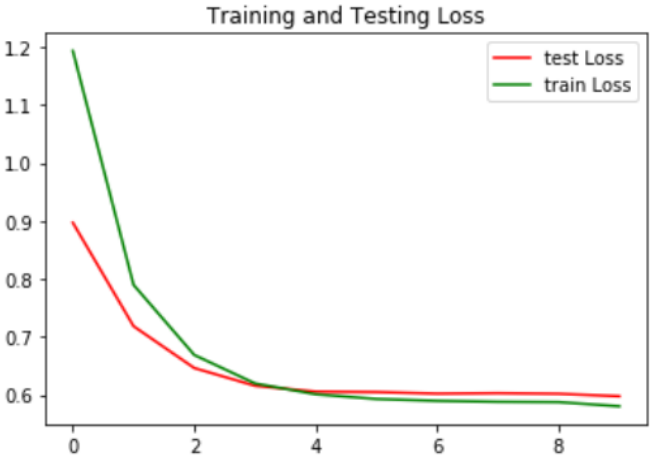

Figure 16. Training and Testing loss for positive classifier

VIII. Conclusion

The analysis of textual data has been in the interest of researchers in the recent years. Multi-class emotion recognition is the most complex variant of the problem and the existing solutions have an accuracy varying from $55-60\%$. However, the authors of this paper managed to propose a multi-level multi-class architecture that was hypothesized to generate better accuracies. The hypothesis has been proven effective reaching an accuracy of 72.44% comparing it to the accuracy of 62.5% using a single-level architecture performing on the same dataset. Further research is recommended applying the multi-level architecture on existing solutions putting into consideration the factors affecting the models regarding the labels, dataset features and number of classes.

IX. Future Work

The authors recommend trying a different architecture for the positive emotions classifier since a better accuracy could have been achieved. However, the authors decided to stop at this accuracy due to timing constraints. Moreover, trying larger datasets with more classes and labels for both positive and negative classifiers is highly encouraged to validate the hypothesis on massive datasets and comparing the results with a single multiclass classifier. Finally, further research is encouraged in the field of labels selection since the affect of the selection of well-defined non-overlapping labels for emotions should not be underestimated.

References

[1] Sentiment Analysis Applications and Examples. (2020, April 09). Retrieved November 11, 2020, from https://monkeylearn.com/blog/sentiment-analysis-applications/

[2] Multi-class Emotion Classification for Short Texts. (n.d.). Retrieved November 11, 2020, from https://tlkh.github.io/text-emotion-classification/

[3] Hana, L. (2020, January 22). A step by step Guide on Sentiment Analysis with RNN and LSTM. Retrieved November 11, 2020, from https://medium.com/@lamiae.hana/a-step-by-step-guide-on-sentiment-analysis-with-rnn-and-lstm-3a293817e314

[4] Jayachandra1221. (2018, July 19). Text Classification Multi class. Retrieved November 11, 2020, from https://www.kaggle.com/jayachandra1221/text-classification-multi-class

[5] Mayank, M. (2017, October 07). Best AI algorithms for Sentiment Analysis. Retrieved December 13, 2020, from https://www.linkedin.com/pulse/best-ai-algorithms-sentiment-analysis-muktabh-mayank

[6] Turney, P.D. (2002). Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on association for computational linguistics (pp. 417–424). doi:10.3115/1073083.1073153.

[7] Hana, L. (2020, January 22). A step by step Guide on Sentiment Analysis with RNN and LSTM. Retrieved November 11, 2020, from https://medium.com/@lamiae.hana/a-step-by-step-guide-on-sentiment-analysis-withrnn-and-lstm-3a293817e314

[8] Recurrent neural network with pooling operation and attention mechanism for sentiment analysis: A multi-task learning approach

[9] Ravi, K. and Ravi, V., 2020. A Survey On Opinion Mining And Sentiment Analysis: Tasks, Approaches And Applications.

[10] Nguyen, T.-L., Kavuri, S., Lee, M., Hwang, S. O. (2018). A fuzzy convolutional neural network for text sentiment analysis. Journal of Intelligent Fuzzy Systems, 35(6), 6025–6034. https://doi-org.libproxy.aucegypt.edu/10.3233/JIFS-169843

[11] Sadegh, M., Ibrahim, R., Othman, Z. A. (2012). Opinion mining and sentiment analysis: A survey. International Journal of Computers Technology, 2(3), 171–178.

[12] GoTrained Python Tutorials. 2020. Twitter Sentiment Analysis Using TF-IDF Approach - Gotrained Python Tutorials. [online] [Accessed 12 December 2020].