Multi Class Sentiment Analysis using Convolutional Neural Networks

Harika Kudarvalli
08807033
hkudarva@lakeheadu.ca
Lakehead University

Abstract— The main goal is to develop a robust and scalable convolutional neural network to overcome the problems faced in Sentiment Analysis for text-based movie reviews. Bag of Words and TF-IDF are the two feature extraction methods considered in this experiment. Both the methods have been compared and have been plotted for visual representation. Accuracy, Loss, Precision, F1 Score and Recall are the comparative measures. Multiple models were constructed, and the results have been discussed in this paper. Though convolutional neural networks are not the absolute solution, but it has shed light on some interesting aspects. Rotten Tomatoes dataset has been used for training and testing. Dataset contains the phrase and sentiment of the review. The results and observation have been tabulated. The paper has been concluded with comparison between BoW and TF-IDF and after evaluating the results, TF-IDF has better accuracy than BoW.

Keywords— Sentiment Analysis, BoW, CNN, TF-IDF, Accuracy.

I. INTRODUCTION

Convolutional Neural Network (CNNs) is typically associated with Computer Vision. CNNs were responsible for major breakthroughs in Image Classification and are the core of most Computer Vision systems today, from Facebook's automated photo tagging to self-driving cars.
More recently, CNNs have been applied to problems in Natural Language Processing and gotten some interesting results.

CNNs specifically are inspired by the biological visual cortex. The cortex has small regions of cells that are sensitive to the specific areas of the visual field. This idea was expanded by a captivating experiment done by Hubel and Wiesel in 1962. The research showed that some individual neurons in the brain activated or fired only in the presence of edges of an orientation like vertical or horizontal edges. For example, some neurons fired when exposed to vertical sides and some when shown a horizontal edge. Hubel and Wiesel found that all these neurons were well ordered in a columnar fashion and that together they were able to produce visual perception. This idea of specialized components inside of a system having specific tasks is one that machines use as well and one that you can also find back in CNNs.
The entire experiment has been defined by following steps:

1. Data Retrieval
2. Data Preprocessing
3. Data Cleaning
4. Data Visualization
5. Bag of Words (BoW)
6. Term-frequency-inverse document frequency (TF-IDF)
7. CNN Model
8. Training Model
9. Evaluation Metrics

II. RELATED WORK

Opinion Mining and Sentiment Analysis by B. Pang and L. Lee;
The focus of our information-gathering behavior has always been to find out what other people think. With the growing availability and popularity of opinion-rich resources such as online review sites and personal blogs, new opportunities and challenges arise as people now can, and do, actively use information technologies to seek out and understand the opinions of others. This survey covers techniques and approaches that promise to directly enable opinion-oriented information-seeking systems. Our focus is on methods that seek to address the new challenges raised by sentiment-aware applications. We include material on summarization of evaluative text and on broader issues regarding privacy, manipulation, and economic impact that the development of opinion-oriented information-access services gives rise to. To facilitate future work, a discussion of available resources, benchmark datasets, and evaluation campaigns is also provided. B. Pang and L. Lee[1], in their paper, has explained deeply about the architecture and techniques of sentiment analysis which can be very useful in the study of this project. They also describe the challenges face by the program to execute efficiently.

Robust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks by Quanzeng You;
Sentiment analysis of online user generated content is important for many social media analytics tasks. Researchers have largely relied on textual sentiment analysis to develop systems to predict political elections, measure economic indicators, and so on. Recently, social media users are increasingly using images and videos to express their opinions and share their experiences. Sentiment analysis of such large-scale visual content can help better extract user sentiments toward events or topics, such as those in image tweets, so that prediction of sentiment from visual content is complementary to textual sentiment analysis. Motivated by the needs in leveraging

large scale yet noisy training data to solve the extremely challenging problem of image sentiment analysis, we employ Convolutional Neural Networks (CNN). We first design a suitable CNN architecture for image sentiment analysis. We obtain half a million training samples by using a baseline sentiment algorithm to label Flickr images. Furthermore, we improve the performance on Twitter images by inducing domain transfer with a small number of manually labeled Twitter images. We have conducted extensive experiments on manually labeled Twitter images. The results show that the proposed CNN can achieve better performance in image sentiment analysis than competing algorithms. In this paper, by Quanzeng You[2], different layers of CNN have been analyzed and how data is processed in every layer is studied and documented. In conclusion, this paper indicates that CNN is better than other deep learning methods, and hence useful for our project.

Comparison between BoW and TF-IDF by Pio Calderon;

Bag of words and TF-IDF refer to the different approaches of categorizing body of document. In Bag of words, you can extract only the unigram words to create unordered list of words without syntactic, semantic and POS tagging. This bunch of words represent the document. In Vector space model, it is algebraic model used for representing documents as vectors. From the given bag of words, you can create a feature document vector where each feature is a word and its value are term weight. In TF-IDF, it is the term weight which is represented in Vector space model. Thus, entire document is a feature vector. which points to a point in vector space such that there is an axis for every term in our bag. This paper helps us to understand how BoW and TF-IDF works in the system. It also helps us to compare the differences between the two, eventually making it easier to study the functioning of these programs.

III. METHODOLOGY

One of the widely used natural language processing task in different business problems is “Text Classification”. The goal of text classification is to automatically classify the text documents into one or more defined categories. Fig. 1. depicts the step by step procedure of how text is being classified into different labels in training and predicting phases. An end-to-end text classification pipeline is composed of following components:

1. Training text: It is the input text through which our supervised learning model can learn and predict the required class.
2. Feature Vector: A feature vector is a vector that contains information describing the characteristics of the input data.
3. Labels: These are the predefined categories/classes that our model will predict.
4. ML Algorithm: It is the algorithm through which our model can deal with text classification (In our case: CNN, RNN, HAN).
5. Predictive Model: A model which is trained on the historical dataset which can perform label predictions.

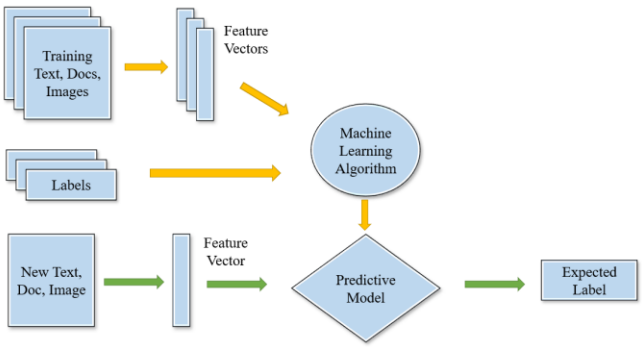

Fig. 1. Text Classification Methodology.

IV. IMPLEMENTATION

A. *Data Retrieval*

The first step for any classification is to retrieve data from a source. In this experiment, a dataset has been taken which consists of multiple reviews from the website Rotten Tomatoes. The dataset has two important columns i.e. PHRASE and SENTIMENT. Each sentence is parsed into its tree structure and each node is assigned a fine-grained sentiment label ranging from $1 - 5$ where the numbers represent very negative, negative, neutral, positive and very positive response respectively.

B. *Data Pre - Processing*

Raw data obtained from users must be pre-processed before feeding it to a model. This increases accuracy and less performance time. There are many pre-processing stages in text classification. In this experiment, we have converted all uppercase letters into lowercase letters. Stop words are removed too. Removal of unnecessary data reduces dataset, but the intent is not impacted.

C. *Data Cleaning*

Once data is pre-processed, it must go through a function to replace old data with cleaned data. Without performing this step, the model might take uncleaned data as input. Fig. 2. shows the process of data cleaning in its programming phase. Following are some preprocessing stages applied to our model.

1. Lowercase Conversion – Similar words will be treated as different words due to case sensitivity. Therefore, convert the entire data into lower case.
2. Removing HTML Tags – As these reviews have been collected from Rotten Tomatoes website, there might be some html tags and definitions which are irrelevant.
3. Removing Punctuations – Punctuations do not have any effect on the sentiment of a sentence, can be excluded.
4. Removing Stopwords – The most crucial step in data cleaning is to remove stop words. Irrelevant words which do not change the meaning of the sentence can be avoided for better accuracy and testing time.
5. Stemming – As the true meaning of the word is in its root therefore rest of the letters can be reduced to its root.

```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
temp =[]
snow = nltk.stem.SnowballStemmer('english')
for sentence in phrase:
    sentence = sentence.lower() #Lowercase Conversion
    cleanr = re.compile('<.*?>')
    sentence = re.sub(cleanr, ' ', sentence) #Removing HTML Tags
    sentence = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    sentence = re.sub(r'[.|,|)|(|\|/]',r' ',sentence) #Removing Punctuations

    #Removing Stop Words & Stemming
    words = [snow.stem(word) for word in sentence.split() if word not in stopwords.words('english')]
    temp.append(words)

phrase = temp
```

Fig. 2. Code Snippet for Data Cleaning.

D. *Data Visualization*

Once the data is cleaned and ready to be fed to the model, understand the data more closely with visualizing its factors in graphs or plots. Graphs give a clear picture about the dataset being dealt. The cleaned data will be split into training and testing sets. By default, we have taken 80-20 ratio for splitting purpose. Fig. 3. depicts the distribution of reviews with respect to its sentiment.

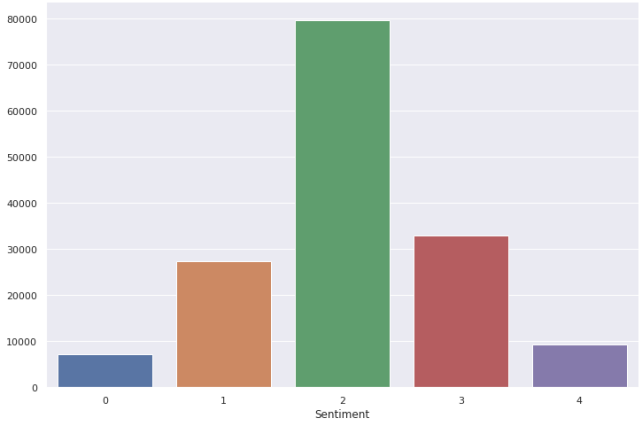

Fig. 3. Bar Graph for Sentiment.

E. *Feature Extraction (BOW)*

The approach is relatively simple: given a set of topics and a set of terms associated with each topic, determine which topics exist within a document. While other, more exotic algorithms also organize words into “bags,” this technique does not create a model or apply mathematics to the way in which this “bag” intersects with a classified document. A document’s classification will be polymorphic, as it can be associated with multiple topics. Fig. 4. is a snippet of code from the working of BoW.

```
#Bag of Words Feature Extraction Method
BOW_vectorizer = CountVectorizer(strip_accents='unicode',
                                 ngram_range=(1,3),
                                 analyzer='word',
                                 min_df=5,
                                 max_df=0.5)
BOW_vectorizer.fit(list(fullSent['Phrase']))
```

Fig. 4. Code Snippet for BOW.

F. *Feature Extraction (TF-IDF)*

Term-frequency-inverse document frequency (TF-IDF) is another way to judge the topic of an article by the words it contains. With TF-IDF, words are given weight – TF-IDF measures relevance, not frequency. That is, wordcounts are replaced with TF-IDF scores across the whole dataset.

G. *Convolutional Neural Network*

CNN’s are efficient for sentence classification tasks as the convolution layers can extract features horizontally from multiple words. These characteristics are essential for classification tasks as it is difficult to find clues about class memberships especially when these clues can appear in different orders in the input. CNN has also been used for document topic classifications where a single local phrase could aid in establishing the topic regardless of the position where it appears in the document. They found that CNN is powerful enough to find these local indicators due to the powerful combination of the convolution and pooling layers.

We initialized the model with Keras Sequential layer. Each filter in the CNN will detect specific features or patterns and then it will be pooled to a smaller dimension in the max-pooling layer.

H. *Training Model*

Once the network has been built, the model can be trained using the above feature extraction methods. This model can be fitted in multiple representations with their average accuracy, f1 score, precision and recall.

I. *Evaluation Metrics*

Precision - is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV).

Recall - is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. F1 Score - The F1 Score is the 2 * ((precision * recall) / (precision + recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall. Fig. 5. is the code that runs behind the evaluation matrix.

```
#Evaluation Metrics
from keras import backend as K

def recall_m(y_true,y_pred):
  true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,1)))
  possible_positives = K.sum(K.round(K.clip(y_true, 0,1)))
  recall = true_positives / (possible_positives + K.epsilon())
  return recall

def precision_m(y_true, y_pred):
  true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,1)))
  predicted_positives = K.sum(K.round(K.clip(y_pred, 0,1)))
  precision = true_positives / (predicted_positives + K.epsilon())
  return precision

def f1_m(y_true,y_pred):
  precision = precision_m(y_true,y_pred)
  recall = recall_m(y_true,y_pred)
  return 2*((precision * recall)/(precision + recall + K.epsilon()))
```

Fig. 5. Code Snippet for Evaluation Metrics.

V. ARCHITECTURE

CNN is a class of deep, feed-forward artificial neural networks where connections between nodes do not form a cycle. CNNs are generally used in computer vision, however they've shown promising results when applied to various NLP tasks as well. Fig. 6. Is the flow chart depiction of the architecture or built of Convolutional Neural Network.

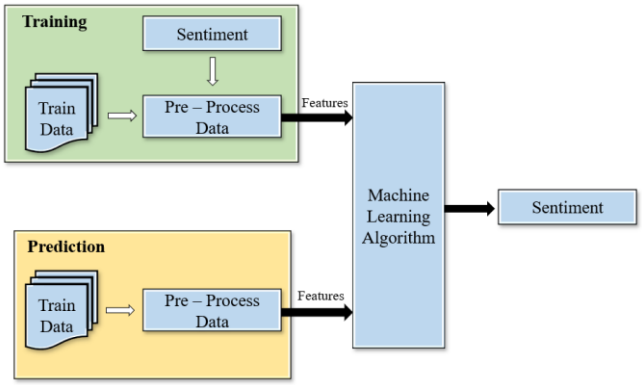

Fig. 6. Architecture of CNN.

A. *Training and Prediction Phases*

In the training process (a), our model learns to associate an input (i.e. a text) to the corresponding output (tag) based on the test samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and tags (e.g. positive, negative, or neutral) are fed into the machine learning algorithm to generate a model.

B. *Feature Extraction*

The first step in a machine learning text classifier is to transform the text into a numerical representation, usually a vector. Usually, each component of the vector represents the frequency of a word or expression in a predefined dictionary (e.g. a lexicon of polarized words). This process is known as feature extraction or text vectorization and the classical approach has been bag-of-words or bag-of-ngrams with their frequency. More recently, new feature extraction techniques have been applied based on word embeddings (also known as word vectors). This kind of representations makes it possible for words with similar meaning to have a similar representation, which can improve the performance of classifiers.

C. *Classification Algorithms*

The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks:

1. Naïve Bayes: a family of probabilistic algorithms that uses Bayes's Theorem to predict the category of a text.
2. Linear Regression: a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).
3. Support Vector Machines: a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. These examples are mapped so that the examples of the different categories (sentiments) belong to distinct regions of that space. Then, new texts are mapped onto that same space and predicted to belong to a category based on which region they fall into.
4. Deep Learning: a diverse set of algorithms that attempts to imitate how the human brain works by employing artificial neural networks to process data.

VI. DATASET

The dataset is comprised of tab-separated files with phrases from the Rotten Tomatoes dataset. The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. Each Sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId while each sentence has a SentenceId. Phrases that are repeated are only included once in the data. train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence. test.tsv contains just phrases. Fig. 7. depicts the dataset in the project before pre-processing. We must assign a sentiment label to each phrase. The sentiment labels are:

0 - negative
1 - somewhat negative
2 - neutral
3 - somewhat positive
4 – positive

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
63	64	2	This quiet , introspective and entertaining in...	4
81	82	3	Even fans of Ismail Merchant 's work , I suspe...	1
116	117	4	A positively thrilling combination of ethnogra...	3
156	157	5	Aggressive self-glorification and a manipulati...	1
166	167	6	A comedy-drama of nearly epic proportions root...	4
198	199	7	Narratively , Trouble Every Day is a plodding ...	1
213	214	8	The Importance of Being Earnest , so thick wit...	3
247	248	9	But it does n't leave you with much .	1
259	260	10	You could hate it for the same reason .	1
271	272	11	There 's little to recommend Snow Dogs , unles...	1

Fig. 7. Dataset before Pre – Processing.

VII. RESULTS

Below are the results and observations found after training and testing model. The model has been evaluated using evaluation metrics. As you can see in Table 1, TF-IDF has better accuracy of 33.27 %.

Metrics	Loss	Accuracy	F1 Score	Precision	Recall
TF-IDF	1.4376	0.3327	0.1528	0.6809	0.0865

Table 1. Observations using TF – IDF.

Following are different graphs of evaluation metrics with respect to its epoch. Blue line has been indicated as Training set whereas orange is Testing set.

Fig. 8. Model Accuracy vs Epoch (TF-IDF).

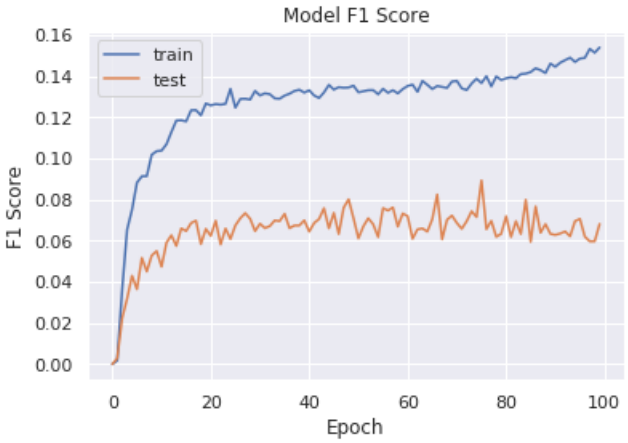

Fig. 9. Model F1 Score vs Epoch (TF-IDF).

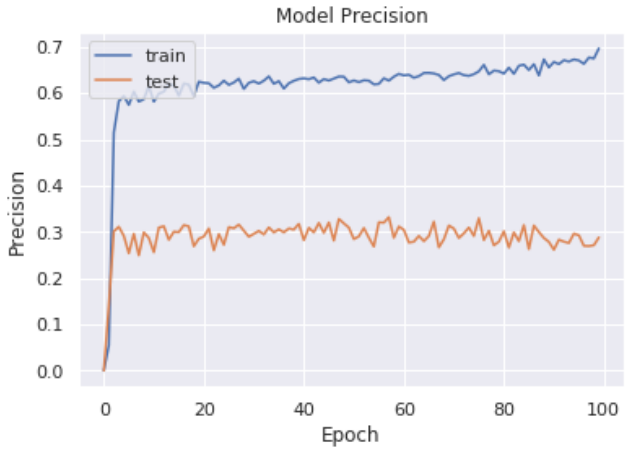

Fig. 10. Model Precision vs Epoch (TF-IDF).

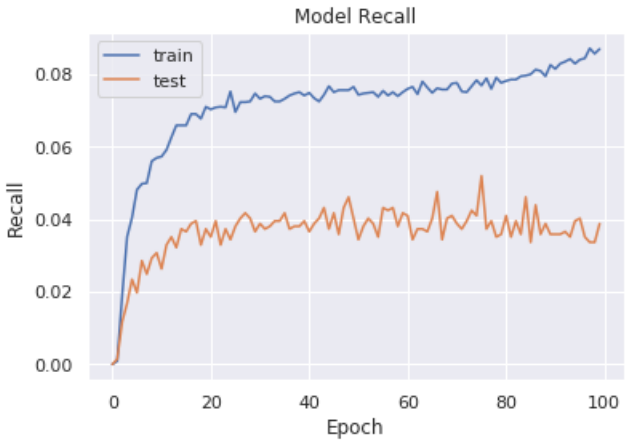

Fig. 11. Model Recall vs Epoch (TF-IDF).

The model has been evaluated using evaluation metrics. As you can see in Table 2, BOW has accuracy of 30.63 %.

Metrics	Loss	Accuracy	F1 Score	Precision	Recall
BOW	1.4441	0.3063	0.1387	0.6476	0.0781

Table 2. Observations using BOW.

Following are different graphs of evaluation metrics with respect to its epoch. Blue line has been indicated as Training set whereas orange is Testing set.

Fig. 12. Model Accuracy vs Epoch (BOW).

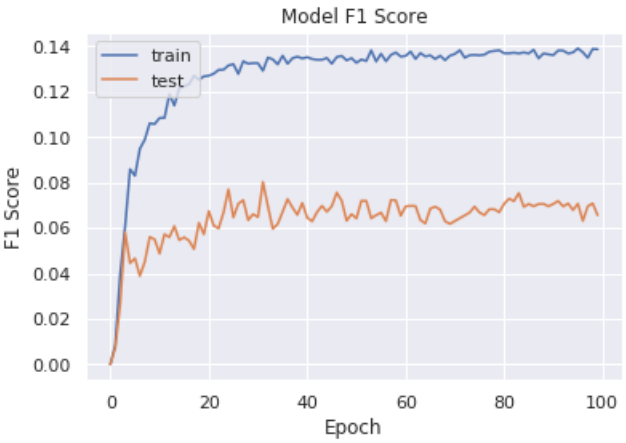

Fig. 13. Model F1 Score vs Epoch (BOW).

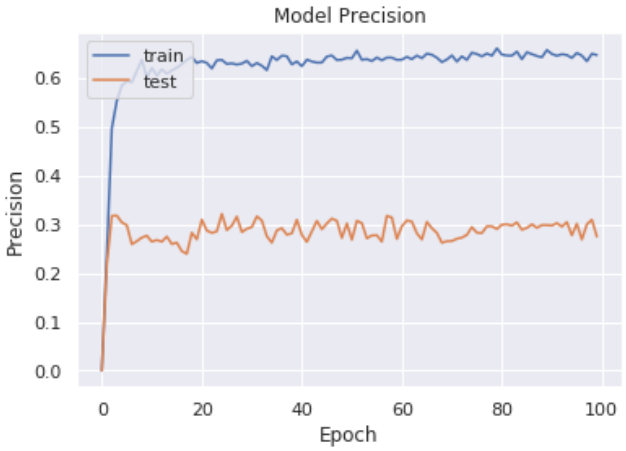

Fig. 14. Model Precision vs Epoch (BOW).

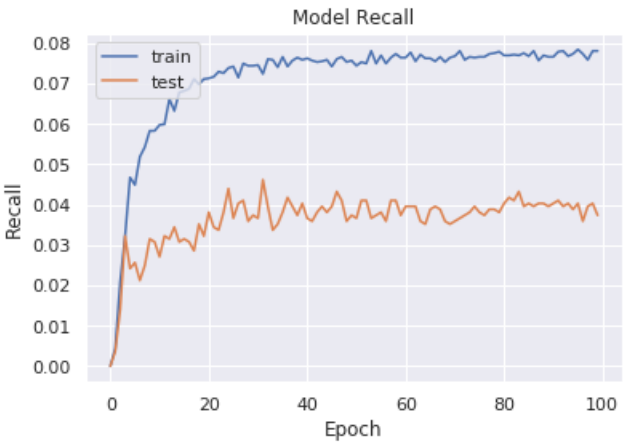

Fig. 15. Model Recall vs Epoch (BOW).

Though the difference is small but TF-IDF has better performance accuracy than BOW.

VIII. Future Work

In this experiment we have observed results from different feature extraction methods after multiple pre – processing steps. In future, I would like to further study other deep learning technologies like Recursive Neural Network and Long Short-Term Memory. In the end, I want to compare all the models in terms of their accuracy and other evaluation matrix.

IX. Conclusion

Sentiment analysis can be applied to countless aspects of business, from brand monitoring to product analytics, from customer service to market research. By incorporating it into their existing systems and analytics, leading brands can work faster, with more accuracy, toward more useful ends. In this project, I constructed a convolutional neural network for text classification using neural analysis and have successfully compared the two feature extraction methods, BoW and TF-IDF and come to conclusion that TF-IDF has better accuracy when compared to BoW. The model can be made more accurate by adding different types of layers in CNN and rigorous data cleaning. Sentiment analysis has its own defects, like failure to differentiate irony, sarcasm and its intended meaning. Though the difference is small but TF-IDF has better performance accuracy than BOW. Sentiment analysis has moved beyond merely an interesting, high-tech whim, and will soon become an indispensable tool for all companies of the modern age. Ultimately, sentiment analysis enables us to glean new insights, better understanding of consumer needs, and empower companies more effectively so that they do better and more productive work.

Acknowledgement

I would like to thank Dr. T. Akilan for guiding me through this experiment and enlightening me with in depth knowledge. I am also thankful to Mr. Shiqi Wang for his supervision as Teaching Assistant.

References

[1] Bo Pang and Lillian Lee, “Opinion Mining and Sentiment Analysis”, 2008

[2] Quanzeng You, Jiebo Luo, Hailin Jin, Jianchao Yang, “Robust Image Sentiment Analysis Using Progressively Trained and Domain Transferred Deep Networks”, 2015

[3] Pio Calderon, “Bag of Words and Tf-idf Explained” web site: www.datameetsmedia.com

[4] Ricky Kim, “Another Twitter sentiment analysis with Python”

[5] Nikolai Janakiev, “Practical Text Classification With Python and Keras”

[6] Denny Britz “Implementing a CNN for Text Classification in TensorFlow”