Sentiment Analysis on Web-services

Siddharth Tarey
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
st2476@rit.edu

Abstract—The use of web-services has been on the rise recently. The quality of the service such as speed, security, ease of access etc. provided by these web-apis have become a primary concern for the user as well as the provider; one way to improve the web-service is to analyze the feedback generated by user reviews. User generated reviews for web-services contain a lot of technical content that expresses the sentiment of the user very subtly, thus it becomes a challenge for any classification model to detect these subtleties and classify the sentiment of the review accurately. This paper proposes a model to accurately detect these subtleties and accurately classify user sentiments. We also discuss the scoring scheme and the classification technique used in this model.

I. INTRODUCTION

Web services have gained a significant importance since the inception of Web 2.0. Almost all organizations use web-services which are either purchased or freely available. Some of the popular web services are the Google Cloud Platform, the Facebook graph API and the Twitter API. There are a lot of factors that amount to the popularity of the API for e.g. the response time of the API, the ease of usage, the amount of information provided etc. The best way to know if a web service is doing well or not is from user feedback. Users of those web service will often provide a feedback on which features they liked or did not like, more often the users of those APIs will post questions requesting more information on the API. A feedback generally gives an idea whether a user likes the web service or not, and if the user does not like it what are the factors that contributed to the negative feedback. Although feedback is important for improving a web-service, the process of scanning through feedback is a cumbersome and a manual task. Thus an automated way of learning the sentiment polarity (positive or negative) would be quite helpful for the organization to identify negative and positive feedbacks. Thus, this paper introduces an experiment of Sentiment Analysis on web-services reviews. Sentiment Analysis is the study of people’s emotions towards certain factors, these factors can be people, shopping products or any topic in general. The goal of Sentiment Analysis is to identify the emotions in a text document and hence detect the polarity of a document, whether its a positive sentiment document or a negative one. Thus sentiment analysis is mainly a classification task where a list of text is given and then we classify if a specific text is of a positive sentiment or a negative. Sentiment Analysis on a high level proceeds in three steps. The first is feature selection, in which the text is preprocessed and converted into mathematical vector notations. This is important since a math representation of a text would be much easier to process and work with rather than the raw text itself. The second step is to use a classification algorithm and train it with the features extracted from the corpus of text. The final step is to store the model generated from the training phase, test it and then deploy it. Many of Sentiment Analysis techniques use lexicon based approaches where pre-existing dictionary exist to convert a word or a phrase to a vector. A popular such library is called the word2vec\(^1\). This dictionary converts a word to a vector. Thus a document can be represented as a \( N \times M \) (If the document is expressed as a unigram) vector where \( N \) is the length of the document and \( M \) is the dimension of the output vector produced by Word2Vec. Consequently this matrix can then be used to train a learning model to perform Sentiment Analysis on a corpus of documents.

Although this technique provides a fair-good amount of accuracy, we have to rely on the accuracy of this dictionary to provide an accurate vector representation of the data. In this paper we will introduce a non-lexicon based technique to classify a document to a sentiment. The \( \text{delta-tf.idf} \)\(^1\) scoring scheme relies on the label assigned to a document to assign the score for a document. By not relying on a dictionary we provide a full autonomy to the to the scoring scheme and thus to the entire learning model. Thus effectively we hope to cut out the middleman for Sentiment Analysis.

II. CHALLENGES FACED

There are several challenges while performing this experiment, some of them are discussed here.

A. Data Retrieval

There are no benchmark datasets available that contains reviews of benchmark datasets. Thus the first challenge is to obtain a suitable dataset for this experiment. There are thousands of web services that are available around the web spanning over several domain, therefore it was decided to focus on just one domain i.e. the domain of social media. This domain was selected simply because of the popularity of various social media websites and their corresponding API, thus the chances of obtaining excellent amount of data is

\(^1\)https://deeplearning4j.org/word2vec.html
high in this particular domain. Since there are no benchmark datasets, various forums that contain feedback on APIs were identified and crawled. The primary source of the dataset was stackoverflow.com

B. Data Analysis

The data that is being extracted is a text data that contains a lot of technical notations. There are reviews which contain code snippets and links, these had to eliminated since they do not contribute towards Sentiment Analysis. As this data has been crawled, it is unlabeled, thus the data has to be labeled manually. Each review that was extracted was either labeled as a positive, negative, or neutral sentiment.

C. Building the Neural Net

The feature vector that will be generated by our learning model will be \( V \times N \) matrix, where \( N \) is the number of documents in the corpus and \( V \) is the number of terms that have a \( tf.idf \) score in the entire corpus, thus effectively \( V \) could represent the total number of nouns adjectives and phrases present in the document corpus. The neural net has to trained on each row of the matrix as each row represents a document, as a result the input neurons for our Neural Net would be the size of \( V \). Thus effectively the neural net could have a large number of input neurons which is difficult to build.

III. RELATED WORK

Opinion mining or Sentiment Analysis is a field in Artificial Intelligence which computes the attitude of a text towards an entity, this entity can be individuals or topics. Topics can be explained using this example. This phone makes a scratching sound all the time it’s not good. In this example the topics that are discussed is the phone and scratching sound. Sentiment Analysis can be considered as a classification problem, where the sentiment polarity (positive, negative or neutral) of a text is found based on some analysis performed. There are three broad levels at which Sentiment Analysis operates, the first is the document level where the sentiment analysis is performed on the entire text assuming that the text is one whole unit and has only one topic in it. The second is the sentence level, here each sentence is treated as one unit, therefore each sentence is assumed to be having one topic and the sentiment polarity is found for that sentence. The third level goes a little deeper performs sentiment analysis at the aspect level, this means that the topics that are discussed in the document are extracted and the sentiment polarity towards each of these topics are then discovered and analyzed.

A. Feature Selection

1) Lexicon Based Methods: As sentiment classification is a classification problem, feature selection is a primary step. There are two types of feature selection in sentiment analysis, one is a lexicon based feature selection and the other is a statistical method.

In one lexicon based approach proposed by Thellwall et. al[2] compiles a dictionary of weighted positive and negative words to form a sentiment score for a text. The weights are manually assigned to every positive or negative words, these weights are then optimized by slightly varying the weight of words and then comparing the accuracy of the subsequent classifier, if there as improvement in the accuracy the modified weight is retained else the original weight is retained. This method is popularly known as SentiStrength.

A similar approach(Vader) was adopted by Gilbert et. all [3] but their included slangs that were popularly used in social media, they hypothesized that slangs are popularly used on social media hence including them into the dictionary can have a significant impact on the accuracy of the output of Sentiment Analysis. Similar to SentiStrength[2] Vader too trains the score of its dictionary using a rule based classifier, thus if a document is marked as positive(or negative) the score of the word is adjusted. Another difference between the two methods is that SentiStrength assigns the score of the most positive or the most negative word to the document, however Vader sums up the scores in the document and then classifies a document as positive(or negative).

A more advanced lexical based method proposed by Bacianella et. all [4] considers "concepts" rather than singleton words to assign a score to a document. The authors hypothesized that if two words are grouped together they may have a completely different sentiment than when considered individually. For e.g. Dog Bite has a negative sentiment as a group of words but when consider Dog and Bite individually they have a neutral sentiment. Thus SentiWordNet[4] uses a graph based approach where the dictionary not only contains sentiments of individual words but also contains the sentiment score grouping of words.

2) Statistical feature selection: A statistical approach does not use dictionary of words to produce a feature vector but rather relies on other aspects of the documents to produce a feature vector. This feature vector is then to model a classifier, to classify a document as a positive or a negative sentiment. A probabilistic approach used by Mass et. all [5] to weigh and score the text document, the approach was to use a probabilistic distribution of words assuming that each word is conditionally independent of the other word. This approach resulted in a significant improvement over the traditional bag of words approach.

Hu et. all [6] chose a different approach for feature selection in a micro-blogging environment like Twitter, they hypothesized Social contacts which are made by users on these websites are usually people who would think alike. For eg. A person who likes Jeffery Archer as an author will be friends with other people who like Jeffery Archer. Similarly, the author of h tweet will have more contacts who have similar traits in technology, this is the first type of theory that can be exploited upon to extract information on the document and is called the Emotional Contagion. The second set of information is derived from the fact that a user generally posts tweets with similar emotions, For e.g If a person has a habit of posting happy tweets, he/she will only post happy tweets. This behavior is
called Sentiment Consistency
The whole data can be modeled into four variables. They are as follows:
- **Matrix X**: which contains \( m \) features for \( n \) documents.
- **Matrix Y**: which contains \( c \) sentiments for \( n \) documents.
- **Matrix F**: which contains \( d \) documents for \( d \) users.
- **Matrix U**: which contains \( d \times d \) matrix which represents the relation between different users. If there is a relation present, the row and column is marked as 1.

In one statistical based approach, by Kucuktunc et. all [7], represented demographic features such as age, location etc. along with the sentiment score of the text as a feature vector for a text document. This corpus of feature vector was then used as an input to classifier. This method however worked well only for a sample of population, the overall model does not represent a general classifier for the entire world population.

A tf.idf based approach is also used to generate a feature vector for a document. The term tf.idf stands for term frequency and inverse document frequency. The term frequency denotes the number of times a word or a phrase has been repeated in the text, the inverse document frequency is the counterweight for the term frequency, it denotes the inverse frequency of the word(or phrase) spanning the entire document. Bespalov et. all. [8] proposed that using \( tf.idf \) in conjunction with n-gram representation of the document could result in an effective in classifying a document sentiment, their experiments with this feature vector along with multi-layer neural net showed results which were comparable with some lexicon-based methods.

**B. Models for Classification**

1) **Supervised Algorithms**: As mentioned discriminative classifier uses the conditional probability to train its classifier. In layman’s terms, a discriminative classifier forms a model using attributes of an observation whereas a generative model uses both the attributes as well as the output class to model a classifier. A Support Vector Machine(SVM) is a popular method used in Sentiment Analysis. The main principle of SVM is to find a linear separator in the search space. Text data are ideally suited for SVMs since text data contain a lot of features and hence causes the data to be sparse('Curse of Dimensionality'). Moreover a non-linearly separable data in one dimension can be linearly separable in a higher dimension[9]. SVM was used by Paltoglou[10] using tf.idf values frequencies as the feature vector, he proposed assigning weights to words depending on their frequency and the sentiment polarity of the document.

\[
weight = \frac{tf.idf \text{ value}}{\text{Number of tweets at a given instance of time}}
\]  

Using this feature and the Naive Bayes classifier the author produced good results on a dataset which had a lot of sentiment variation over a short span of time.

A Multi-stage classifiers are used when the data is very sparse or if the data is skewed towards one class or the other. Mejova et. all[13] used a multi-stage classifier on a Twitter data consisting of election micro-blogs. They pointed out that the data is skewed in positive or negative depending on the number of active users on twitter. If a user constantly tweets negatively/positively on a candidate, you tend to have skewed data. They used a SVM classifier for the first stage, where the tweets were classified and given positive or negative scores, in the second stage a logistic regressors were used to finally classify the tweet into positive or negative sentiment. The primary task of the first classifier is to assign weight to the tweets depending on the user, for e.g if the user tweets very frequently, then the weight assigned to that tweet is less compared to an non-frequent user. Their results showed that including the demographic of a tweet is an effective way of classifying a tweet as a positive, negative or neutral.

2) **Sem-supervised Algorithm**: The semi-supervised models use unlabeled data for training along with a small set of labeled data set. The reason behind using a semi-supervised learning algorithm is that a generating a labeled data set is a painstaking task, hence we us label a small amount of data and then use this to label the unlabeled data and model a classifier. One type of semi supervised learning that is discussed here is the Expectation Maximization algorithm.

The expectation maximization algorithm proceeds in two steps, the first step is the expectation where the data instances are evaluated using the current parameters supplied to the classifier, the second step is the maximization step where the classifier parameters are evaluated again to maximize the accuracy of the classifier model. In[14] the authors uses naive bayes classifier in their expectation maximization process. The algorithm proceeds in two steps first is the Expectation step and the second is the Maximization step.

SVM was also used by Canuto et. all[11] with their meta-features as an input to the SVM classifiers. They compared their results with the traditional bag of words approach and found that their methods out performed the bag of words approach.

Naive Bayes is a popular discriminating classifier, which uses conditional probability derived from Bayes rule, with a naive assumption that the attributes of an instance are independent of each other. The authors in [12] used Naive Bayes classifier with a twitter dataset over features derived from tf.idf values of a document tweet and the time window in which they were written. They argued that in certain domain like sports there is a significant spike in the number of tweets at certain instances of time, thus to correctly weigh a feature in a tweet we should consider the time the tweet was posted. Thus the features can be represented as.

\[
weight = \frac{tf.idf \text{ value}}{\text{Number of tweets at a given instance of time}}
\]  

The authors in [14] used a multi-stage classifier on a Twitter data consisting of election micro-blogs. They pointed out that the data is skewed in positive or negative depending on the number of active users on twitter. If a user constantly tweets negatively/positively on a candidate, you tend to have skewed data. They used a SVM classifier for the first stage, where the tweets were classified and given positive or negative scores, in the second stage a logistic regressors were used to finally classify the tweet into positive or negative sentiment. The primary task of the first classifier is to assign weight to the tweets depending on the user, for e.g if the user tweets very frequently, then the weight assigned to that tweet is less compared to a non-frequent user. Their results showed that including the demographic of a tweet is an effective way of classifying a tweet as a positive, negative or neutral.

The algorithm proceeds in two steps first is the Expectation step and the second is the Maximization step. The expectation
In this case 'fuel efficiency' and 'engine' can be good engine features. For e.g the first classifier will work on other and label the data. Each of the classifiers works on its own set of input features. For example, classifier 1 will work on features denoted by \( X_1 \), and classifier 2 will work on features denoted by \( X_2 \). \( X_1 \) and \( X_2 \) denote the entire feature list, thus in essence each classifier models on its own set of features. While evaluating a class for an unlabeled dataset, the confidence calculated by both the classifiers are compared, if the first classifier has a higher confidence than the second one then the observation is labeled as calculated by the first classifier. Thus in essence the two classifiers provide feedback to each other to train a better model.

**C. Topic Modeling Techniques**

A topic model represents an abstract topic that is being discussed in a document. A document in general will contain multiple topics. For example, 'This car has good fuel efficiency and good engine'. In this case 'fuel efficiency' and 'engine' can be modeled as topics. Topics in general are generated using a generative machine learning algorithm by identifying recurring patterns in a document corpus. One of the most basic topic modeling technique is the Latent Dirichlet Allocation (LDA) [16], it is a generative model that uses a joint probability distribution to identify the topics that are presented in the document. The probability of a word belonging to a topic is the product of number of times a topic appears in a document and the number of times the word occurs in the topic. LDA assumes a Dirichlet distribution for every word in the beginning, this distribution is then improved by sampling the joint probability distribution using sampling methods such as Gibbs sampling. Another more advanced technique was proposed by Kawamae[17] called the Latent Evaluation topic model. This model was developed for a product review type setting. The LET like the LDA is a bayesian model which uses Dirichlet probabilistic distribution to derive. The difference between LDA and LET is that, LDA uses bag of words and uses a Dirichlet prior based on these bag of words and the document, in this case the Dirichlet prior is based on the words, the item that is discussed and the rating given to the item. Thus in LDA the directly observable entity is the Bag of words, but in case of LET the directly observable entities are the Bag of Words, the item under consideration and the rating given to the item. A clustering based topic modeling was proposed by Rajagopal et. all. [18] where they used a concept of Bag of Concepts to initiate the feature representation. The bag of concept may be a single lexical term or a group of lexical terms for e.g. 'have good food' can be considered as one concept, the difference between a concepts bag of words is that the bag of words uses consecutive words in the form of \( n \)-grams, but this might not be in case of concepts, thus the concepts can be visualized as a graph with different words interconnected to each other. Thus the bag of concepts and the sentiment score (obtained from pre-complied sources such as SentiWordNet[4] and INTELNET[19]). This feature vector is then simply used in k-means clustering to obtain topics in a document corpus.

**IV. Hypothesis**

Lexicon based approach are popular for sentiment analysis since they provide good results for sentiment classification where sentiments can be easily identified by words containing a strong sentiment polarity such as good, best and bad, horrible, such type of settings can be found in corpus of movie reviews or hotel reviews. The reviews on web-service contain very technical content and express emotions very subtly. For example, I have been trying to authenticate through the api since yesterday, but I have no success. In this case, the user very subtly expresses his emotion of frustration. Lexicon based approaches may or may not be able to detect these subtlety. Thus we hypothesize that relying on the labels assigned to the reviews itself might help us to classify the documents to an appropriate sentiment polarity. We propose to use the \( \text{delta-tfidf} \) to score our text which is a different variation of the traditional \( \text{tf-idf} \) scoring scheme. \( \text{tf-idf} \) is the most popular approach towards sentiment analysis, in this approach the frequency of words in a document and the inverse of its frequency in the document corpus is used as a feature vector. Generally the \( \text{tf-idf} \) can be represented as:

\[
\text{wordweight} = \frac{tf}{\log(df/N)}
\]

where \( tf \) represents the frequency of a word in a document, and \( df \) represents the frequency of the word in the entire document corpus, \( N \) represents the number of documents and acts as the normalizing factor for the \( df \) term. Although this is successful...
in information retrieval and extracting relevant text, it does not help in classifying a document to positive or a negative sentiment. Thus previous researches have used SentiStrength, which contains a dictionary of words that contain a weighted score for a word based on its polarity. For e.g. the word Good will have a sentiment polarity of 4, whereas the word Bad will have a sentiment polarity of -4. Thus using a tf.idf with SentiStrength gives a useful feature vector that can be used for classifying a document to a positive or a negative sentiment. In this approach we will use the delta version of tf.idf. The delta version of tf.idf works best on labeled data corpus, instead of using SentiStrength for weighing the word, the delta tf.idf utilizes the class assigned to the document to weigh the document. The equation for the delta tf.idf can be given as follows.

\[
\text{wordweight} = tf \times \log (\frac{N_1 \times idf_2}{N_2 \times idf_1})
\]  

Where tf represents the term frequency in the document. idf_1 represents the frequency of the word in documents of class 1, and idf_2 represents the frequency of the word in documents of class 2. Similarly N_1 and N_2 represents the number of documents in class 1 and class 2. Since the weight of the word will depend on the class of the document, I hypothesize that the weighting dictionary schemes like SentiStrength can be discarded for this approach. For e.g a sentence This movie has very good visual effects, this sentence will be labeled as positive, and the word good will automatically and indirectly receive a positive score, since words like ‘good’ will always occur in documents having positive sentiments or a neutral sentiment.

For Classification models, we propose to use a a Convolution Neural Network(CNN) due to its success with other approaches. Previously authors have used a SentiWordNet or a Word2Vec weighting scheme to a CNN, however to our knowledge this is the first time anyone has attempted to use the delta version of tf.idf along with a CNN to classify documents to positive or a negative sentiment.

V. Scoring Schemes

Weighing schemes for a text is the most vital part in Sentiment Analysis and Information Retrieval. The basic idea is to have a scoring scheme that would indicate the importance of a word in a document, adjectives like good or important should have more weight than words like the, it etc. One the preliminary approach to do this was to use the term frequency. The term frequency represents the frequency of every word in the document. Consider this example: This is a good book, each word in the above example will have a frequency of one. Although the intuition behind the intuition behind the term frequency seems logical but in practicality the most significant word generally have a very low frequency, the prepositions (i.e, and articles) are words that often dominate a document and hence end up having a greater frequency than adjectives. To counter the large frequency that entails less significant words, a normalizing factor was introduced called the inverse document frequency.

A. Term Frequency & Inverse Document Frequency

tf.idf is one the most widely used scoring schemes in Sentiment Analysis and Information Retrieval. It consists two terms, the first is the term frequency which measures how frequently a term occurs in the document, and the inverse document frequency which will help us determine how important the term is for the document. The inverse document frequency is the frequency of the term in the entire document corpus. This frequency acts as a counterweight for the term frequency, thus its role is to diminish the score of frequently occurring words such as is, a, the as described earlier and increase the weight of the other non-frequently occurring words such as adjectives and nouns. The classic tf.idf scoring scheme for a document D can be represented as follows:

\[
w_i = tf \times idf = tf \times \log \frac{N}{df}
\]  

where tf represents the frequency of the i_th term in the document itself and df represents the frequency of the term in the entire document corpus. This indicates that the frequency value of the log term will be very small, which will result in diminishing the value of the term frequency. The less frequent words however will have a large value and will then help retain or amplify the term frequency value. The N term can be considered as a normalizing factor and it represents the number of documents in the corpus. Although tf.idf gives a good scoring scheme for a document it does not give enough information regarding the Sentiment...
The sentiment orientation is generally judged by words which give an indication of positivity or negativity. For e.g., the words like good, excellent give an indication of positivity, whereas bad, horrible, thus in tf.idf the words like bad and good will have the same score thus making it difficult to identify the bad sentiment from the good.

VI. Delta tf.idf

The solution to the above mentioned issue is provided by Martineau and Finin [10] by localizing the idf to the documents of the positive or the negative class. We calculate how a term is biased towards a class rather than finding out how rare the term is the document corpus, as a result for a given term we find the idf for a corpus of documents belonging to each class. Thus the delta-tf.idf will look like as follows.

\[ w_i = t f \ast \left( \log N_1 \frac{d f_1}{d f_2} - \log N_2 \frac{d f_2}{d f_1} \right) \]

where \( tf \) is the term frequency of the term in the current document and \( df_i \) is the idf for the document corpus with class 1, similarly \( df_2 \) is the idf for the document corpus with class 2.

The intuition behind this is that positive words are more distributed towards one class and the negative words would be distributed towards the negative class. Thus if a word has an uneven distribution between the two classes it means that the term has a significant contribution towards determining the sentiment of the document, whereas a term that has an even distribution will not have a significant score and thus will have no significant contribution in classifying the sentiment of the document.

VII. Classification Technique

Neural Networks have recently been quite successful in Sentiment Analysis. Kim[20] and Zhang et. al [21] use Convolution Neural Networks(CNN) for sentence classification. They first convert the tokens of each sentence into word vectors, which is achieved by using popular tools called word2Vec. Thus each word of a sentence will have a vector representation \( v \) which will have a dimension of \( d \). If each sentence contains \( s \) words, then for each sentence we will get a matrix of \( s \times d \). This matrix can be then treated as an image and is convolved using filters. Since each row represents a word, the filters usually have the width of the row and a variable height. Thus if the height of the filter is \( h \) and its width is \( d \) the convolution with a filter \( w \) can be read as:

\[ C = w \ast A[i : i + h] \]

where \( i \) is the row number. The subsequent activation function, pooling and softmax can be achieved using the usual ReLu, 1-max-pooling and regularization.

Although CNN provides fantastic results for classification, it is used due to the matrix representation of the data. The matrix representation is the result of using a vector representation of words. In case of delta-tf.idf a word is not represented as a vector but as a score having a single dimension, as a result we could have a sentence/document representation as a \( s \times 1 \) vector, where \( s \) is the length of the document.

A. Fully Connected Neural Net

Since we have a one-dimensional vector for every instance it is better to use a fully connected neural net rather than a CNN. An Artificial neural net can be visualized as shown in Figure 1.

In a fully connected neural network every neuron in one layer is connected to every neuron in the next layer. A neuron can be considered as a black box, that takes an input and throws out an output. The function that governs the behavior of the output is governed by two factors, the first is the weight assigned to the neuron and second is the activation function. The output to the neuron can be governed by the following function.

\[ w_i = g \left( \sum w_{(i-1)} \ast a_{(i-1)} \right) \]

where \( w_i \) represents the weight of the neuron in one layer and \( w_{(i-1)} \) represents the weight of the neuron in the previous layer, \( a_{(i-1)} \) represents the input from the previous layer. The summation represents the sum of all neurons from the previous layer. The entire summation is then given to an activation function \( g \). We would use ReLu(Rectified Linear Unit) as our activation function which is given by

\[ f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \]

The activation function plays an important role in neural network as it introduces a non-linearity in each hidden layer in the Neural Network, as a result we can map non-linear classifications for our input data.

The feed forward network will not have accurate results in the first iteration of the data, this error in the output will then be sent as a feedback to the neural network, using this feedback the weights assigned to the neurons can be adjusted to get a more accurate result in the subsequent iteration. This process is called back propagation.

The back propagation for this feed forward network can be done by taking the partial derivative of the output layer of a neuron and then adjusting the weight of the neuron by subtracting the current weight and the partial derivative obtained. Thus the new assigned weight during back propogation can be defined as.

\[ w_i = w_i + \alpha \ast a_i \ast \Delta_j \]

where \( w_i \) is the weight of the current neuron and \( \Delta_j \) is the partial derivative obtained from the output, and \( \alpha \) can be defined as the learning rate for the back propagation which generally set between 0 and 1.

Using this framework the neural net will be trained for the given dataset for several epochs, where 1 epoch represents a complete iteration of the training data.
VIII. The Experiment

An overview of the model is given in Figure 1. The experiment is conducted on a real world dataset. All experiments are performed on Windows-10 operating on a Intel i-7 CPU with 2.40 Ghz clock speed and an 8GB RAM. The accuracy is measured using 10-fold cross validation and the performance will be compared with other benchmark techniques.

A. Dataset

For the dataset, we crawled a popular question and answer website called the stackoverflow.com. We performed search queries on various web-services and extracted questions and answers related to the API. We crawled hundreds of posted questions and answers for a single API. We focused on extracting question and answers in the social media domain, thus focusing on extracting data only for APIs such as Facebook, Twitter, LinkedIn etc. We extracted answers which were marked as correct by the community and also extracted the corresponding question. This was done since the answers marked as correct will be more relevant to the feedback or reviews that we intend to classify. Thus in total we have 3700 instances of reviews.

B. Preprocessing

Like any data-set extracted, the data needs to be pre-processed and prepped to use it in a model. The following preprocessing steps are taken to ensure an efficient use of our model.

1) Text Cleaning: We cleaned the input data by removing invalid inputs and removed code snippets and HTML links from the text, leaving only English text in the corpus, next the data was manually labeled to either a Positive sentiment or a Neutral sentiment. Also reviews that were very short (less than 10 words) were removed.

2) Stop Word Removal: Stop words do not contribute a lot in sentiment analysis, hence keeping this in mind, we removed stop words from the documents.

3) Stemming: we also performed word stemming using the Porter Stemmer\(^2\). Stemming is a process to reduce words to its

\(^2\)https://tartarus.org/martin/PorterStemmer/

root form, For e.g. words like "argue", "argued", "argues" and "arguing" reduce to the stem "argu". This will help to reduce the number of terms whose weight has to be calculated, if we did not have stemming words like "argue" and "arguing" would be represented uniquely in our feature vector, thus word stemming helps in dimensional reduction of the data.

IX. Results

Since we have a complete foundation and model to be built, the next steps would be to complete the implementation of the delta-tf-idf, and the Fully Connected Neural Net using TensorFlow library in Python. The overall accuracy of the model will be evaluated as follows:

\[
\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}
\]

where tp, tn, fp and fn represents true positive, true negative, false positive and false negative.

The delta-tf.idf is implemented for bi-grams, this is done as unigrams do not capture all the phrases that contribute to a negative sentiment, For e.g. a phrase such as not good will contribute to a negative sentiment, but a unigram will consider not and good as separate and unrelated terms. The result of this weighting scheme is a scoring matrix where the row represents the documents, and the column represents the score of the bi-gram if it is present.

The neural net is trained over each row of the matrix and the output consists of two nodes, one will be activated if the sentiment is neutral and the other one will be activated if the sentiment is negative. We estimated the number of hidden neurons empirically by increasing the number of hidden neurons and calculating the Sum of Squared Error for each increment of neurons. We increased the number of hidden units from 300 to 2000, with each model running for 100 epochs. In Figure 3. The SSE error decreases as the number of hidden nodes increase, the minimum error is obtained at around 800 to 950 hidden units. The minimum error indicates that the error between the predicted output and the actual output is the least with 800 hidden units and 100 epochs. Using this information, we select this model to further train our classifier with more number of epochs.
We experimented with two models one containing 900 hidden nodes and another containing 950 hidden nodes for the neural net. Thus the model had and input and an output layer, with one hidden layer consisting of 900 or 950 nodes. For the model containing 900 hidden units he results are as shown in Figure 4. We achieve a stable accuracy of about 65% for 1000-7000 epochs. For 950 hidden nodes the accuracy achieved is shown in Figure 5. The stable accuracy obtained was between 55-60%. In both cases the accuracy was unstable in the initial rounds of epochs, but it finally stabilized after 1000 epochs. One reason for this low range of accuracy could be traced to the length of the document. We observed that the document containing negative sentiment are usually large and the negative sentiment is usually confined to just one or two sentence of the document. Thus it is possible that the negative sentiment of this document may get masked by the other neutral sentiment of the other sentences in the document. We compared our results with two lexicon based method, the first is the Aspect Sentiment Unification Model (ASUM)[22] and the second is SentiStrength.[2].

(ASUM) which is generative model used for topic modeling and sentiment classification. The model draws a Dirichlet distribution for the neutral and negative sentiment for a document. The distribution of sentiment is then improved using Gibbs sampling and a dictionary of words that the model uses to estimate the negativity and the neutrality of the document. We found that for this dataset our model outperforms ASUM. We speculate that it is due to the fact that ASUM models the sentiments at a sentence level, these sentiments discovered at the sentence level then contribute to the overall score of the document. Thus when a document is too long, it may or may not reflect the overall sentiment of the document. Some sentences may overshadow the overall sentiment of the document.

SentiStrength uses a dictionary of words for classify a text into positive and negative sentiment. It uses a unigram representation of the text and scores each word according to the sentiment score assigned in the dictionary. The scores are assigned from -1 to -5 for negative sentiment, where -1 represents weakly negative and -5 represents strongly negative. Similarly, +1 represents a neutral sentiment and +5 represents a strongly positive sentiment. In our case, if the score of the document is greater than 0, then we have labeled the document as a neutral sentiment and if it is less than 0, we have labeled the document as a negative sentiment. Our model of Delta-tf.idf outperforms SentiStrength. As the emotions in the reviews are subtly mentioned, SentiStrength may not be able to detect this and thus gives a lower accuracy than our model.

We also experimented delta-tf.idf with the SVM classifier.
using a linear kernel and ten-fold cross validation. The SVM with delta-tf.idf outperforms all of the models including the ANN model. The frequency matrix obtained from delta-tf.idf is very large and sparse, thus as SVM provides a better separation on sparse data than ANN, we obtain a high accuracy for SVM.

X. FUTURE WORK

In the future, we could thing about considering the document length during the scoring process. This could ensure that the non-negative terms do not overshadow the negative terms in a long document. We could also consider modeling the neural net on a GPU for faster training and testing phases.

XI. ACKNOWLEDGEMENTS

I would like to thank Dr. Xumin Liu, who advised and encouraged me on the technical aspects and polish the fundamental ideas of this project. I would also like to thank Dr. Joe Geigel for his guidance on the milestone requirements and poster requirements.

REFERENCES