

Sentimental Analysis on Social Data Using Machine Learning and Supervised Learning Techniques

Pillala Sri Harsha Devi

Dept. of Computer Science and Engineering
Jawaharlal Nehru Technological University

Kakinada, Andhra Pradesh.

Abstract-Social Website provides a convenient platform, on which users can share information with their friends and post their timely status or attitude. There exists the most up to date and heterogeneous data of users, different social websites provide various functions that users can upload, comment and repost messages with different media types. Sentiment analysis aims at computationally identifying people's opinion or attitude towards entities such as events, topics or product features. This work is also known as opinion mining or subjectivity analysis. Sentimental analysis on social data i.e., sentiment analysis can apply to almost all possible domains like products, services for social events and political elections, market research, social media, advertising, recommendation systems, email filtering, stock market prediction, upcoming movie reviews sentiment prediction, book reviews sentiment, etc. In the existing system sentiment orientation from review text documents using machine learning and supervised learning algorithm techniques. The algorithms that are projected and implemented are Naive Bayes (NB), Support vector Machine (SVM) and K-means clustering algorithmic methods. We enhanced these techniques to develop more accurate results using Neural network and deep learning algorithms and features.

Keywords-Convolution Neural Networks (CNN), Recurrent Neural Networks (RNN), Activation Functions, Sigmoid Function, Rectified Linear unit(ReLU) .

1. INTRODUCTION

Data mining research has successfully shaped numerous methods, tools, and algorithms for handling huge volume of data to solve real world problems. The key objectives of the data mining process are to effectively handle large scale data, mine actionable rules, patterns and gain insightful knowledge. As the internet and its technology is growing, people got the freedom of expressing their views, interests and opinions about the things they see around or use regularly in the form of reviews and feedback. Presently, a day's loads of individuals are utilizing the web and doing web based shopping and in the long run they will search for good things. Today's service providers or product providers are more interested in the reviews of their customers because they contain the opinion of the customer and/or, his/her interest about that product or service. Service providers are faced challenging issues in finding behavior or interest of their customer.

Since people have different blogs, twitter, forum discussions, data analysts require more attention to get the sentiment from the reviews which were posted by the people in the comments. In order to grow up the business of the data analysts, they require special attention to extract opinion of the people about the particular entity.

Definition of Sentiment is an attitude, emotion or mood, etc., and Definition of Sentiment analysis is to identify the polarity of review text or the subjective and the emotions of a particular topic in document or sentence. Mine people's review and feelings toward any subject matter of interest, which is the task of sentiment analysis. Now a day, sentiment analysis can apply to almost all possible domains like products, services for social events and political elections, market research, social media, advertising, recommendation systems, email filtering, stock market prediction, upcoming movie reviews sentiment prediction, book reviews sentiment, etc.

1.1 Sentiment Analysis

Sentiment analysis is also called as opinion mining to extract people's opinion from web. The sentimental analysis have become an integral part of the product marketing and user experience as both businesses and consumers turn to online resources for feedback on products and services.

In short, sentiment analysis is the automated process that allows machines to identify and extract opinions within text, such as tweets, emails, support tickets, product reviews, survey responses, etc. Usually, besides identifying the opinion, these systems extract attributes of the expression example:

- **Polarity:** If the speaker express a positive or negative opinion,
- **Subject:** The thing that is being talked about,
- **Opinion holder:** The person, or entity that expresses the opinion.

At present, Sentiment analysis is a topic of great interest and development since it has many practical applications. Since publicly and privately available information over Internet is constantly growing, a large number of texts expressing opinions are available in review sites, forums, blogs, and social media.

With the help of sentiment analysis systems, this unstructured information could be automatically transformed into structured data of public opinions about products, services, brands, politics, or any topic that people can express opinions about. This data can be very useful for commercial applications like marketing analysis, public relations, product reviews, net promoter scoring, product feedback, and customer service.

1.2 Opinion Mining

Opinion mining is extracting people's opinions from web. It mainly analyzes people's opinions, emotions, appraisals, emotions towards organizations, entity (product), topics, issues etc. It involves in building a system to collect and categorize the opinions about a product. Opinion mining is also called as sentimental analysis. Opinion mining has been an emerging research field in Computational Linguistics, Text Analysis and Natural Language Processing (NLP) in recent years. It is the computational study of people's opinions towards entities and their aspects. Entities usually refer to individuals, events, topics, products and organizations.

Aspects are attributes or components of entities. In the last few years, social media has become an excellent source to express and share people's opinion on entities and their aspects. With the availability of vast opinionated web contents in the form of comments, reviews, blogs, tweets, status updates, etc. it is harder for people to analyze all opinions at a time to make good decisions. So, there is a need for effective automated systems to evaluate opinions and generate accurate results.

Sentiment Analysis, Emotion Analysis, Subjectivity Detection has also become an active research area in recent years along with opinion mining. Some people express their opinions in binary scale (i.e. Positive or Negative) and some other expresses their opinions explicitly in terms of ratings (i.e. one to three or five stars). The term "Polarity" in opinion mining refers to the orientation scale. The polarity is predicted on either a binary (positive or negative) or multivariate scale using sentiment polarity classification or polarity classification techniques

1.3 Levels of Opinion Mining

According to Bing Liu, Opinion mining tasks are generally classified into three major levels: document-level, sentence-level, and phrase-level.

1.3.1. There are three levels of opinion mining.

1.3.1.1 Document Level: In this approaches whole document is considers as a single entity and the analysis approaches in applied on the whole document. The result generated in document level sometimes not appropriate.

1.3.1.2 Sentence Level: In the sentence level approaches every sentence is considered as an entity and analysis approaches is applied on individual sentence then their result is summarized to provide the overall result of the document.

1.3.1.3 Aspect Level: Phrase-level opinion mining is also known as aspect based opinion mining. It performs fine grained analysis and directly looks at the opinion. The goal of this level of analysis is to discover sentiments on aspects of items. Nouns or noun phrases which are explicitly mentioned are called explicit aspects. The proposed system is based on aspect level sentimental analysis. Phrase level sentiment analysis or aspect-based sentiment analysis is the best solution to mine people's interests from online reviews. Aspects are the attributes of the service or product and we have named service or product as an entity. It is a context dependent since people are using different type of text or data in different types of media. Aspect level sentiment analysis works by finding all the aspect terms and mining those aspect terms to get the opinion.

For example, consider a sentence "I impressed by the actors performance but storyline is weak". In this sentence, aspect terms are 'actors' and 'story'. Opinion or sentiment expressed towards those aspects is 'impressed' and 'weak'. By using our school English knowledge we can say that audience or viewer is satisfied with actor's performance but unsatisfied with the story and narration. Phrase or Aspect based sentiment analysis can solve the problems of data analysts while making important decisions. In order to get the sentiment, we need to make a machine to learn and this can be done by supervised learning and unsupervised learning. Classification predicts the class labels of categorical data.

Class labels are predetermined and it builds the classifier model. For some data we train the classifier then we perform testing for new data. This classifier tries to find out the relationship among attributes and classifies based on the maximum relationship in terms of probability. Supervised text classification algorithms for their famous are Naive-Bayes, Support Vector Machines. The accuracy of classification is different for these algorithms and way of classification is also different. Naive Bayes classification algorithms are the more scalable algorithm to classify documents based on the frequency of words.

Naive Bayes classifiers consider more number of features for training the classifier. Naive Bayes algorithm follows Bayes theorem by assuming independence of its features. Support Vector Machines (SVM) was another classifier in the competition in this field. It is better classifier than Naive Bayes. SVM provides a unique solution. In view of the above we have developed a system that is used to classify opinion using aspect level classification to get positive, negative and neutral aspects. Then to get the sentiment of the testing data, Naive Bayes and Support Vector Machine are used.

1.4 Aspect Based Opinion Mining

People not only express their opinion on documents and sentences, but also in aspects and entities. Level of information provided in document level or sentence level is not sufficient for making a good decision and therefore looking in-depth into aspects and entities gave a new direction for research called aspect or feature based opinion mining. A drawback in document or sentence level is that they cannot provide complete information of a product, For example, a positive or negative review of a particular product doesn't mean that the reviewer likes or dislikes all aspects of that product.

A person who needs to buy a mobile with excellent camera quality will search only for the reviews about that particular aspect i.e. "picture quality" rather than overall review of that mobile.

Single aspect opinions are reviews where people focus only on one aspect of the product, whereas in multi-aspect, people express differing opinions on more than one aspect simultaneously in the same review and even in same sentence of that product. For example, "This book had a good storyline, but the paper quality is bad". Here, the reviewer gave a positive mention on the "story" aspect and negative mention on the "paper quality" aspect of the book. Segmenting multi aspect sentence into multiple single aspect sentences is called sentence segmentation and it is a challenging task in aspect based opinion mining.

Two major tasks in aspect based opinion mining are aspect extraction and aspect sentiment classification. Process of identifying the opinion words from the given sentence is called aspect extraction and categorizing the extracted opinion words into one of the polarity scales is called aspect sentiment classification. Study shows that adjectives, adverbs and subjective nouns are considered to be aspect related words in most cases and gives high performance. Consider this review "Truly enjoyed the action sequences and the screen play but weak story line made it a missed opportunity, music is neutral not too bad or good".

Table 1 Polarity of aspect.

Aspect	Polarity		
	Positive(+)	Negative(-)	Neutral(=)
Screenplay	*		
Music			*
Story line or narration		*	

II. SYSTEM ARCHITECTURE

1. Existing System Architecture

The author provide a survey on hybrid approach to analyze aspect based sentiment of social data. It covers work on explicit and implicit aspects which is crucial for aspect based sentiment of data. Previous researches have assumed that social data i.e tweets or facebook data level classification which only determines general sentiment of data. The approach is done by using Dependency Parsing, Association rule mining. Senti wordnet is used to solve aspect based sentimental analysis. The work is done on Hate crime sentiment data set and Stanford Twitter sentiment dataset. For racial aspects from hate crime, the approach successfully classified 28% as neutral, 53% as hate and 19% as free data. Besides, it also classified 24% as neutral , 58% as hate and 18% as free towards sexual aspects. Finally, for religion aspects, the hybrid approach classified 32% as neutral, 48% as hate and 20% as free.

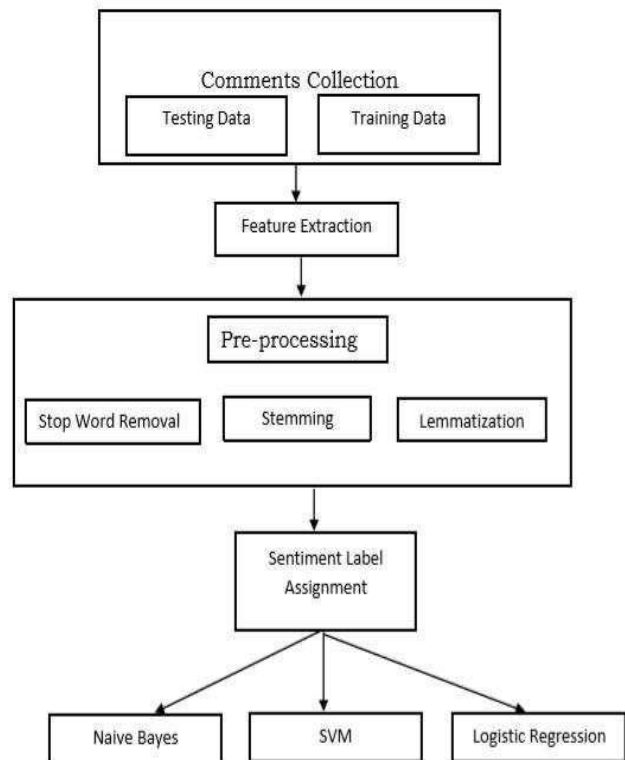


Figure 1 Existing System Architecture.

2. Proposed System Architecture

To develop a system that is used to classify opinion using aspect level classification to get positive, negative and neutral aspects. The aspect based opinion mining is done based upon the polarity. Most interesting aspects will be extracted based upon the relative importance. Positive, negative and neutral aspects will be extracted. Naive Bayes and Support Vector Machine and neural network algorithms are used to get the sentiment of the test datasets.

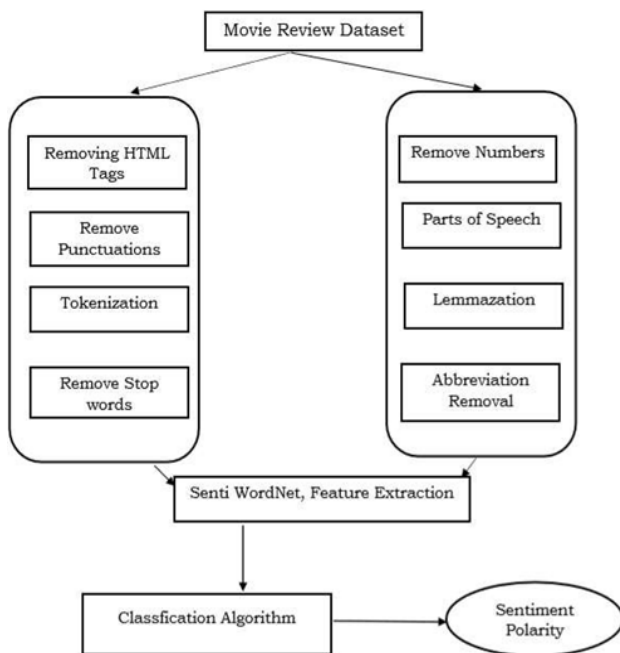


Figure 2 Proposed System Architecture.

3. Overview of the Project

The Overview of the project contains in detail implementation of both existing and proposed systems. The movie review dataset is taken for the implementation and results are analyzed accordingly.

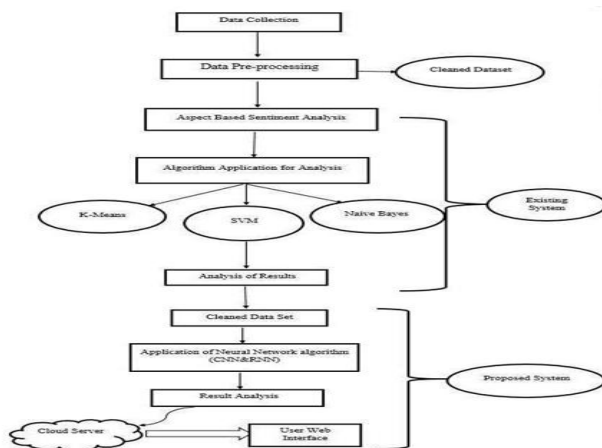


Figure 3 Flow chart of the project.

III. SOFTWARE DESCRIPTION

1. System Requirements:

1.1. Hardware Requirements:

- PROCESSOR : Any Intel or AMD x86 or x64 processor.
- RAM : At least 2048 MB.
- HARD DISK : 700 MB for a typical installation of PYCHARM.
- KEY BOARD : Multimedia Keyboard.

1.2. Software Requirements:

- OPERATING SYSTEM : Windows XP or higher.
- FRONT END : Jupyter Notebook.

2. Technical Requirements

1.2. Choice of language:

1.2.1. Introduction to Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter.
- Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Flexible and powerful, Python was originally developed in the late 1980s at the National Research Institute for Mathematics and Computer Science by Guido van Rossum as a successor to the ABC language. Since its introduction, Python has grown in popularity thanks to what is seen as a clear and expressive syntax developed with a focus on ensuring that code is readable. Python is a high-level language. This means that Python code is written in largely recognizable English, providing them with commands in a manner that is quick to learn and easy to follow. This is in marked contrast to low-level languages, like assembler, which are closer to how the computer "thinks" but almost impossible for a human to

follow without experience. The high-level nature and clear syntax of Python make it a valuable tool for anyone who wants to learn to program. It is also the language that is recommended by the foundation for those looking to progress from the simple Scratch.

Example: Hello World

The easiest way to learn a new programming language is to create a project that prints “Hello World!” on the screen. In Scratch, you just had to drag and drop bricks of prewritten code, but in Python, you need to write this program entirely by hand. A Python project is, at heart, nothing more than a text file containing written instructions for the computer to follow. This file can be created using any text editor. For example, if you enjoy working at the console or in a terminal window, you can use nano; or if you prefer a graphical user interface (GUI), you

can use Leaf pad. Another alternative is to use an integrated development environment (IDE) such as IDLE, which provides Python-specific functionality that’s missing from a standard text editor, including syntax checking, debugging facilities and the ability to run your program without having to leave the editor. This chapter gives you instructions on how to create Python files using IDLE, but of course, the IDE program that you choose to use for programming is up to you. The chapter also includes instructions for running your created files directly from the terminal, which can be used in conjunction with any text editor or other IDE. To begin the Hello World project, open IDLE from the Programming menu in the Debian distribution’s desktop environment. If you’re not using IDLE, create a blank document in your favourite text editor and skip the rest of this paragraph. By default, IDLE opens up in Python shell mode, so anything you type in the initial window will be immediately executed. To open a new Python project which can be executed later, click on the File menu and choose New Window to open a blank file. The IDLE Python Shell window.

It’s good practice to start all Python programs with a line known as a shebang, which gets its name from the # and ! characters at the beginning of the line. This line tells the operating system where it should look for the Python files. Although this is not entirely necessary for programs that will be run from within IDLE or will call Python explicitly at the terminal, it is required for programs that are run directly by calling the program’s filename. To ensure the program runs regardless of where the Python executable is installed, the first line of your program should read as follows: `#!/usr/bin/env python`.

This line tells the operating system to look at the \$PATH environment variable — which is where Linux stores the location of files that can be executed as programs—for

the location of Python, which should work on any Linux distribution used on the Pi. The \$PATH variable contains a list of directories where executable files are stored, and is used to find programs when you type their name at the console

or in a terminal window. To achieve the goal of printing out a message, you should use Python’s print command. As its name suggests, this command prints text to an output device—by default, to the console or terminal window from which the program is being executed. Its usage is simple: any text following the word print and placed between quotation marks will be printed to the standard output device. Enter the following line in your new project:

```
print “Hello, World!”
```

The final program should look like this: `#!/usr/bin/env python`
`print “Hello, World!”`

If you’re creating the example program in IDLE rather than a plain text editor, you’ll notice that the text is multi colored, (where colors are represented as differing shades of grey in the print edition). This is a feature known as syntax highlighting, and is a feature of IDEs and the more-advanced text editing tools. Syntax highlighting changes the color of sections of the text according to their function, in order to make the program easier to understand at a glance. It also makes it easy to spot so-called syntax errors caused by forgetting to put an end-quote in a print command or forgetting to comment out a remark.

For this short example, syntax highlighting isn’t necessary—but in larger programs, it can be an invaluable tool for finding errors. Syntax highlighting in IDLE. Before you run your program, save it as `helloworld.py` using the File menu. If you’re using IDLE, the file will be given the extension `.py` automatically. If you’re using a text editor, be sure to type `.py` at the end of the filename (not `.txt`) when you save it. This extension indicates that the file contains Python code—although Python is clever enough to run the program even if it’s saved with a different file extension. How you run the file will depend on whether you’re using IDLE or a text editor. In IDLE, simply choose Run Module from the Run menu, or press the F5 key on the keyboard. This will switch IDLE back to the

Python shell window and run the program. You should then see the message `Hello, World!` appear onscreen in blue. If not, check your syntax—in particular, check that you have quotation marks at both the beginning and end of the message on the print line.

1.2. Choice of platform:

Running helloworld.py in IDLE

If you created the helloworld.py program in a text editor, you'll need to open a terminal window from the Accessories menu on the desktop. If you saved the file anywhere except your home directory, you'll also have to use the cd command to change to that directory (see Chapter 2, "Linux System Administration"). Once you're in the right directory, you can run your program by typing the following: python helloworld.py. This tells the operating system to run Python and then load the helloworld.py file for execution. Unlike the Python shell in IDLE, Python will quit when it reaches the end of the file and return you to the terminal. The result, however, is the same: the message Hello, World! is printed to the standard output.

3. Processing steps

- Preparing the data set on Movie reviews.
- Preprocessing the data set
- Extracting the aspect terms.
- Differentiating aspect terms as positive, negative and neutral.
- Train the dataset using classification technique.
- Test the dataset.
- Calculate efficiency using performance metrics.
- Train and test the dataset.
- Calculate the efficiency using performance metrics.
- Compare the result with other algorithms.

4. Code

Libraries importing import pandas as pd

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from string import punctuation
from collections import Counter
%matplotlib inline

sentiment_data = pd.read_csv('training.txt', sep='\t')
sentiment_data.columns = ['Class', 'Data']
unlabeld_data = pd.read_csv('testdata.txt', sep='\t')
unlabeld_data.columns = ['Data']
Preprocessing pipeline

sentiment_data.head()

from sklearn.utils import shuffle
sentiment_data = shuffle(sentiment_data)
unlabeld_data = shuffle(unlabeld_data)
reviews_processed = []
unlabeled_processed = []
for review in reviews
review_cool_one = ".join([char for char in review if char
not in punctuation])
reviews_processed.append(review_cool_one) for review
in unlabeled_reviews:
review_cool_one = ".join([char for char in review if char
not in punctuation])
```

```
unlabeled_processed.append(review_cool_one)
```

5. CNN

```
import numpy

from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten

from keras.layers.convolutional import Conv1D

from keras.layers.convolutional import MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility seed
```

6. RNN

```
hidden_layer_size = 512 # how many nodes LSTM cells
will have number_of_layers = 1 # how many RNN layers
the network will use batch_size = 100 # how many
reviews we feed at onces learning_rate = 0.001 # learning
rate
number_of_words = len(vocab_to_int) + 1 #how many
unique words do we have in vocab (+1 is used for 0 -
padding)
dropout_rate = 0.8

embed_size = 300 #how long our word embeddings will be
epochs = 6 # how many epochs do we use for tr
for I in range(epochs):
    training_accuracy = []
    ii = 0
    epoch_loss = []
    while ii + batch_size <= len(X_train):
        X_batch = X_train(ii:ii+batch_size)
        Y_batch = Y_train(ii:ii+batch_size).reshape(-1,1)
        A,o,_ = Session.run([accuracy,cost,optimizer],
        Feed_dict={input:x_batch,targets:y_batch})

        Training_accuracy.append(a)
        Epoch_loss.append(o)
        ii += batch_size
    print('Epoch: {}/{}'. Format(i,epochs), '|Current loss:
    {}'. Format (np.mean(epoch_loss)),
    '| Training accuracy:
    {:.4f}'. format(np.mean(training_accuracy)*100))
```

IV. SYSTEM ANALYSIS

1. Existing Systems

The author provide a survey on hybrid approach to analyze aspect based sentiment of social data. It covers work on explicit and implicit aspects which is crucial for aspect based sentiment of data. Previous researches have assumed that social data i.e, tweets or Facebook data level

classification which only determines general sentiment of data. The approach is done by using Dependency Parsing, Association rule mining, Senti word Net is used to solve aspect based sentimental analysis. The work is done on Hate crime sentiment data set and Stanford Twitter sentiment dataset.

For racial aspects from hate crime, the approach successfully classified 28% as neutral, 53% as hate and 19% as free data. Besides, it also classified 24% as neutral, 58% as hate and 18% as free towards sexual aspects. Finally, for religion aspects, the hybrid approach classified 32% as neutral, 48% as hate and 20% as free.

2. Approaches in aspect based opinion mining

In general not all people like or dislike all aspects of a product, it differs from every individual, from a review of 1000 sentences it is difficult for potential customer to read reviews of all aspects at a time and make a decision whether to purchase the product or not. Many research works have been carried out in order to overcome this difficulty. Most of the early works of aspect based opinion mining are categorized as one of these approaches. Frequency-based, Relation-based and Model-based approaches.

2.1. Frequency-based approach: This approach identifies the frequent aspects of product on which many people have expressed their opinion. If the occurrence of aspect related terms is more than that of the predefined threshold value then that term is considered to be frequent aspect.

2.2. Relation-based approach: This approach finds the relationship between the words and sentiments to identify aspects. For example, the sentence “This phone has a good display” denotes [sentiment, aspect] that the aspect is display of the phone and the sentiment, “good” is said to be positive. This approach usually engages part-of-speech (POS) patterns to extract aspects.

2.3. Model-based approach: In this approach, several models are created for aspect extraction so that it can be applied to domain independent data sets. This approach overcomes the limitation of frequency and relation based approach. Most commonly used mathematical model based on supervised learning techniques are Hidden Markov Model (HMM) and Conditional Random Field (CRF), and based on unsupervised topic modeling techniques are Probabilistic Latent Semantic Indexing (PLSI) and Latent Dirichlet Allocation (LDA).

3. Social media

68% of user’s will often go straight to a business’s social media profile to read reviews. Regardless of whether you sell products or offer a service, customers want to see real

people talking about your brand, what you have to offer and if it’s worth their time and money. Each social media platform has its own individual way of providing genuine information about your business to potential customers who may be using your services or products for the first time, or thinking about using your business.

- Reading online reviews is part of the decision process for customers. Roughly 77% of people read customer reviews before committing to purchase.
- Reviews written online are trusted. Around 88% of customers will trust online reviews more than a personal recommendation. Customers can see on your Facebook page for example, that it’s a genuine person that’s actively talking about your brand, on their own accord.
- Online reviews allow for the opportunity to provide good customer service. There may come a time when someone is unhappy with your product or service and because social media is so accessible, many will choose to voice their opinion on your Facebook or Twitter page. In the proposed system we have collected reviews about Ipod. As proposed system is based on aspects, first we need to extract aspects then work on it.

4. Technical Description

4.1. Input Dataset:

Input dataset is a collection of reviews on movies. For example:

Table 2 Input dataset of movies.

##Don't get this...keep browsing.
##The movie has so far been one of the most advertised and populated one so far.
##I'm still figuring out why.
Features [-2] ## There isn't much interesting in movie at all, except songs.
##My friend saw this movie but he didn't like it much, I look at other reviews on this site and tried out the, and it's definitely the most overrated in history.
Music [+2] ##MUSIC: This movie music is pretty good. ## The music is slow poisons.
screenplay[-1]## the movie screenplay is very slow.

Input data set is a collection of reviews. Data set is collected from NLTK python library. It contains nearly about 800 reviews. These reviews need to be preprocessed to remove unwanted and noisy data. Preprocessing is the main step in natural language processing. It involves several steps. After preprocessing extraction of aspects will be done. By using relative importance aspects will be extracted and separated as positive, negative and neutral. In training phase we have taken a few aspects. Then based on that we have given another few aspects for testing phase. Then by applying classification algorithms like Naive Bayes Classifier and Support Vector Machine an accuracy of classification can be determined.

4.2. Data preprocessing

Data pre-processing is an important step in the data mining process. Data-gathering methods are often loosely controlled, resulting in out-of-range values, impossible data combinations, missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data cleaning is the process of detecting and correcting corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

After cleaning, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores. Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

In this project we have taken reviews on ipod. These reviews are not cleaned. So preprocessing should be done. The entire project is done in python language with Pycharm as IDE by using NLTK package. Natural Language Tool Kit (NLTK) is a comprehensive Python library for natural language processing and text analytics. NLTK is often used for rapid prototyping of text processing programs and can even be used in production applications. In preprocessing the following steps will be followed. Stop word removal, stemming and lemmatizing. Then preprocessed sentences will be produced.

4.2.1. Stop word removal:

Stop words are common words that generally do not contribute to the meaning of a sentence, at least for the purposes of information retrieval and natural language processing. These are words such as 'the' and 'a'. Most search engines will filter out stop words from search queries and documents in order to save space in their index. NLTK comes with a stop words corpus that contains word lists for many languages.

4.2.2. Stemming:

Stemming is a technique to remove affixes from a word, ending up with the stem. For example, the stem of cooking is cook, and a good stemming algorithm knows that the "ing" suffix can be removed. Stemming is most

commonly used by search engines for indexing words. Instead of storing all forms of a word, a search engine can store only the stems, greatly reducing the size of index while increasing retrieval accuracy. One of the most common stemming algorithms is the Porter stemming algorithm by Martin Porter. It is designed to remove and replace well-known suffixes of English words.

4.2.3. Lemmatization:

Lemmatization is very similar to stemming, but is more akin to synonym replacement. A lemma is a root word, as opposed to the root stem. So unlike stemming, you are always left with a valid word that means the same thing. However, the word you end up with can be completely different.

4.3. Algorithm for Data-Preprocess:

Input: A text file containing all the reviews

Output: Sentences free from stop words, and stemmed, lemmatized words.

Method: Tokenize the given data set D into sentences $S_1, S_2, S_3, \dots, S_i$ again tokenize the sentence into words $W_1, W_2, W_3, \dots, W_j$.

For each word W in a sentence, S look for the presence of it in stop word. If you found it, skip that word else include it and go for another word.

5. Supervised learning:

Supervised learning is the machine learning task of inferring a function from labeled training data. Each example is a pair consisting of an input object and a desired output value. The data are labelled with pre-defined classes. It is like a teacher gives the classes. Test cases are into these classes too. Use training data to infer model. Apply model to test data.

Also, it is used in many situations of natural learning. A set of input and output patterns called a training set is required for this learning mode. Typically, supervised learning rewards accurate classifications or associations and punishes those which yield inaccurate responses. The teacher estimates the negative error gradient direction and reduces the error accordingly.

In many situations, the inputs, outputs and the computed gradient are deterministic. However, the minimization of error proceeds over all its random realizations. As a result, most supervised learning algorithms reduce to stochastic minimization of error in multidimensional weight space.

Supervised learning networks represent the main stream of the development in neural networks. Some examples of well-known pioneering networks include the Perceptron, ADALINE/MADALINE, and various multilayer networks. Two phases are involved in a supervised learning network: retrieving phase and learning phase.

Some of the most predominant Supervised Learning techniques in

Sentiment Analysis have been SVM, Nave Bayesian Classifiers and other Decision Trees. A Naive Bayes classifier is a simple probabilistic model based on the Bayes rule along with a strong independence assumption. Support Vector Machines has outperformed other classifiers such as Naive Bayes.

5.1. Naive Bayes Classifier:

Bayesian classifiers are based around the Bayes rule, a way of looking at conditional probabilities that allows you to flip the condition around in a convenient way. A conditional probably is a probably that event X will occur, given the evidence Y. That is normally written $P(X | Y)$. The Bayes rule allows us to determine this probability when all we have is the probability of the opposite result and of the two components individually.

$$P(X | Y) = P(X) P(Y | X) / P(Y).$$

This restatement can be very helpful when we are trying to estimate the probability of something based on examples of it occurring.

In this case, we are trying to estimate the probability that a document is positive or negative, given its contents. We can restate that, so that is in terms of the probability of that document occurring if it has been predetermined to be positive or negative. This is convenient, because we have examples of positive and negative opinions from our data set above.

The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems.

Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data. The point of view that renders this process a “naive” Bayesian one is that we make a large assumption about how we can calculate the probability of the document occurring; it is equal to the product of the probabilities of each word within its occurrence. This implies that there is no link between one word and another word. Independence assumption it is called. We can estimate the probability of a word occurring, given a positive or negative sentiment by looking through a series of examples of positive and negative sentiments and counting how often it occurs in

each class. This is what makes this supervised learning, the requirement for pre classified examples to train on.

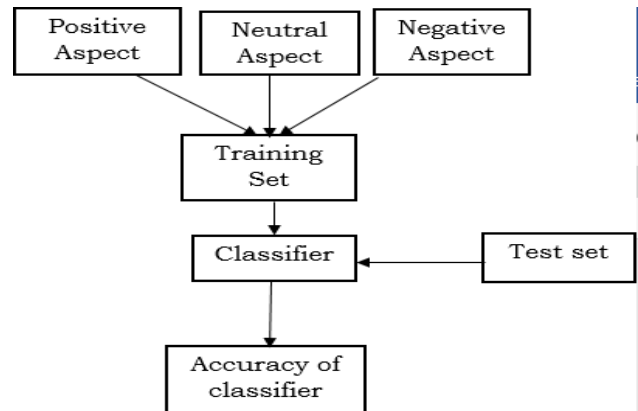


Figure 4 Main Methodology of Naive-Bayes classifier.

Assume independence among attributes A_i when class is given:

- $P(A_1, A_2, \dots, A_n | C) = P(A_1 | C_j) P(A_2 | C_j) \dots P(A_n | C_j)$
- Can estimate $P(A_i | C_j)$ for all A_i and C_j .
- New point is classified to C_j if $P(C_j) \prod P(A_i | C_j)$ is maximal.

For continuous attributes:

Discretize the range into bins. One ordinal attribute per bin. Violates independence assumption. Two-way split: $(A < v)$ or $(A > v)$. Choose only one of the two splits as new attribute. Probability density estimation: Assume attribute follows a normal distribution. Use data to estimate parameters of distribution (e.g., mean and standard deviation). Once probability distribution is known, can use it to estimate the conditional probability $P(A_i | c)$

For discrete attributes:

$$P(A_i | C_k) = |A_{ik}| / N_c$$

where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C .

Advantages of the Naive Bayes Classification technique can be summarized as follows:

Robust to isolated noise points. Handle missing values by ignoring the instance during probability estimate calculations. Robust to irrelevant attributes. It is fast and space efficient. It is not sensitive to irrelevant features. Independence assumption may not hold for some attributes. Fast to train (single scan). Fast to classify. Not sensitive to irrelevant features. Handles real and discrete data. Handles streaming data well. Use other techniques such as Bayesian Belief Networks (BBN).

Disadvantages of the Naive Bayes Classification technique:

Assumes independence of features. It makes a very strong assumption on the shape of your data distribution, i.e. any

two features are independent given the output class. Due to this, the result can be (potentially) very bad - hence, a "naive" classifier. This is not as terrible as people generally think, because the NB classifier can be optimal even if the assumption is violated, and its results can be good even in the case of sub-optimality.

5.2. Support Vector Machine:

SVM utilizes an optimum linear separating hyper plane to separate two data sets in a feature space. This optimum hyper plane is produced by maximizing minimum margin between the two sets. Therefore the resulting hyper plane will only be depended on border training patterns called support vectors. The support vector machine operates on two mathematical operations:

- Nonlinear mapping of an input vector into a high-dimensional feature space that is hidden from both the input and output.
- Construction of an optimal hyper plane for separating the features. SVMs are a new technique suitable for binary classification tasks, which is related to and contains elements of non-parametric applied statistics, neural networks and machine learning.

Like classical techniques, SVMs also classify a company as solvent or insolvent according to its score value, which is a function of selected financial ratios. But this function is neither linear nor parametric. The formal basics of SVMs will be subsequently briefly explained. The case of a linear SVM, where the score function is still linear and parametric, will first be introduced, in order to clarify the concept of margin maximization in a simplified context. Afterwards the SVM will be made non-linear and non-parametric by introducing a kernel. As explained further, it is this characteristic that makes SVMs a useful tool for credit scoring, in the case the distribution assumptions about available input data cannot be made or their relation to the PD is non-monotone.

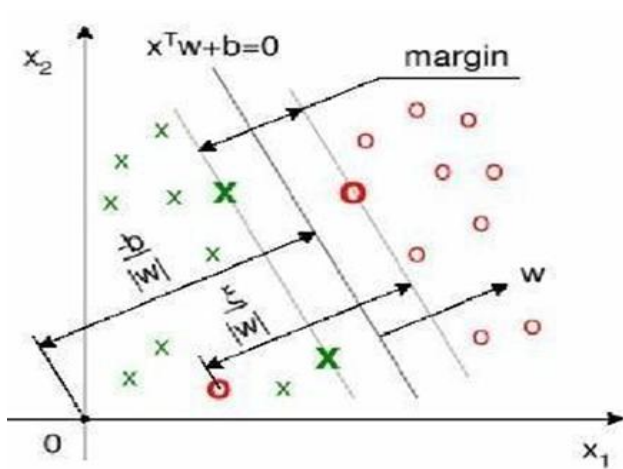


Figure 5 Geometrical Representation of SVM Classification.

The advantages of the SVM technique can be summarized as follows:

- By introducing the kernel, SVMs gain flexibility in the choice of the form of the threshold separating solvent from insolvent companies, which needs not be linear and even needs not have the same functional form for all data, since its function is non-parametric and operates locally.
- SVMs provide a good out-of-sample generalization, if the parameters C and r (in the case of a Gaussian kernel) are appropriately chosen. This means that, by choosing an appropriate generalization grade, SVMs can be robust, even when the training sample has some bias. By choosing different r values for different input values, it is possible to rescale outlier.
- SVMs deliver a unique solution, since the optimality problem is convex. This is an advantage compared to Neural Networks, which have multiple solutions associated with local minima and for this reason may not be robust over different samples.

Disadvantages of SVM:

A common disadvantage of non-parametric techniques such as SVMs is the lack of transparency of results. SVMs cannot represent the score of all companies as a simple parametric function of the financial ratios, since its dimension may be very high. It is neither a linear combination of single financial ratios nor has it another simple functional form. The weights of the financial ratios are not constant. Thus the marginal contribution of each financial ratio to the score is variable. Using a Gaussian kernel each company has its own weights according to the difference between the value of their own financial ratios and those of the support vectors of the training data sample.

6. Baseline Models, Techniques and Algorithms used

- Logistic Regression
- KNN (K-Nearest Neighbours)
- Naive Bayes
- SVM (Support Vector Machine)
- TF-IDF (Term Frequency - Inverse Document Frequency)
- N-Gram
- Word cloud
- Confusion Matrices
- 5-Fold Cross Validation etc.,

7. Proposed System

Neural Networks are a class of models within the general machine learning literature. Neural networks are a specific set of algorithms that have revolutionized machine learning. They are inspired by biological neural networks and the current so-called deep neural networks have proven to work quite well. Neural Networks are themselves general function approximations, which is why they can

be applied to almost any machine learning problem about learning a complex mapping from the input to the output space. Neural networks are one of the most beautiful programming paradigms ever invented. In the conventional approach to programming, we tell the computer what to do and break big problems up into many small, precisely defined tasks that the computer can easily perform. In contrast, we don't tell the computer how to solve our problems for a neural network. Instead, it learns from observational data and figures out its own solution to the problem. Today, deep neural networks and deep learning achieve outstanding performance for many important problems in computer vision, speech recognition, and natural language processing. They're being deployed on a large scale by companies such as Google, Microsoft, and Facebook. I hope that this post helps you learn the core concepts of neural networks, including modern techniques for deep learning.

7.1. Enhanced Techniques:

- Data Set and data preprocessing
- Convolution Neural Networks(CNN)Algorithm
- Recurrent Neural Network (RNN) Algorithm
- Activation Functions
- Cloud Deployment

7.1.1. Data set and Data preprocessing

Sourcing the labelled data for training a deep learning model is one of the most difficult parts of building a model. The data set "cleandata.txt" consists of 239,233 lines of sentences with an index for each line. The index is used to match each of the sentences to a sentiment score in the file "labels.txt". The score ranges from 0 to 1, 0 being very negative and 1 being very positive.

The data is split into 3 parts:

- **Train.csv:** This is the main data which is used to train the model. This is 50% of the overall data.
- **Val.csv:** This is a validation data set to be used to ensure the model does not overfit. This is 25% of the overall data.
- **Test.csv:** This is used to test the accuracy of the model post training. This is 25% of the overall data.

We will use CNN&RNN and in particular LSTMs, to perform sentiment analysis in Keras . Conveniently, Keras has a built-in IMDB movie reviews data set that we can use.

7.1.2. Convolution Neural Networks

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects

detections, recognition faces etc, are some of the areas where CNNs are widely used.

CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modelling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

7.1.2.1. CNN Architecture

Convolutional Neural Networks (CNN) is one of the variants of neural networks used heavily in the field of Computer Vision. It derives its name from the type of hidden layers it consists of. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. Here it simply means that instead of using the normal activation functions defined above, convolution and pooling functions are used as activation functions

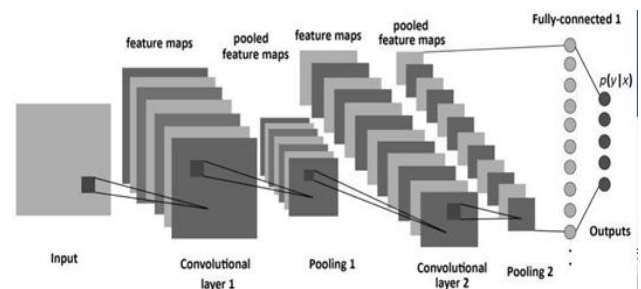


Figure 6 CNN architecture.

To understand it in detail one needs to understand what convolution and pooling are. Both of these concepts are borrowed from the field of Computer Vision and are defined as

7.1.2.2. Convolution: Convolution operates on two signals (in 1D) or two images (in 2D): you can think of one as the "input" signal (or image), and the other (called the kernel) as a "filter" on the input image, producing an output image (so convolution takes two images as input and produces a third as output).

In layman terms it takes in an input signal and applies a filter over it, essentially multiplies the input signal with the kernel to get the modified signal. Mathematically, a convolution of two functions f and g is defined as

$$(f * g)(i) = \sum_{j=1}^m g(j) \cdot f(i - j + \frac{m}{z})$$

which, is nothing but dot product of the input function and a kernel function.

7.1.2.3. Pooling: Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

There are 2 main types of pooling commonly known as max and min pooling. As the name suggests max pooling is based on picking up the maximum value from the selected region and min pooling is based on picking up the minimum value from the selected region.

Thus as one can see A Convolutional Neural Network or CNN is basically a deep neural network which consists of hidden layers having convolution and pooling functions in addition to the activation function for introducing non-linearity.

7.1.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks. RNN's make use of sequential information such as text. In a "traditional" feed forward neural network we assume that all inputs are independent of each other. But for many tasks that's a very bad idea. A sentence, for example, has a clear grammatical structure and order, where each word depends on the previous word. If you want your neural network to learn the meaning (or sentiment in our case) the network must know which words came in which order. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far.

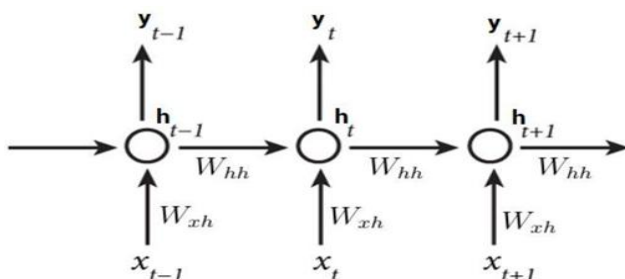


Figure 7 Recurrent Neural Networks Architecture.

$x(t-1)$, $x(t)$, $x(t+1)$ are sequential inputs that depend on each other (such as words in a sentence). $y(t-1)$, $y(t)$, $y(t+1)$ are the outputs. Unique for RNN is the fact that the calculation of the current hidden state $h(t)$ of the neurons for the input $x(t)$ depends on the previous hidden state $h(t-1)$ For the previous input $x(t-1)$. W_{xh} and W_{hh} are weight matrices that connect the input $x(t)$ with the hidden layer $h(t)$, and $h(t)$ with $h(t-1)$ respectively. This way we introduce a recurrence to the neural network which can be considered as a memory on the previous inputs. In theory, this way "vanilla" RNNs can make use of information in arbitrarily long sequences, but in practice, they are limited to looking back only a few steps.

7.1.4. Activation Function

7.1.4.1. Sigmoid Function

A sigmoid function or logistic function is defined mathematically as

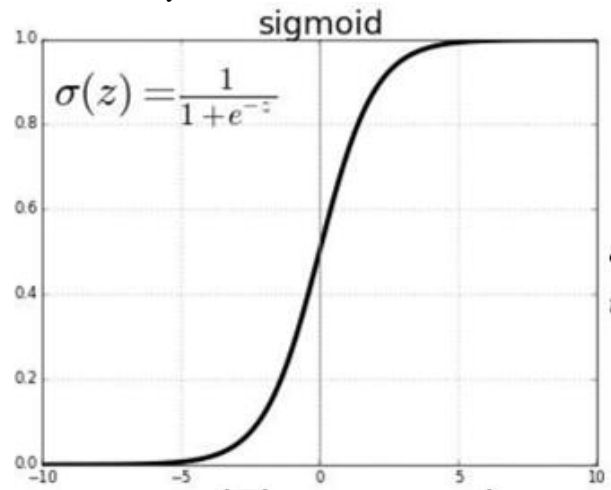


Figure 8 Sigmoid Function.

The value of the function tends to zero when z or independent variable tends to negative infinity and tends to 1 when z tends to infinity. It needs to be kept in mind that this function represents an approximation of the behaviour of the dependent variable and is an assumption. Now the question arises as to why we use the sigmoid function as one of the approximation functions. There are certain simple reasons for this.

- It captures non-linearity in the data. Albeit in an approximated form, but the concept of non-linearity is essential for accurate modelling.
- The sigmoid function is differentiable throughout and hence can be used with gradient descent and backpropagation approaches for calculating weights of different layers.
- The assumption of a dependent variable to follow a sigmoid function inherently assumes a Gaussian distribution for the independent variable which is a general distribution we see for a lot of randomly

occurring events and this is a good generic distribution to start with. However, a sigmoid function also suffers from a problem of vanishing gradients. As can be seen from the picture a sigmoid function squashes its input into a very small output range [0,1] and has very steep gradients. Thus, there remain large regions of input space, where even a large change produces a very small change in the output. This is referred to as the problem of vanishing gradient. This problem increases with an increase in the number of layers and thus stagnates the learning of a neural network at a certain level.

7.1.4.2. Tanh Function

The $\tanh(z)$ function is a rescaled version of the sigmoid, and its output range is $[-1, 1]$ instead of $[0, 1]$. The general reason for using a Tanh function in some places instead of the sigmoid function is because since data is centered around 0, the derivatives are higher. A higher gradient helps in a better learning rate.

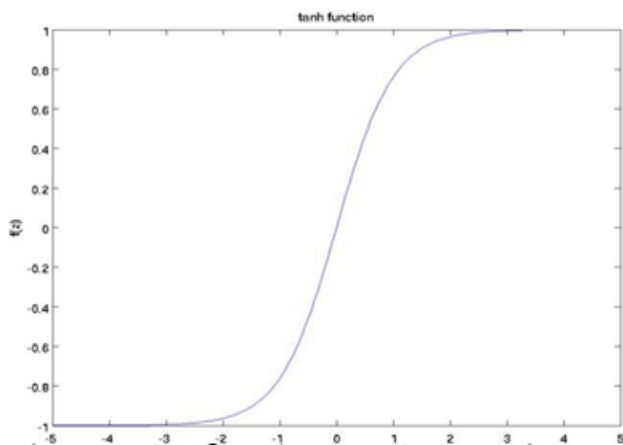


Figure 9 Tanh Function.

7.1.4.3. ReLU Function

The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x , it returns that value back. So, it can be written as $f(x) = \max(0, x)$. Graphically it looks like this

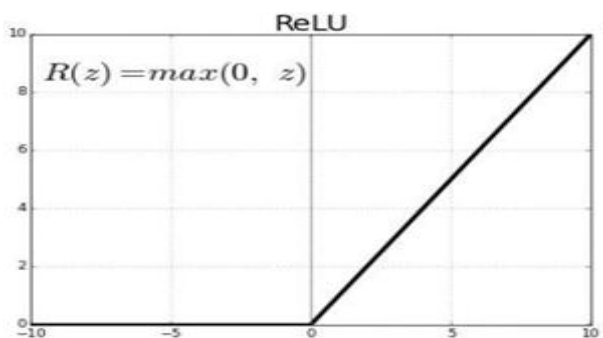


Figure 10 ReLU Function.

The Leaky ReLU is one of the most well-known. It is the same as ReLU for positive numbers. But instead of being 0 for all negative values, it has a constant slope (less than 1). That slope is a parameter the user sets when building the model, and it is frequently called α . For example, if the user sets $\alpha=0.3$, the activation function is $f(x) = \max(0.3*x, x)$. This has the theoretical advantage that, by being influenced, by x at all values, it may make more complete use of the information contained in x . There are other alternatives, but both practitioners and researchers have generally found an insufficient benefit to justify using anything other than ReLU. In general practice as well, ReLU has found to be performing better than sigmoid or tanh functions.

V. RESULTS

1. Output Screens of Existing System

The following are the output screens of feature extractions:

- N-grams features 76682
- The cross validation Scores are as follows:
Logistic Regression : 0.344152
Naïve Bayes : 0.360470
Linear SVC : 0.535262

1.1. Naive Bayes

Multinomial Naive Bayes score : 0.511397807740809
Logistic Regression score : 0.5121091602806609

```

Train Score for Naive Bayes for number of features 1000 : 0.551714939332455
Test Score for Naive Bayes for number of features 1000 : 0.5521712419568662
[[ 13 255 1124 43 0]
 [ 6 471 4785 185 0]
 [ 2 177 15095 487 2]
 [ 0 59 5018 1437 23]
 [ 0 5 1067 692 61]]
Train Score for Naive Bayes for number of features 2000 : 0.5711560380576686
Test Score for Naive Bayes for number of features 2000 : 0.5670126426746855
[[ 45 363 985 42 0]
 [ 19 751 4415 179 3]
 [ 10 306 14850 589 8]
 [ 2 70 4640 1771 54]
 [ 0 10 890 806 119]]
Train Score for Naive Bayes for number of features 10000 : 0.6222202462269718
Test Score for Naive Bayes for number of features 10000 : 0.5915219710932195
[[ 82 592 732 29 0]
 [ 45 1328 3809 184 1]
 [ 18 571 14303 855 16]
 [ 3 85 3962 2403 84]
 [ 0 10 602 1035 178]]
Train Score for Naive Bayes for number of features 20000 : 0.6424858738794086
Test Score for Naive Bayes for number of features 20000 : 0.5973097940311055
[[ 90 631 689 25 0]
 [ 63 1454 3679 168 3]
 [ 13 640 14173 917 20]
 [ 0 78 3814 2566 79]
 [ 0 13 538 1084 190]]
Train Score for Naive Bayes for number of features 50000 : 0.6840437485348444
Test Score for Naive Bayes for number of features 50000 : 0.6103404791929382
[[ 101 705 609 19 1]
 [ 51 1699 3498 116 3]
 [ 4 699 14071 983 6]
 [ 1 69 3577 2813 77]
 [ 1 8 478 1146 192]]
Train Score for Naive Bayes for number of features 60000 : 0.6915534286661223
Test Score for Naive Bayes for number of features 60000 : 0.6148672680829049
[[ 98 712 608 16 1]
 [ 41 1737 3474 113 2]
 [ 3 666 14099 989 6]
 [ 1 67 3508 2887 74]
 [ 1 7 447 1175 195]]
Train Score for Naive Bayes for number of features 66292 : 0.6931782356697681
Test Score for Naive Bayes for number of features 66292 : 0.6161606363371811
[[ 95 715 606 18 1]
 [ 37 1714 3503 111 2]
 [ 2 67 3440 2877 74]
 [ 1 7 447 1175 195]]

```

Fig 11 Naive Bayes.

1.2. SVM

KNN score : 0.4932583179745853

Linear SVM score : 0.5121091602806609

```

Train Score for svc for number of features 1000 : 0.5394035907426419
Test Score for svc for number of features 1000 : 0.526045203220487
[[ 596 298 383 83 75]
 [ 848 1388 2379 453 299]
 [ 896 1361 11344 1447 715]
 [ 292 446 2458 2056 1285]
 [ 56 53 314 517 885]]
Train Score for svc for number of features 2000 : 0.5844778387641767
Test Score for svc for number of features 2000 : 0.5537232838619977
[[ 661 370 281 75 48]
 [ 921 1619 2208 432 187]
 [ 688 1477 11487 1558 553]
 [ 248 356 2245 2373 1315]
 [ 30 42 218 550 985]]
Train Score for svc for number of features 10000 : 0.6792824981609772
Test Score for svc for number of features 10000 : 0.59767978045073884
[[ 805 444 133 37 16]
 [ 886 2309 1777 325 70]
 [ 453 1752 11377 1822 359]
 [ 111 306 1902 2970 1248]
 [ 14 31 106 650 1024]]
Train Score for svc for number of features 20000 : 0.717049156474573
Test Score for svc for number of features 20000 : 0.6062663691919682
[[ 780 500 115 27 13]
 [ 860 2509 1644 298 56]
 [ 396 1820 11330 1896 321]
 [ 75 251 1893 3090 1228]
 [ 8 23 82 671 1041]]
Train Score for svc for number of features 50000 : 0.7858973218977099
Test Score for svc for number of features 50000 : 0.6228214828467036
[[ 772 561 81 15 6]
 [ 830 2746 1528 224 39]
 [ 344 1787 11518 1869 245]
 [ 56 242 1859 3220 1160]
 [ 3 18 62 736 1006]]
Train Score for svc for number of features 60000 : 0.8019513851277615
Test Score for svc for number of features 60000 : 0.6259255666569664
[[ 763 570 85 12 5]
 [ 812 2800 1505 216 34]
 [ 323 1813 11507 1893 227]
 [ 52 227 1853 3294 1111]
 [ 3 16 66 746 994]]
Train Score for svc for number of features 66292 : 0.8109888688594825
Test Score for svc for number of features 66292 : 0.6266045849904615
[[ 760 579 80 11 5]
 [ 800 2700 1500 210 30]
 [ 320 1800 11500 1890 220]
 [ 50 220 1850 3290 1110]
 [ 3 16 66 746 994]]

```

Fig 12 SVM.

1.1. Successratios of SVM, Naïve Bayes, Logistic Regression

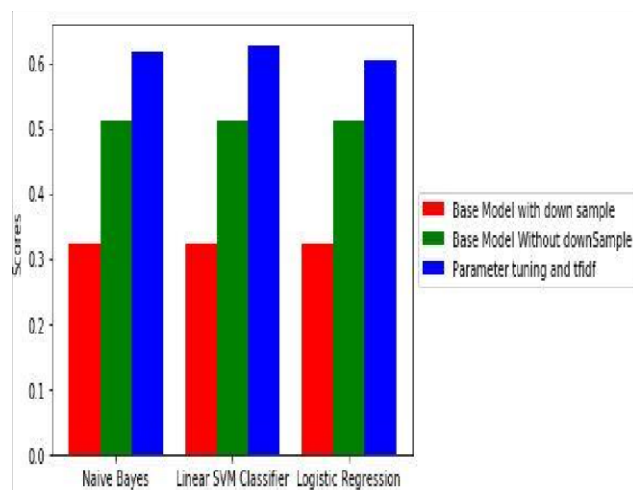


Figure 13 success ratios.

1.2. Accuracy for Different Models

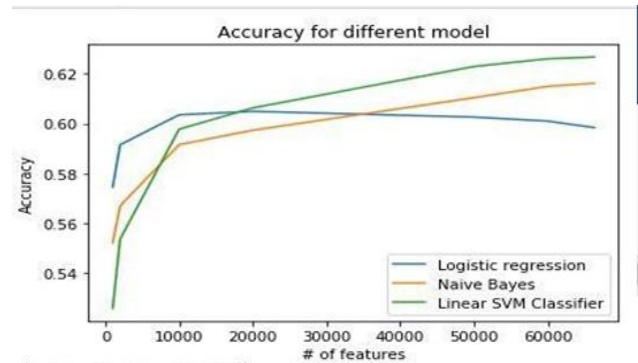


Figure 14 Accuracy of Different Models.

1.3. Naive Bayes confusion Matrix:

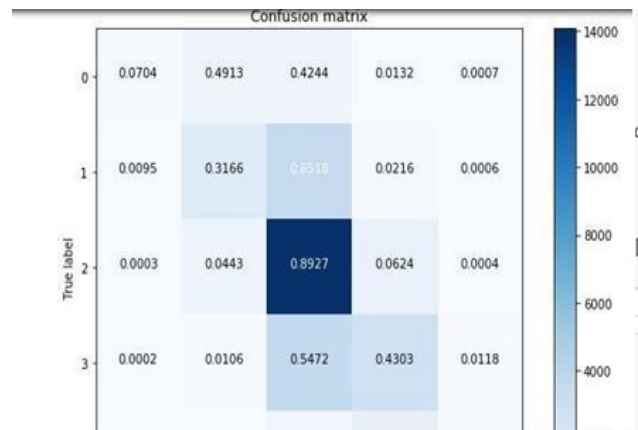


Figure 15 Confusion Matrix of Naive Bayes.

1.4. SVM Confusion Matrix

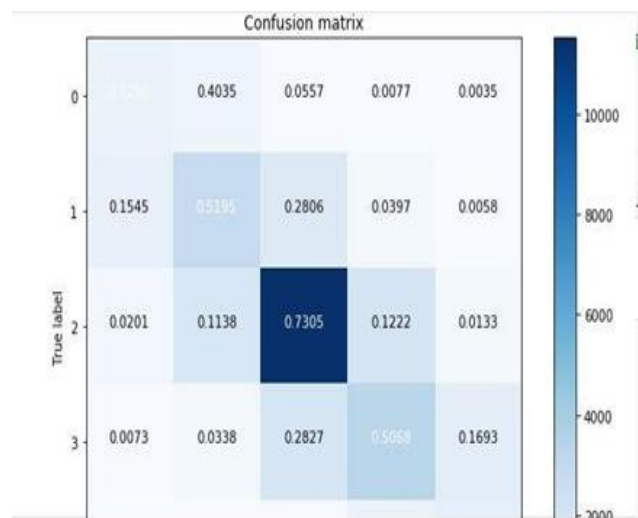


Figure 16 SVM Confusion Matrix.

1.5. Final Accuracy of Existing System

The Final Accuracy we got 79%

```

) binary--raise,
)
pipeline_svc = Pipeline(
    ('fidf', tfidf),
    ('svc', svc))

pipeline_svc.fit(X, y)

train_score = pipeline_svc.score(X, y)
print("Final Accuracy after training", train_score)
y_pr = pipeline_svc.predict(X)
return y_pr

#training on the whole data set and saving the predicted file in "output.csv"
numpy.savetxt("output.csv", final_model[train_150k['Cleaned_Phrase'], train_150k['Sentiment']], delimiter=',')

```

The final accuracy we got by training the data is 0.796

Fig 17 Final Accuracy of Existing System.

2. Proposed System Results

2.1. Data preprocessing

```

features = np.zeros((len(reviews_to_ints), seq_len), dtype=int)
for i, review in enumerate(reviews_to_ints):
    features[i, -len(review):] = np.array(review)[:seq_len]

features_test = np.zeros((len(unlabeled_to_ints), seq_len), dtype=int)
for i, review in enumerate(unlabeled_to_ints):
    features_test[i, -len(review):] = np.array(review)[:seq_len]

```

Step 1.9 Split into training and testing parts

```

In [16]: X_train = features[:6400]
        y_train = labels[:6400]

        X_test = features[6400:]
        y_test = labels[6400:]

        X_unlabeled = features_test

        print('X_train shape {}'.format(X_train.shape))
        print('X_unlabeled shape {}'.format(X_unlabeled.shape))

        X_train shape (6400, 250)
        X_unlabeled shape (28936, 250)

```

Done with preprocessing pipeline

Fig 18 Data preprocessing.

2.2. CNN Result Accuracy

The Accuracy we got by implementing CNN algorithm is 88.86%.

```

In [20]: import warnings
        warnings.filterwarnings('ignore')

        model = Sequential()
        model.add(Embedding(top_words, 32, input_length=max_words))
        model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
        model.add(MaxPooling1D(pool_size=2))
        model.add(Flatten())
        model.add(Dense(250, activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
        print(model.summary())

In [21]: import warnings
        warnings.filterwarnings('ignore')

        # Fit the model
        model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=2, batch_size=128, v
        # Final evaluation of the model
        scores = model.evaluate(X_test, y_test, verbose=0)
        print("Accuracy: %.2f%%" % (scores[1]*100))

WARNING:tensorflow:From c:\program files\python36\lib\site-packages\tensorflow\python\ops
orflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
- 118s - loss: 0.4350 - acc: 0.7756 - val_loss: 0.2761 - val_acc: 0.8836
Epoch 2/2
- 58s - loss: 0.2042 - acc: 0.9204 - val_loss: 0.2677 - val_acc: 0.8886
Accuracy: 88.86%

```

Fig 19 CNN Result Accuracy.

2.3. RNN result Accuracy

The Accuracy we got by implementing RNN algorithm is 99.00%

```

1318     return self.call_tf_sessionrun(
-> 1319         options, feed_dict, fetch_list, target_list, run_met
1320     )
1321     def _prun_fn(handle, feed_dict, fetch_list):
c:\program files\python36\lib\site-packages\tensorflow\python\client\s
t, fetch_list, target_list, run_metadata)
1405     return tf_session.TF_SessionRun_wrapper(
1406         self.session, options, feed_dict, fetch_list, target
-> 1407         run_metadata)
1408     def call_tf_sessionrun(self, handle, feed_dict, fetch_list
1409
KeyboardInterrupt:

```

```

In [32]: test_accuracy = []

        ii = 0
        while ii + batch_size <= len(X_test):
            X_batch = X_test[ii:ii+batch_size]
            y_batch = y_test[ii:ii+batch_size].reshape(-1, 1)

            a = session.run([accuracy], feed_dict={inputs:X_batch, targets:y_ba
            test_accuracy.append(a)
            ii += batch_size

In [29]: print("Test accuracy is {:.4f}%".format(np.mean(test_accuracy)*100))

Test accuracy is 99.0000%

```

Fig 20 RNN result Accuracy.

2.4. Cloud Deployment

In cloud Deployment, We run the code in Cloud but RAM is only of 512 MB and so it takes more time in order to run the code and get the accuracy.

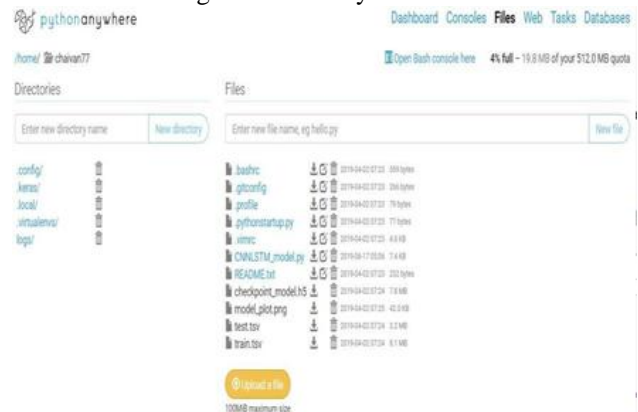


Fig 21 Cloud Deployment.

VI. CONCLUSION

The proposed system extracts the aspects from the movie reviews. The reviews are on the different movie reviews. These reviews contained noisy and some unwanted data. Those unwanted data is removed during preprocessing so that training the classifier will be easy. Nouns are considered as the aspect terms and extracted based on POS tagging. By using the threshold value, classification of the aspects into positive, negative and neutral is done. On the aspects Naive-bayes and Support Vector Machine are applied. In Proposed System Neural Network algorithms are applied. These classifiers are applied to find out the correct classification of aspects. Finally

experimental results are presented. These experimental results shows that accuracy in analyzing the sentimental state of people on movie. Support Vector Machine uses more complex functions to extract features, It consume more memory and time due to its algorithmic complexity. In proposed System Neural Network algorithms are applied yields more accuracy and time complexity compared to Existing System. Hence, the experimental result can provide that the proposed system yields greater performance rather than the existing system. We can experiment with different preprocessing techniques, heterogeneous features, supervised and unsupervised algorithms for developing a more accurate system. For handling large data, we can use proposed heterogeneous features and deep learning features such as Word2Vec, Doc2Paragraph and Word Embedding apply to deep learning algorithms such as Recurrent neural networks (RNNs) and Convolutional deep neural networks (CNNs) to get remarkable result.

1. Future Scope:

In future, it will be proposed to work on strong positive and strong negative aspect terms .This will be implemented by using the Fuzzy concepts. By using the Fuzzy logic, the results will be better and accurate when compared to existing methods.In the future, we can experiment with different preprocessing techniques, heterogeneous features, supervised and unsupervised alorithms for developing a more accurate system. For handling large data;We can use proposed heterogeneous features and deep learning features such as Word2vec,Doc2paragraph and word Embedding apply to deep learning algorithm.

REFERENCES

- [1] Rajalaxmi Hegde, Dr. Seema. S “Aspect Based Feature Extraction and Sentiment Classification of Review Data sets using Incremental Machine learning Algorithm” 3rd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB17)
- [2] Toqir A. Rana and Yu-N Cheah “Improving Aspect Extraction Using Aspect Frequency and Semantic Similarity-Based Approach for Aspect-Based Sentiment Analysis” Springer International Publishing AG 2018 P. Meesad et al. (eds.), Recent Advances in Information and Communication Technology 2017, Advances in Intelligent Systems and Computing 566.
- [3] Nurulhuda Zainuddin, Ali Selamat, and Roliana Ibrahim “Improving Twitter aspect based sentimental analysis using hybrid approach “ on 2016.
- [4] D V Nagarjuna Devi, Chinta Kishore Kumar, Siriki Prasad, “A Feature Based Approach for Sentiment Analysis by Using Support Vector Machine” 2016 IEEE 6th International Conference on Advanced Computing.
- [5] Shwetha Rana, Archana Singh “Comparative Analysis of Sentiment Orientation Using SVM and Naive Bayes Techniques” 2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016) Dehradun, India 14-16 October 2016.
- [6] Povoda, Lukas, Radim Burget, and Malay Kishore Dutta, "Sentiment analysis based on Support Vector Machine and Big Data", Telecommunications and Signal Processing (TSP), 2016 39th International Conference on, IEEE, 2016.
- [7] Edison Marrese-Taylor, Juan D. Velasquez, Felipe Bravo-Marquez and Yutaka Matsuo “Identifying Customer Preferences about Tourism Products using an Aspect-Based Opinion Mining Approach” 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.
- [8] Jingbo Zhu, Huizhen Wang, Muhua Zhu, Benjamin K. Tsou, Matthew Ma “Aspect-based opinion polling from customer reviews” IEEE Transactions on affective computing, Vol. 2, No.1, JanuaryMarch 2011.
- [9] Raisa Varghese and Jayasree M “Aspect Based Sentiment Analysis using Support Vector Machine Classifier” 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI).
- [10] Kim Schouten, Onne van der weiide, Flavius Frascinar, and Rommert Dekker “Supervised and Unsupervised Aspect Category Detection for Sentiment Analysis with Co-occurrence Data” IEEE Transactions on cybernetics.
- [11] Medhat, W., Hassan, A., Korashy, H.: Sentiment analysis algorithms and applications: a survey. Ain Shams Eng. J. 5(4), 1093–1113 (2014)
- [12] Bhuta, S., Doshi, A., Doshi, U., Narvekar, M.: A review of techniques for sentiment analysis of twitter data. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), pp. 583–591, February 2014
- [13] Khan, F.H., Bashir, S., Qamar, U.: Tom: twitter opinion mining framework using hybrid classification scheme. Decis. Support Syst. 57, 245–257 (2014)
- [14] Lek, H.H., Poo, D.: Aspect-based twitter sentiment classification. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 366–373, November 2013
- [15] Feldman, R.: Techniques and applications for sentiment analysis. Commun. ACM 56, 82–89 (2013)
- [16] Brychcin, T., Konkol, M., Steinberger, J.: Uwb: machine learning approach to aspect-based sentiment analysis. SemEval 2014, 817 (2014)
- [17] [17]. Liu, K.-L., Li, W.-J., Guo, M.: Emoticon smoothed language models for twitter sentiment analysis. In: AAI, vol. 2, pp. 1678–1684 (2012). cited By (since 1996)1.
- [18] Li, S., Zhou, L., Li, Y.: Improving aspect extraction by augmenting a frequencybased method with web-based

- similarity measures. *Inf. Process. Manage.* 51(1), 58–67 (2015)
- [19] Kansal, H., Toshniwal, D.: Aspect based summarization of context dependent opinion words. In: *Procedia Computer Science*, vol. 35, pp. 166–175, Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference, KES2014 Gdynia, Poland, Proceedings, September 2014.
- [20] Camara E.M., Martin-Valdivia, M.T., Lopez, L.A.U., Montejo-Raez, A Sentiment analysis in twitter. *Nat. Lang. Eng.* 20(1), 1-28 (2014). By (since 1996) Pak, A., Paroubek, P : Twitter as a corpus for sentiment analysis and opinion mining. In: *Proceedings of the seventh Conference on International Language Resources and Evaluation (LREC 2010)*, (Valletta, Malta), European Language Resources Association (ELRA), May 2010.