Sentiment Classification Of Movie Review And Twitter Data Using Machine Learning

Prafulla Mohapatra[1], Rohit Kumar Singh[2], Shashank Pandey[3], PrashanthAnand Kumar[4], Mrs.Asha K N[5]

[5]Assistant Professor

[1,2,3,4,5]*Department of Computer Science & Engineering , Dr. Ambedkar Institute of Technology, Bengaluru-56, India*

Abstract*— *Over three billion people use some form of social media in their day to day lives. Therefore, it is not unwise to say that social media is one of the single largest collection of data about humans present, in the world currently. Sentiment analysis is one of the most common operations done on social media data. In this paper, we perform sentiment analysis, using a variety of vectorizers and classifiers to see which combination yields the highest accuracy. Analysis is performed on Twitter and movie review data. The two data sets are inherently different and therefore there could be a difference between the accuracies. The front end of this application is web based. Twitter and movie review data are collected from two API's in real time and then the different tweets/reviews are classified as either being positive or negative. This is then presented in the form of a donut graph.

Keywords — *Sentiment Analysis, Machine Learning, Information Retrieval, Opinion Mining and Natural language processing.*

I. INTRODUCTION

Sentiment is the emotion attached to a sentence; it can be a positive emotion or a negative emotion. Sentiment analysis is described as the ability to associate a specific sentiment (or emotion) to a particular sentence, paragraph, or even a whole document. Sentiment analysis offers any organization, the ability to monitor various social media sites in real time and act accordingly. It can be used to get to know about the current trends and topics and then at the end, make an educated choice while selecting among different products, services or even human beings!

In the past decade, new forms of communication, such as micro blogging and text messaging have emerged and becomeubiquitous. While there is no limit to the range of information conveyed by tweets and texts, it's difficult to deduce any valuable inferences from this corpus. These _tweets and texts' are often short and are used to share opinions and sentiments that people have, about what is going on in the world around them. There has been another major development in the last twenty years - reviews. The number of avenues to put forth one's opinion about a certain thing has increased tenfold. The _things' here can refer to anything from food/décor of a restaurant, movie or product reviews from Amazon. These reviews can be collated, pre-processed and then finally analysed to find interesting, relevant patterns. Patterns that will help us discern general public discourse associated with a particular subject.

Our paper aims to use huge amounts of data from two digital API's – Movie db and Twitter API, and along with sophisticated machine learning techniques, creates a web application which uses sentiment analysis to show the opinion / polarity associated with a subject. Data here refers to the thousands of online posts available. The result is then visually represented using various graphs which any user can use, to get the relevant information by a mere glance. The user would enter a word and theapplication would show the public opinion associated with that word or a phrase through appealing visuals.

Thus our application acts as a one stop shop at which user can get information about a person, product or a service. Information is presented in an aesthetically pleasing way and is gotten through analysing experiences shared by thousands of other people on a digital platform. People need not have multiple apps and have to personally go through several reviews to get a clear picture about a particular thing; they can achieve it in a fraction of the time by a click.

II. OVERVIEW

This project involves making a Web application that performs sentiment analysis on social media and movie review data. Data analytics where sentiment analysis takes place is the backend of the system. We use python language to build this, mainly using Jupyter notebooks, for their simplicity and robust performance. Data was collected, and then pre-processed, to make sure it's in the optimal state before advanced algorithms were applied to it. Then different vectorizers were used along with different classifiers to check which combination yields the best possible accuracy on the data sets. Each classifier's optimal value of the select parameter was found and then, that value was applied in its final iteration. Various natural language processing

operations like stemming, lemmatization etc. were performed to see if there was an increase in accuracy. Finally, the accuracies achieved on the two data sets were compared to see if there was a disparity present. Then different operations were performed, to try and reduce the disparity

.

Web application was chosen as the frontend, because, the data is safer against the instances of personal system crashes and the data would still be accessible in the cloud next time we log into the web application system. To develop the python web application we have used Flask framework. We have created an instinct_flask.py file, which acts as a local server, which is regularly used to set up the routes that will become the application's points of interaction. This file also acts as a REST (Representational State Transfer) API which allows communication between a web-based client and server that employs representational state transfer (REST) constraints. We have used basic html and css files to design the front end. We represent the sentiment using a donut chart with proper illustration for positive and negative tweets. The donut chart is drawn using the Google Chart API which is an interactive Web service that creates graphical charts from user-supplied data.

The real time tweets are acquired using Tweetpy which is an open-sourced, hosted on GitHub and enables Python to communicate with Twitter platform and use its API. It helps us to acquire a certain number of tweets from the Twitter based on a given query. Movie db is a similar API used to acquire movie reviews. Our web application allows the user to search twitter or movie sites for relevant topics (The topic here can be a name, place, person or an event) and present the user with the perception of the said topic by the people, based on our sentiment classifier.

III. DATA

A. Twitter data

The dataset is taken by combining 1,600,000 tweets with emoticons pre-removed. The dataset was collected using the Twitter API for use in the paper [1].
In their approach, they assumed that any tweet with positive emoticons, like :), were positive, and tweets with negative emoticons, like :(, were negative. They used the Twitter Search API to collect these tweets by using keyword search. The no of Positive and negative tweets are the same.

B. Movie review data

25000 reviews are provided and as the twitter data, the positive and negative reviews are equally divided. The reviews are roughly 4-5 lines each.

IV. DATA PRE-PROCESSING

A. Pre-processing for twitter data

a) HTML encoding: In general, HTML encoding has not been converted to text, that is:

```
'inLOveeee&lt;3 and it hurts '
```

&lt‘ is an html tag(less than), we have to convert it into the desired form (<‘).
The following command decodes it:

$ex = BeautifulSoup(dftrain.text[2204], 'lxml')$
$print(ex.get\_text())$

```
'in LOveeee <3 and it hurts '
```

b) Removal of mentions:Mentions (@name) don‘t add to the sentiment of the tweet.

```
'@angry_barista I baked you a cake
but I atedit '
```

The following code removes them:

$re.sub(r'@[A-Za-z0-9\_]+', '', dftrain.text[22])$

```
' I baked you a cake but I ated it '
```

c) Removal of links:Links also don't add value to the sentiment of the tweet, hence must be removed.

```
"@switchfoothttp://twitpic.com/2y1zl
- Awww, that's a bummer.  You shoulda
got David Carr of Third Day to do it.
;D"
```

The following code removes them:

$re.sub('ttps?://[^]+', '', dftrain.text[0])$

```
"@switchfoot -Awww, that's a bummer.
You shoulda got David Carr of Third
Day to do
it. ;D"
```

Links can also start with _www‘.
The following code removes such links:

$re.sub('www.[^]+', '', dftrain.text[79])$

```
'wonders why someone that u like so
much can make you so unhappy in a
split seccond . depressed . '
```

d) Removal of non-letter characters:Numbers, punctuations or any other special characters are not useful, however the content of the hash tag could be useful, therefore only the hash tag is let go, the content succeeded by the hash tag is kept just as it is.

```
'@Kenichan I dived many times for the
ball. Managed to save 50%     The rest
go out of bounds'
```

The following code removes them:

$re.sub("[^a-zA-Z]", " ", dftrain.text[2])$

```
' Kenichan I dived many times for the
ball  Managed to save       The rest
go out of bounds'
```

e) Expansion of negation words: While removing punctuations, _can't' becomes _can t', which is problematic as can and can't are understood as similar by the learning process. To take care of that all such words, we manually expand them into their constituent words.

"@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there. "

The following code handles them:

negations_dic = {"isn't":"is not", "aren't":"are not", "wasn't":"was not", "weren't":"werenot","haven't":"havenot","hasn't":"hasnot","hadn't":"hadnot","won't":"will not", "wouldn't":"would not", "don't":"do not", "doesn't":"doesnot","didn't":"did not", "can't":"cannot","couldn't":"couldnot","shouldn't":"shouldnot","mightn't":"might not", "mustn't":"must not"}

$$neg\_pattern = re.compile(r'\b(' + '|'.join(negations\_dic.keys()) + r')\b')$$

neg_handled = neg_pattern.sub(lambda x: negations_dic[group()], dftrain.text[4])

"@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can not see you all over there. "

f) Removal of extra spaces: Sometimes unnecessary white spaces have been created because of the removal of unwanted characters, we will tokenize and join together to remove unnecessary white spaces.

$extra\_spaces = re.sub("[^a-z-Z]", " ", dftrain.text[2])$ '

Kenichan I dived many times for the ball

Managed to save The rest go out of bounds'

B. Pre-processing for movie review data

The following code is used to remove the punctuations which contribute nothing to the sentiment of the text.

reviews = [re.sub("(\.)|(\;)|(\:)|(\!)|(\?)|(\,)|(\")|(\()|(\))|(\[)|(\])|(\d+)","",line.lower()) *for line in reviews*]

The following code is used to remove break tags (<br \>), which are found throughout the reviews.

reviews = [re.sub("(<br\s/><br\s*/>)|(\-)|(\/)"," ",line) for line in reviews]*

V. TRAINING ANDTEST DATA SPLIT

The data is firstly divided to training and test sets. Then the some of the data is further taken out of the training data is to be validation data. The difference between validation and test being that validation is part of the training data when it's fit into the vectorizer and transformed, however the test data isn't present, and is transformed based on the training data. Since there are 1.6 million tweets, 1 per cent is enough to be test data and validation data. In the case of movie reviews there are 25,000 data instances, hence we use a 25-75 split for both validation sets. 25,000 reviews were used as test cases. Accuracies were checked for different values of the select parameter in each classifier to see which the optimal value for the parameter is. For logistic regression the parameter is c (Inverse of regularization strength; must be a positive float, smaller values specify stronger regularization), for multinomial NB its alpha (Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing)) and finally for linear svc it's again c (Penalty parameter C of the error term).

VI. RESULTS

Two vectorizers namely, Count and Tf-Idf vectorizers were used along with 3 classifiers namely Logistic regression, Multinomial NB and Linear SVC. Different values of c/alpha parameters were used for each of the classifiers for the validation data sets to see which yields the highest accuracy. Then various nlp (Natural language processing) techniques were applied before Logistic regression was used to classify to see which technique yields the highest accuracy, then that technique was applied before classifiers to compare accuracies between the classifiers. These techniques are: removal of stopwords as shown in [5], Stemming (Porter stemmer is used as in [2]) and finally, Lemmatization (as used in [3]) and n-grams (shown in [4]). For the final accuracy, stopwords were removed, lemmatization was performed with ngram= (1, 2) and test data was used.

A. Accuracies found with count vectorizer on Twitter data.

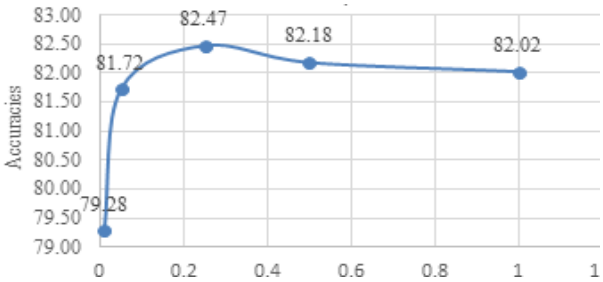

Fig.1 Logistic regression, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

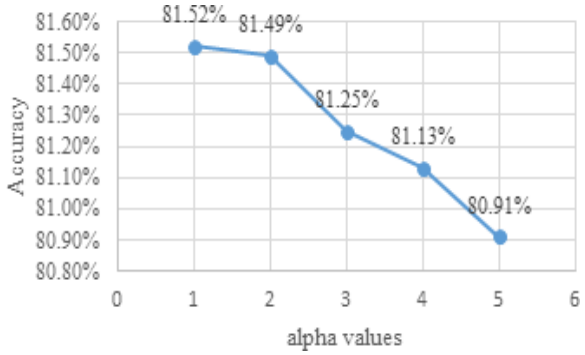

Fig. 2 Multinomial NB, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

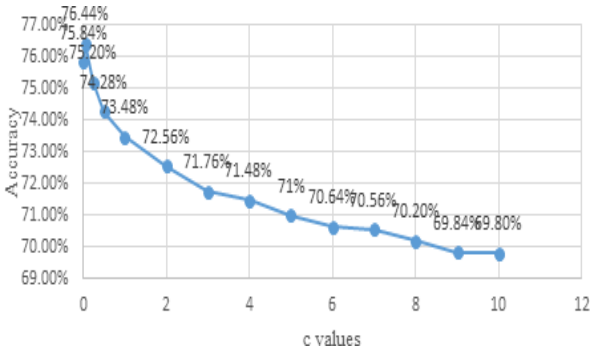

Fig. 3 Linear SVC, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

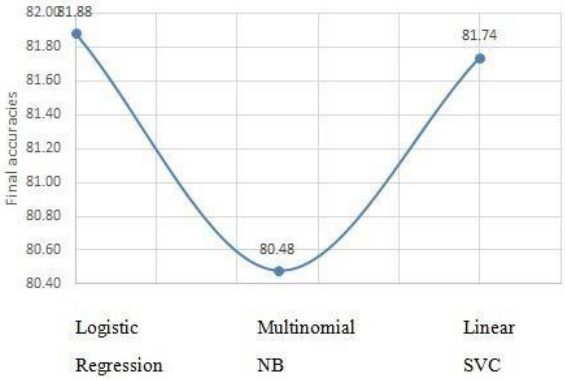

Fig. 4 Final Accuracy

B. Accuracies found with Tf-Idf vectorizer on Twitter data.

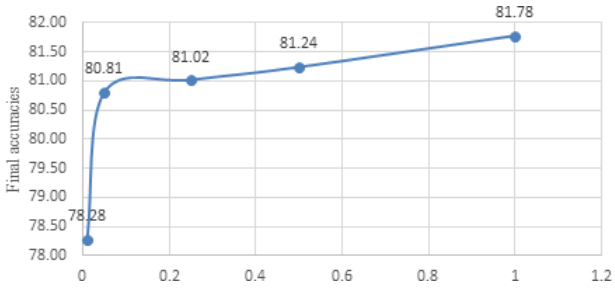

Fig. 5 Logistic regression, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

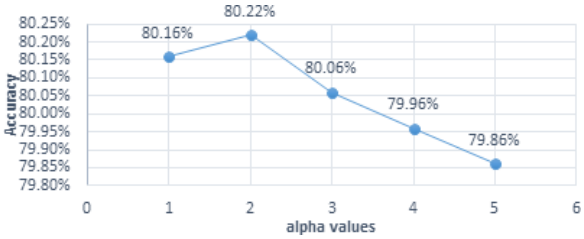

Fig. 6 Multinomial NB, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

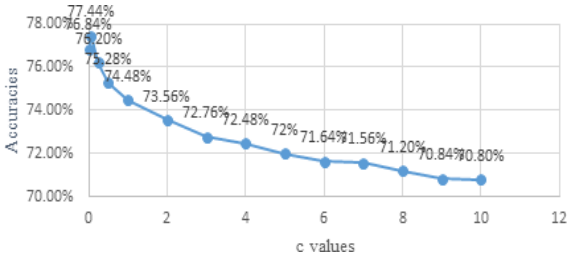

Fig. 7 Linear SVC, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

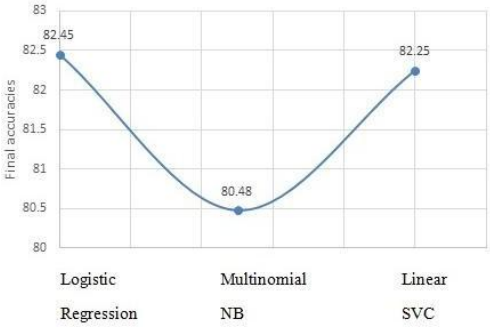

Fig. 8 Final Accuracy

TABLE 1: ACCURACIES WITH DIFFERENT NLP TECHNIQUES USING LOGISTIC REGRESSION WITH COUNT VECTORIZER

	Accuracies for different values of —c‖				
	0.01	0.05	0.25	0.5	1
Initial	78.87%	79.65%	79.96%	79.93%	79.84%
After removing stopwords	76.64%	77.45%	77.62%	77.68%	77.50%
With Stemming	78.93%	79.72%	79.87%	79.84%	79.86%
With Lemmatization	79.41%	79.92%	80.26%	80.31%	80.17%
With Stemming and ngram=(1,2)	81.09%	82.10%	82.45%	82.17%	82.05%
With Lemmatization and ngram=(1,2)	79.28%	81.72%	82.47%	82.18%	82.02%

TABLE 2: ACCURACIES WITH DIFFERENT NLP TECHNIQUES USING LOGISTIC REGRESSION WITH TF-IDF VECTORIZER

	Accuracies for different values of —c‖				
	0.01	0.05	0.25	0.5	1
Initial	76.02%	77.24%	77.90%	77.91%	78.02%
After removing stopwords	76.11%	77.41%	78.11%	78.11%	78.18%
With Stemming	78.39%	79.39%	79.86%	80.05%	80.09%
With Lemmatization	77.55%	78.88%	79.55%	79.76%	79.89%
With stemming and ngram=(1,2)	77.18%	79.62%	81.08%	81.62%	82.17%
With Lemmatization and ngram=(1,2)	78.28%	80.81%	81.02%	81.24%	81.78%

C. Accuracies found with count vectorizer on Movie Review data.

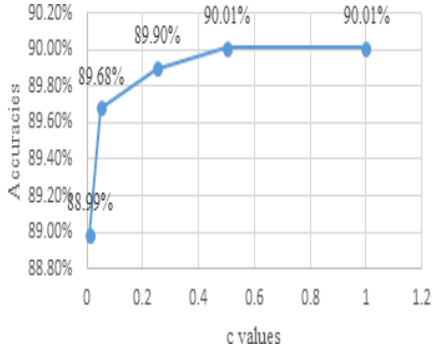

Fig. 9 Logistic regression, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values

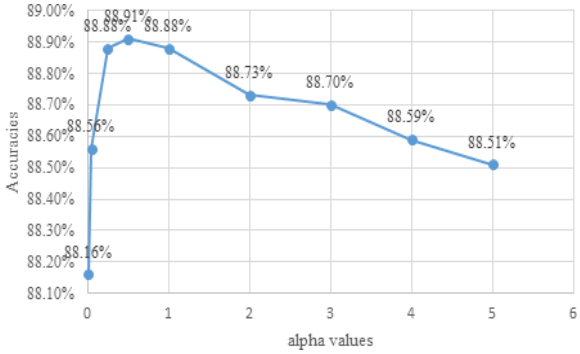

Fig. 10 Multinomial NB, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

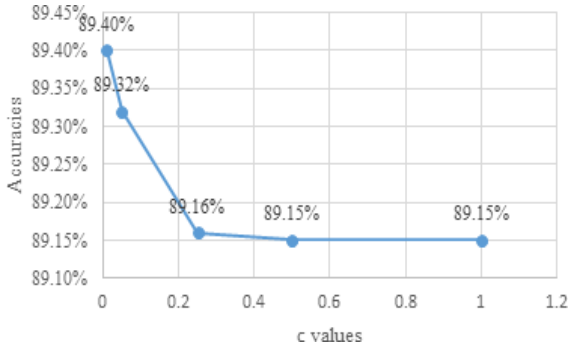

Fig. 11 Linear SVC, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

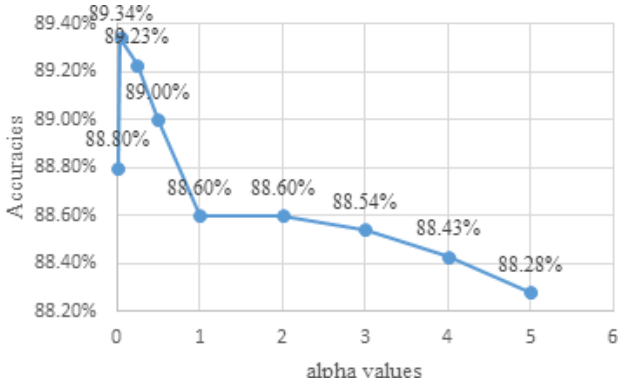

Fig. 14 Multinomial NB, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

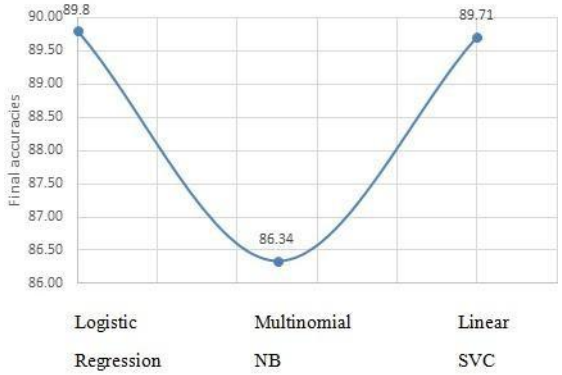

Fig. 12 Final Accuracy

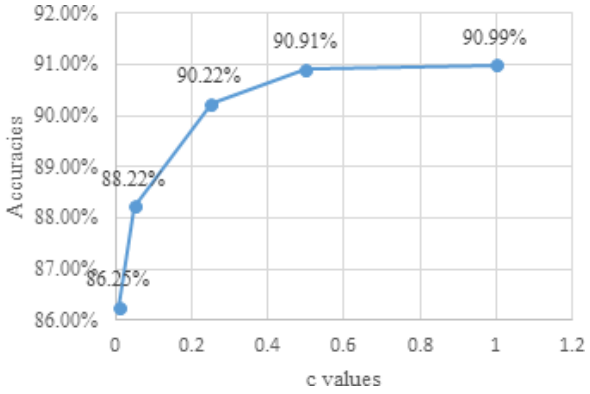

Fig. 15 Linear SVC, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

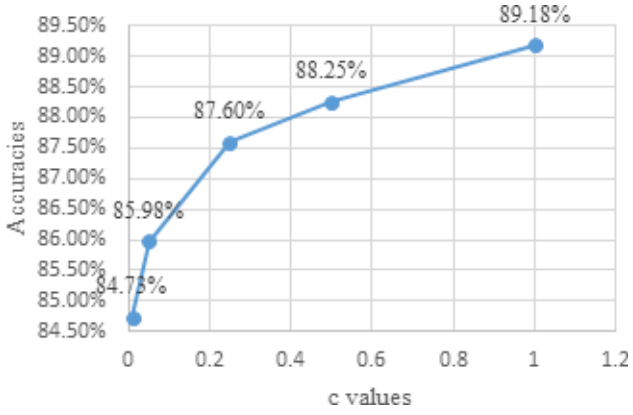

Fig.23 Logistic regression, for ngram= (1, 2) together with lemmatization, accuracies for different 'c' values.

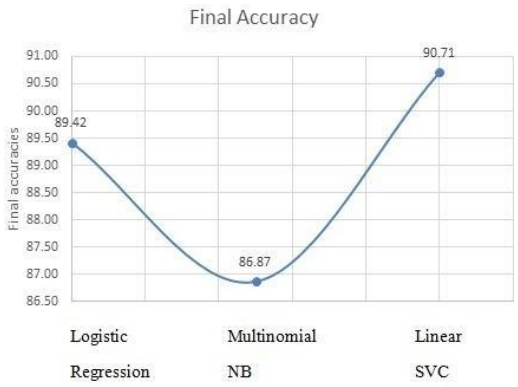

Fig.16 Final Accuracy

TABLE 3: ACCURACIES WITH DIFFERENT NLP TECHNIQUES USING LOGISTIC REGRESSION WITH COUNT VECTORIZER

	Accuracies for different values of —c‖				
	0.01	0.05	0.25	0.5	1
Initial	86.89%	88.81%	88.22%	88.06%	87.82%
After removing stopwords	87.44%	88.13%	87.68%	87.39%	87.17%
With Stemming	87.36%	88.06%	87.68%	87.6%	87.23%
With Lemmatization	88 %	88.52 %	88.51 %	88.32 %	88.04 %
With stemming and ngram=(1,2)	88.99%	89.64%	89.84%	89.71%	89.71%
With Lemmatization and ngram=(1,2)	88.99%	89.68%	89.90%	90.01%	90.01%

TABLE 4: ACCURACIES WITH DIFFERENT NLP TECHNIQUES USING LOGISTIC REGRESSION WITH TF-IDF VECTORIZER

	Accuracies for different values of —c‖				
	0.01	0.05	0.25	0.5	1
Initial	82.76%	85.2%	87.16%	87.96%	88.36%
After removing stopwords	84.97%	85.72%	87.24%	88.06%	88.65%
With Stemming	83.61%	85.53%	87.68%	88.19%	88.75%
With Lemmatization	83.88%	85.37%	87.61%	88.19%	88.86%
With stemming and ngram=(1,2)	85.42%	86.11%	87.79%	88.64%	89.36%
With Lemmatization and ngram=(1,2)	84.73%	85.98%	87.6%	88.25%	89.18%

VII. OBSERVATIONS

A. Observations on the accuracies across the two data sets

Although stopwords help increase the accuracy for some parameter values, this doesn‘t happen for all values with their being a decrease in the case of some. However removing stopwords does speed up the fitting and transformation process(understandable, since removing words means less features which in turn means less time needed for fitting and transformation). Across the two data sets, there is an increase in accuracy with stemming and lemmatization, in both data sets, although which one shows better results is arguable.

There‘s a marked increase in accuracy when ngram= (1, 2) is applied. From the graphs we can see that, the trend of accuracies for different parameter values remains roughly the same for each classifier (this statement remains true across the two datasets). For example Fig 1, Fig 5, Fig 9 and Fig 13 all look similar because they all implement logistic regression classifier.

B. *Observations on the similarities and differences between the two data sets.*

Between the 2 data sets, the accuracy is significantly more in the case of movie reviews compared to the tweets, even though, there are significantly more tweets. There could be many reasons for this, one being, the words are spelled correctly in reviews and hence there is one feature made for one word, whereas in the case of tweets, a word can be spelled in multiple (often wrong!)ways. Another reason could be that there are simply more words (i.e. features) there in each review than those are there in each tweet. This would help classify a review better.

The _final accuracy‘ graphs are roughly similar, showing that the behaviour of classifier remains similar throughout different vectorizers and datasets (for example, accuracies for Multinomial NB are always (marginally) lesser than that of the other two classifiers).

The optimal value of the parameter (be it c or alpha) doesn‘t remain roughly similar across the two data sets but varies when different vectorizers are used, with Tf-Idf Vectorizer having larger optimal parameter values than their Count Vectorizer counterparts.

VIII. ADDITIONAL NOTES

The accuracy was checked on the twitter data without any pre-processing or nlp operations with Textblob‘s inbuilt _sentiment‘ method. It was found to be about 61.17%.

To bridge the gap between twitter and movie review data accuracies, two different spell checking functions were applied to each word. Since it was estimated to take more than 2 days to complete the process, the experiment was performed with 10,000 tweets. There was no significant improvement to be found with either function. The two spell checking functions are, textblob‘s _correct‘ function and a function based on Peter Norvig‘s algorithm (https://norvig.com/spell-correct.html). The latter has a supposed accuracy of around 80% to 90%.

IX. CONCLUSION AND FUTURE SCOPE

More research can be made on how the parameter values of each classifier affect the accuracy. The reason behind why Tf-Idf vectorizers have a larger optimal parameter value than that of Count Vectorizer can thus be found out. Since there was no improvement with spelling correction, it can be inferred that the most likely reason for the higher accuracy is the larger number of words present in the reviews. Although it could also be the correct grammar (with the right syntax) that yields the movie reviews its higher accuracy. A lot of research is present in literature for detecting sentiment from the text. Still however, there is a huge scope of improvement of these existing sentiment analysis models. Existing sentiment analysis models can be improved further with more semantic and common sense knowledge.

X. FRONT END SNIPPETS

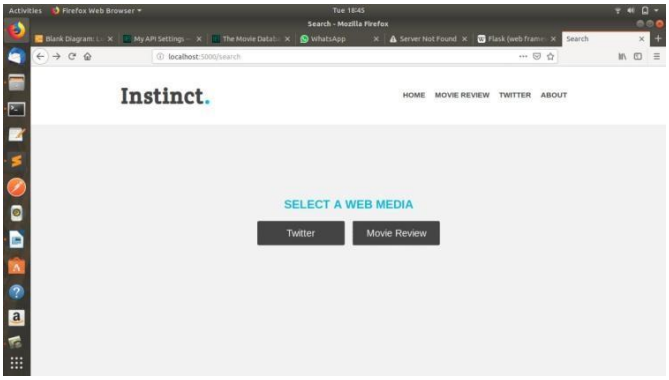

Fig. 17

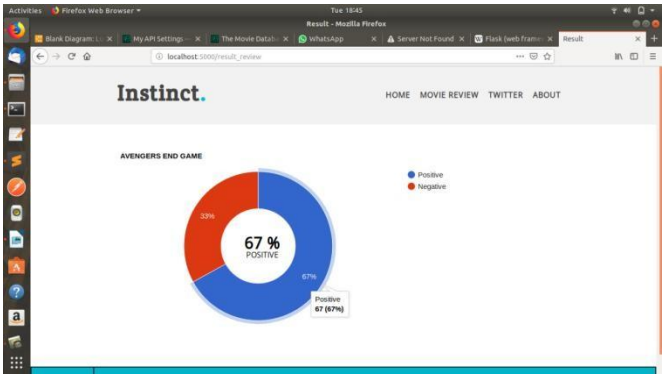

Fig. 18

REFERENCES

[1] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12), 2009.

[2] Willett, P. (2006). The Porter stemming algorithm: then and now. Program, 40(3), 219-223.

[3] Karimov, R., Samkova, M., Nikitina, S., & Akinin, A. (2016). Using a hybrid algorithm for lemmatization of a diachronic corpus. In CEUR workshop proceedings (Vol. 1886, pp. 1-8).

[4] Church, K. W. AT&T Bell Laboratories Murray Hill, NJ USA kwc@ research. att. com.

[5] Saif, H., Fernandez, M., He, Y., & Alani, H. (2014). On stopwords, filtering and data sparsity for sentiment analysis of twitter.