UNIVERSITY OF WINDSOR

Master of Applied Computing

Advanced Software Engineering (COMP 8117)

Report on

SENTIMENT ANALYSIS OF TWITTER DATA ON MOVIE REVIEW PREDICTION

submitted by

Group - 17

Harshitha Inti (105218827)

Prakash Kothapalli (105158404)

Sai Sireesha Tirumala Pudi (105167550)

Abstract: *Sentiment Analysis deals with opinion-based mining, where its main concern is to analyze the emotions and opinions from text basically handles the sentiment from the text. For a given source of data, Sentiment analysis processes the data and produces the sentiment of the person towards a particular target either any specific product or topic. In other terms, the Sentiment Analysis is the contextual mining of text to computationally determine the opinion of the person from a specific source which helps the business to interpret the review of their service/product. Social media has large amounts of unstructured data like blogs, tweets, posts, comments or status updates, etc. which is one of the sources for sentiment analysis to grasp the opinions of people. Twitter is one of the platforms, where sentiment analysis plays a crucial role in handling it as it has data in different slangs, words with typing errors and recurrent words. In our project, we have developed a website which gives more accurate sentiments from the tweets for the Hollywood movie reviews. We have used Tweepy(Twitter API), TextBlob library using Python which differentiates the positive, negative and neutral tweets.*

General overview:

In an era of big data, it is hard to extract the actual meaning of the unstructured data for business. The accumulation of larger sets of data through social networking, media, blogging websites, micro blogging, online communications forums and many more is making this more complex. The way of expressing thoughts has changed its explicit meaning in today's era of the internet. The companies often focuses on the voice of the customer, by extracting the meaningful insights in order to derive integrated customer intelligence decisions. In our project, we provided precise reviews based on the latest and live tweets for the movies of any language. Twitter plays a vital role in the current world that gives a platform for any user to express their viewpoints on any topic. We have used Twitter data, which was extracted using Tweepy API for our project to give the user more genuine reviews for the movies from the latest tweets.

The motivation of this project is to predict movie reviews and provide users the more actual review based on the critics using twitter data. One of the Text Analytics solutions that emphasize the data at a more detailed level by correlating various opinions is "Sentiment Analysis".

Generally, before watching a movie, people look for reviews either in websites like IMDB or ask for public talk. This information is not sufficient for the audience to analyze by themselves using naive methods. Sentiment Analysis concentrates on identifying and categorizing the emotions for each tweet. It can be divided into two broad categories, one is feature or aspect-based sentiment analysis and the other is objectivity based sentiment analysis. In our project scenario, the extraction of movie review tweets from the complete data set is feature-based sentiment analysis and the further digging focusing on the emotions like good, bad, etc is objectivity based sentiment analysis.

To analyze the sentiment from the twitter data, multiple symbolic and machine learning algorithms are used in general. But we have used Tweepy API for extracting Tweets and then used TextBlob, a Python library that parses the input data using sentiment analysis and outputs a sentiment of classified data based on opinions. By taking this data we have trained further to split into positive, negative and neutral tweets in a user-friendly environment using HTML and CSS.

Project Details and Methodology

Definitions HTML: Hyper Text Markup Language (HTML), create documents on the World Wide Web. Using different tags and attributes, HTML defines the layout and structure of a web page.

Python: It is one of the high level programming languages focusing on the core functionality long with the regular programming tasks. Python is used for developing desktop GUI applications, websites and web applications.

Specifications

1. Functional Specification

- RAM: 8 GB
- Processor: intel i5
- Compatibility: Desktop or Laptop
- Operating System: macOS (Linux)

2. Non Functional Specification

Twitter developer tool "Tweepy" API extracts movie-related tweets**.**

Text processing tool "TextBlob" categorizes extracted tweets into positive, negative and neutral tweets.

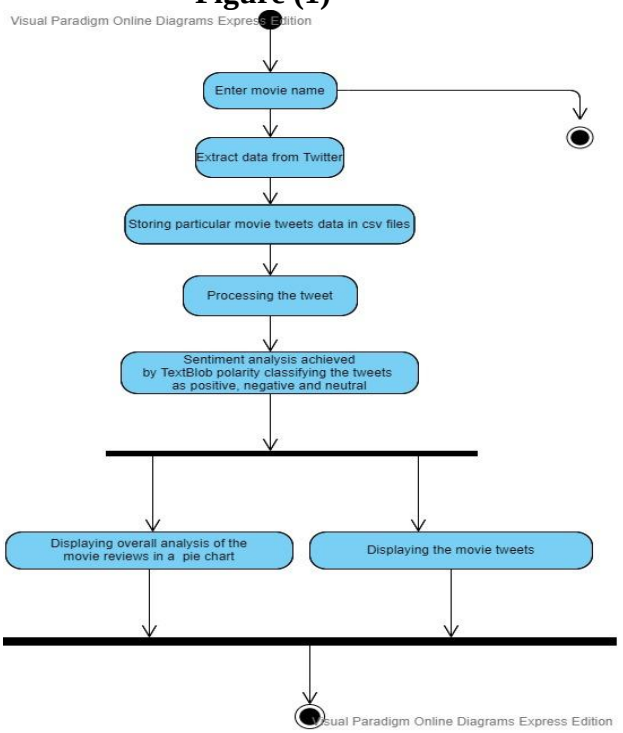
Architecture

Platform

Framework: **Django** - is an open-source Python-based web framework. Django can be defined as a high-level Python Web framework that encourages quick development, secure and pragmatic design.

Design : Activity Diagram

Figure (1)



Methodology:

The tweets are analyzed using sentiment analysis to predict the review of the movies Our methodology consists of four steps:

A. Data Collection: The Pre-requisite is the creation of Twitter API(Tweepy) for collecting the data (Movie tweets)

Data fields stored for each tweet are listed below

i. Tweet Id
ii. Username of person who tweeted
iii. Tweet text
iv. Geo Tagging

The data that has been collected for each movie is stored in a CSV file.

B. Data Pre-processing: The data has been filtered to get tweets of the movie through regular expression(RE) pattern matching.

In the first step is to filter the data as per our required target(movie reviews) and the second step is to translate the data format into the input format required for sentiment analysis.

C. **Sentiment analysis:** TextBlob library has been used for sentiment analysis polarity to analyze and classify the data into positive, negative and neutral tweets.

Experimental setup

i) Implementation Details

Frontend Implementation:

Created a responsive webpage using HTML, CSS that can automatically enlarge, resize according to the platform as to make it look good among all devices. Cascading Style Sheets (**CSS**) is a stylesheet language which is used to describe the presentation of a document written in HTML or XML. In CSS, we used background-color, border-color, color, label, padding, margin, height, width, textarea, font-size, font-weight, font-family, line-height, hover for styling our webpages.

Head tag is used to define the heading portion of the document which contains information regarding the document. This head tag contains other related elements such as title, meta, scripts, links.

Used **title tag** for declaring the title of Html document. This title is usually displayed in the browsers title bar.

Responsive webpage was made using the **<meta> element** with name "viewport". This viewport gives the browser instructions on how to control dimensions and scaling of the page.

Syntax:

<meta name="viewport"

Content="width=device-width, initial-scale=1.0">

We used link tag to link between the document and external resource. i.e., link tag is used to link external style sheets. The href used in the link tag specifies the location of linked document.

Used html **charset** attribute to display html page correctly. This charset used in the page must be known to the web browser and this is specified in the meta tag.

Body tag element contains all the contents of html document such as text, hyperlinks, images, videos, etc. We used **Body tag** to define the structure of the webpage.

Using the **id** attribute, created unique ids for the three responsive webpages. This id value is referenced in CSS and javascript inorder to perform certain tasks for the elements. Created unique id i.e., "banner" for background video play. This is performed using a **video** tag which specifies a standard way to embed video in the webpage. Created source element is used for the selection of videos to display.

Used **class** attribute in order to define equal styles for elements within the same class name i.e., all the elements within the same class attribute will get the same style. Html **section tag** is a standalone section. It doesn't have a specific semantic element to represent it. This tag defines section in a document (footer, header) or any other sections of the document. **Anchor tag** is used to refer another webpage link. In the below syntax, href is referenced to tweets.html page using anchor tag.

Href attribute: it is the most important attribute of anchor element which indicates the we blink destinations. We used block

quote tag to specify a section that is quoted from another source. Used **span** element inorder to group other elements for styling purpose with the help of class or id attribute. This tag is similar to div tag but the difference is “div is block-level element where as span is an inline element). We used the **footer tag** that defines footer for document. Our footer element contains contact information and team. In order to run javascript on several pages in a website, an external javascript file is to be created. We referred the script file using src attribute in **script tag.**

Backend Implementation:

Pre-requisite:

As a pre-requisite, got approval for extracting data(tweets) from twitter and created Twitter API in **developer.twitter.com**

Editor used : Vi, Python
Languages used : Python

Installed tweepy module using pip.
Syntax: pip install tweepy

Program to extract tweets:
Imported tweepy module and created four variables for consumer and secret keys

```
consumer_key = "**********"
consumer_secret = "**********"
access_token = "**********”
access_token_secret = "**********"
```

created authentication for accessing Twitter

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
#initialize Tweepy API
api = tweepy.API(auth,wait_on_rate_limit=True)
```

Used Cursor function which is predefined in tweepy in a for loop to extract tweets belonging to the hashtag phrase.

```
for tweet in tweepy.Cursor(api.search, q=hashtag_phrase+' -filter:retweets', \
                           lang="en", tweet_mode='extended').items(1000):
```

Imported csv python module and opened a csv file with write permissions. Using for loop, each tweet is written inside the csv file tweets.csv with tweet information containing tweet, user name, geotagging.

Figure (2.1)

Extracted tweets data for particular hashtag value and stored them in tweets.csv.

Sentiment Analysis:
We implemented sentiment analysis using TextBlob Python library for text data processing. TextBlob provides an API to perform Natural Language Processing tasks like classifying, extracting, analyzing on sentiments, etc.

Imported TextBlob module in views.py(connectivity file) to perform sentiment analysis on the tweets.

The next step is to open the csv file which we created earlier and each tweet is read and calculated the **polarity** of the tweet i.e., positive, negative or zero.
Stored no of positives, negatives, and zeroes in 3 variables and each positive tweet, negative tweet, and neutral tweets are stored in lists.

Figure (2.2)

```
positive_tweets=[]
negative_tweets=[]
neutral_tweets=[]
for line in list1:
    analysis=TextBlob(line)
    if(analysis.sentiment.polarity>0):
        positive+=1
        positive_tweets.append(line)
    elif(analysis.sentiment.polarity<0):
        negative+=1
        negative_tweets.append(line)
    elif(analysis.sentiment.polarity==0):
        neutral+=1
        neutral_tweets.append(line)
```

Connectivity:

For connecting backend and frontend, we used **Django framework**.

Django - is an open source Python based web framework.
Django can be defined as a high-level Python Web framework that encourages quick development, secure, and pragmatic design.

As the first step, we installed Django in local server using the below command

pip install Django

Next step is to create a directory structure for our web application.

Django-admin startproject mywebsite

We have created our app using manage.py with the below command.

python manage.py startapp filmatory

Migrate frontend files which we created to templates directory.

Created url patterns to provide connectivity links to template files which are index.html, tweets.html and oa.html

Figure (2.3)

```
# mywebsite/Filmatory/urls.py

from django.conf.urls import url, include
from Filmatory import views

urlpatterns = [
    url(r'^$', views.HomePageView,name="home"), # Notice the URL has been named
    url(r'^tweets$', views.tweetsPageView,name="tweets"),
    url(r'^oa$', views.oaPageView,name="oa"),



]
```

To run Html files in local server, we have written views for each Html file in views.py.
We imported the render module.

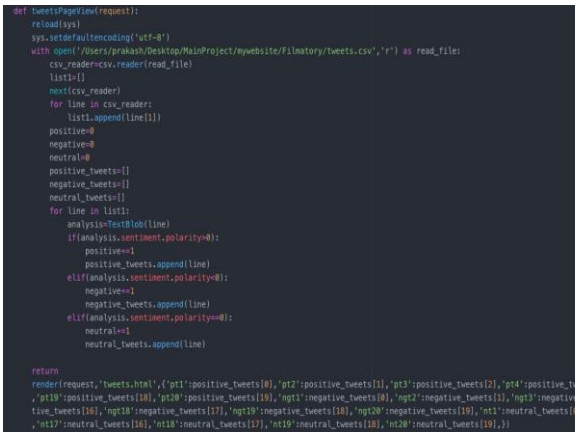
Figure (2.4)

```
def HomePageView(request):

    return render(request,'index.html')
```

Backend is also written in views.py files for sending sentiment analysis data to frontend files. Dictionary is created in request function to send information.

Figure (2.5)



Created a directory with name “static” and migrated css and js files which we created earlier.

Each dictionary value is referenced in Html files in the below format.

Figure (2.6)

```
{% block content %}{{content}}{% endblock content %};
{% block content1 %}{{content1}}{% endblock content1 %};
{% block content2 %}{{content2}}{% endblock content2 %};
```

Anchor links are referenced with below format.

Figure (2.7)

```
<li><a href="{% url 'home' %}">Home</a></li>
<li><a href="{%url 'tweets' %}">Tweets</a></li>
<li><a href="{% url 'oa' %}">Overall Analysis</a></li>
```

CSS and java files are referenced in html files in below format.

Figure (2.8)

```
<script src="{% static 'js/jquery.min.js' %}"></script>
<script src="{% static 'js/browser.min.js' %}"></script>
<script src="{% static 'js/breakpoints.min.js' %}"></script>
<script src="{% static 'js/util.js' %}"></script>
<script src="{% static 'js/main.js' %}"></script>
```

Used Google APIS for generating charts with the backend data

Figure (2.9)

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {

        var list1= {% block content %}{{content}}{% endblock content %};
        var list2= {% block content1 %}{{content1}}{% endblock content1 %};
        var list3={% block content2 %}{{content2}}{% endblock content2 %};
        var data = google.visualization.arrayToDataTable([
          ['Review', 'No of Tweets'],
          ['Negative Reviews', list2],
          ['Neutral Reviews', list3],
          ['Positive Reviews',list1],
        ]);


        var options = {
          title: 'Twitter Analysis'
        };

        var chart = new google.visualization.PieChart(document.getElementById('piechart'));

        chart.draw(data, options);
      }
```

Screenshots:

Figure (3)

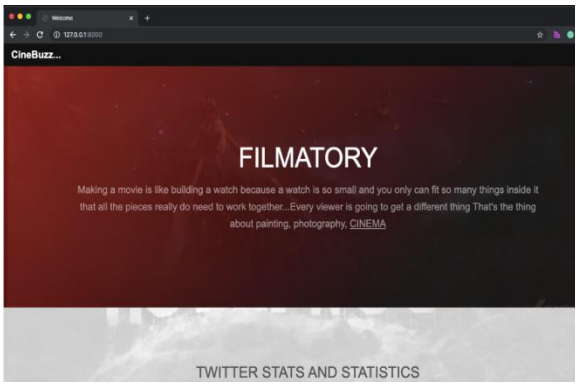


Figure (3.1)

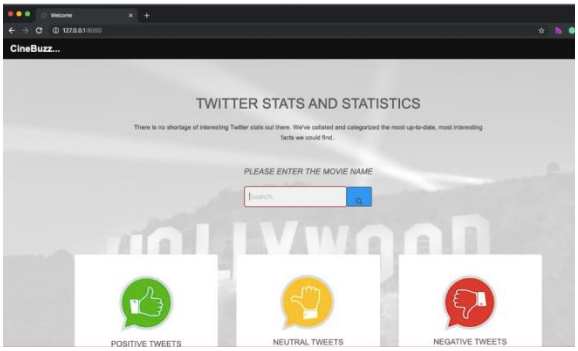


Figure (3.2)

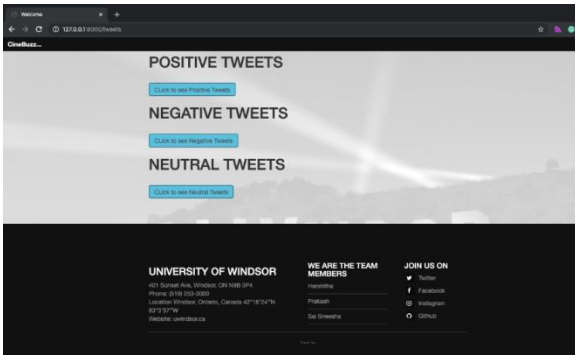


Figure (3.3)

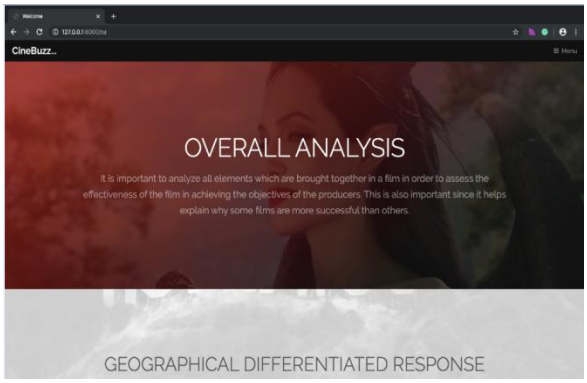


Figure (3.4)

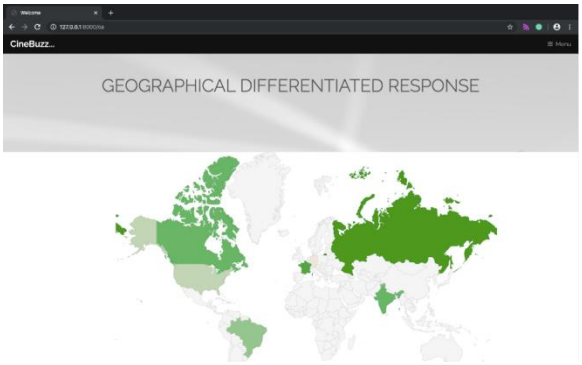
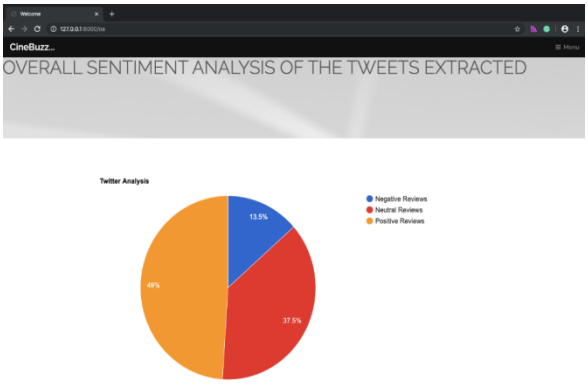


Figure (3.5)



ii) Testing

The types of testing we have used in our project are listed below:

Reliability Testing: It is a measure of reliability for failure-free testing over a period of time. In our project, we faced an issue while we are loading videos and images to the website, then we have changed it to the correct format and tested it. We haven't faced any such failure again while testing.

Performance Testing: This testing is used to checked for system stability and responsiveness to determine the performance of a program or software. The response time for displaying an overall analysis of the tweets for any movie is minimal.

iii) Discussion of your findings, and challenges

One of the challenges is the extraction of Twitter data which required access keys. We have then found the information related to access keys, we finally got the access keys and customer keys by the following those procedures

The connection between the front end and back end is another challenge. We achieved by using the Django framework.

Conclusion

Our website gives a genuine overall review of the movie based on the latest tweets along with an individual rating of positive, negative and neutral tweets. Users can make their decision to watch a movie easily instead of using naive techniques. It also provides the tweets on the homepage along with the twitter user details. Sentiment Analysis is efficient to implement Twitter data sentiment analysis. Usually, it is difficult to predict the sentiment for every single movie tweet as it has repeated words, words using slang, hashtags, misspelled words, etc. which is an unstructured data. Using TextBlob Python library, the sentiment analysis easily analyzes the emotion of each tweet and classifies the positive, neutral and negative tweets.

Future work:

We are planning on our future work to enhance more efficient feature on our website.

Our future work includes

I. Providing the list of movies in a specific region along with the theatre details and location of the theatre where the movie is playing.
II. A chatbot feature called "Tweet bot" to make more user-friendly.
III. Enlarging the targets of sentiment - To analyze the sentiments of all current trending topics in every location along with the movies.

References

1. Amolik, N. Jivane, M. Bhandari, and V. M, “Twitter Sentiment Analysis of Movie Reviews using Machine Learning Techniques.,” Twitter Sentiment Analysis of Movie Reviews using Machine Learning Techniques.
2. H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, “A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle,” A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle, Jul. 2012.

3. P. Savaram, S. Fatima, and S. Bandu, “Analysis of sentiment on newspaper quotations: A preliminary experiment,” Analysis of sentiment on newspaper quotations: A preliminary experiment, Jul. 2013.
4. S. Gupta, “Sentiment Analysis: Concept, Analysis and Applications,” Sentiment Analysis: Concept, Analysis and Applications, Jan. 2018.
5. S. Hu and T. Dai, “Online Map Application Development Using Google Maps API, SQL Database, and ASP.NET.
6. Z. Kobti, “ Introduction to Software Testing.”