Sentiment Analysis of Product Reviews on Social Media

Velam Thanu and David Yoon

1 Introduction

In the world of e-commerce, product reviews are of great importance to both buyers and sellers. Most of the products are bought online, and reviews are one of the most important and easy ways for a customer to gain trust in the product. From the sellers' point of view, they need to know the performance of their products. The feedback of their customers is crucial to improve quality and service.

There are many ways to gather product reviews (like looking at the review section on the e-commerce website where the product is sold or having a portal for customer issues). But another important platform wherein product reviews pour in is social media. Social media is increasingly used by humans to express their feelings and opinions in the form of short text messages to reach a large audience. Hence, it is important for both the buyer and seller to find a way to analyze people's comments about products on a social media platform.

As there is an immense amount of data available on social media, it is impossible to analyze them manually in real time. We need tools that can do this for us and give an analyzed output. In this project, we have developed an application which gives the sentiment of tweets about the product.

V. Thanu · D. Yoon (✉)
CIS Department, University of Michigan – Dearborn, Dearborn, MI, USA
e-mail: dhyoon@umich.edu

H. R. Arabnia et al. (eds.), *Advances in Software Engineering, Education, and e-Learning*, Transactions on Computational Science and Computational Intelligence, https://doi.org/10.1007/978-3-030-70873-3_64

1.1 Purpose

The main purpose of this application is to give a report on the number of positive, negative, and neutral tweets on Twitter posted by humans around the world for any product. It is very simple to use. The user must just enter the name of the product and submit it, and this application gives a graph as an output which shows the percentages of positive, negative, and neutral tweets. It also shows a few sample tweets under the positive and negative categories.

Hence, the user can get a good idea of the latest sentiments of people on a product which they tweeted without spending any time to go through the tweets. In fact, the user need not even have a Twitter account.

1.2 Motivation

There is an immense amount of valuable data available on social media in the form of product reviews. It is important to analyze them for the benefit of both the buyer and seller. For this, we need good tools which can analyze data and give results about the sentiment of users for different products.

The attempt made in this project is to create an application that analyzes the tweets of a product on Twitter and gives the sentiment analysis result. This is highly automated and needs very little effort from the user. From just a click, the user can get valuable data which can impact their buying or selling decisions having known that social media is the platform that is used immensely to express oneself in recent times.

1.3 Brief Description

The application is basically a tool that gives the user the sentiment analysis of products.

There is an immense amount of valuable data available on social media in the form of product reviews. It is important to analyze them for the benefit of both the buyer and seller. For this, we need good tools which can analyze data and give results about the sentiment of users for different products.

The attempt made in this project is to create an application that analyzes the tweets of a product on Twitter and gives the sentiment analysis result. This is highly automated and needs very little effort from the user. From just a click, the user can get valuable data which can impact their buying or selling decisions having known that social media is the platform that is used immensely to express oneself in recent times.

2 Technical Specification

2.1 System Architecture Diagram

Below is the system architecture design diagram of the application. From the front-end webpage, the user submits the keyword/product name. The Flask server receives this keyword and passes it to the sentiment analysis server. The sentiment analysis server is the Python script that is the heart of this application.

It uses the Tweepy library that enables this Python script to communicate with the Twitter API to authenticate and get the tweets based on search keywords. It then uses the TextBlob library to perform sentiment analysis on the tweets. The sentiment analysis results are now passed to the Flask server which displays them on the front-end web page (Fig. 1).

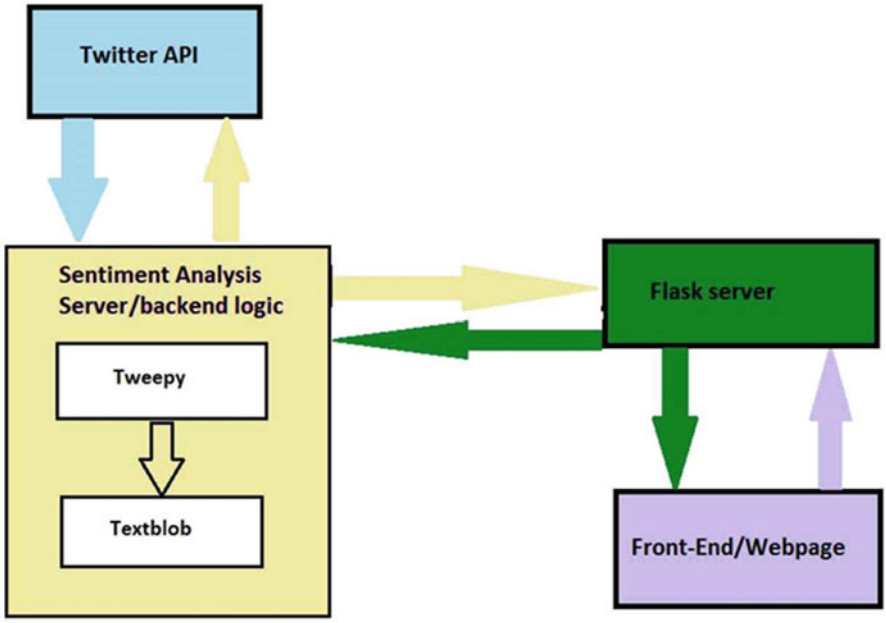

Fig. 1 System Architecture

2.2 Description Libraries and Web Framework Used

Twitter

Twitter is an online news and social networking platform where people communicate and express their feelings in short messages called *tweets*. Twitter restricts every tweet to *280 characters*, which keeps the tweets short and scan-friendly. Users can get the crux of the tweet by just giving a glance which makes Twitter very popular in today's attention-deficit world (Fig. 2).

There are about 600 million daily searches and 10 billion tweets on Twitter. Twitter and its third-party apps offer a way for the users to stay in the loop on what people are saying about a company or brand, directly or indirectly. Twitter helps in putting a face to a company and functions as a customer service platform to many companies.

Due to its openness in sharing and the immense amount of data, Twitter is a prime example of social media in which users can mine interesting patterns and build real-world applications. Sentiment analysis of tweets gives real-time information for disaster relief, using Twitter analytics for improving businesses.

Twitter APIs

An application program interface (API) is a set of routines, protocols, and tools for building software applications. An API makes it easier to develop a program by providing all the building blocks which a programmer puts together to develop a new application altogether.

Twitter allows access to parts of its service via APIs to allow people to build software that integrates with Twitter, like a solution that helps a company.

One of the APIs is the standard search API which is used in this project that returns a collection of relevant tweets matching a specified query.

I have used the following two parameters of this API:

Search(q, count) where

Fig. 2 Twitter

Fig. 3 Tweepy

q *is A UTF-8, URL-encoded search query of 500 characters maximum, including operators.*

count *is the number of tweets to return per page, up to a maximum of 100.*

2.3 Tweepy

Tweepy is a Python library for accessing the Twitter API (Fig. 3).

It is open-sourced and hosted on GitHub and enables Python to communicate with the Twitter platform and use its API.

TextBlob

Sentiment analysis employs natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to the voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

In recent years, sentiment analysis has become a hot-trend topic of scientific and market research in the field of natural language processing (NLP) and machine learning. In this project, the sentiments of tweets are analyzed as either positive, negative, or neutral.

For this, we have used TextBlob, which is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun-phrase extraction, *sentiment analysis*, classification, translation, and more.

We are more concerned about the sentiment analysis functionality of the API. The sentiment property returns a named tuple of the form Sentiment (polarity, subjectivity). The polarity score is afloat within the range [−1.0, 1.0].

Looking into the source code of *textblob.en.sentiments* (which is the sentiment analysis module of TextBlob), we get to know that it has a training set with preclassified movie reviews. When we give a new text for analysis, it uses the *Naïve Bayes classifier* to classify the new text's polarity as positive or negative (Fig. 4).

```
class NaiveBayesAnalyzer(BaseSentimentAnalyzer):                    [docs]
    """Naive Bayes analyzer that is trained on a dataset of movie reviews.
    Returns results as a named tuple of the form:
    ``Sentiment(classification, p_pos, p_neg)``

    :param callable feature_extractor: Function that returns a dictionary of
        features, given a list of words.
    """

    kind = DISCRETE
    #: Return type declaration
    RETURN_TYPE = namedtuple('Sentiment', ['classification', 'p_pos', 'p_neg'])

    def __init__(self, feature_extractor=_default_feature_extractor):
        super(NaiveBayesAnalyzer, self).__init__()
        self._classifier = None
        self.feature_extractor = feature_extractor

    @requires_nltk_corpus                                           [docs]
    def train(self):
        """Train the Naive Bayes classifier on the movie review corpus."""
        super(NaiveBayesAnalyzer, self).train()
        neg_ids = nltk.corpus.movie_reviews.fileids('neg')
        pos_ids = nltk.corpus.movie_reviews.fileids('pos')
        neg_feats = [(self.feature_extractor(
            nltk.corpus.movie_reviews.words(fileids=[f])), 'neg') for f in neg_ids
        pos_feats = [(self.feature_extractor(
            nltk.corpus.movie_reviews.words(fileids=[f])), 'pos') for f in pos_ids
        train_data = neg_feats + pos_feats
        self._classifier = nltk.classify.NaiveBayesClassifier.train(train_data)


    def analyze(self, text):                                        [docs]
        """Return the sentiment as a named tuple of the form:
        ``Sentiment(classification, p_pos, p_neg)``
        """
        # Lazily train the classifier
        super(NaiveBayesAnalyzer, self).analyze(text)
        tokens = word_tokenize(text, include_punc=False)
        filtered = (t.lower() for t in tokens if len(t) >= 3)
        feats = self.feature_extractor(filtered)
        prob_dist = self._classifier.prob_classify(feats)
        return self.RETURN_TYPE(
            classification=prob_dist.max(),
            p_pos=prob_dist.prob('pos'),
            p_neg=prob_dist.prob("neg")
        )
```

Fig. 4 TextBlob

Naïve Bayes Classifier

It is a classification technique based on Bayes' theorem with an assumption of independence among predictors.

Bayes' theorem is stated as:

$$P\,(y|X) = \frac{P\,(X|y)\,P(y)}{P(X)}$$

where ***X*** is a problem vector

$$X = (x_1, x_2, x_3 \ldots .., x_n)$$

P(y|X) is the probability of hypothesis y given the vector x. This is called the posterior probability.

P(X|y) is the probability of vector X given that the hypothesis y was true.

P(y) is the probability of hypothesis y being true (regardless of X). This is called the prior probability of y.

P(X) is the probability of the vector (regardless of the y).

The naïve assumption of independence among predictors is now applied, and we can come up with the below Naïve Bayes theorem.

$$P\,(\,y|\,x_1, \ldots, x_n)\;\;\alpha\;\;P(y)\prod_{i=1}^{n} P\,(x_i|\,y)$$

Due to good results in multiclass problems and independence rules, Naïve Bayes classifiers are mostly used in text classification and have a higher success rate as compared to other algorithms. As a result, it is widely used in social media sentiment analysis, to identify positive and negative customer sentiments just like it's used in TextBlob.

- In TextBlob, they have first trained the analyzer using a movie review dataset. An overview of how they could have trained is explained below.
- The reviews are preprocessed as punctuations, special characters, etc. and are not needed for sentiment analysis.
- The next step is creating a list of all the words we have in the training set, breaking it into word features. Word features are basically a list of distinct words, each of which has its frequency (number of occurrences in the set) as a key.
- The next step is to go through all the words in the training set, comparing every word against the review at hand and giving a label of 1 if it is present in the review or 0 if it is not present in the review. Also, we can give a positive or negative label for each review. We thus create a matrix of values which are.
- 0 or 1 and positive or negative, respectively, which is called the feature
- Vector.

Fig. 5 Flask

- After creating the feature vector, the NaiveBayesClassifier.train() function in python trains the analyzer, and the probability model is created.
- Next, if we pass a review from testing data, it will analyze the review and give us an output of whether it has a positive, negative, or neutral sentiment.

Flask

Flask is a *web application framework* written in Python.

Web application framework or simply Web framework represents a collection of libraries and modules that enable a web application developer to write applications without having to bother about low-level details such as protocols and thread management (Fig. 5).

Flask is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and *Jinja2 template* engine. Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications. Werkzeug is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinga2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages. By convention, templates are stored in subdirectories within the application's Python source tree, with the name templates.

In this project, I have used Flask with templates to create a dynamic web page that can take user inputs, process them using the backend python code (sentiment analyzer), and then publish the output from it dynamically on the webpage.

3 Conclusion

To conclude, in this project, the sentiment analysis of product tweets on Twitter was performed successfully, and the results were displayed to the user. The sentiment

analysis was performed using libraries available in python. Also, APIs provided by Twitter were used to communicated and fetch data.

In this way, the immense amount of data available on social media platforms like Twitter can be put to constructive use.