CSCI 5408 ASSIGNMENT 2

By: Bhargav Dalal (B00785773) & Hardik Galiawala (B00777450)

MAY 23, 2018

DALHOUSIE UNIVERSITY

Table of Contents

1. Objective:

The main purpose of this assignment is to perform the sentimental analysis by understanding the extracting, transforming and loading the data process into NoSQL database. Furthermore, this processed data is analyzed using basic machine learning algorithm.

2. Task Description:

The task was to understand the process of extracting, transforming and processing the data which is extracted from different web resources such as website, online database, and services, etc. In this assignment, we are extracting the data from a social media platform (twitter). The data is extracted with the help of python language and later it was stored in CSV format.

The extracted data is then transformed into the meaningful data by defining the data type as well as handling the missing data. While transformation, the data type of the fields is also set and hence, all the data in the future will be converted into defined data type and store it in a CSV file. This CSV file is later loaded into the NoSQL database using a python script.

The python script will load the data in NoSQL database using Elastic Search API. Furthermore, machine learning techniques are used on such for sentimental analysis. The machine learning algorithm was trained based on lexicon provided to it. The lexicon is obtained from the website(link). Thus, the NoSQL database can be referred here as Data Warehouse as the data was first extracted, then transformed and finally loaded in it.

We have installed python 2 and elastic search on the server instance on Microsoft Azure. Once the environment for python is set up, we install libraries such as tweepy and elasticsearch-dsl to load the data in NOSQL database. Finally, model based on lexical analysis was used to analyze the data.

3. Twitter Tweet Extraction:

We have worked on python to extract, transform and load the data. “tweepy” is the package used to communicate with Twitter API. First, we install the tweepy package in python. Further, from the twitter, noted the require consumer & access key as well as consumer & access secret to establish and authorize the connection.

```
# Establish connection

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth)
```

Fig 3.1 – Establishing Connection with Twitter

Once the connection is established, the tweets are extracted by cleaning/filtering the tweets and stored in a file named as “tweetsUnencoded.csv”. The content stored in this CSV file is id, username, posted time and the tweet. The tweet was stored after applying the filter such as fetching the tweets, removing all the emojis, removing links, removing hashtags etc. The functions for obtaining tweets, removing emojis and removing hashtag and creating new CSV file is as follows:

```
def get_tweets_lab(query):
    api = tweepy.API(auth)
    try:
        tweet = api.search(query, count = 30)
    except tweepy.error.TweepError as e:
        tweet = json.loads(e.response.text)

    return tweet
```

Fig 3.2 – Query function to fetch tweets

```
# Remove emojis
# Reference: https://stackoverflow.com/questions/33404752/removing-emojis-from-a-string-in-python#33417311
def discard_emoji(tweet_text):

    emojis_removed = re.compile(u"[^\U00000000-\U0000d7ff\U0000e000-\U0000ffff]", flags=re.UNICODE)
    return emojis_removed.sub(u'', unicode(tweet_text, 'utf-8'))
```

Fig 3.3 – Removing Emojis Function

```
with open('tweetsUnencoded.csv', 'wb') as outfile:

    writer = csv.writer(outfile)
    #writer.writerow(['id', 'user', 'created_at', 'text'])
    for query in queries:
        t = get_tweets_lab(query)
        for tweet in t:


            hashtags = re.sub(r'#\S+', '', tweet.text.encode('utf-8'))
            tags = re.sub(r'@\S+', '', hashtags)
            url = re.sub(r'http\S+', '', tags)
            retweet = re.sub(r'RT ', '', url)
            retweet_emoji = discard_emoji(retweet)
            writer.writerow([tweet.id_str, tweet.user.screen_name,
                             tweet.created_at, retweet_emoji.encode('ascii', errors = 'ignore')])
```

Fig 3.4 – Fetching username, post time, and filtering the tweets before storing in csv file

So, the new CSV file will contain only the tweets along with the username, created time, and id of the tweet.

4. Sentimental Analysis Algorithm:

The sentimental analysis is stored in the csv file named as “sentimentAnalysis.csv”. We have used lexical_easy.csv file obtained from the website, to analysis the sentiments. The fields of the CSV file are the words, and its respective score. The words are divided into three categories: Positive, Neutral and Negative. The respective scores for each category of words are: 1, 0, -1.

The lexical_easy.csv is used as a training set for the model and the tweets fetched were used as the test data.

The noticeable analysis of the tweets was the accuracy for identifying whether the tweet was positive, neutral or negative was not up-to the mark. The reason behind it is, the tweets are categorized based on the score of all the type of words in the tweet. But, this accuracy can be improved. Instead of categorizing the tweet based on the score of the words, if we provide the probability of the word being positive, neutral or negative.

5. Loading Data into Elastic Search DB:

Before diving into loading the data in the elastic search database, let’s review what we did until now. We started with establishing the connection with the twitter. After authorization, the tweets were fetched from the twitter based on the query. These tweets were stored in a separate CSV file which was processed and analyzed using lexicons. Finally, the analyzed data is stored in another CSV file.

This CSV file is inserted into the Elastic Search (NoSQL) Database. The elastic search is installed on the Microsoft Azure server instance. The data was inserted from CSV file named as “sentimentAnalysis.csv” which was obtained after the analysis on the tweets using python. “elasticsearch_dsl” package is required to install to communicate with the elastic search via python.

The connection was established initially with the server using DNS name on which elastic search is installed. The code snippet for loading the data into Elastic Search Database is as follows:

```
class Tweet(DocType):
    #user = Text(analyzer='snowball', fields={'raw': Keyword()})
    tweet = Text(analyzer='snowball')
    score = Text()
    sentiment = Text(analyzer='snowball')
    #hashtags = Keyword()
    created_at = Date()

    class Meta:
        index = 'tweet_sentiments'

        def save(self, ** kwargs):
            '''
            self.lines = len(self.tweet.split())
            self.hashtags = [tag for tag in
                             self.tweet.split()
                             if tag.startswith('#')]
            '''
            return super(Tweet, self).save(** kwargs)
with open('sentimentAnalysis.csv', 'rb') as csvfile:

    id_number = 0
    sentiment = csv.reader(csvfile, delimiter=',')
    for i in sentiment:

        id_number = id_number + 1
        Tweet.init()
        tweet = Tweet(meta={'id': id_number})
        tweet.tweet = i[0]
        tweet.sentiment = i[1]
        tweet.score = i[2]
        tweet.created_at = datetime.now()
        tweet.save()
```

Fig 5.1 – Loading the data in Elastic Search Database

The important thing to be noted for querying the elastic search database is, the index is “tweet_sentiments” and “doc” is the name of the document.

The snapshot of the output based on the query on the entered data in the database can be seen below:

Query : POST http://<ip address>:9200/tweet_sentiments/doc/_search

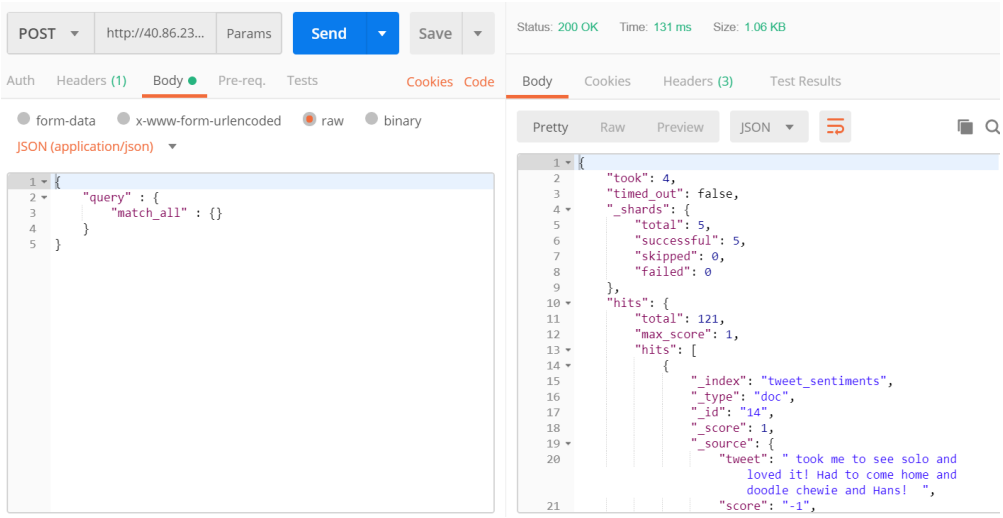

Fig 5.2 – Output based on the query of the Elastic Search Database

6. ETL as Batch Process:

The shell script was developed on the server to automate the process of the extracting, transforming and loading (ETL) the data.

```
#!/bin/sh

echo 'Extrating and cleaning the data'

python EC.py

echo 'Done with data extraction and cleaning'

python sentiment_analysis.py

echo 'Processing the data Done'

python load_elasticsearch.py

echo 'Loaded the data in the Elastic Search'
~
~
~
~
~
```

Fig 6.1 – Script for automation of ETL process

Here, EC.py is a script which was explained in section 3. It is responsible for extracting and cleaning the data. It will also create another CSV file. This CSV file will become the input for the sentiment_analysis.py (discussed in Section 4) where data is analyzed, and another CSV file named "sentimentAnalysis.csv" is created. This file will become the input to the load_elasticsearch.py (discussed in Section 5) and it will finally load the data in the elastic search database.

7. Code Submission:

GitHub Repo link: Assignment2Repo

8. Conclusion:

Based on the points emphasized in section 4. of this report we conclude that given the dataset, the accuracy for categorizing tweet was moderate. Given the computation power and resources, we used a lexicon file which has less words in its dictionary. It also has an integer score which makes it less efficient. However, we understand that it would be a nice idea to use a lexicon file which has the probability distribution of a word being positive or negative given its occurrence. The accuracy can further be improved by predicting the probability distribution of the tweets being positive, negative or neutral given our prior knowledge.