Automated Sentiment Analysis of Text Data with NLTK

Jiawei Yao
Department of Communication, Beijing University of Posts and Telecommunications, China
chelsea.bean@yeah.net

ABSTRACT At present, most of researches in natural language processing primarily are focused on deep learning way to solve accuracy problems. This paper discusses a branch of natural language processing, sentiment analysis. In order to implement the function of sentiment analysis efficiently, we built an algorithm by calling the NLTK library firstly. As a result, this method can roughly obtain the emotional scores of different sentences and compare them with the scores given by human being who create these sentences. The results are presented by correlation calculation and images such as boxplot. The analysis of the images demonstrates the deficiency of the method. In addition, for the results, we found the direction of learning to solve the accuracy problems and made reasonable arrangements and planning.

1. BACKGROUND

1.1 Natural Language Processing
Natural language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyze large amounts of natural language data. Natural language processing is a model for studying language ability and language application. A computer (algorithm) framework is built to implement such a language model, and it is perfected, evaluated, and finally used to design various practical systems. Its branches are Automatic Speech Recognition (ASR), Named entity recognition (NER), Optical character recognition (OCR), Sentiment analysis and so on.

1.2 Sentiment Analysis
Sentiment analysis (sometimes known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study subjective preferences and affective states. Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event.

1.3 Comma-separated Values File
In computing, a comma-separated values (CSV) file is a delimited text file that uses a comma to separate values. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. The CSV file format is not fully standardized. The basic idea of separating fields with a comma is clear, but that idea gets complicated when the field data may also contain commas or even embedded line-breaks. CSV implementations may not handle
such field data, or they may use quotation marks to surround the field. Quotation does not solve everything: some fields may need embedded quotation marks, so a CSV implementation may include escape characters or escape sequences.

In addition, the term "CSV" also denotes some closely related delimiter-separated formats that use different field delimiters. These include tab-separated values and space-separated values. A delimiter that is not present in the field data (such as tab) keeps the format parsing simple. These alternate delimiter-separated files are often even given a .csv extension despite the use of a non-comma field separator. This loose terminology can cause problems in data exchange. Many applications that accept CSV files have options to select the delimiter character and the quotation character.

1.4 The Natural Language Toolkit
The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

1.5 Random matrix
In mathematics, random matrices (also called probability matrices, transition matrices, substitution matrices, or Markov matrices) are the matrices used to describe the transformation of a Markov chain. Each of its terms is a non-negative real number that represents the probability. It applies to probability theory, statistics and linear algebra and is used in computer science and population genetics.

The random matrix describes the Markov chain in a finite state space S. If the probability of moving from i to j within a time step is Pr(j|i)=\(p_{i,j}\), the i-th row and j-th column elements of the random matrix P are given by \(p_{i,j}\).

Since the sum of the probabilities from state i to the next state must be 1, this matrix is a right random matrix, so we can get \(\sum pt_{i} = 1j\). The probability of a two-step transition from i to j is given by the \((i,j)\) number element of the given square matrix of P: \((p 2)_{i,j}\) [4].

2. DESIGN AND IMPLEMENTATION

2.1 Targets
For each dataset, we need to complete analysis as follows

- Import modules
- Open the input file (csv) using the csv module and read content (texts and ratings).
- Run the sentiment analysis function to each text review and retrieve a score.
- Collect all the scores from the entire dataset.
- Evaluate correlation between user-generated ratings and NLTK-generated scores.
- Visualize the result using R-language.

The basis of these tasks is to extract texts and read content (texts and ratings). Now let me introduce the relevant codes and methods that I used.
2.2 Import Modules
In the Python script, I need to import the modules I need, which contains the functions I am going to use. The codes and libraries as follow:

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk import tokenize
from numpy import *
import numpy
import sys
import os
import csv
```

Take sys for example, first, we use the import statement to enter the sys module. Basically, this statement tells Python that we want to use this module. The sys module contains functions related to the Python interpreter and its environment. When Python executes the import sys statement, it looks for the sys.py module in the directory listed in the sys.path variable. If the file is found, the statement in the main block of the module will be run, and then the module will be available to you. Note that the initialization is only done when we first input the module. In addition, "sys" is short for "system."

2.3 Datasets
In this project, we will use the NLTK's sentiment analysis function to analyze text sentiment using three datasets: 1) Amazon product review, 2) beer review, and 3) movie review. Each dataset provides a list of pairs of a review content and a numeric rating like figure 1. For instance,

- **Text:** “I like this move”
- **Rating:** 5

![Figure 1. beer-short.csv file](image)

2.4 Content Extraction
Read content (texts and ratings) and store them in two lists. Part of codes for implementing these features is as follows:

```python
with open(filename,'rt') as csvfile:
    reader = csv.reader(csvfile)
    listx = [row[1] for row in reader]
    del listx[0]
```
floatlistx = [float(_s) for _s in listx]
for index1 in range(0, len(floatlistx)):
    floatstrx.append("%.4f" % (floatlistx[int(index1)] / 20))
print(floatstrx)

Open the CSV file and use a loop to read the second list in the file, meanwhile listx can store all elements about the second list. We use the same method to extract sentences and store all elements about the first list in col.

2.5 NLTK-generated Scores

sid = SentimentIntensityAnalyzer()
for sentence in col:
    ss = sid.polarity_scores(sentence)
    listy.append(ss["compound"])  

floatlisty = [float(_s) for _s in listy]

Using the function polarity_scores(x) [1] to calculate floats for sentiment strength based on the input text which is a single string text data. The output is a dictionary that has four fields, {‘compound’, ‘neg’, ‘neu’, ‘pos’}. We can select any one of the properties for the next study.

2.6 Evaluate Correlation

Evaluate correlation between user-generated ratings and NLTK-generated scores. Moreover, the correlation is Pearson correlation coefficient. Pearson correlation coefficient, used to measure the correlation (linear correlation) between two variables, is between [-1, +1].

cor = numpy.corrcoef(floatlistx, floatlisty)
print "The correlation is: ", cor

Two lists containing numbers. The shape of floatlistx and floatlisty should be same. We can get the return of the correlation coefficient matrix of the variables. For instance:

   Import numpy
   numpy.corrcoef([1,2,3], [1,3,2])
> array([[1. , 0.5],
        [0.5, 1. ]])

2.7 Collect All the Scores

Create a new file to store all outputs about sentence, ratings and sentiment.

   newfile = file(outputname,'wb')
   writers = csv.writer(newfile)
   writers.writerow(['sentence', 'ratings', 'sentiment'])
   for index in range(len(floatlisty)):
       data=[col[int(index)], floatlistx[int(index)], floatlisty[int(index)]]
       writers.writerow(data)
   newfile.close()

Return a writer object responsible for converting the user’s data into delimited strings on the given file-like object. csvfile can be any object with a write() method. If csvfile is a file object, it must be opened with the ‘b’ flag on platforms where that makes a difference. An optional dialect parameter can
be given which is used to define a set of parameters specific to a particular CSV dialect. It may be an instance of a subclass of the Dialect class or one of the strings returned by the list_dialects() function. The other optional fntparams keyword arguments can be given to override individual formatting parameters in the current dialect. For full details about the dialect and formatting parameters, see section Dialects and Formatting Parameters. To make it as easy as possible to interface with modules which implement the DB API, the value None is written as the empty string. While this isn’t a reversible transformation, it makes it easier to dump SQL NULL data values to CSV files without preprocessing the data returned from a cursor.fetch* call. Floats are stringified with repr() before being written. All other non-string data are stringified with str() before being written. The output.csv file’s screenshot as follows: Figure 2.

Figure 2. Outputs.csv file

2.8 Markov Matrix
Suppose M is a map from time series X(X = {x(t) | t ∈ N, x(t) ∈ R}) to a network g ∈ G(g = {N, A} is a set which have nodes N and arcs A). We can assume the Q quantiles, and then M assigns each quantile q_i to a node n_i ∈ N in the relevant network with weight w_i^k_j as long as two values x(t) and x(t + k) belong to quantiles q_i and q_j, with t = 1, 2, ..., T and the time differences k = 1, ..., k_max < T.

Weights w_i^k_j are simply given by the number of times a value in quantile q_i at time t is followed by a point in quantile q_j at time t + k, normalized by the total number of transitions. Repeated transitions through the same pair increase the value of the corresponding weight. With proper normalization, the weighted adjacency matrix becomes a Markov transition matrix W_k, with ∑_j w_i^k_j = 1[2].

We can use an example to learn this algorithm, see Figure 3. When k = 1, Q = 4 quantiles and X have T = 60 time points (three coloured lines mean four quantiles) [5].
3. RESULTS AND DISCUSSION

3.1 Boxplot and Histogram-R language

The American statistician John Tukey invented the box map in 1977. It consists of five numerical points: minimum (min), lower quartile (Q1), median, upper quartile (Q3), and maximum (max). You can also add averages (mean) to the box diagrams, as shown above. The next quartile, median and upper quartile make up a "box with compartments." An extension line is established between the upper quartile and the maximum value. This extension line becomes a "whisker" [6].

Since there are always "dirty data" and "outliers" in real-life data, these outliers are remitted separately so as not to shift the overall characteristics due to these few outliers. The two levels of whiskers in the box diagram were modified to the minimum and maximum observations. The maximum (minimum) observation value is set an experience here to 1.5 IQR distances from the quartile values.

When analysing the data through the box diagram, the box diagram can help us to identify the characteristics of the data effectively:

- Visually identify outliers in the data set (see outliers).
- Determine the degree of data dispersion and bias in the data set (observe the length of the box, the shape of the upper and lower compartments, and the length of the beard).

In addition, a histogram is an accurate representation of the distribution of numerical data.

Related codes:

```r
#boxplot
library(ggplot2)
acs = read.csv("C:/Users/46025/Desktop/SentimentDetection.csv",header=T)
ggplot(acs,aes(x = ratings,y = sentiment)) + geom_boxplot()
#histogram
library(ggplot2)
acs=read.csv("C:/Users/46025/Desktop/SentimentDetection.csv",header=T)
ggplot(acs, aes(x = ratings)) + geom_histogram(binwidth = 1, fill = "lightblue", colour = "black")
```

The ggplot2 is a data visualization package for the statistical programming language R. Geom represents a geometric object, which is an important layer control object in ggplot because it is responsible for the type of graphics rendering. Therefore, we can figure the boxplot and histogram by using this package easily.
3.2 Boxplot and Histogram-display

Figure 4 is the boxplot and histogram results I generated using the dataset movie.csv. According to the boxplot, we can see that rating 4 and 5, the number of outliers are very much. In addition, we can know that the number of rating 4 and 5 in the histogram is the most, from which we can know that our method is limited to a small amount of text. If we handle a large number of sentences using this method, it can reduce accuracy and get many of the wrong scores. Meanwhile, the calculation of correlation also proves this conclusion. For instance, we calculated the correlation of beer.csv and beer-short.csv, the results as follow:

![Figure 5. beer.csv correlation](image)

![Figure 6. beer-short.csv correlation](image)

We can get the same conclusion from figure 5 and figure 6, because beer-short.csv just have 10 sentences, beer.csv have 5000 sentences. Compared with the correlations, we can know if we handle a large number of sentences by using this method, it can reduce accuracy and get many of the wrong scores.

4. CONCLUSION AND FURTHER WORK

The sentiment prediction system we used in this paper works just by looking at words in isolation, giving positive points for positive words and negative points for negative words and then summing up these
points. Through evaluating correlation between user-generated ratings and NLTK-generated scores and figuring the boxplot and histogram, we can know the order of words is ignored and important information is lost, ex: This movie was actually neither that funny, nor super witty. It caused the results maybe have lower accuracy with the number of sentences increasing. In contrast, the fact that we want to build a new deep learning model actually builds up a representation of whole sentences based on the sentence structure. It computes the sentiment based on how words compose the meaning of longer phrases. We can get scores that are more accurate by constantly training data sets and adjusting weights like figure 7. This way, the model is not as easily fooled as previous models.

The next step I am going to do is to understand TensorFlow and TFlearn, which is a modular and transparent deep learning library built on top of Tensorflow. Based on these fundamental knowledges, I will build a new deep learning model which have higher performance in Sentiment Analysis [3].

REFERENCES
[1] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.
[2] A. Campanharo, E. Doescher, F. Ramos, "Automated EEG signals analysis using quantile graphs," 14th Int. Work-Conf. Artif. Neural Netw. IWANN, Proceed. 2017.
[3] M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. Computational Linguistics, 36(4):673–721.
[4] TAO Wenbing, JIN Hai. A novel based on graph theory image segmentation method [J]. Chinese Journal of computers, 2007, 30(1): 110-119.
[5] MathWorks, (1994-2018). Documentation - Quantile-quantile plot. [online]Available from: http://cn.mathworks.com/help/stats/qqplot.html?s_tid=gn_loc_drop [(Accessed: 5/1/2018)].
[6] John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team; 2012 - 2018 The Matplotlib development team. Last updated on Aug 11, 2018. Created using Sphinx 1.7.6. Doc version v2.2.3-1-gd47e15e7a.