Author: M. Kolaksazov,

Main source: https://marcobonzanini.com/2015/03/02/mining-twitter-data-with-python-part-1/

ANALYSIS OF THE SENTIMENT OF DATA FROM THE SOCIAL MEDIA

ANALYSIS OF DATABASES FROM “TWITTER”, THE ACCOUNT OF NETFLIX

INTRODUCTION: SOCIAL MEDIA AND TEXT ANALYSIS

- Social media contains a large amount of data in the form of messages, distributed between users, containing information about different subjects.
- The text data can contain different categories of emotional meaning, as well. Consequently, this emotional meaning is analysed in the field of text analysis.
- The field of text analysis includes the areas of “Natural language processing” and the “Sentiment analysis”. It is used in order to categorize, quantify and study semantic information in text.
- On the basis of the analysis of text carried out by the means of algorithms for natural language processing, this text data can be further sorted in categories, e.g.: positive or negative sentiment.
- For Python, a set of libraries exist, called NLTK, or Natural language toolkit, which contains libraries for processing and analyzing of the data from text.
- Other option for the analysis of the text can be the “so-called” pointwise mutual information (PMI) of combination of two words, that occur next to each other in the text data. This parameter is used to calculate what is the semantic relation of two word, that can be found one after other in text.

PREPARATION OF A PROGRAM FOR TEXT DATA ANALYSIS FROM TWITTER

PLAN, DESCRIBING THE STEPS IN THE TEXT ANALYSIS:

- Obtaining the text data from the Twitter API
- Pre-processing the text data. Creating a Python DataFrame containing different arguments from a tweet as columns. Removing unnecessary words and symbols, such as numbers, hashtags, @-mentions, url addresses, very short words, stop words and the own personal names.
- Sorting the data from the tweet on the basis of the sentiment, such as containing positive and negative sentiment. This can be realized by the means of the algorithms from TextBlob sentiment analysis.
- Calculating of the PMI for every combination of words from text.
- Carrying out the sentiment analysis by the means of comparing words from the text, as well as from two vocabularies, containing positive and negative word and calculation of the PMI.
- Comparing between the sentiment analysis, performed by the means of the TextBlob algorithms, as well as the PMI calculation.

OBTAINING THE TEXT DATA

- Text data from Twitter could be obtained by two ways:
  - Accessing the Twitter API and importing a certain number of tweets (limited to 200). Tweets are imported with their specific attributes, such as id number, date of publishing, count of likes, etc., or
  - Reading the data from a *.txt or *.json file and transform it as Python DataFrame object. In this case, the text can be pre-sorted by sentiment, in order supervised training with a machine learning algorithm to be carried out .
  - In the current task, tweets from the channel of “Netflix”, were used.

PRE-PROCESSING OF THE TEXT

- Data is next transformed in a Python DataFrame, every tweet being a row from the DataFrame, and the specific attributes of the tweet are arranged in the columns.
- After that, the text from the tweet undergoes the so-called pre-processing. This includes cleaning the tweet from the unnecessary words and symbols, such as:
  - Stopwords. This is a special category of words, taken from NLTK vocabulary and includes commonly used words: “the”, “a”, “in”, “on”, “just”, etc.
  - Hashtags, url-addresses and @-mentions, as well as the numbers inside text
  - Names, e.g. “John Smith”, “Los Angelis”, “San Francisco” that can almost always be found together.
  - We want the special characters inside words, used in different languages to remain, for example accents or umlaut.

SORTING OF TEXT DATA BY SENTIMENT WITH TEXTBLOB

- TextBlob 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.

Features:

- Noun phrase extraction
- Part-of-speech tagging
- Sentiment analysis
- Classification (Naive Bayes, Decision Tree)
- Language translation and detection powered by Google Translate
- Tokenization (splitting text into words and sentences)
- Word and phrase frequencies
- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction
- Add new models or languages through extensions
- WordNet integration

CALCULATION OF THE POINTWISE MUTUAL INFORMATION

- Sentiment Analysis is one of the interesting applications of text analytics. Although the term is often associated with sentiment classification of documents, broadly speaking it refers to the use of text analytics approaches applied to the set of problems related to identifying and extracting subjective material in text sources.
- Pointwise mutual information (PMI) in theory of informatics and statistics is related to the probability of one event to occur simultaneously with other. In contrast to mutual information (MI) which builds upon PMI, it refers to single events, whereas MI refers to the average of all possible events.

$$\mathrm{PMI}(t_1, t_2) = \log\left(\frac{P(t_1 \wedge t_2)}{P(t_1) \cdot P(t_2)}\right)$$

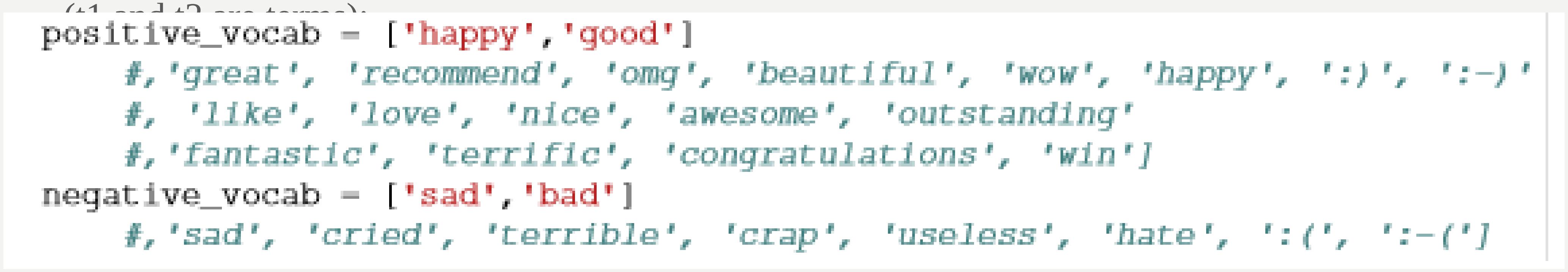
- The probability of observing the term *t* (P*(t)*) and the probability of observing the terms *t1* and *t2* occurring together (P*(t1^t2)*) can be computed on the basis of the set of documents (tweets) *D*. We define the Document Frequency (DF) of a term as the number of documents where the term occurs. The same definition can be applied to co-occurrent terms.
- Hence, we can define our probabilities as:

$$P(t) = \frac{\mathrm{DF}(t)}{|D|}$$

$$P(t_1 \wedge t_2) = \frac{\mathrm{DF}(t_1 \wedge t_2)}{|D|}$$

SENTIMENT ANALYSIS WITH PMI

- The technique we’re discussing in this post has been elaborated from the traditional approach proposed by Peter Turney in his paper „Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews“. A lot of work has been done in Sentiment Analysis since then, but the approach has still an interesting educational value. In particular, it is intuitive, simple to understand and to test, and most of all *unsupervised*, so it doesn’t require any labelled data for training.
- Firstly, we $\mathrm{SO}(t) = \sum_{t' \in V^+} \mathrm{PMI}(t,t') - \sum_{t' \in V^-} \mathrm{PMI}(t,t')$ SO) of a word as the difference between its associations with positive and negative words. In practice, we want to calculate “how close” a word is with terms like *good* and *bad*. The chosen measure of “closeness” is Pointwise Mutual Information (PMI), calculated as follows (t1 and t2 are terms):

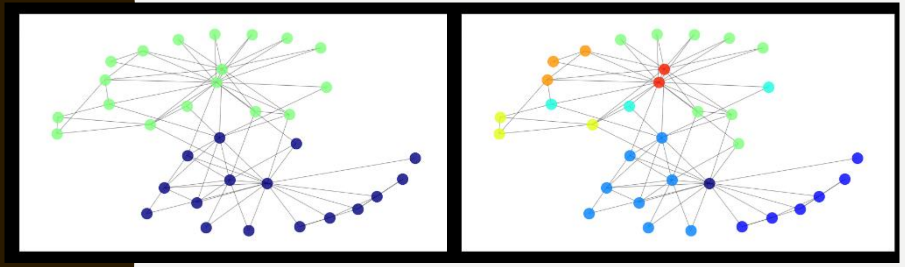


```
positive_vocab = ['happy','good']
    #,'great', 'recommend', 'omg', 'beautiful', 'wow', 'happy', ':)', ':-)'
    #, 'like', 'love', 'nice', 'awesome', 'outstanding'
    #,'fantastic', 'terrific', 'congratulations', 'win']
negative_vocab = ['sad','bad']
    #,'sad', 'cried', 'terrible', 'crap', 'useless', 'hate', ':(', ':-(']
```

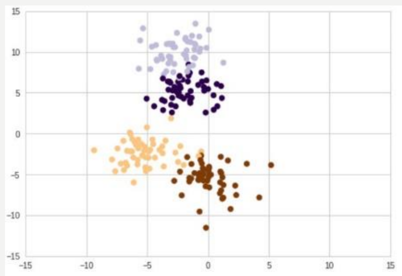
CATEGORIZING TEXT DATA BY THE MEANS OF THE ALGORITHM FOR UNSUPERVISED LEARNING, THE CLUSTER ANALYSIS

- When the data is not labeled, it can not be described by the means of supervised learning algorithms for machine learning. Therefore, other type of learning is required, which works with unlabeled, uncategorized data, or the so-called “unsupervised” learning.
- Unsupervised learning is used to classify data, based on its structure. In cluster analysis the algorithm observes patterns in the data and after that separates the data in clusters. The amount of clusters can be known in advance, or the algorithm can be set to calculate the analysis with a sequence of number, and after that to choose the best way to divide the data into different clusters.
- Depending on how many clusters we want to separate the data, the algorithm can give different results.
- In the current task, we know in advance, that three types of data exist: positive, negative and neutral, so we used the k-means algorithm to categorize data into these three classes.

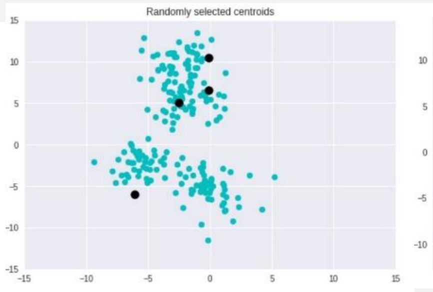
Organizing data
into different number of clusters

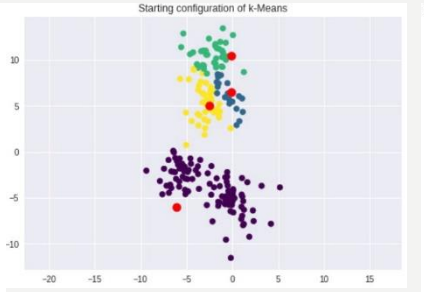


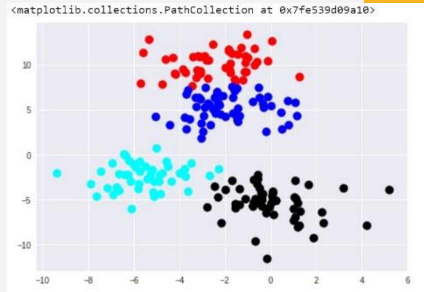
original data



organizing data into clusters by different means



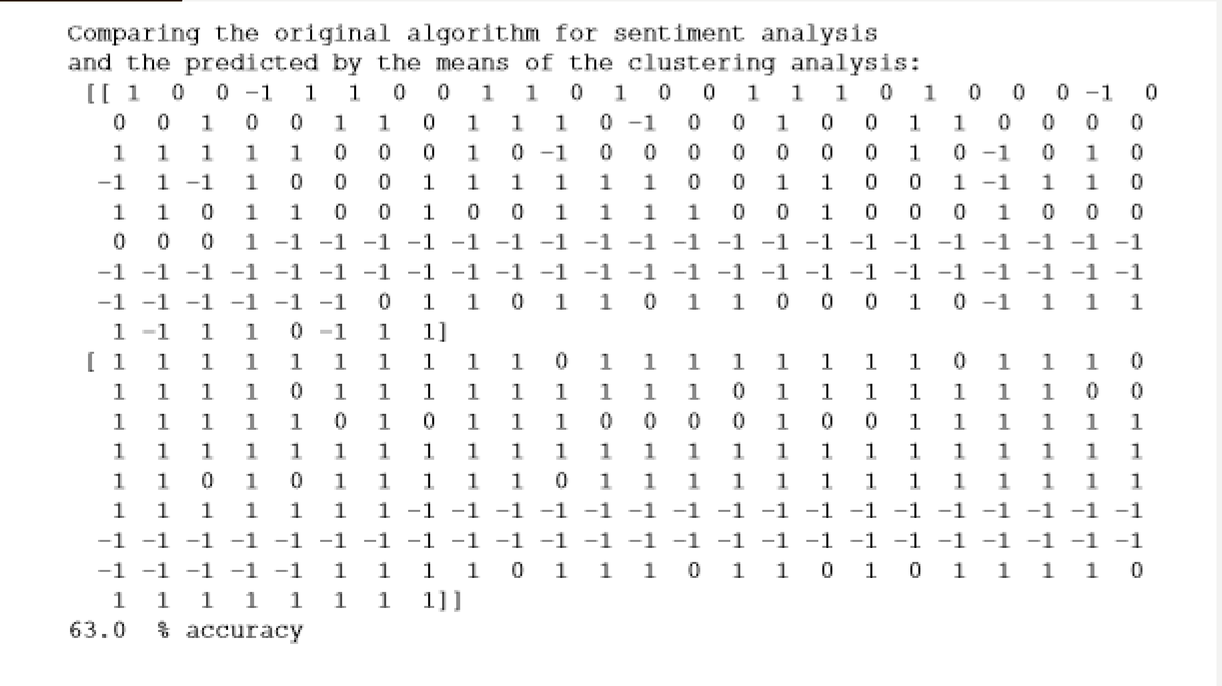




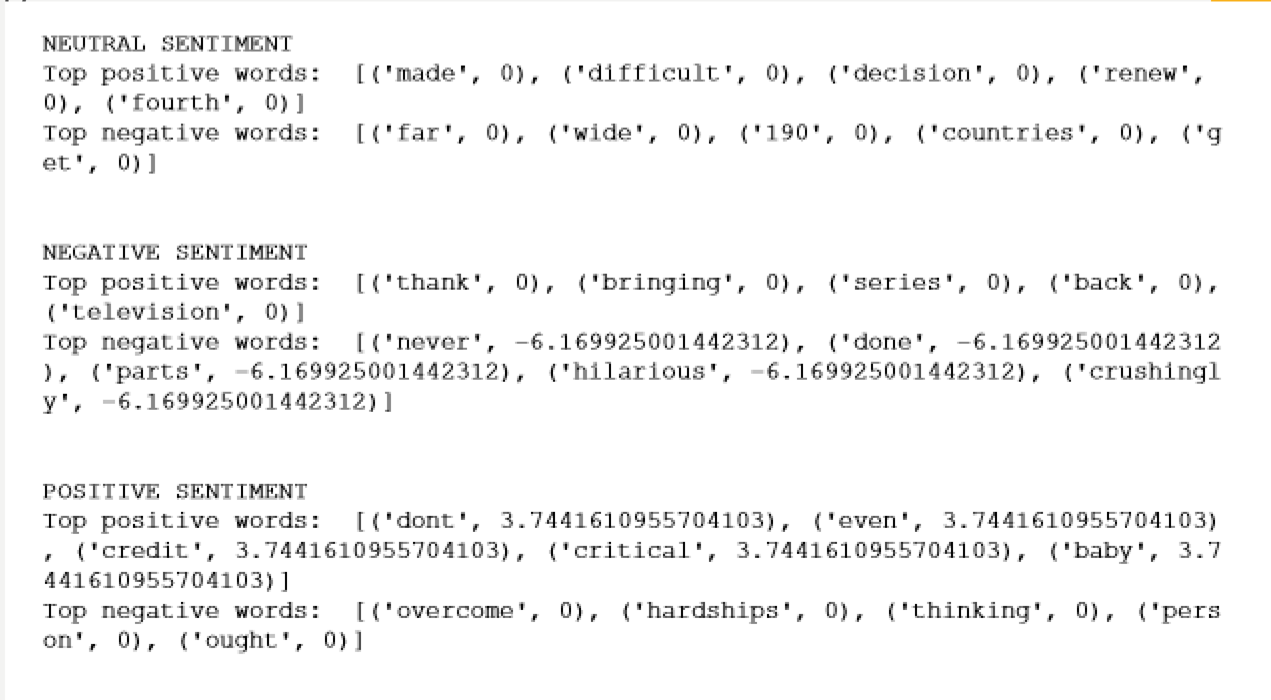
```
<matplotlib.collections.PathCollection at 0x7fe539d09a10>
```

COMPARING BOTH ALGORITHMS

- It was very interesting to compare both algorithms for sorting by sentiment
- For this purpose tweets were separated into three different groups (positive “1”, negative “-1” and neutral “0”) by the sentiment algorithm from TextBlob and the PMI of word combinations from each group was calculated.
- It can be clearly seen, that inside “positive sentiment” category there are words with positive PMI, but not negative, whereas inside “negative” category it is the reverse. This category contains only words with negative PMI. In the third class, or the so-called neutral sentiment lacks words with positive or negative PMI.
- It is also worth to note, that after sorting the data by the means of the TextBlob and after sorting by the means of the cluster analysis, both means of sorting were compared. It was found out, that the accuracy of the cluster analysis (by k-means) was 63%, compared with the TextBlob sorting.



```
Comparing the original algorithm for sentiment analysis
and the predicted by the means of the clustering analysis:
 [[ 1  0  0 -1  1  1  0  0  1  1  0  1  0  0  1  1  1  0  1  0  0  0 -1  0
   0  0  1  0  0  1  1  0  1  1  1  0 -1  0  0  1  0  0  1  1  0  0  0  0
   1  1  1  1  1  0  0  0  1  0 -1  0  0  0  0  0  0  0  1  0 -1  0  1  0
  -1  1 -1  1  0  0  0  1  1  1  1  1  1  0  0  1  1  0  0  1 -1  1  1  0
   1  1  0  1  1  0  0  1  0  0  1  1  1  1  0  0  1  0  0  0  1  0  0  0
   0  0  0  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1 -1  0  1  1  0  1  1  0  1  1  0  0  0  1  0 -1  1  1  1
   1 -1  1  1  0 -1  1  1]
 [ 1  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1  1  1  0  1  1  1  0
   1  1  1  1  0  1  1  1  1  1  1  1  1  1  0  1  1  1  1  1  1  1  0  0
   1  1  1  1  1  0  1  0  1  1  1  0  0  0  0  1  0  0  1  1  1  1  1  1
   1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
   1  1  0  1  0  1  1  1  1  1  0  1  1  1  1  1  1  1  1  1  1  1  1  1
   1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
  -1 -1 -1 -1 -1  1  1  1  1  0  1  1  1  0  1  1  0  1  0  1  1  1  1  0
   1  1  1  1  1  1  1  1]]
63.0  % accuracy
```

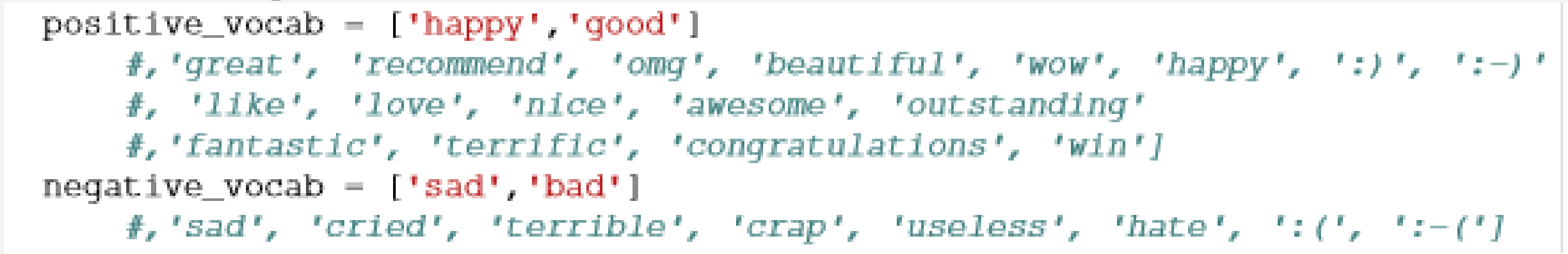


```
NEUTRAL SENTIMENT
Top positive words:  [('made', 0), ('difficult', 0), ('decision', 0), ('renew',
0), ('fourth', 0)]
Top negative words:  [('far', 0), ('wide', 0), ('190', 0), ('countries', 0), ('g
et', 0)]


NEGATIVE SENTIMENT
Top positive words:  [('thank', 0), ('bringing', 0), ('series', 0), ('back', 0),
('television', 0)]
Top negative words:  [('never', -6.169925001442312), ('done', -6.169925001442312
), ('parts', -6.169925001442312), ('hilarious', -6.169925001442312), ('crushingl
y', -6.169925001442312)]


POSITIVE SENTIMENT
Top positive words:  [('dont', 3.7441610955704103), ('even', 3.7441610955704103)
, ('credit', 3.7441610955704103), ('critical', 3.7441610955704103), ('baby', 3.7
441610955704103)]
Top negative words:  [('overcome', 0), ('hardships', 0), ('thinking', 0), ('pers
on', 0), ('ought', 0)]
```

- In the described algorithm, vocabularies with only two (positive and negative) words were used. Thus, the searching criteria is reduced, as well as the PMI, which has comparatively low values.
- When more words were included inside vocabularies, the differences of PMI increases. It can be seen, that inside negative sentiment class can be found words with positive PMI, however, its values were lower, as compared with the PMI of words from positive sentiment class.



```
positive_vocab = ['happy','good']
    #,'great', 'recommend', 'omg', 'beautiful', 'wow', 'happy', ':)', ':-)'
    #, 'like', 'love', 'nice', 'awesome', 'outstanding'
    #,'fantastic', 'terrific', 'congratulations', 'win']
negative_vocab = ['sad','bad']
    #,'sad', 'cried', 'terrible', 'crap', 'useless', 'hate', ':(', ':-(']
```

POSITIVE				NEGATIVE				NEUTRAL SENTIMENT			
Top positive words:		Top negative words:		Top positive words:		Top negative words:		Top positive words:		Top negative words:	
'boy'	10.13218	'absolutely'	0	'100'	6.169925	'believe'	0	'made'	0	'movie'	0
'critical'	8.81025	'away'	0	'class'	6.169925	'download'	0	'difficult'	0	'split'	0
'netflix'	5.066089	'tear'	0	'morning'	6.169925	'days'	0	'decision'	0	'half'	0
'cry'	5.066089	'eye'	0	'finis'	6.169925	'sorry'	0	'renew'	0	'officially'	0
'laugh'	5.066089	'better'	0	'lifespan'	4.584963	'better'	0	'fourth'	0	'production'	0
'dies'	5.066089	'personifies'	0	'goldfish'	4.584963	'future'	0	'season'	0	'things'	0
'harnessed'	5.066089	'resilience'	0	'feel'	4.584963	'dont'	0	'choice'	0	'value'	0
'facts'	5.066089	'human'	0	'answer'	4.584963	'shot'	0	'come'	0	'distributes'	0
'grisly'	5.066089	'spirit'	0	'grilled'	4.584963	'one'	-9.17	'way'	0	'black'	0
'watched'	4.481127	'power'	0	'cheese'	4.584963	'binged'	-12.34	'harsh'	0	'work'	0
'last'	4.481127	'overcome'	0	'job'	4.584963	'never'	-12.34	'tai'	0	'far'	0
'night'	4.481127	'hardships'	0	'thank'	0	'done'	-12.34	'fuck'	0	'wide'	0
'basically'	4.481127	'thinking'	0	'bringing'	0	'parts'	-12.34	'one'	0	'190'	0
'hours'	4.481127	'person'	0	'series'	0	'hilarious'	-12.34	'fish'	0	'countries'	0
'stomach'	4.481127	'ought'	0	'back'	0	'crushingly'	-12.34	'people'	0	'get'	0

- ***As a conclusion***, the sentiment analysis is a very subjective method for text analysis, which is highly dependent to the words, that were included as a mean of classification. One way to reduce subjectivity is to use unsupervised learning as a method for categorization. It should be noted, that, standard algorithms for unsupervised learning, by the means of cluster analisys, such as k-means are not very efficient (63% in the current task).