OULU UNIVERSITY OF APPLIED SCIENCES

Cao Khac Sang

DATA MINING FOR SOCIAL NETWORK DATA

DATA MINING FOR SOCIAL NETWORK DATA

Cao Khac Sang
Bachelor's Thesis
Spring 2019
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information Technology

Author: Cao Khac Sang
Title of Bachelor´s thesis: Data Mining for Social Network Data
Supervisor: Kari Laitinen
Term and year of completion: Spring 2019 Number of pages: 35

Today, people are possessing huge databases whose potentials have not been fully exploited. In particular, the development of information technology and the application of information technology in many fields have made that data treasure increase rapidly. Besides, social networks, such as Facebook, Twitter and Instagram are growing and having a great impact on social life. Typically, Twitter is a special social network where each message is sent with no more than 140 characters but having an amazing spread rate. Many of these social network users are singers, actors and especially politicians and only one of their status on social networks has an impact on real-world society. This explosion has led to an urgent need for new techniques and tools to automatically convert that huge amount of data into useful knowledge. Conducting such jobs is the process of discovering knowledge in the database, in which the data mining technology has become a current trend of information technology in the world in general. The application was created with the purpose of classifying the emotional status and discovering the main concern of a politician.

The project was developed by using Python for every single task including scrapping data and analysis. In addition, the collected data was stored in the CSV format for the convenience of data visualization and analysis. The thesis will show key topics of messages that a particular politician sends to people via Twitter.

The crucial part for the whole project is its source code which is taken apart in this thesis and is accessible at the following GitHub link: https://github.com/ilookforme102/bachelor_thesis

---

Keywords:

Data Mining, Natural Language Processing, Machine Learning, Social Network, Python

PREFACE

The basis for this thesis originally stemmed from my internship at ABCD Agency in Berlin. During the internship, I had experience working with a wide range of data collected from social networks and I found that there was a lot of potential from exploiting information hidden in the text form. Therefore, I wanted to realize part of my ideas in this thesis.

I would like to express my sincere gratitude to the support of many people. First of all, my parents, who supported me under all circumstances. Secondly, my tutoring teacher, who helped me with the thesis. Finally, my colleagues at the ABCD Agency, who gave me many ideas during the application development process.

Helsinki, 1.5.2019
Cao Khac Sang

CONTENTS

VOCABULARY

API: Application Programming Interface

CSV: Comma-Separated Values

DL: Deep Learning

LDA: Latent Dirichlet Allocation

ML: Machine Learning

NLP:Natural Language Processing

NLTK: Natural Language Tookit

LSA: Probabilistic Latent Semantic Analysis

URL: Uniform Resource Locator

1 INTRODUCTION

Data mining is a set of techniques that are used to automatically exploit and find out the mutual relationships of data in a huge and complex dataset and also to find patterns hidden in that dataset. The overall aim of the data mining process is to extract, exploit and use potentially valuable data from inside a large amount of data stored in the data storage system to create an easy-to-understand information structure.
Aside from the raw analysis step, it also involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. (1).

A data mining application was created in this thesis work. The entire application is deployed in Python based on the author's experience and basic needs in using this programming language daily. The focus of this thesis is to analyze data collected from Twitter. The theory of sentiment analysis and topic modeling as well as the workflow will be discussed in Chapter 2. On the other hand, Chapter 3 will provide a brief introduction of the tools involved in the application deployment process. The sample code will be shown in Chapter 4 for the purpose of evaluation and further discussion. The final chapter will illustrate an overview of this thesis.

This topic was selected based on the author's passion for mathematics and programming, both seemingly completely different fields that have now been combined to develop one of the most important industries in the next era of humanity - data science.
Hopefully, in the future, this topic will be further expanded and strengthened by more advanced algorithms with a higher accuracy.

2 THEORY

The ultimate aim of the thesis was to solve two main categories - sentiment analysis and topic modeling for the collected data.

2.1 Sentiment Analysis

Sentiment analysis or opinion mining is the process of computationally identifying and categorizing opinions, attitudes, and emotions expressed in a piece of text (2). The main purposes of sentiment analysis are to detect and extract subjective information for determining whether the writer's attitude towards a particular topic is positive, neutral, or negative. The sentiment analysis is a hot subject in Natural Language Processing with a large number of works and it is performed on various levels: document level, sentence level, and word level.

There can be two approaches to the sentiment analysis.
1. Lexicon-based methods
2. Machine Learning-based methods.
In this problem, the Lexicon-based approach was used.
The lexicon-based approach calculates the sentiment of a given text from the polarity of the words or phrases in that text (3). For this method a lexicon (a dictionary) of words with the polarity assigned to them is required. Examples of the existing lexicons includes: Opinion Lexicon, SentiWordNet, AFINN Lexicon, NRC-Hashtag, General Inquirer Lexicon3.

The workflow of a lexicon-based sentiment analysis process is illustrated in FIGURE 1.

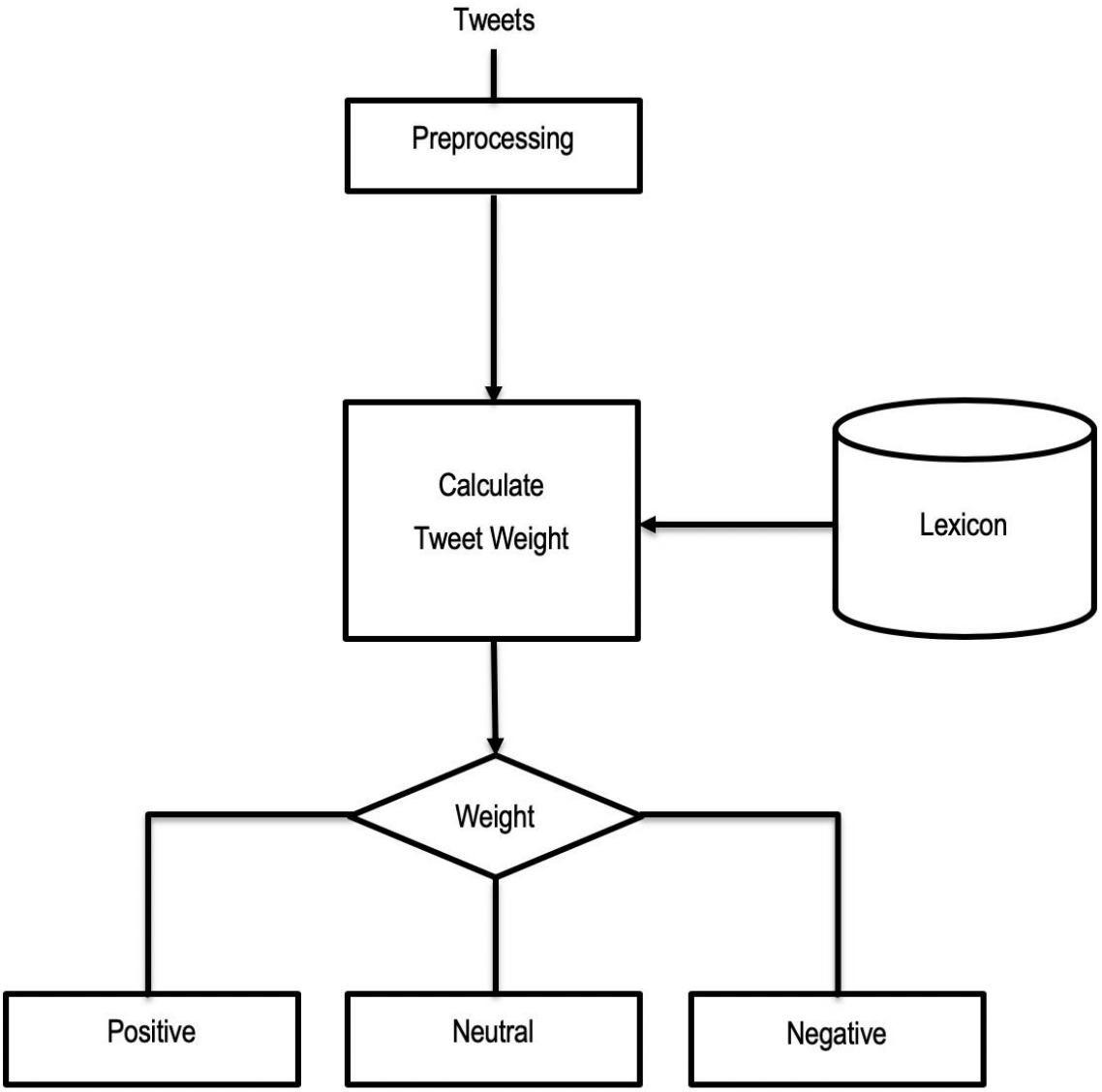


FIGURE 1. Architecture of the framework of lexicon-based sentiment analysis.

The sentiment score of the text T can be computed as the average of the polarities conveyed by each of the words in the text. The methodology for the sentiment can be described with the following steps:

- Pre-processing. The text undergoes pre-processing steps that were described in the previous section: POS tagging, stemming, stop-words removal, negation handling, tokenizing into N-grams. The outcome of the pre-processing is a set of tokens or a bag-of-words.
- Checking each token for its polarity in the lexicon. Each word from the bag-of-words is compared against the lexicon. If the word is found in the lexicon, the polarity W_i of that

word is added to the sentiment score of the text. If the word is not found in the lexicon, its polarity is considered to be equal to zero.

- Calculating the sentiment score of the text. After assigning polarity scores to all words comprising the text, the final sentiment score of the text is calculated by dividing the sum of the scores of words caring the sentiment by the number of such words:

$$Score_{AVG} = \sum_{i=1}^{m} W_i$$

The averaging of the score allows obtaining a value of the sentiment score in the range between -1 and 1, where 1 means a strong positive sentiment, -1 means a strong negative sentiment and 0 means that the text is neutral.

For example, for the text: "*lexicon*[+0.52] *is*[0] *very*[+0.89] *great*[+0.97] *method*[0]"

the sentiment score is calculated as follows:

$$Score_{AVG} = \frac{0.52 + 0 + 0.89 + 0.97 + 0}{5} = 0.48$$

The Score 0.48 represents the positive in the semantics for the sentence *"lexicon is very great method"*

2.2 Topic Modeling

The idea behind Topic Modeling is to build new documents based on the probability distribution. First of all, to create a new document, it is necessary to select a distribution of topics. This means that documents are made up of different topics, with different distributions. Besides, to generate words for a document, words can be randomly selected based on the probability distribution of the words on the topics. In contrast, given a set of documents, it is possible to define a set of hidden topics for each document and distribute the probability of the words on each topic.

There are many ways to approach this issue; however, the author will use LDA as a solution within the scope of this project.

2.2.1 Latent Direchlet Allocation

LDA is a probability model for idea-based discrete data sets: each document is a mixture of multiple topics. In essence, LDA is a three-level hierarchical Bayes model: the corpus level, document level, word level in which each part of the model is treated as a finite mixing model based on the set of topic probabilities. FIGURE 2 illustrates the idea of LDA: a document is a probability distribution of different topics.

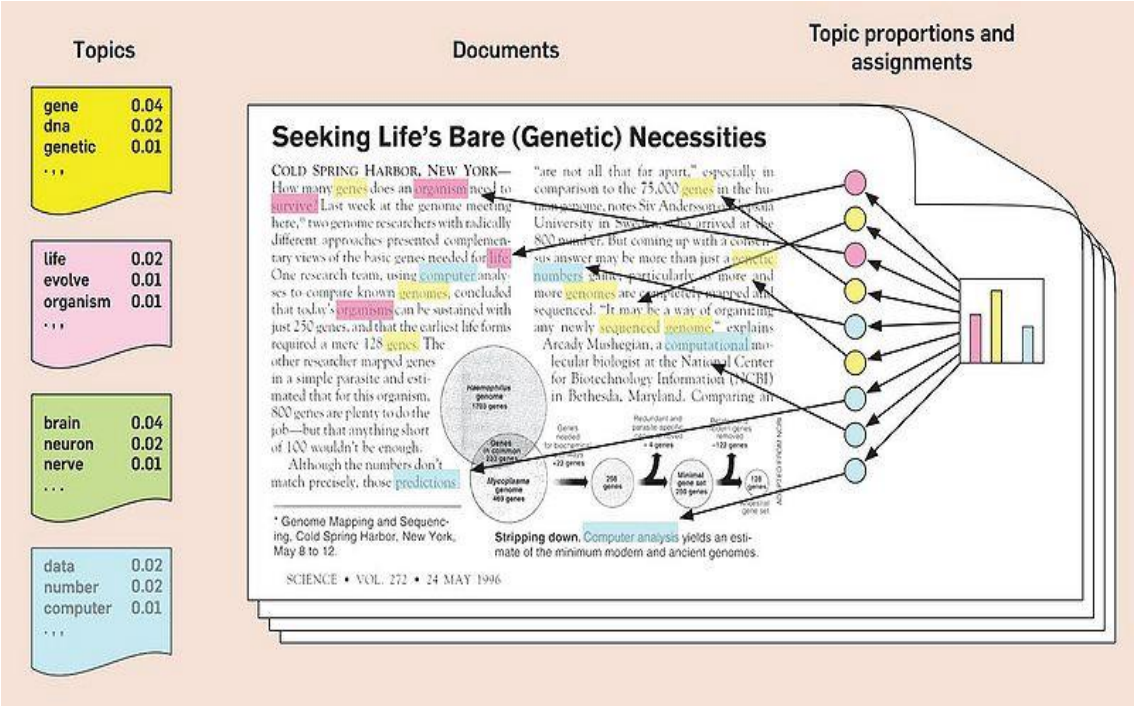


FIGURE 2. Each document is a mixture of topics (4)

2.2.2 Parameter Estimation Process

Formally,the following terms were defined:

• A corpus is a collection of M documents denoted by D = $\{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_M\}$

• A document is a sequence of N words denoted by $\boldsymbol{d}$ = $(w_1, w_2, ..., w_N,)$, where w_n is the n-th word in the sequence.

• A word is the basic unit of discrete data, defined to be an item from a vocabulary indexed by {1,...,V}. We represent words using unit-basis vectors that have a single component

equal to one and all other components equal to zero. Thus, using superscripts to denote components, the vth word in the vocabulary is represented by a V-vector w such that $w^v = 1$ and $w^u = 0$ for $u \neq v$.

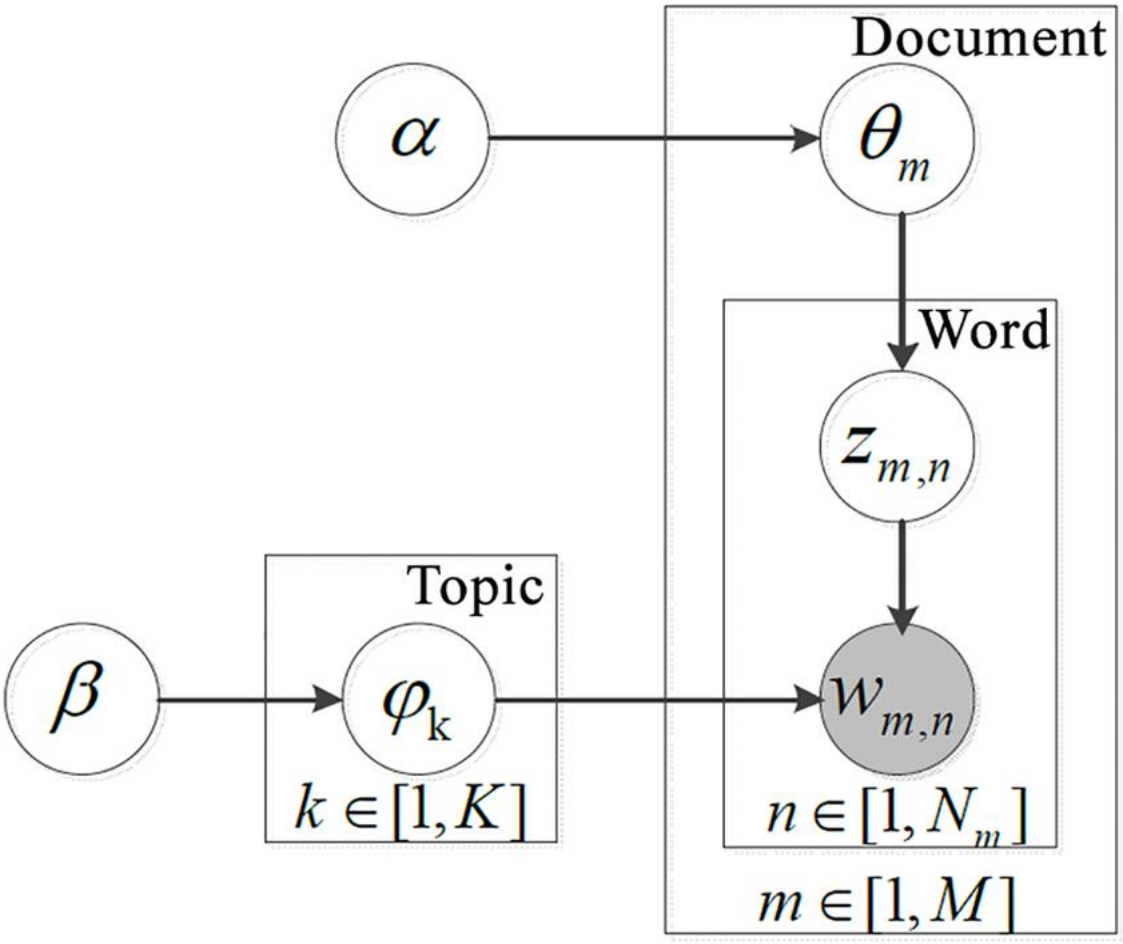


FIGURE 3. Graphical model representation of LDA (5).

The graphical model representation of LDA is clearly described in FIGURE 3. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

Notations:

α: Dirichlet prior on θ_m

β: Dirichlet prior on φ_{k}

M: Number of documents in corpus D = $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_M\}$

K: Number of latent topics

V: Number of words in Vocabulary

N_m: Number of words in the m-th document or length of document $\boldsymbol{d}_m$

$Z_{m,n}$: Topic of word W_n in document $\boldsymbol{d}_m$ (topic index)

$W_{m,n}$: n-th word in document $\boldsymbol{d}_m$ index by $Z_{m,n}$

θ_m: Distribution of topic in m-th document

φ_k: Distribution of word in topic $Z_{m,n}$

Process of topic modeling by LDA:

- For each document m, create the distribution of topic θ_m
- Topic index $Z_{m,n}$ is created from the first step.
- For each topic index and the distribution φ_k then $W_{m,n}$ is created.
- φ_k is sampled once for the entire corpus

3 TECHNOLOGY

This chapter provides technologies used in this thesis project. It also gives an overview of the context and work process of the thesis.

3.1 Python

Python is a very popular object-oriented programming language for writing system utilities and code on the Internet. The latest version of this programming language is 3.7. Created by Guido van Rossum in Amsterdam in the late 80s, Python completely created dynamics and used automatic memory allocation mechanisms. Python was developed in an open source project and is currently managed by the Python Software Foundation (6).
Some typical characteristics of Python:

- Python is Interpreted: Thanks to the interpreter function, Python's interpreter can handle commands at runtime. Thus, there is no need to compile the program before executing it (similar to Perl and PHP).
- Python is Interactive: Python's interactive feature makes it possible to interact directly with its interpreter right at the command prompt. Specifically, it can execute the command directly at the Python prompt.
- Python is Object-Oriented: Python strongly supports the object-oriented programming style and coding package programming techniques in the object.
- Python is a Beginner's Language: Although Python is considered a programming language for those who are new to computer programming, it strongly supports the development of many different types of applications, from scientific and numeric applications to software development application and game.

There are many libraries in Python for science and numerical calculations, such as SciPy and NumPy, which are used for general purposes in computing. And, there are specific libraries, for example, EarthPy for earth science, AstroPy for Astronomy. In addition, Python is also heavily used in machine learning, data mining and deep learning.

3.2 Data Visualization Tools

Data visualization is an important step in pre-processing data to build an effective machine learning model.

Data visualization helps to better understand input data including data distribution, characteristics and correlation of features as well as the visual representation of data with noise or missing. These insights help greatly in the process of selecting or training model.

In the field of data science, the data visualization process plays an important role, helping to show the relationship and meaning of data more clearly. Python, the most popular language in the field of data science now, has a lot of options for doing this task besides many native Python libraries, such as Matplotlib, Seaborn and other libraries.

3.2.1 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms (7). Matplotlib provides many types of visualizations for presenting data with just few lines of code. It can also be used to draw three-dimensional graphs or add interactive effects to diagram.

Because the popularity of Python is increasing, Matplotlib also receives the same attention. Since Matplotlib is a combination of Numpy and Scipy, it is considered a perfect alternative to MATLAB.

3.2.2 Seaborn

Seaborn is a Python visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics (8). Seaborn is an improvement in the complexity of the graph and simpler in syntax when compared to Matplotlib. It is tightly integrated with the PyData stack including support for NumPy and Pandas data structures and statistical routines from scipy and statsmodels.

3.2.3 Plotly

Plotly provides online graphing, analytics, and statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST (9). Built

on top of plotly.js, plotly.py is a high-level, charting library. plotly.js ships with over 30 chart types, including e.g. scientific charts, 3D graphs, statistical charts, SVG maps, financial charts.
Plotly is not only a data visualization tool but a platform that provides online graphing, analytics, and statistics tools for individuals and collaboration.

3.3 Data Analysis Libraries

Python is also one of the two most commonly used programming languages by data scientists.
In addition to the data visualization role, in computer science alone, Python can be used for data analysis, web programming, machine learning, natural language processing, and other things.
Python is a very flexible programming language. It is suitable for handling many tasks including not only statistics or analysis. For example, you can do both statistics and programming on the same Python program so that the data analysis program can integrate into other code sections in the production chain easily.
Python has many outstanding, relevant data science libraries, such as NumPy, Pandas and Scikit-learn.

3.3.1 Numpy and Pandas

The most basic package, when the scientific computation stack was built, is NumPy (shortened from Numerical Python), which provides a lot of useful features for parts operations in n-arrays and matrix in Python. This library provides the ability to vectorize math operations in the NumPy array, which improves performance along with the execution speed.

On the other hand, the Pandas library in Python is an open source library for reading or writing data between memory and many file formats: CSV, text files, Excel, SQL, HDF5. It is also a powerful set of Python programming and data analysis and processing tools that intelligently link data, handle missing data and automatically unstructured data about a structured form. Pandas uses a separate data structure called DataFrame, which provides flexibility and efficiency in data manipulation and indexing as well as provides a lot of functions to handle and work on this data structure.
Today, this package is widely used in both research and development of data science applications.

3.3.2 NLTK and Textblob

Written by Steven Bird and Edward Loper, Computer Department, University of Pennsylvania, USA and 2001, NLTK or Natural Language Toolkit is a library written in Python that supports Natural Language Processing. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum (10).

NLTK is increasingly improving and integrating new tools by thousands of programmers and collaborators around the world. Python in conjunction with NLTK is the most powerful toolkit for Natural Language Processing.

Built on top of NLTK and Pattern, Textblob is a library for processing textual data. It provides a simple API for diving into common NLP tasks, such as sentiment analysis, pos-tagging and noun phrase extraction.

3.3.3 Gensim

As an open source library for Python, Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. The target audience is the natural language processing (NLP) and information retrieval (IR) community (11). This library is designed to be suitable for large text data, only in in-memory processing.

Gensim is designed to be used with raw and unstructured digital textual data. Gensim implement algorithms such as Dirichlet processes by hierarchy (HDP), Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA), as well as tf-idf, random projections, Word2Vec and Document2Vec as well as supports re-rendering patterns words in set of documents (often referred to as Corpus).

4 IMPLEMENTATION

The Twitter platform provides access to its data via Twitter API. The Twitter API allows programmers to access some of the original Twitter cores including timeline, status updates and user information and allows developers to write and expand their applications in a complete new and creative way.

Developers can build user profiles based on usernames, content, picture profiles, and people they are following on Twitter with only a few requests to RESTful API. In summary, the Twitter API provides programmers with many integrated methods to communicate with Twitter.

4.1 Working with Twitter API

To get started, the following steps need to be done:

- Signing in Twitter.
- Using the Twitter account, in Developer Access, creating an application that will generate the API credentials to access Twitter from Python.
- Importing the Tweepy package.

The dashboard in FIGURE 4 allows the user to view, edit, create as well as generate access tokens for the application.

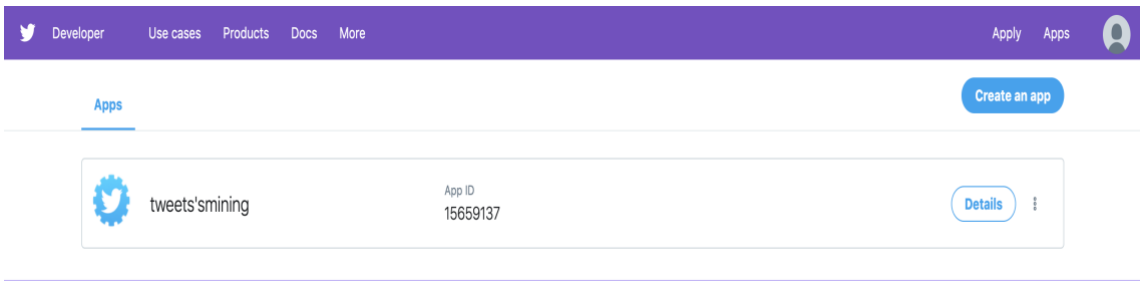


FIGURE 4. Create application in Twitter

To access the Twitter API, there are 4 different keys needed to be generated from the Twitter Dashboard page. These keys are located in application settings after clicking on the Keys and Access Tokens tab.

- Consumer key
- Consumer secret key

- Access token key
- Access token secret key

This can be seen clearly in FIGURE 5.

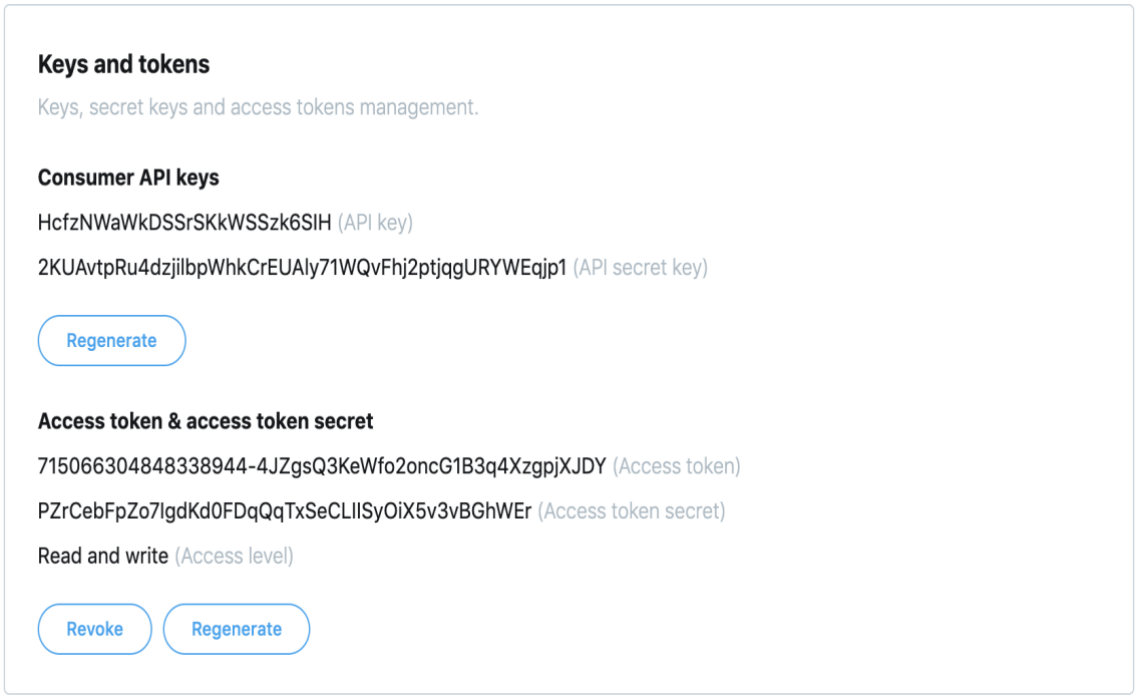


FIGURE 5. Keys and Access Tokens for the application

FIGURE 6 shows that the keys obtained from the first step are stored as string variables and the packages needed for the data collection are also imported. These keys are injected to the Tweepy function in the FIGURE 7 to request user authorization from the Twitter API.

```
In [3]: # At the beginning of the process, we need to scrapping the data from Twitter through it's API
        # by package Tweepy

In [5]: # -*- coding: utf-8 -*-
        """
        Created on Sun Jul 15 02:45:39 2018
        @author: ILOOKFORME102
        """
        import tweepy
        import csv
        from tweepy import OAuthHandler

        consumer_key = 'HcfzNWaWkDSSrSKkWSSzk6SIH'
        consumer_secret = '2KUAvtpRu4dzjilbpWhkCrEUAly71WQvFhj2ptjqgURYWEqjp1'
        access_token = '715066304848338944-4JZgsQ3KeWfo2oncG1B3q4XzgpjXJDY'
        access_token_secret = 'PZrCebFpZo7IgdKd0FDqQqTxSeCLIISyOiX5v3vBGhWEr'
```

FIGURE 6. Define access token

```python
tweet_name = "realDonaldTrump"
def get_all_tweets(screen_name):
    #Twitter only allows access to a users most recent 3240 tweets with this method

    #authorize twitter, initialize tweepy
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth)
```

FIGURE 7. Request user authorization from Twitter API

Twitter provides an API called “user timeline” to retrieve data from a specific user by username. It allows the progam to retrieve data from the most recent 3,200 tweets and every request sent to the server can only retrieve up to 200 tweets . Therefore, a Python loop is created to get all of 200 tweets for each request and it will only stop when the server stops returning data. The results of the entire process are printed in the terminal, which can be seen in the FIGURE 8:

```
getting tweets before 1106987911185162244
...400 tweets downloaded so far
getting tweets before 1102002565116542975
...600 tweets downloaded so far
getting tweets before 1094714204328771585
...800 tweets downloaded so far
getting tweets before 1086987568074424319
...1000 tweets downloaded so far
getting tweets before 1080616636363743233
...1200 tweets downloaded so far
getting tweets before 1073210823936495616
...1399 tweets downloaded so far
getting tweets before 1065225940836458495
...1599 tweets downloaded so far
getting tweets before 1058825551090237440
...1799 tweets downloaded so far
getting tweets before 1052888451199262724
...1998 tweets downloaded so far
getting tweets before 1047156358343282687
...2196 tweets downloaded so far
getting tweets before 1040563888125894655
...2392 tweets downloaded so far
getting tweets before 1034990940593823745
...2592 tweets downloaded so far
getting tweets before 1029332350969237503
...2792 tweets downloaded so far
getting tweets before 1022936690267176970
...2992 tweets downloaded so far
getting tweets before 1016634257857568768
...2992 tweets downloaded so far
```

FIGURE 8. Getting data from Twitter

At the present, a dataset with 3,200 rows and 7 columns equivalent to 7 features is built. The most important factors are:

- Lenght: Lenght of a tweet
- Tweets: Content of a tweet

- Date: Published date of a tweet
- Likes: Number of likes a tweet gets from followers
- Retweets: Number of retweets

All data will be saved to a Python's list (similar to an array in Javascript).
Since this data will change every time, the function is run to facilitate data processing and to make it persistent by saving all the collected data to a CSV format file: *realDonaldTrump_tweets.csv*
The first 10 rows and all features of dataset are shown in FIGURE 9:

```
In [67]: # Let's take an overview of data:
         data.head(10)
```

	Length	Tweets	Date	Source	Likes	Retweets
0	140	The First Step Act proves that our Country can...	2019-04-03 17:09:19	Twitter for iPhone	66584	17444
1	140	Congress must get together and immediately eli...	2019-04-03 13:45:03	Twitter for iPhone	88367	22880
2	140	...This will be a great campaign issue. I neve...	2019-04-03 13:37:14	Twitter for iPhone	60627	15416
3	139	I was never planning a vote prior to the 2020 ...	2019-04-03 13:26:53	Twitter for iPhone	92488	20834
4	124	Today, it was my great honor to welcome @NATO ...	2019-04-02 19:59:57	Twitter for iPhone	44017	9603
5	140	Today, we celebrate the tremendous accomplishm...	2019-04-02 19:42:48	Twitter for iPhone	59116	14176
6	140	“I haven’t seen any Democrats down here at the...	2019-04-02 14:52:09	Twitter for iPhone	116937	34460
7	23	https://t.co/jVIodsTNNH	2019-04-02 14:51:17	Twitter Media Studio	37450	11708
8	140	After many years (decades), Mexico is apprehen...	2019-04-02 14:41:50	Twitter for iPhone	96701	24188
9	140	There is no amount of testimony or document pr...	2019-04-02 12:54:11	Twitter for iPhone	88366	21057

FIGURE 9. First 10 rows from dataset

4.2 Data Visualization

From the dataset and based on the number of tweets and the time of posting, some simple statistics and visualization are represented.
By using Plotly, the distribution of tweets over time is plotted as shown in FIGURE 10.
The graph of this distribution will not be generated in the terminal of Jupyter Notebook but will display interactively through an HTML file so that the user can zoom in to see the distribution of tweets every week or every day of the month.

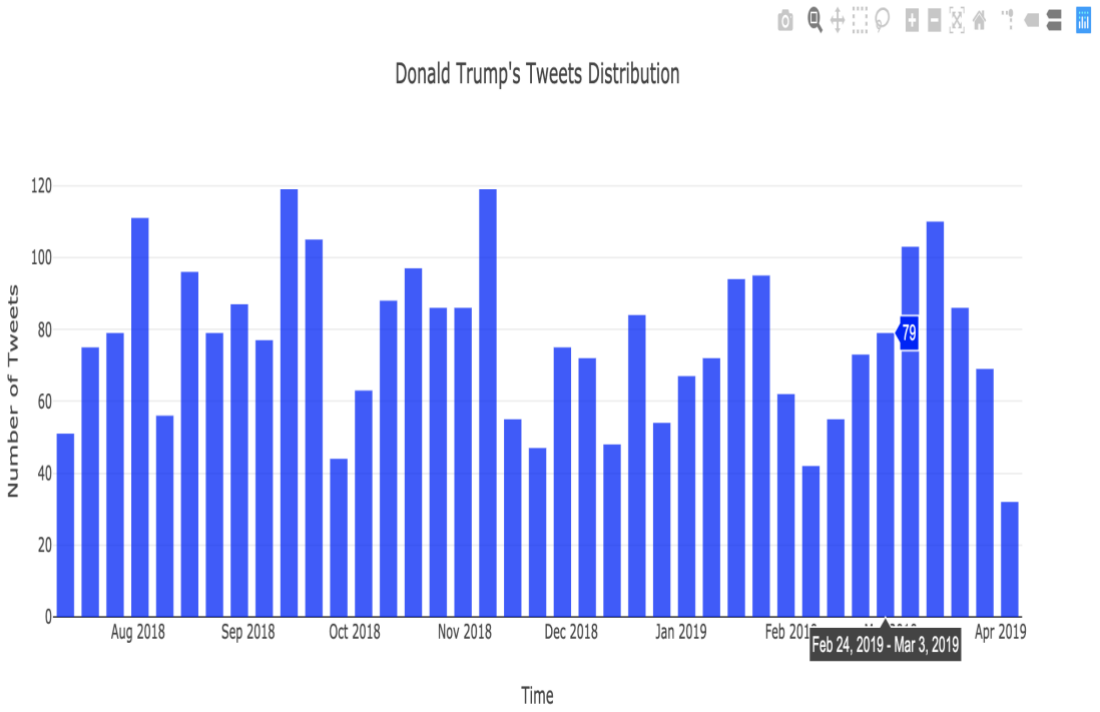


FIGURE 10. Histogram of Tweets distribulition

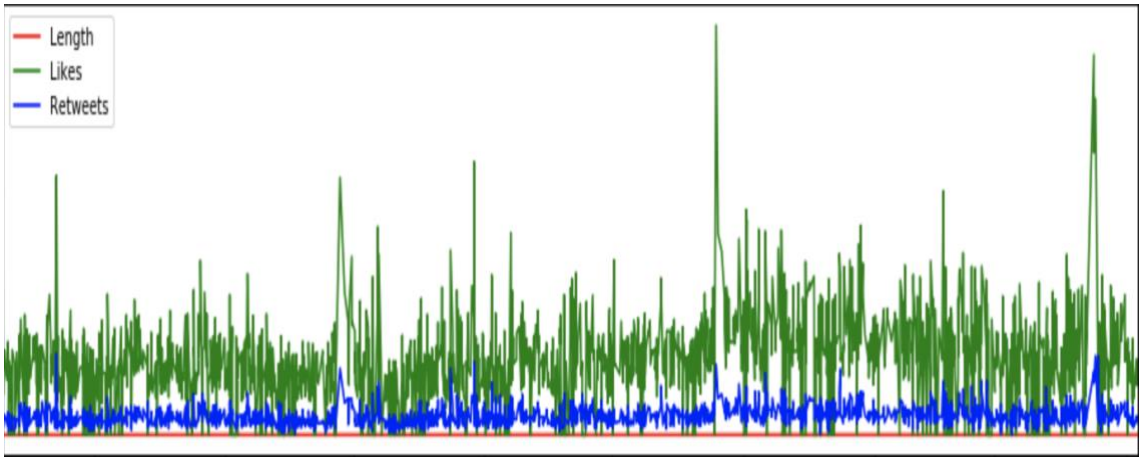


FIGURE 11. Distribution of likes and retweets

From the histogram in FIGURE 10, it can easily be seen that the number of Donald Trump's tweets fluctuates in a way that regulates the time around a certain average value. He tends to be twitting more in the middle of the month and less at the beginning and end of the month.

In another development, the time series graph of likes, retweets and length of tweets is reflected in FIGURE 11.

The number of likes and retweets from his tweets is very high and there is a consistent variation. From the graph, it can be seen that the influence of Donald Trump on this social network is

tremendous. The people who follow him on social networks are always ready to receive and spread the information that he updates daily via Twitter.

By using the Numpy library, simple but interesting statistics were discovered as follows:

- Average length of tweets: 121.61764705882354
- Most favorite tweet by number of likes: "*Merry Christmas!*"
- Length: 16 characters.
- Number of likes: 498,558
- Most favorite tweet by number of retweets: "*They just didn't get it, but they do now* "
- Number of retweets: 118,104
- Length: 86 characters.

4.3 Data Pre-processing

Before conducting data analysis, cleaning and processing data is extremely important. Since the collected data is in a text form, the aim of this process is to simply remove non-text characters such as emotion icons, HTML, URLs and redundant whitespace and convert all to a standard form for a more convenient analysis in the next step.
This can be done by using the regular expressions package of Python.
FIGURE 12 demonstrates the process of cleaning data effectively with the package RE.

```
In [72]: def clean_data(tweet):
             # using regex to clean the text by removing the link from text
             tweet = re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([
             # Remove HTML special entities (e.g. &amp;)
             tweet = re.sub(r'\&\w*;', '', tweet)
             # To lowercase
             tweet = tweet.lower()
             # Remove Punctuation and split 's, 't, 've with a space for filter
             tweet = re.sub(r'[' + string.punctuation.replace('@', '') + ']+', ' ', tweet)
             # Remove words with 2 or fewer letters
             tweet = re.sub(r'\b\w{1,2}\b', '', tweet)
             # Remove whitespace (including new line characters)
             tweet = re.sub(r'\s\s+', ' ', tweet)
             # Remove single space remaining at the front of the tweet.
             tweet = tweet.lstrip(' ')
             #Remove additional white spaces
             tweet = re.sub('[\s]+', ' ', tweet)
             tweet = re.sub('[\n]+', ' ', tweet)
             #Remove not alphanumeric symbols white spaces
             tweet = re.sub(r'[^\w]', ' ', tweet)
             #Replace #word with word
             tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
             #Remove :( or :)
             tweet = tweet.replace(':)','')
             tweet = tweet.replace(':(','')
             #trim
             tweet = tweet.strip('\'"')
             tweet = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())
             return tweet
         data['clean_tweet'] = data.Tweets.apply(clean_data)
```

FIGURE 12. Data cleaning using regular expression.

Another important step before doing a sentiment analysis is to eliminate stopwords and punctuation. By definition, a stop word is a commonly used word such as “the”, “a”, “an”, “in” that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query (12). The removal of these words does not only changes the semantics of the document but also frees up space and reduces the database processing time.

NLTK makes this step more effective by providing a list of stopwords as well as punctuation for any language.

As pointed out in FIGURE 13, the list of stopwords as well as punctuation in English is stored in a variable called “stoplist”. The function loops through the entire data, filtering out words that are not present in the stoplist to get the final valid words.

```
In [86]:
#cleaning da
# remove common words and tokenize
list1 = ['RT','rt']
stoplist = stopwords.words('english') + list(punctuation) + list1

texts = [[word for word in str(document).lower().split() if word not in stoplist] for document in corpus]
dictionary = corpora.Dictionary(texts)
# store the Trump dictionary
dictionary.save(os.path.join(TEMP_FOLDER, 'trump.dict'))
corpus = [dictionary.doc2bow(text) for text in texts]
corpora.MmCorpus.serialize(os.path.join(TEMP_FOLDER, 'trump.mm'), corpus)
```

FIGURE 13. Eliminate stopwords and punctuation.

4.4 Data Analysis

4.4.1 Sentiment Analysis with Textblob

As introduced briefly in the theory section, the thesis will conduct a lexicon-based method for a sentiment analysis using the Textblob library (shown in FIGURE 14).

The sentiment score calculation for each tweet is done by Textblob and only output is classified as follows:

- If a sentiment score is greater than 0, assigning it to a value 1 also understands that the sentence is positive.
- If a sentiment score is equal to 0, keep the value and it is semantically neutral
- In case this number is smaller than zero, assign it to -1 and it is negative in semantics.

Notice the values of 1.0 and -1 are assigned to the classification but meaningless in terms of arithmetic.

```
def sentiment_tweets(tweet):
    model = TextBlob(tweet)
    if model.sentiment.polarity > 0:
        return 1
    elif model.sentiment.polarity == 0:
        return 0
    else:
        return -1

data['Sentiment']= np.array([sentiment_tweets(tweet) for tweet in data['clean_tweet']])
display(data.head(10))

#anlysing the result of sentiment analysis:
pos_tweets = [ tweet for index, tweet in enumerate(data['clean_tweet']) if data['Sentiment'][index] > 0]
neu_tweets = [ tweet for index, tweet in enumerate(data['clean_tweet']) if data['Sentiment'][index] == 0]
neg_tweets = [ tweet for index, tweet in enumerate(data['clean_tweet']) if data['Sentiment'][index] < 0]

print("percentage of positive tweets: {}%".format(len(pos_tweets)*100/len(data['clean_tweet'])))
print("Percentage of neutral tweets: {}%".format(len(neu_tweets)*100/len(data['clean_tweet'])))
print("Percentage de negative tweets: {}%".format(len(neg_tweets)*100/len(data['clean_tweet'])))
```

FIGURE 14. Sentiment analysis by using Textblob.

The final result shows:

- Percentage of positive tweets: 51.036096256684495%
- Percentage of neutral tweets: 30.84893048128342%
- Percentage de negative tweets: 18.114973262032084%

Only 18% of Donald Trump's tweets are negative. This number is not bad for a controversial president like Donald Trump. Objectively, it can be said that Donald Trump has a tendency to send positive and objective messages to the followers.

4.4.2 Topic Modeling with Gensim

The aim of the LDA model is “learning” a topic mix in each document and a word mix in each topic so that the input data must be a Document-Term Matrix which includes the vector of the vocabulary and the frequency of each word in a document.

By definition, the Document-Term Matrix or Term-Document Matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents. In a document-term matrix, rows correspond to documents in the collection and columns correspond to terms.
The main inputs of the process of topic modeling is the Document-Term Matrix, which can be prepared by Gensim from the beginning as seen in FIGURE 15:

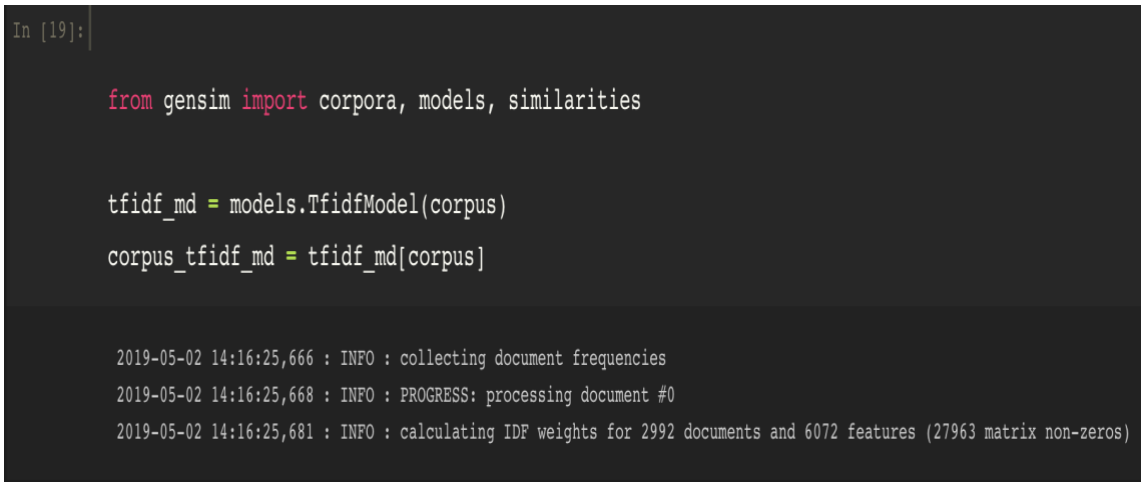


FIGURE 15. Document-Term Matrax is input of LDA model.

All materials required to train the LDA model have been prepared. Apart from that, alpha and beta are hyperparameters that affect the sparsity of the topics. According to the Gensim document, the default value of these parameters is equal to 1.0/*num_topics* (13). Variable *num_topics* is set equal to 5 which is evident in FIGURE 16.

```
#main part of LDA topic modeling:
number_topics = 5
lda_model = models.LdaModel(corpus, id2word=dictionary, num_topics=number_topics)
corpus_lda = lda_model[corpus_tfidf_md]
#Show first n important word in the topics:
lda_model.show_topics(number_topics,5)
from collections import OrderedDict

data_lda = {i: OrderedDict(lda_model.show_topic(i,25)) for i in range(number_topics)}
```

FIGURE 16. Build LDA model.

The ouput of the whole process is the top 10 words in each topic. The human work in this process is to adjust the parameters until results make sense. If not, this process is repeated again and the parameters and number of topics are tried to be altered.

As a result of the LDA model from the previous step, keywords of the topics are shown in FIGURE 17:

```
2019-05-03 12:39:31,764 : INFO : topic #0 (0.200): 0.028*"great" + 0.012*"thank" + 0.012*"border" + 0.008*"america" + 0.008*"president" + 0.007*"realdonaldtrump" + 0.007*"get" + 0.006*"people" + 0.006*"wall" + 0.006*"trump"
2019-05-03 12:39:31,766 : INFO : topic #1 (0.200): 0.009*"people" + 0.009*"great" + 0.008*"country" + 0.007*"fake" + 0.005*"news" + 0.005*"democrats" + 0.005*"new" + 0.005*"many" + 0.005*"media" + 0.005*"really"
2019-05-03 12:39:31,768 : INFO : topic #2 (0.200): 0.015*"president" + 0.010*"realdonaldtrump" + 0.010*"trump" + 0.006*"great" + 0.005*"time" + 0.005*"thank" + 0.005*"first" + 0.004*"new" + 0.004*"wall" + 0.004*"years"
2019-05-03 12:39:31,771 : INFO : topic #3 (0.200): 0.012*"people" + 0.010*"great" + 0.010*"today" + 0.009*"border" + 0.007*"vote" + 0.006*"president" + 0.005*"big" + 0.004*"state" + 0.004*"country" + 0.004*"china"
2019-05-03 12:39:31,773 : INFO : topic #4 (0.200): 0.014*"democrats" + 0.010*"border" + 0.007*"great" + 0.007*"new" + 0.007*"trump" + 0.006*"security" + 0.006*"realdonaldtrump" + 0.005*"one" + 0.005*"president" + 0.005*"news"
```

FIGURE 17. Topic and its keywords.

These keywords characterized the topic 0 and corresponding to each keyword is its weight. This number shows the importance of each word for the topic. The bigger the weight, the higher the importance of the keyword with the topic.

The weight reflects how important a keyword is to that topic thus in this case “trump” is the most important keyword. Data analyses can base on keywords to predict the main topic for all these

keywords. In this case, the topic may be "election" or " presidency" or more generally "political". Likewise, the same process is conducted with the remaining topics.

The FIGURE 17 shows the wordcloud of keywords in each topic.

FIGURE 18. Wordcloud of topic's keywords.

There are 5 different colors to distinguish the keywords of 5 topics. The size of the keywords in each topic is also different. High-level keywords are represented with larger sizes and deeper colors.

Visualizing topics and keywords in the form of word-cloud only shows qualitative, to understand the importance and quantitative information for these topics, PyLDAvis package was used in this phase.

PyLDAvis is a dedicated package for topic model visualization. PyLDAvis is designed to help users interpret the topics in a topic model that has been fit to a corpus of text data. Hereby, the distribution of topics was visualized in a more intuitive way.

As shown in FIGURE 19, the package extracts information from a fitted LDA topic model to an interactive web-based visualization one.

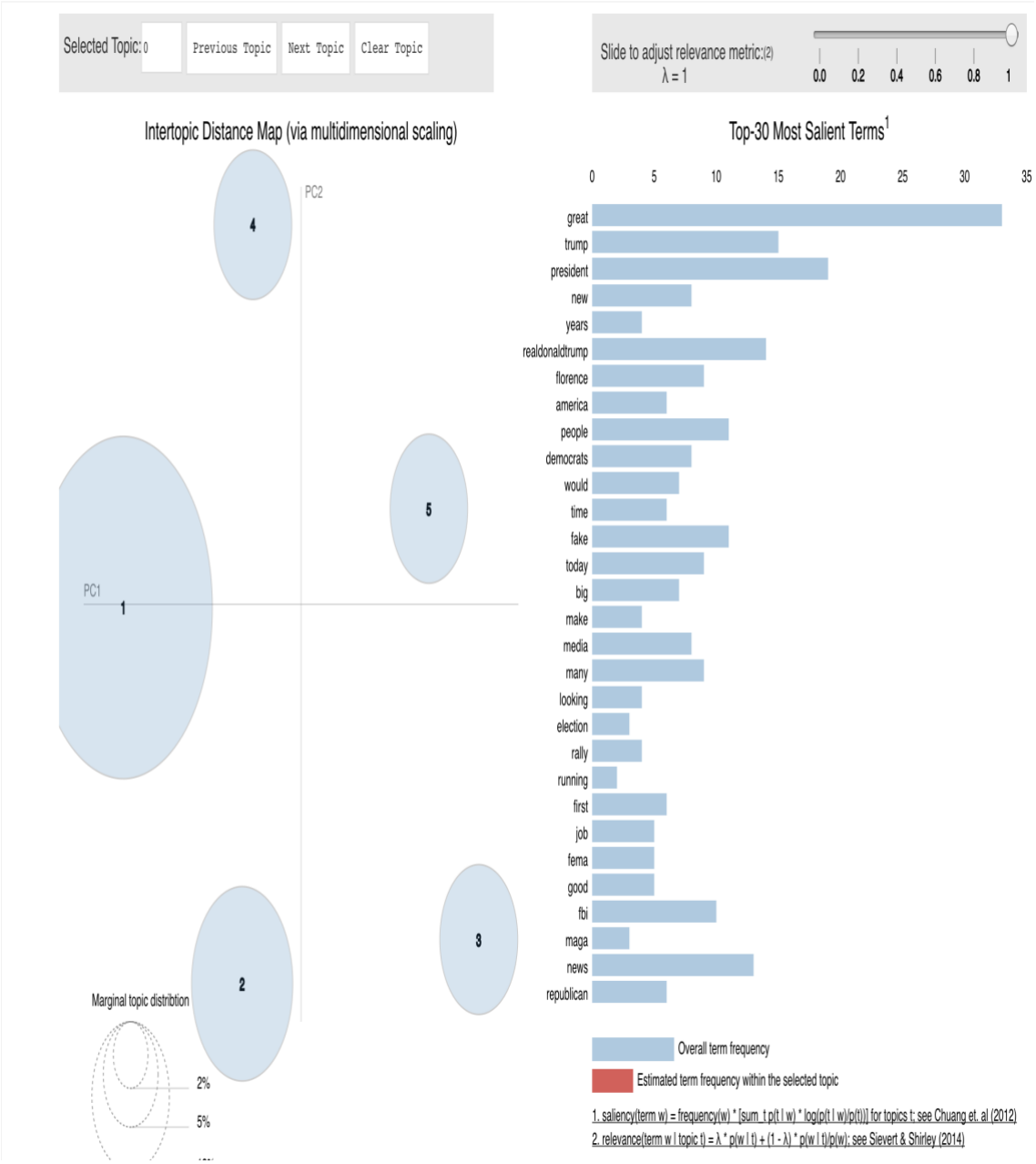


FIGURE 19. Topics distribution.

The parameters on the graph can be explained as follows:

- Saliency: a measure of how much the term tells you about the topic.
- Relevance: a weighted average of the probability of the word given the topic and the word given the topic normalized by the probability of the topic.
- The size of the bubble measures the importance of the topics, relative to the data.

It can be seen that the first topic has the most importance, the remaining topics have the same importance level.

Based on the keywords synthesized, it is assumed that the topics created are as follows:

Topic 1: border security

Topic 2: trump and medias

Topic 3: trade war

Topic 4: political parties

Topic 5: election

5 CONCLUSION

During the completion of this project, the author start to understand the whole process of a data mining project in practice. As usual, when participating in online courses, the author only needed to learn the theory of machine learning models and apply algorithms with Python to produce the final results. In fact, to solve a problem one must go through a lot of stages. Fortunately, Twitter provides an open API for developers to be able to collect data for building applications based on their data.

After taking the data in hand, the next step is pre-processing and cleaning the data. This stage actually occupies a considerable amount of work and time. The final work is to apply analytical models to existing data to produce the desired results.

The author's approach to this subject is not based on machine learning models but a statistical interference and parameter estimation because it requires the author to have a lot of knowledge about statistics and probability to understand how models work and to apply models in practice.
The progress and results of the project are clearly stated in the previous steps. The author has found positive or negative emotional attitudes in President Donald Trump's tweets and analyzed them further to find the key topics in this data.

Things that have been done from this project are still limited and in fact, it has a lot of potentials. In the future, the project can continue by conducting real-time data processing and deploying in the form of a web application for easier access for ordinary Internet users.

REFERENCES

1. Wikipedia. Data Mining. Date of retrieval 01.01.2019
https://en.wikipedia.org/wiki/Data_mining

2. Wikipedia. Sentiment Analysis. Date of retrieval 10.01.2019
https://en.wikipedia.org/wiki/Sentiment_analysis

3. Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, Manfred Stede. 2011. Lexicon-Based Methods for Sentiment Analysis. Association for Computational Linguistics, 268. Date of retrieval 10.01.2019
https://dl.acm.org/citation.cfm?id=2000518

4. Soumya Ghosh. 2018. Topic modelling with Latent Dirichlet Allocation (LDA) in Pyspark. Date of retrieval 20.01.2019
https://medium.com/@connectwithghosh/topic-modelling-with-latent-dirichlet-allocation-lda-in-pyspark-2cb3ebd5678e

5. Yuanchao Liu, Ming Liu, Xin Wang. 2015. Towards Semantically Sensitive TextClustering: A Feature Space ModelingTechnology Based on Dimension Extension. Date of retrieval 22.01.2019
https://www.researchgate.net/figure/The-LDA-model_fig1_273777934

6. What is Python. Date of retrieval 10.11.2018
https://opensource.com/resources/python

7. Matplotlib Documentation. Date of retrieval 10.11.2018
https://matplotlib.org/index.html

8. Official Seaborn Tutorial. Date of retrieval 10.02.2019
https://seaborn.pydata.org/index.html

9. Plotly Python Open Source Graphing Library. Date of retrieval 15.02.2019
https://plot.ly/python/

10. Natural Language Toolkits. Date of retrieval 19.02.2019
https://www.nltk.org/

11. Gensim - Project description. Date of retrieval 11.03.2019
https://pypi.org/project/gensim/

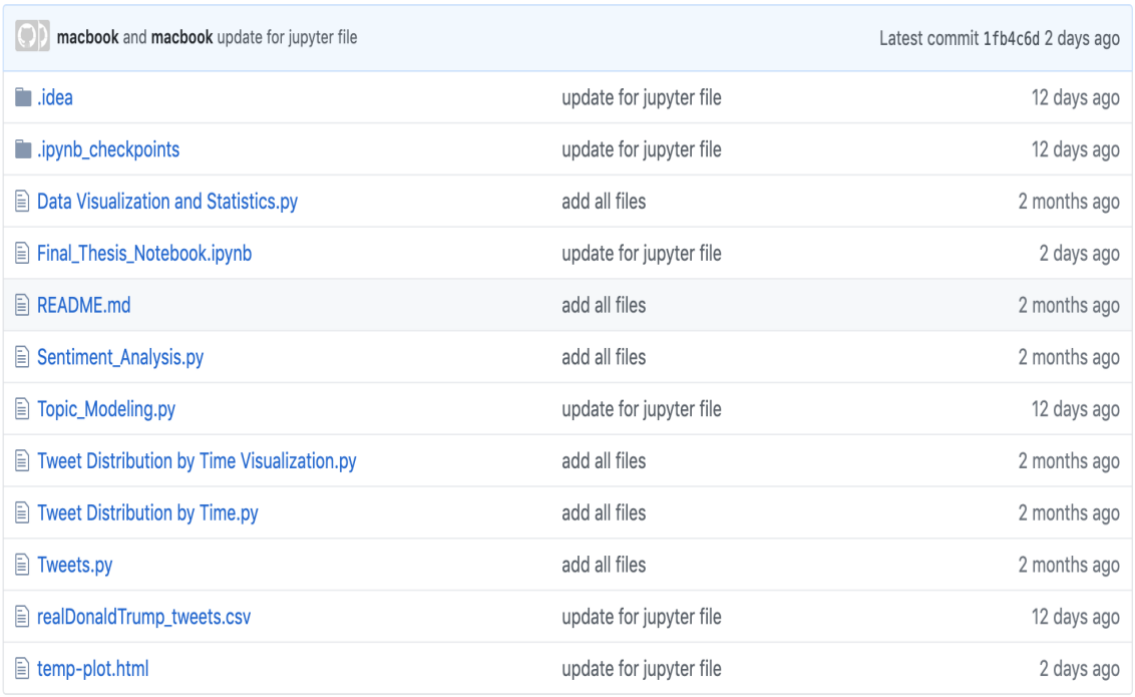
12. Removing stopwords with NLTK in Python. Date of retrieval 20.03.2019
https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

13. Gensim Documentation. Date of retrieval 14.03.2019
https://www.pydoc.io/pypi/gensim-3.2.0/autoapi/models/ldamulticore/index.html

APPENDIX

APPENDIX 1

The folder structure of the project.



macbook and macbook update for jupyter file		Latest commit 1fb4c6d 2 days ago
.idea	update for jupyter file	12 days ago
.ipynb_checkpoints	update for jupyter file	12 days ago
Data Visualization and Statistics.py	add all files	2 months ago
Final_Thesis_Notebook.ipynb	update for jupyter file	2 days ago
README.md	add all files	2 months ago
Sentiment_Analysis.py	add all files	2 months ago
Topic_Modeling.py	update for jupyter file	12 days ago
Tweet Distribution by Time Visualization.py	add all files	2 months ago
Tweet Distribution by Time.py	add all files	2 months ago
Tweets.py	add all files	2 months ago
realDonaldTrump_tweets.csv	update for jupyter file	12 days ago
temp-plot.html	update for jupyter file	2 days ago

The main part of the project was included in Jupyter Notebook file "*Final_Thesis_Notebook.ipynb*" Data was stored in "*realDonaldTrump_tweet.csv*"