Topic Monitoring in the Pharmaceutical Industry

Master Team Project

presented by
Hailian Hou (123456789)
Chia-Chien Hung (123456789)
Lu Lifei (123456789)
Olga Pogorelaya (123456789)
Ngoc Nam Trung Nguyen (123456789)
Md. Raziul Hasan Al Tariq (123456789)
Alexander Weiß (123456789)

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

August 2017

Contents

List of Algorithms

List of Figures

List of Tables

Listings

Acronyms

API	Application Programming Interface.
HTTP	Hypertext Transfer Protocol.
REST	Representational State Transfer.
SA	Sentiment Analysis.

Chapter 1

Introduction

Today in 2017 social media is ubiquitous and is taken for granted in our day-to-day life. From single shops around the corner to large companies own social media accounts to communicate with customers or try to engage new ones. The number of active users on the biggest social media platforms is tremendously high. On Facebook we have around 1.871 million active users as of January 2017. Whatsapp and Facebook messenger share the number of 1.000 million active users. For Twitter in comparison to Facebook, the numbers may seem small but with 317 million active users it belongs to the biggest social media platform of our time. [2]
More than ever people share their thoughts about current global events like conflicts between different countries, political discussions, terrorism, rising diseases as well as the opinions to companies and products. For companies nowadays it is easier than ever to find out what customer think and talk about them and their products. With Twitter we have one the highest reactive social media platforms of our time. Almost instantly people write messages (tweet) to react accordingly to new events in the world. Due to this fact companies try to effectively monitor Twitter and other social media platforms to get insights into the current customer situation regarding their product and company reputation. In the course of the Master Team Project - Topic monitoring in the pharmaceutical industry by the University of Mannheim , this work especially focuses on the monitoring of social media for the pharmaceutical context. It is processed in cooperation with AbbVie Inc. located in Ludwigshafen, Germany, a pharmaceutical company focused on both biopharmaceuticals and small molecule drugs. The goal of this project is to find efficient and effective ways to retrieve data from different social media platforms to analyse sentiments, public opinions, emotions towards different events, find suitable topics and detect rising trends in social media. To accomplish this task, this work utilizes

different machine learning algorithms to detect the requested features in the messages of users. The declared aim is to find a good combination of those algorithms to provide reliable results and findings so that especially AbbVie can use and incorporate them into their attempt to monitor social media.
This report describes how to gather data from different social media platforms and what restrictions will apply to them. It focuses on Facebook and Twitter, because this selection covers the most used and the most reactive social media platform today. Furthermore it deals with different machine learning algorithms, explains how they could be applied to different use cases and suggests situation when and how to use them.

1.1 Problem Statement

1.2 Contribution

1.3 Related Work

Chapter 2

Theoretical Framework

At first we want to provide an explanation to some fundamental terms which will be used in this report. It should give an overview what the different terms mean and in which context they are used.

2.1 Social Media Data

The term *social media* refers to all web application which are built on the technological foundations of the Web 2.0. In comparison to the beginning of the web, Web 2.0 is characterized by a huge improvement of communication channels between users and increased communications between those. Social media platforms heavily rely on this progressive development of user interaction. They offer a place where users can collaborate and interact with each other, share and create content with ease. Unlike traditional static websites, the content of social media platforms is nearly entirely made by the users, which lead to a very reactive and accurate picture of the community who is using the network. These platforms can be divided in different types. The following shows a non-exhaustive list of those. [12][13]

- Forums (Reddit)
- Microblogging services (Twitter)
- Social networks (Facebook, Google+)
- Wikis

- Live casting (YouTube, Twitch)

- Video sharing (YouTube, Twitch)

Social media is an integral part of everyones life and a world without it is nearly no longer possible. For companies it offers a broad variety of possibilities to engage and get to know their customers better. Social media analytics helps to support business decisions, mostly based on the customer sentiment, which makes it a great datasource to analyze data. [12]

In this work we will mainly referring to the term social media in the context of two platforms, Twitter and Facebook, as those are building the base for our data gathering process.

2.2 Sentiment Analysis

Sentiment Analysis (SA) belongs to the broad field of data mining techniques. With the help of Natural Language Processing and text analytics subjective information are extracted and quantified from texts and messages, mostly on social media sources. The results of those techniques can quantify the sentiment of people's opinions and feelings towards a certain subject of the text. Mostly those are products, companies, other people or current events. Sometimes sentiment analysis is also known as opinion mining or emotion AI. With the help of SA companies can detect subjective key values for their organization and implement measures which take advantage of these findings. Some of these values are the brand reputation along with the popularity of it, the company reputation overall and new product perception. This is essentially useful when new products are introduced and the acceptance of those needs to be measured as fast as possible. [15]

In this report sentiment analysis is used to determine the opinions of people regarding the pharmaceutical industry. It tries to apply suitable data mining algorithms to quantify the opinions regarding drugs, diseases and companies which are active in this sector.

2.3 Topic Detection

Topic Detection copes with the problem of identifying topics to unseen or unidentified events. It can be applied to a static collection of data ("retrospective detection")

or even on real-time news (”on-line detection”) feed from different sources. To detect topics miscellaneous data mining algorithms can be used. They all are based on using unlabeled data to train them. Unlabeled data means, that it is non curated by human annotators which detect topics beforehand. The algorithms just take raw data and try to cluster them into different groups (topics) based on common attributes among different data objects. [14]

Topic Detection within this work is primarily used on real-time news. It is first trained on pre-crawled data from social media platforms but later applied to the continuously stream of new data.

2.4 Trend Detection

Trend Detection techniques try to find out what is suddenly increasing in popularity. This is especially applicable to highly reactive social media platforms. For some types organizations it is tremendously important to know what are trends among the people, especially for news organizations, governments and retailers. At most of the time a trend is coupled with a specific topic. If it possible to detect trending topics, organization can recognize what people think is noteworthy and can react to those events appropriately. [9]

Chapter 3

Methodology

Here we tell you something

3.1 Data Collection

The first step we take is to collect data from related social media platforms, in our case Facebook and Twitter. We utilize the Application Programming Interface (API)s of those to retrieve posts, comments (Facebook) and tweets (Twitter) from users. This is done simultaneously for both Facebook and Twitter. We constructed a list of keywords related to pharmaceuticals industry, companies, diseases and products as ”anchors” for the crawlers, to collect suitable and appropriate data. These keywords were used for both APIs.

Products	Companies	Diseases
adalimumab	abbvie	ankylosing spondylitis
humira	amgen	arthritis
enbrel	johnson&johnson	hepatitis
ibrutinib		psoriasis
		rheumatoid arthritis
		trilipix

Table 3.1: Anchor keywords for collecting data

3.1.1 Facebook API

Overview

Like every major web based platform, Facebook offers an Application Programming Interface (API) to programmatically interact with its data. This allows developers to build applications which can utilize the social connection and profile information to make them more involving. Furthermore they have access to public data on Facebook as well as have the opportunity to publish posts and messages to the news feed. Officially the API is called Graph API, because it is follows the concept of a Social Graph [4]. It consists of three key objects:

1. Nodes
    - Describe all things that are shown on the Facebook web page: User Pages, Posts, Comments and Images
2. Edges
    - Describe the relation between these things. For example the comments to an Image posted on a Page
3. Fields
    - Contain additional information to nodes. For example the name of pages or the relationship status of users

The Graph API is based on the simple Hypertext Transfer Protocol (HTTP) and can therefore work with every programming language which implements a HTTP library. Due to this fact it is very easy to implement the API into an application. It supports common request methods like, GET for retrieving information, POST for uploading data and DELETE for deleting data. With this report we exclusively use GET requests to get data from the Graph, as we do not want modify data on Facebook.

Restrictions

Facebook offers a highly efficient and very good documented API for getting data from the social network. So it would be actual a very good data source for social media monitoring, but unfortunately there are some restrictions which apply. In the following the most serious ones are listed.

1. Data scraping terms of service

- Facebook has a dedicated terms of service document for collecting data from Facebook through automated means. Before any automatic data collection takes place, a written permission has to be obtained by Facebook. Without this permission it is not allowed to scrape any data. Also other restrictions like not selling, or transferring the collected data apply at any given time. So it is absolutely necessary to read through these terms of services and contact Facebook before choosing it as a reliable datasource. [3]

2. Streaming API

   - Facebook API offers no streaming endpoint, so no real time feed about new content on the platform can be obtained programmatically. It is up to the developer to create processes to constantly crawl for updated data on Facebook with static endpoints.

3. Rate limits

   - The Graph API has some rate limits which will apply if too many requests are sent to it. During this work the rate limit only appeared once while a lot of pages for one keyword were crawled. So we assume that the rate limitation will not apply many times while using the API. [5]

4. No access to public posts

   - On April 30th, 2015 made a breaking change to their search API endpoint. By then it was possible to search for public posts on Facebook through `/search?type=post&q=foobar`. After this day the endpoint was deprecated and exclusively available for approved companies. From then on, it was not possible to get any public user post. With this change in the API, it lost the most promising endpoint for companies for monitoring the network. Without the opportunity to crawl public posts from users the only available reliable source for getting posts is the page endpoint. With this it is possible to search the pages directory regarding specific keywords and retrieve posts from them. In fact, a company could try to apply for a license to get access to the post search endpoint, but it seems that this is only available to a limited set of media publishers. [6]

Workflow

The restrictions explained in section 3.1.1 lead us to nearly drop Facebook from our workflow, but we tried to find a suitable way to get the most out of the remaining available endpoints. To overcome the deprecation of the post search endpoint, we tried to get as much information out of the pages search endpoint as we could. This serves the base endpoint for our work. It offers the opportunity to specify keywords which the pages need to match to be returned by the API. We used the keywords from Table 3.1 to crawl the pages directory. Afterwards it was necessary to restrict the retrieved pages to only those which were in a pharmaceutical context. Then the posts and comments from the pages can be crawled and saved for later work.

One major drawback of this approach is that the amount of pages and posts on these pages are very low. Only a few conversation about adverse drug reaction, opinions about drugs and companies take place on them. Furthermore the conversation inside those pages seems to be very limited and biased related to the keyword of the page. So this approach may not lead to a good overview about opinions on different keywords but for us it was the most suitable way to get data from Facebook.

The dataset

Conclusion

At first Facebook seemed to be a good and reliable data source for this project. But quickly it became clear that this is not the case. With the restriction of the public post search it is impossible to get all user posts about a keyword. Its only possible to get posts of pages and the corresponding comments to them. But if often occurs that posts on pages are very biased regarding the keyword of the page. Also a lot of pages in the pharmaceutical context seem to not get updated very often and some seem to be very abandoned. So the breaking point here definitely is the depreciation of the API endpoint for searching posts. This leads to the fact, that Facebook can not be used as a suitable data source for topic monitoring social media. Therefore we dropped the Facebook dataset out of the calculation and algorithms.

3.1.2 Twitter API

Overview

Like Facebook, Twitter also offers APIs to programmatically access data. They are divided into two different parts. One part are static Representational State Transfer (REST) conform ones which use simpleHTTP requests to create new tweets, read user profile, search for tweets and more.

The second part are dynamic APIs, better known as the Streaming API. They give applications access to the global real-time stream of Twitter data, without any overhead or by pulling data from the static API endpoints. In total Twitter offers three different streaming endpoints:

1. Public stream: For public Twitter data
2. User stream: For data associated with only one user
3. Site stream: Multi user version of user streams

This streaming endpoints differ significantly from the static endpoints. They are keeping a persistent HTTP connection open between client and server. The server then is streaming tweets as they occur and the client is responsible for processing and storing the result. This resolves into a realtime stream of twitter data which makes any application working with this model reactive and everytime up-to-date.[16]

Restriction

Unlike Facebook, Twitter does not have that many restrictions we have to respect in our use case.

1. Date restriction in the search API (Static REST API)
    - When you want to use the static search endpoint, where you can search for tweets with specific keywords only the latest two weeks from the date of the request are available to search. So it is generally impossible to search for historical tweets through the API. There are some companies like GNIP [7], which are offering service to get access to historical data. To get access to this data you have to contact those companies to get a custom historical dataset, but this option was not suitable for this project as it would had involve payments and our goal is to avoid that.
2. Requests per minute (Static REST API)

- The REST endpoints, in contrast to the streaming endpoints are rate limited. So developers have to take care of the amount of requests that are sent to the API. A full overview about the different limits can be found at `https://dev.twitter.com/rest/public/rate-limits` [17]

3. Concurrent streaming APIs (Streaming API)

   - Any application which is using any of the different streaming APIs can only open only one stream to Twitter. This have to be kept in the mind if a developer wants to use multiple streaming endpoint.

Workflow

While getting to know the API better we decided to utilize the public streaming API to get the tweets for specific keywords mentioned in Table 3.1. This endpoint accepts a filter value, which defines what Tweets we will receive from the API. The keywords for that filter can either be separated by a comma, which will work as an **OR** concatenation or separated by an white space, which will work as an **AND** concatenation. An example request to the endpoint could look like the following:

```
1 https://stream.twitter.com/1.1/statuses/filter.json?&
    track=abbvie,humira,adalimumab...
```

Listing 3.1: Twitter streaming endpoint

This leads very convenient way to crawl data from Twitter only for specific topics. Another advantage of using the streaming API is that rate limits will not apply to them. Therefore is possible to receive a arbitrary number of tweets from it without getting rate limited, which is especially useful if peaks of tweets appear at certain events. The only restriction which applies is that it is not possible to open multiple data streams to the stream endpoint at a given time, but in our case this is not necessary as only the public one is used. Furthermore the response from this endpoint is very comprehensive so it is not necessary to get data from the static endpoints, which will benefit in not going to reach any rate limits. A truncated example response can be seen Listing 3.2.

```
1 { created_at: 'Mon Aug 21 13:17:14 +0000 2017',
2   id: 899621415120494600,
3   id_str: '899621415120494597',
```

```
4  text: 'RT @NRAS_UK: Rheumatoid Arthritis: What You
       Need to Know. Read article by @Treated_com here:
       https://t.co/Z036Mzepee https://t.co/bIh4kREAIi',
5  source: '<a href="http://twitter.com/#!/download/
       ipad" rel="nofollow">Twitter for iPad</a>',
6  truncated: false,
7  in_reply_to_status_id: null,
8  in_reply_to_status_id_str: null,
9  in_reply_to_user_id: null,
10 in_reply_to_user_id_str: null,
11 in_reply_to_screen_name: null,
12 user: [Object],
13 geo: null,
14 coordinates: null,
15 place: null,
16 contributors: null,
17 retweeted_status: [Object], // !null if this is a
       retweet, contains information about original
       tweet
18 is_quote_status: false,
19 retweet_count: 0,
20 favorite_count: 0,
21 entities:
22  { hashtags: [],
23    urls: [ [Object] ],
24    user_mentions: [ [Object], [Object] ],
25    symbols: [],
26    media: [ [Object] ] },
27 extended_entities: { media: [ [Object] ] },
28 favorited: false,
29 retweeted: false,
30 possibly_sensitive: false,
31 filter_level: 'low',
32 lang: 'en',
33 timestamp_ms: '1503321434179' }
```

Listing 3.2: Example Twitter API response

The dataset

As of August 20th 2017 we stopped crawling data from the Twitter API. Our base dataset for Twitter then consists of 2 types of objects: Tweets and the authors of those. The most important one of these two is the tweet object with it's message field. The message is the fundamental part which is used to build models for sentiment analysis and topic detection. It contains the the tweet message along with URLs, Emojis and hashtags. The author object was only crawled for the sake of completeness and in case if it will be needed in some circumstances. The definition of the tweet object can be seen in Listing 3.3.

```
id: {
      type: Sequelize.STRING,
      primaryKey: true,
  },
  keywordType: {
      type: Sequelize.STRING
  },
  keyword: {
      type: Sequelize.STRING
  },
  created: { // when was the tweet created
      type: Sequelize.DATE
  },
  createdWeek: {
      type: Sequelize.INTEGER
  },
  toUser: {
      type: Sequelize.STRING
  },
  language: {
      type: Sequelize.STRING
  },
  source: {
      type: Sequelize.STRING
  },
  message: {
      type: Sequelize.STRING
  },
  hashtags: { //hashtags inside the message
```

```
30          type: Sequelize.STRING
31      },
32      latitude: { // latitude of the tweet author
33          type: Sequelize.STRING
34      },
35      longitude: { // longitude of the tweet author
36          type: Sequelize.STRING
37      },
38      retweetCount: {
39          type: Sequelize.INTEGER
40      },
41      favorited: {
42          type: Sequelize.BOOLEAN
43      },
44      favoriteCount: {
45          type: Sequelize.INTEGER
46      },
47      isRetweet: { // is the tweet a retweet
48          type: Sequelize.BOOLEAN
49      },
50      retweeted: { // was this tweet retweeted
51          type: Sequelize.INTEGER
52      }
```

Listing 3.3: Tweet object definition for the database

Type	Amount
Tweets	250732
Retweets	27631
Answer to tweets	597
User	48555

Table 3.2: Common numbers of the Twitter dataset

Conclusion

After we had this massive disappointment with the Facebook dataset, Twitter is by far the better source for building up a dataset for data mining. It offers the oppor-

Keyword	Amount	Proportion in %
johnson & johnson	70240	28.0140
psoriasis	46070	18.3742
hepatitis c	35089	13.9946
rheumatoid arthritis	28940	11.5422
amgen	21404	8.5366
abbvie	14149	5.6431
bristol myers	13355	5.3264
johnson&johnson	5641	2.2498
arthritis	3512	1.4007
ankylosing spondylitis c	3473	1.3851
humira	3029	1.2081
ibrutinib	1851	0.7382
hepatitis	1662	0.6629
NULL	764	0.3047
enbrel	669	0.2668
adalimumab	667	0.2660
imbruvica	172	0.0686
trilipix	45	0.0179

Table 3.3: Twitter tweets per keyword

tunity to crawl tweets in real-time with no need to access the static endpoints. The API gives us access to every user tweet matching the keywords, which ensures that at most of the time only appropriate and suitable tweets are crawled. In addition to those advantages the streaming endpoint do not have an rate limitation, which makes it even easier to programmatically crawl tweets. This advantages make sure that Twitter is the primary data source for this project.

3.1.3 Technology Stack

The following description should emphasize how the tools we used to gather data from social media changed during the progress of this project.

In the first four months of the project we used corresponding R packages to access the data from Facebook and Twitter. For Facebook this was a package called *RFacebook* [1] and for Twitter the package was called *twitteR* [8]. The data was crawled manually by executing different R-Scripts which were saving their results

into *.csv* files. These files then were uploaded to *Google Drive* to have common database-like location for our data. But with the progress of this work the organization of the different files becomes unmanageable and we decided to save our data in a MySQL database. Simultaneously to that we switched our crawling stack and implemented it based on Node.js server which is directly connected to the database. With the database we had a structured place for our data which offered a better way to have access to our data than the *.csv* files. Furthermore the *.csv* files were encoding dependent which lead to a lot of issues during the initial crawl of data, the database solved this by not being affected of encodings. The implementation of the crawling processes was done with the help of different Node packages: *twitter* [10] for retrieving data from the streaming API of Twitter and *fbgraph* [11] for Facebook.

In contrast to the *twitteR* package *twitter* for Node.js is able to crawl the public stream of Twitter data. With that it was able to set up an automatic crawl mechanism which saved suitable tweets, regarding the keywords in Table 3.1, into our database. Because of that the manual data collection for Twitter was postponed and only served as a backup mechanism. A truncated (left out database saving) part of the code, which was responsible for the crawling and saving can be found in Listing 3.4

```
1  /**
2  * @function track
3  * @param {String} filters The filters which are used
       to track tweets from the Twitter API
4  * @description Starts using the Twitter API with the
       specified filters. When a new tweet is crawled it
       upserts the author data
5  * and inserts the raw tweet data into the database.
6  * @see {@link module:Connectors~TwitterAuthor}
7  * @see {@link module:Connectors~Tweet}
8  * @memberof TwitterCrawler
9  * @return {void}
10 */
11 track(filters) {
12     logger.log('info', 'Start streaming Twitter tweets
           with filters: ' + filters);
13     this.client.stream(
14         'statuses/filter', {
15             track: filters
16         },
```

```
17          stream => {
18              stream.on('data', event => {
19                  if (event.lang == 'en') {
20                          //Save event (tweet) into the
                               database
21                  }
22              });
23              stream.on('error', function (error) {
24                  logger.log('error', error);
25                  //throw error;
26              });
27          }
28      );
29  }
```

Listing 3.4: TwitterCrawler.track

As listed in section 3.1.1 Facebook does not offer a real-time stream to their data. Therefore the crawling of the Facebook data still had to be done manually by invoking methods from the *fbgraph* package on our server. The result of those manual crawls then was upserted into the database as well. A sample code of this process can be seen in Listing 3.5

```
1   /**
2  * @function searchAndSaveFBPages
3  * @param  {String} keyword Keyword for searching
       Facebook pages
4  * @description Searches for Facebook pages which match
       the keyword. It also inserts the found pages and
       their posts and comments into the database.
5  * Only pages with matching categories (.env file
       config) are saved.
6  * @see File /server/.env
7  * @memberof FacebookCrawler
8  * @returns {void}
9  */
10 searchAndSaveFBPages(keyword) {
11          logger.info("Started Facebook search");
12
13      var pages = new Array();
14
```

```
15          var searchOptions = {
16                  q: keyword,
17          type: "page",
18          fields: "name,id,feed{id,link,message,story,
               likes,comments{id,message},created_time},
               category"
19      }
20
21      graph.search(searchOptions, async (err, res) => {
22      //Iterate over results and save them into the db
23      });
24  }
```

Listing 3.5: FacebookCrawler.searchAndSaveFBPages

The update of our technology stack during the project led to a suitable and reliable way of crawling data from Twitter and Facebook. The centralised data consolidation inside the MySQL database additionally improved the workflow for our machine learning processes.

The final technology stack based on Node.js (JavaScript):

1. Node.js server running at least 8 hours a day
2. Twitter-Crawler for crawling data form the public stream API *twitter* package [10] (automatic)
3. Facebook-Crawler for crawling data from the GraphAPI *fbgraph* package [11] (manual)
4. MySQL database for saving tweets and authors (Twitter), pages,posts and comments (Facebook)

Bibliography

[1] Pablo Barbera et al. *Package "Rfacebook"*. May 25, 2017. URL: https://cran.r-project.org/web/packages/Rfacebook/Rfacebook.pdf (visited on 08/21/2017).

[2] Dave Chaffey. *Global social media research summary 2017*. Ed. by Dave Chaffey. Apr. 27, 2017. URL: http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/ (visited on 08/18/2017).

[3] Facebook. *Facebook Site Scraping Terms of Service*. Ed. by Facebook. Apr. 10, 2010. URL: https://facebook.com/apps/site_scraping_tos_terms.php?hc_location=ufi (visited on 08/19/2017).

[4] Facebook. *Graph API Overview*. Ed. by Facebook. Feb. 18, 2015. URL: https://developers.facebook.com/docs/graph-api/overview/ (visited on 08/18/2017).

[5] Facebook. *Graph API Rate Limit*. Ed. by Facebook. June 9, 2016. URL: https://developers.facebook.com/docs/graph-api/advanced/rate-limiting?locale=de_DE (visited on 08/19/2017).

[6] Facebook. *Public Feed API*. Ed. by Facebook. Sept. 6, 2013. URL: https://developers.facebook.com/docs/public_feed?locale=en_US (visited on 08/19/2017).

[7] Gnip. *Gnip - Unleash the Power of Social Media*. Ed. by Gnip. Oct. 20, 2015. URL: https://gnip.com/ (visited on 08/19/2017).

[8] Jeff Grentry. *Package "twitteR"*. July 29, 2015. URL: https://cran.r-project.org/web/packages/twitteR/twitteR.pdf (visited on 08/21/2017).

[9] Jeff Kolb. *Trend detection in social data*. Ed. by Jeff Kolb. July 16, 2015. URL: https://blog.twitter.com/official/en_us/a/2015/trend-detection-social-data.html (visited on 08/22/2017).

[10] Desmond Morris. *Node-Twitter*. Jan. 13, 2017. URL: https://github.com/desmondmorris/node-twitter (visited on 08/21/2017).

[11] Cristiano Oliveira. *Node-fbgraph*. Jan. 19, 2017. URL: https://github.com/criso/fbgraph (visited on 08/21/2017).

[12] Margaret Rouse. *Definition of Social Media*. Ed. by Margaret Rouse. Sept. 15, 2016. URL: http://whatis.techtarget.com/definition/social-media (visited on 08/21/2017).

[13] Margaret Rouse. *Definition of Web 2.0*. Ed. by Margaret Rouse. Jan. 15, 2015. URL: http://whatis.techtarget.com/definition/Web-20-or-Web-2 (visited on 08/21/2017).

[14] Young-Woo Seo and Katia Sycara. *Text Clustering for Topic Detection*. Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213, Jan. 31, 2004. URL: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1705&context=robotics (visited on 08/22/2017).

[15] Techopedia. *What is Sentiment Analysis*. Ed. by Techopedia. Jan. 21, 2014. URL: https://www.techopedia.com/definition/29695/sentiment-analysis (visited on 08/22/2017).

[16] Twitter. *Streaming API Overview*. Ed. by Twitter. May 15, 2012. URL: https://dev.twitter.com/streaming/overview (visited on 08/19/2017).

[17] Twitter. *Twitter API Rate Limits Chart*. Ed. by Twitter. Nov. 22, 2016. URL: https://dev.twitter.com/rest/public/rate-limits (visited on 08/20/2017).