Architecture and Algorithms for Filtering Tweets Based on Chosen Countries and Cities

Nemanja Igić*, Vladimir Dimitrieski*, Ivan Luković *, Slavica Aleksić*, and Milan Čeliković*
* University of Novi Sad/Faculty of Technical Sciences, 21000 Novi Sad, Serbia
{nemanjaigic, dimitrieski, ivan, slavica, milancel}@uns.ac.rs

***Abstract*— In this paper we present an algorithm for filtering Twitter data based on the tweet geographic location. Desired geographic locations are provided as a set of parameters, and different properties of a tweet are considered to determine the location. A user may also choose the number of threads and amount of memory used in filtering process. In this way, the user may fine-tune the algorithm performance. Filtered data are stored in the Hadoop distributed file system which runs on a 16 nodes cluster. Therefore, the analysis may be executed in a distributed environment. We also present system architecture which supports the filtering process and analysis of filtered data.**

I. INTRODUCTION

A social network is a structure composed of a series of social actors and a set of connections between these actors. Social actors can be individuals or organizations. Among the most popular social networks today are Google+ with 1.6 billion members, Facebook with 1.28 billion members, Twitter with 645.75 million members and Qzone with 480 million members [1]. Social networks produce large amount of data from different geographic areas on a daily basis making them a perfect platform for such analysis. In this paper, we observe data gathered from Twitter. Twitter is an online social networking service that enables users to send and read short 140-character messages called "tweets" [2]. Twitter provides a rich REpresentational State Transfer (REST) Application Programming Interface (API) allowing programmatic access to read and write Twitter data [3].

The main problems in analyzing data from social networks are that data cover a wide range of topics and there is a lack of metadata information relevant to the analysis (geographic coordinates, tags, etc.). Also, metadata is mostly user-defined, and in many cases caries inaccurate information. To provide accurate analysis, there is a need for collecting large amounts of data and providing a way to compensate for missing metadata. The main motivation behind this work is a lack of efficient algorithms for finding tweet locations when there are no geographic coordinates specified, which are specified in less than 2 percent of tweets [4]. Our goal is to implement a fast algorithm for filtering tweets based on predefined country and city parameters with acceptable accuracy. This kind of filtering may be useful for various analysis, since one will gather approximately 40 times more tweets from the specified locations rather than using only twitter geo tags, as it will be presented further in this paper.

As the amount of collected data grows, there is a need for an easy way to provide horizontal scalability [5]. We choose Hadoop for this purpose. Hadoop is a software framework for distributed storage and distributed data processing on commodity hardware clusters. It represents an implementation of the Google system for executing MapReduce (MR) programs presented in [6]. Hadoop stores data in Hadoop Distributed File System (HDFS) [7]. HDFS represents a distributed file system that has a high resistance to failures and is designed to be used on commodity hardware. This framework is based on the MapReduce programming model [8], which aims at processing large amounts of data through parallel and distributed algorithms running on a cluster. We chose Hadoop since it is an open source system which distributes tasks on all nodes within the Hadoop cluster. Therefore, as amounts of data grow, the performance would not be degraded as extra nodes can be always added to the cluster to make the analysis faster.

In this paper we present an algorithm for filtering data gathered from Twitter. The algorithm provides two processes. One, that fetches tweets provided via Twitter API and second, filtering process that filters data by defined cities and countries. The filtering process is executed in parallel. Filtered data are stored in HDFS for further analysis. In the paper we also present architecture of the cluster where the filtered data are stored. Also, we discuss the influence of a system architecture design on the system performance. The system architecture is based on a cluster with 16 nodes, each with a Hadoop instance installed.

In addition to Introduction and Conclusion, the paper is organized in five sections. Related Work is presented in Section II. In Section III, we present the proposed architecture. In Section IV we describe a part of a tweet structure relevant for the proposed algorithm. The algorithm used for filtering data collected from Twitter by the geographic location is presented in Section V. In the same section we discuss the storing mechanism for data in HDFS. In Section VI we present an example of filtering data using the algorithm presented in Section V.

II. RELATED WORK

To the best of our knowledge, there is only one paper that presents algorithm for filtering tweets by location. Rest of the papers mentioned in this section present similar algorithms and architectures to this paper. However, none of these papers provides parallel, real-time location filter.

Mahmud et al. [9] use an ensemble of statistical and heuristic classifiers to predict tweet locations based on the content of users‟ tweets and their tweeting behavior. Chen et al. [10] analyze user tweets through brand-specific intelligent filters to group users based on specific opinion. Kapanipathi et al. [11] used a Semantic Web

approach to filter public tweets matching interests from personalized user profiles. Mane et al. [12] used to analyze tweets using Hadoop, but without filtering and grouping them before analysis. Siddaraju et al. [13] present different approaches of processing big data with Hadoop in general. In contrast to all these works, we use a parallel algorithm to find tweet locations in real-time without a need to train and use classifiers. Only filtered data are stored in HDFS. Therefore we keep only tweets that are relevant to our analysis.

III. System Architecture

In order to achieve good Hadoop performances, a system based on Hadoop must be set up in distributed environment. For the purposes of this research, we have used an infrastructure consisting of 16 networked computers, as a basis of our system architecture.

The architecture of our system comprises a cluster consisting of 16 nodes. On all nodes we have installed Fedora 19 [14] operating system. Each node has 230 GB of dedicated memory for Fedora 19, of which 180 GB is reserved for Hadoop. On each node there is 4 GB RAM available and four processing cores. The nodes are a part of a single computer network with bandwidth of 100 Mb/s. Within the network, all computers are uniquely identified by their hostname formed by concatenating the letter „S" with a number from the interval [201, 216]. On all computers, Hadoop version 2.4.1 is installed. This version is chosen as it is the latest stable version at the time of writing. Hadoop instances are configured following tutorials presented in [15] and [16]. We chose computer with hostname S206 to be our master node purely for physical accessibility reasons. Master node is responsible for distributing tasks to slave nodes within the cluster where Hadoop is installed. Hadoop instances share data via Secure Shell (SSH) protocol [17]. Architecture organized in this way facilitates horizontal scaling. Therefore, as amounts of collected data grow, by adding extra nodes to our cluster, performance will not degrade. In Figure 1, we present our system architecture.

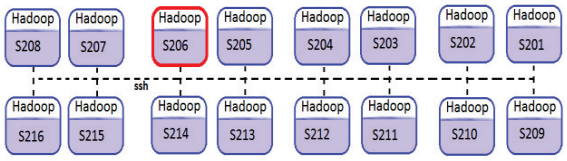

Figure 1The system architecture

IV. Tweet Structure

In this section we present a part of the tweet structure that is used for the filtering algorithm that will be presented in the next section. The entire specification of the tweet structure may be found in [18].

Data fetched from the Twitter data stream may be categorized in one of the following categories: tweets, messages about erased statuses and notifications about exceeded number of connection attempts. From all of these categories, in this paper we only consider tweets. Data from other categories are ignored because they do not carry information relevant to the analysis performed in this paper.

Fields from the tweet structure which have been used in our research are:

- *coordinates* - represent the geographic location of a tweet as reported by the user or client application.
- *lang* - when present, indicates a language identifier corresponding to the machine-detected language of the Tweet text, or "und" if no language could be detected.
- *user* - the user who posted this tweet. A location and a time zone are defined by a user in the *location* and *time_zone* fields.
- *text* - The actual UTF-8 text of the tweet.

Example of the part of the tweet structure described above with values of aforementioned fields is presented in Listing 1.

```
"coordinates":
{
    "coordinates":
    [
        -75.14310264,
        40.05701649
    ],
    "type":"Point"
}
"lang": "en"

"place":
{
    "attributes":{},
     "bounding_box":
    {
        "coordinates":
        [[
                [-77.119759,38.791645],
                [-76.909393,38.791645],
                [-76.909393,38.995548],
                [-77.119759,38.995548]
        ]],
        "type":"Polygon"
    }
}
"text": "Tweet Button, Follow Button, and Web
   Intents javascript now support SSL
   http:\/\/t.co\/9fbA0oYy ^TS"
"user":
{
    "location":"San Francisco, CA",
    "time_zone":"Pacific Time (US &amp;
      Canada)"
}
```

Listing 1 Example of the tweet structure part used in our approach

V. Collecting, Filtering and Storing Tweets

In this section, we present the algorithm for collecting, filtering and storing data, which represents the central part of our system. First, we give a brief overview of the system architecture component which main task is to collect, filter and data from twitter and store it in HDFS. Second, we present the algorithm for filtering data based on the tweet structure presented in Section IV. Finally, we present storing mechanism of filtered data in HDFS.

A. A Component for Collecting, Filtering, and Storing Data

Collecting, filtering and loading data is performed in parallel. The component for collecting, filtering and storing data (CFS) is implemented in Python

programming language. We chose Python as it has many libraries which facilitate efficient implementation of the component. For example, we used library for concurrent programming to implement concurrent execution of CFS component. JSON library was, in our case, used for transforming tweets fetched from the Twitter stream represented in JSON format to objects. In Figure 2 we present a structure of CFS that consists of three segments. In one thread, data are downloaded from the Twitter stream and placed in a queue for further processing. This is presented in Figure 2 in the top segment labeled "part 1". The queue used for this task is implemented in the Python Queue package, as thread safe collection. Other threads, responsible for filtering and storing data, are presented in Figure 2 in the middle segment, labeled "part 2". The number of threads responsible for filtering and storing data is defined by a parameter. Data are stored in a temporary list. The list size is also defined as a parameter. Once the maximum size of the list is reached, a thread responsible for storing data empties the list and transfers data from the list to HDFS. This is presented in Figure 2 in the bottom segment labeled "part 3".

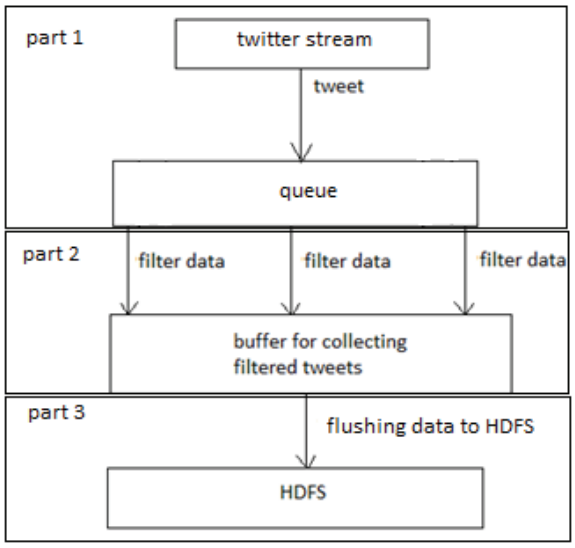

Figure 2 The CFS component

B. Filtering Algorithm

The main goal of the filtering algorithm is to provide a suitable way to extract tweets that are only from countries and cities relevant to the analysis. Countries and cities are defined as parameters in a parameter file. The format of the parameter file is presented in Section VI.

After a tweet is fetched from the Twitter stream, it is necessary to check whether the tweet originates from one of the defined countries. If a tweet is from one of the defined countries and cities it should be stored, otherwise it should be ignored. Data filtering is done in five steps presented in Figures 3 to 7. These steps are sorted by the relevance, where the first step provides the most accurate results, while the last step provides the least accurate results. Each successive step determines the country and city of a tweet with less accuracy. A next step in the algorithm is executed only after the tweet location cannot be determined by the previous step. This way of handling data may lead to inconsistency as the algorithm may not be able to precisely determine a location due to the fact that too much metadata is missing. Complete consistency in this case will result in a rejection of too much information, which would be later reflected on the reliability of the analysis. In the following paragraphs, each step will be explained in detail.

In the first step of our algorithm, a tweet is checked for the *coordinates* field. If the tweet contains the coordinate value, the next action is to check if its value corresponds to coordinates of one of the defined countries. If the coordinates belong to one of the defined countries, the distance between the given coordinates and the coordinates of each city from that country is calculated. A city that is closest to the given coordinates is declared to be the city from where the tweet is originated. Afterwards, country and city names are placed in the *location* field of the tweet. Modified tweet is then placed in a temporary list of collected tweets. In Figure 3 we present a part of the activity diagram for this step. If the aforementioned conditions do not hold, the algorithm proceeds to the second step.

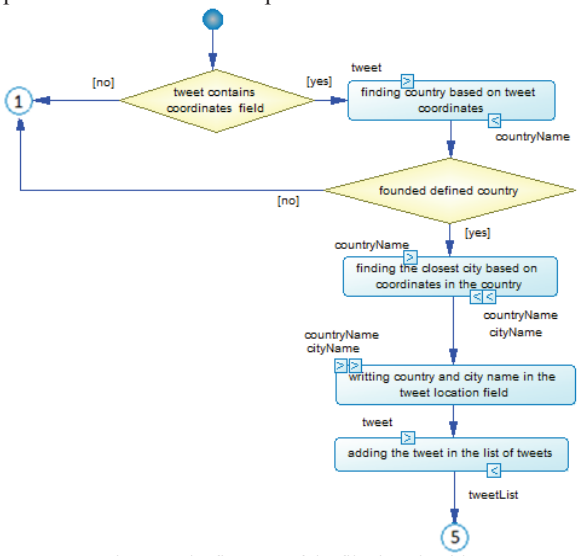

Figure 3 The first step of the filtering algorithm

In the second step of our algorithm, a tweet is checked for the *place* field. If the tweet contains the value of the *place* field, within the *place* field there is the *bounding_box* subfield that represents a polygon consisting of coordinates enclosing the place. The center of the bounding box is declared to be the geo-location of the tweet. The rest of the activities in this step are carried out analogously to the first step of the algorithm. In Figure 4 we present a part of the activity diagram for this step. If the aforementioned conditions do not hold, the third step of the algorithm is performed.

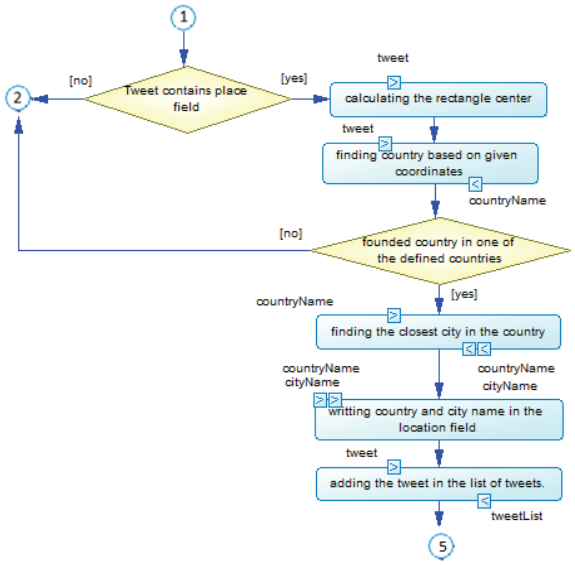

Figure 4 The second step of the filtering algorithm

In the third step of our algorithm, data are filtered by the *location* filed. First, words in the *location* field are parsed. This can be achieved by replacing all characters that are not alphanumeric with whitespace and then by splitting the resulting string by whitespace. Next, it must be checked whether some words include a variation of the name of defined city or country. Name variations are associated with defined country and city names and are defined manually as a parameter. They represent alternative or slang names for those countries and cities. Afterwards, a check for the city name variation is performed. If there is a match, country name from where the city is from and city name is placed in the *location* field of the tweet. In a case when there are no matches found for city name variations and there is a country name variation, only the country name is written into the tweet *location* field. In Figure 5 we present a part of the activity diagram for this step. If the aforementioned conditions do not hold, algorithm proceeds to the fourth step.

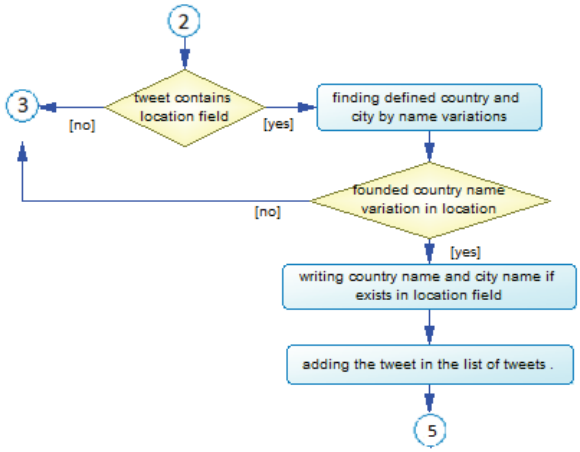

Figure 5 The third step of the filtering algorithm

The fourth step checks whether the language defined in the *lang* field, is a vernacular for one of the countries. Due to errors in the identification of the language in the *lang* field, mutually intelligible languages should be also considered as vernaculars. If the language in the tweet is one of the vernaculars of the defined country, it is assumed that a tweet is from the observed country and country name is written in the *location* field. In Figure 6 we present a part of the activity diagram for this step. If the previous conditions do not hold, algorithm proceeds to the fifth step.

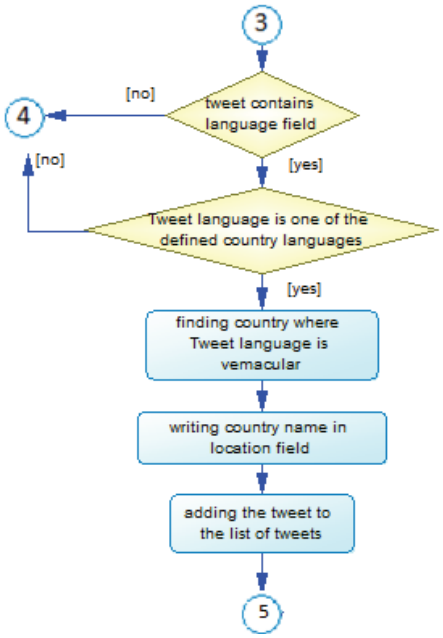

Figure 6 The fourth step of the filtering algorithm

In the final step of the algorithm, the value of the *time_zone* field is checked. The value of this field is parsed as in the third step. Then the country name variations are checked if they appear in one of the parsed words. If there is a country for which this condition is satisfied, the name of that country is written in the *location* field. If the previous conditions do not hold, algorithm finishes filtering given tweet. In that case, a thread executing the algorithm fetches a new tweet and repeats the process again. In Figure 7 we present a part of the activity diagram for this step.

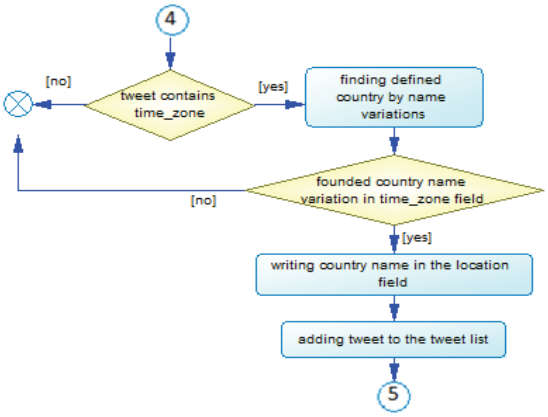

Figure 7 The fifth step of the filtering algorithm

Aforementioned steps are chosen because they reference fields from Twitter structure which contain information about tweet location. Steps are sorted by relevance by using results from analysis presented in

Section 6, where accuracy for each step was measured and by using explanation of results for each step which is also presented in Section 6.

C. Storing data in HDFS

After the list of collected tweets reaches the maximum size, the list values are stored in HDFS. Since storing data to the hard disk is expensive operation, the size of the list should be as large as possible (more than a hundred of elements).

Since Hadoop is based on the MapReduce programming model, data are stored as key-value pairs. After the list reaches defined size, it gets stored in HDFS. In our approach, the key in the key-value pair represents the *id* of the list, which values are incremented by 1, starting from 1. The value in the key-value pair represents the entire content of the list. After the list is stored in HDFS, list elements are removed. In Figure 8 we present a part of the activity diagram for storing data in HDFS.

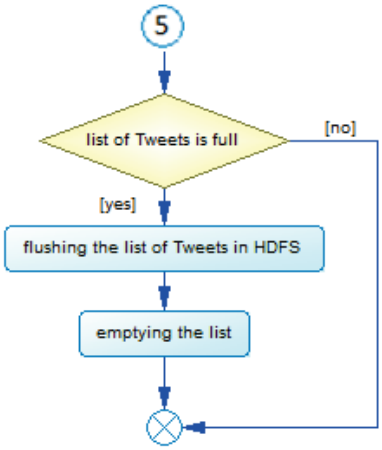

Figure 8 The algorithm steps for storing data in HDFS

Since the first step checks the *coordinates* field, one might be sure that tweet is originated from desired location. Therefore, values of fields used in step 3, 4 and 5, are checked and if certain values for those fields appear predefined number of times, they are also considered in the algorithm. This represents a way to handle synonyms (for example, in *location* field Novi Sad might often appear as Serbian Athens).

VI. Example of the System Usage

In this section we present an example of the system usage by collecting tweets which are originated from Serbia, Croatia, Montenegro or Bosnia and Herzegovina. First, we present set-up of necessary input parameters for this example. Second, we present the results of running the filtering algorithm for the defined set of parameters and the accuracy of the algorithm for this example.

In Listing 2, we present performance parameters. The RAM parameter defines the percentage of free memory that is to be allocated for the temporary list of tweets that gathers filtered tweets. The thread parameter defines the number of threads which are used for filtering data.

```
RAM#60
thread#5
```

Listing 2 Performance parameters

In Listing 3, we present parameters relevant to the cities we observe. First, we define country from where the city is from. Afterwards, we define a city name followed by geographic coordinates in a form of a pair (longitude, latitude). Finally, we define city name variations.

```
RS#Novi Sad#45.253829,19.830435#novi sad,нови сад
RS#Beograd#44.808182,20.452938#београд,beograd,belgrade
RS#Nis#43.325485,21.904346#ниш,nis,niš
ME#Podgorica#42.458755,19.253537#podgorica
BA#Sarajevo#43.861104,18.412421#sarajevo
HR#Zagreb#45.807778,15.976691#zagreb
BA#Banja Luka#44.778541,17.204558#banja luka
```

Listing 3 Parameters relevant to cities

In Listing 4, we present parameters relevant to the observed countries. The first parameter represents country code. Afterwards, we define city variation names, followed by country codes of vernaculars. In the end we define time zone name variations, which are usually comprised of name variations of the given country and cities. Those countries and cities represent time zone boundaries of countries and their subdivisions.

```
RS#србија,pc,срб,srb,srbija,serbia#rs,sr,sl#s
rbija,serbia,beograd,belgrade
BA#bosna,bosna i hercegovina,bih,republika
srpska,република српска#sl,ba#bosna i
hercegovina,bosnia and
herzegovina,sarajevo,sarajevo#
HR#hrvatska,hr,cro,croatia#sl,hr#hrvatska,cro
atia,zagreb
ME#crna gora,montenegro,mne,cg#sl,me#crna
gora,montenegro,podgorica
```

Listing 4 Parameters relevant to countries

Tweets were collected for 6 hours. In that time, the process defined by the algorithm presented in this paper filtered 2,213,098 tweets. 2,180 tweets were recognized as tweets from defined locations, which represented 0.0985% from the total number of tweets.

Our analysis was performed in two phases. First, we take a sample of 4,458 tweets from all gathered tweets. Our algorithm recognizes 5 tweets from that sample to be from defined territories. Afterwards, we manually check all 4,458 sample tweets. In this case there are 4 tweets that match defined countries and cities. One tweet from Campo Grande, city in Brazil, is recognized as Montenegrin due to the *location* field. The *location* field contains a value “cg” which represents acronym for this country name in their native language. However, the acronym of the Campo Grande city is also “cg”. Value “cg” represents country variation name in our case. Thus, we recognize all tweets for defined countries and cities in our sample, but one false positive. There are no false negatives in this example.

Second, we analyze how many tweets each step of our algorithm finds and what is the accuracy of each step in the aforementioned 2,180 tweets. The analysis is done manually, by looking the content of all the aforementioned tweets. In the first step, 58 tweets are found, which makes 2.6% of total number of tweets, and

had accuracy of 100%. This percentage of accuracy is to be expected, because geographic coordinates are specified. The only problem is that we group all tweets by cities which we define and therefore we declare that tweet is from the closest defined city. In our case it is desired behavior, because we want to group tweets in areas around largest cities of each country. In the second step, 29 tweets are found, which makes 1.3% of total number of tweets, with accuracy of 96%. Errors can appear to the following two reasons. First, the *place* field contains bounding box of place which is mentioned in tweet, not an exact geographic coordinate. Second, the *place* field contains places mentioned in tweet, not the actual place from where tweet is from. In the third step, 697 are found which makes 32% of total number of tweets. This makes with the accuracy of 94%. 80% of mistakes are made due to the value "cg" in the *location* field. The value "cg" represents Campo Grande, so to increase the accuracy of this step it is necessary to exclude value "cg" from parameters. Therefore, choice of defined parameters is very important for performance of this step, especially city and country variation names. In the fourth step 1,078 tweets are found with accuracy of 84%. Most of the errors are made due to language identification algorithm. Also, there is a problem with languages which are native languages in more than one country. This step varies for different countries and languages but must be included because in this case there are 49.4% of tweets found due to this step. In the final step, 318 tweets are found, which makes 14.6% of total number of tweets, with accuracy of 81%. The problem with this step is that users define their time zone manually and in many cases it is incorrect. Accuracy of the algorithm is calculated by Formula 1:

$$a = a1 * f1 + a2 * f2 + a3 * f3 + a4 * f4 + a5 * f5$$

Formula 1 Accuracy of the algorithm

where a is the accuracy of the algorithm, ai, $i \in \{1..5\}$, accuracy of step 1 to 5, and fi, $i \in \{1..5\}$, number of tweets for that step divided by total number of tweets. Accuracy for this example is 86%.

VII. CONCLUSIONS

In the paper we present an algorithm for filtering data gathered from Twitter and architecture for the system to analyze that data.

Data filtered by the algorithm may be used for various analyzes that are based on specific locations. Our algorithm showed good performance for the example presented in Section VI. However, the performance may vary for different values of input parameters. Some steps may show very low performance for some values of parameters, like it is the case with step four for the English language. Therefore, there is a need for a preprocessing algorithm, to find the most efficient combination of steps and for efficient way to provide performance analysis.

This platform can be a basis for a powerful decision support system in a form of a platform as a service. In our future research, we plan to add data from multiple new sources such as news web sites, forums, and blogs to collect more data from wider range of people and therefore to improve our analyses. We also plan to implement data mining packages for complex analysis of public opinion. Based on those data mining packages, we plan to build a Domain-Specific Language, which would allow users who do not have any experience in programing to specify and generate MapReduce programs to be executed on our system. With those features, we can provide a fast and accurate system for analyzing public opinion on various topics which can be used by users with no or little programing experience in a simple way.

ACKNOWLEDGMENT

The research presented in this paper was partially supported by Ministry of Education, Science and Technological Development of Republic of Serbia, Grant III-44010.

REFERENCES

[1] "Social network users" [Online] Available: http://en.wikipedia.org/wiki/List_of_social_networking_websites [Accessed: 29.11.2014].

[2] "Tweet definition" [Online], Available: http://whatis.techtarget.com/definition/Twitter [Accessed: 18.11.2014].

[3] "Twitter API" [Online], Available: https://www.dev.twitter.com/rest/public [Accessed: 18.11.2014].

[4] "Twitter geolocation and its limitations" [Online], Available: http://dfreelon.org/2013/05/12/twitter-geolocation-and-its-limitations/ [Accessed: 05.01.2015].

[5] C. Weinstock and J. Goodenough, "On System Scalability", in Software Engineering Institute (CMU/SEI-2006-TN-012), March 2006

[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters" in *Communications of the ACM Volume 51 Issue 1, January 2008*

[7] Ralf Lammel, "Google"s MapReduce Programming Model " in *Science of Computer Programming Volume 70 Issue 1,* January, 2008

[8] "Hadoop HDFS," [Online], Available: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html [Accessed: 20.11.2014].

[9] Jalal Mahmud, Jeffrey Nichols, Clemens Drews, "Home Location Identification of Twitter Users", in *ACM Transactions on Intelligent Systems and Technology (TIST) Volume 5 Issue 3,* September 2014

[10] Jilin Chen, Allen Cypher, Clemens Drews , Jeffrey Nichols, "CrowdE: Filtering Tweets for Direct Customer Engagements" in *Cameron Wynn,* September, 2014

[11] Kapanipathi, Orlandi, Sheth, Passant, "Personalized Filtering of the Twitter Stream" in *Marco de Gemmis*, November, 2011

[12] Sunil B. Mane, Yashwant Sawant, Saif Kazi, Vaibhav Shinde , "Real Time Sentiment Analysis of Twitter Data Using Hadoop " in *International Journal of Computer Science and Information Technologies, Vol. 5* March , 2014

[13] Siddaraju,, Sowmya , Rashmi, Rahul, "Efficient Analysis of Big Data Using Map Reduce Framework " in *International Journal of Recent Development in Engineering and Technology Volume 2, Issue 6*, June 2014

[14] "Fedora project" [Online] Available: http://www.server-world.info/en/note?os=Fedora_19&p=download [Accessed: 28.11.2014].

[15] Mihir Bellare, Tadayoshi Kohno, Chanathip Namprempre, "Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol", in *Ninth ACM Conference on Computer and Communications Security*, ACM, 2002

[16] "Single node cluster tutorial", [Online], Available: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/ [Accessed : 21.11.2014].

[17] "Multi node cluster tutorial", [Online], Available: http://tecadmin.net/set-up-hadoop-multi-node-cluster-on-centos-6/ [Accessed : 21.11.2014].