

UNIVERSITÀ DEGLI STUDI DI  
MILANO-BICOCCA

DATA MANAGEMENT

FINAL PROJECT

---

# Visual Stethoscopes

---

*Authors:*

Juan Carlos Rosito Cuellar - j.rositocuellar@campus.unimib.it

Lida Amalia Follari - l.follari@campus.unimib.it

January 28, 2019



## Abstract

*Visual Stethoscopes title's words well exemplify this project's long shot goal. Listening, in real-time, to online conversations, in order to capture signals, revealing public opinion sensibility to some social indicators.*

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Business Understanding</b>	<b>3</b>
<b>3</b>	<b>Methodology and Result</b>	<b>4</b>
3.1	Data Understanding: Data Discovery . . . . .	5
3.2	Data Understanding: Data Acquisition and Data Exploration . .	6
3.3	Data Preparation and Modeling . . . . .	7
3.4	Evaluation . . . . .	10
3.5	Presentation and Deployment . . . . .	12
<b>4</b>	<b>Discussion</b>	<b>13</b>
<b>5</b>	<b>Conclusions</b>	<b>14</b>
	<b>References</b>	<b>14</b>

# 1 Introduction

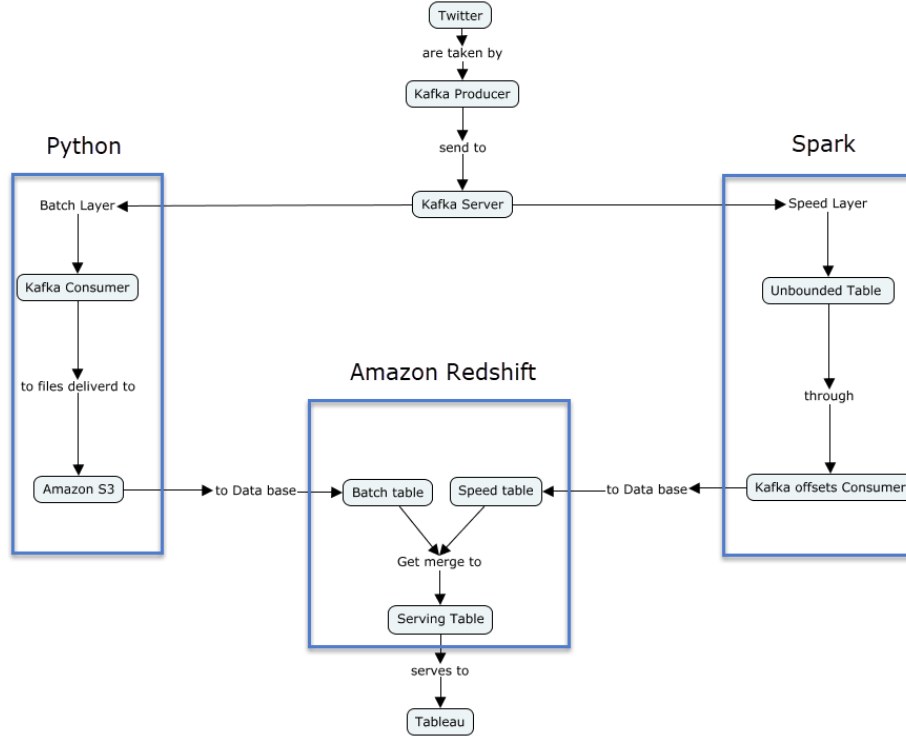
Being able to capture sensitive topics in online public squares conversations, where people meets, exchange ideas and concerns, and being able to exploit those findings in order to feel the population's temperature about certain social indicators, could represent a good mean for administrations, dealing with spending evaluations/choices, and for researchers, studying social issues.

Twitter has been selected in this project as the virtual square to be listened to in real time, since it, more than others social platforms, reminds a place where persons, willing to broadcast messages to audience, meets. The project purpose is to focus the attention on tweets that contains one word (customized any time) and calculating the ratio between the sum of their re-tweets, over the sum of their followers giving, in a certain way, the sensibility's measure of the public opinion about a certain topic.

The technological solution to this business issue must be searched in Data Management ecosystem, providing data discovery, data acquisition, data exploration and data analysis instruments. The implemented platform will have to manage big data, in such a way that the so called 3V will have to be delivered: Velocity, brought by streaming with "lamda architecture", Volume, brought by the amount of data and the number of data transactions, and Variety brought by data formats.

In the planned informative system will come into play the management of the streaming data, with the data reliability offered by the lambda architecture; through a speed layer, managing unbounded tables through incremental queries, and a batch layer taking static tables of the remaining offsets to get merged with the speed layer tables for the deployment. Implementing old and new technologies of data storage RDBMS, CVS, JSON, together with applications for data acquisition of Twitter API implemented by Spark and Python Programming Frameworks, and the stream-processing platform Kafka exploiting to an on-line container Amazon S3 and to an online database Amazon Redshift. The implemented system schema is:

Figure 1: total model



## 2 Business Understanding

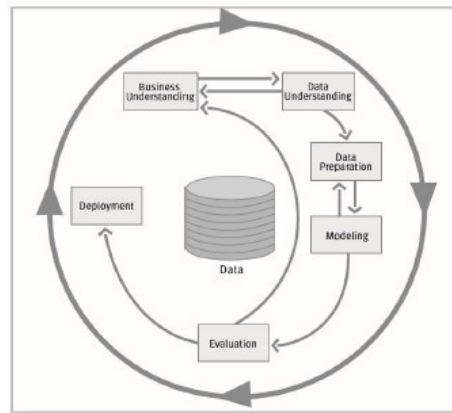
In public administrations, one of the most important activities is to decide where to invest public resources and get a feeling of population well being, satisfaction. Until now, the most common method of investigation has been the statistic poll, prone to mystification, errors, and very expensive process. Being able to dig in the web in order to automatically investigate public sensibilities open new and endless possibilities. Twitter, a social networking and microblogging service, has been chosen, among other social platforms, since it acts quite like a megaphone for people willing to broadcast their opinions/sentiments to a large audience. Users use mobile phones or computers to send and to read messages, called "tweets", while a "re-tweet" is a re-post of another Twitter user's tweet on someone profile to show to your own followers. This case is a special type of data mining problem: Data description and summarizing. Data description and summarizing, aims at the concise description of characteristics of the data, typically in elementary and aggregated form. This gives the user an overview of the structure of the data. Getting the count of tweets, aggregated by location, relative re-tweets and re-tweets over tweets ratio, represents a simple descriptive statistical and visualization technique providing the kind of insight required by

the case goal.

### 3 Methodology and Result

CRISPDM reference model will be used to describe project's elements to be explained.

Figure 2: Crisp-DM Reference Model



- Business Understanding
- Data Understanding: Data Discovery
- Data understanding: Data Acquisition and Data Exploration
- Data Preparation and Modelling
- Evaluation
- Presentation and Deployment

### 3.1 Data Understanding: Data Discovery

Selected data comes from Twitter social platform. It is an external data source. Our needed data is open, free and easy to access thanks to Twitter API. Data returned by the Twitter API provides data encoded, based on key-value pairs, with named attributes and associated values, in JSON format, the structure and the request depends on the method that has been used. The selected Twitter API method for request is "search()", this method provides the power of data semantics in the Tweeter system, bringing the capability to getting the tweeters with a search of a word in a context. Each Tweet has an author, a message, a unique ID, a time-stamp of when it was posted, and sometimes geo metadata shared by the user. Each User has a Twitter name, an ID, a number of followers, and most often an account bio. In addition to the text content itself, a Tweet can have over 150 attributes associated with it. Given the goal, the implemented system doesn't need to get the whole tweet, instead thanks to the API interface, it is enough to query Twitter not for the whole tweet, but just for very little information:

- ID
- number of tweet author followers
- number of tweet's re-tweets
- number of favorites of the twitter
- location defined by the author (much more common data than long-lat geographical info)

The quality of data is good, no missing, no corrupted data. Images of quite real-time data can be visualized in JSON format:

Figure 3: Twitter query result in JSON format

```
{
  "created_at" : "Thu May 10 15:24:15 +0000 2018" ,
  "id_str" : "850006245121695744" ,
  "text" : "Here is the Tweet message." ,
  "user" : {
  } ,
  "place" : {
  } ,
  "entities" : {
  } ,
  "extended_entities" : {
  }
}
```

### 3.2 Data Understanding: Data Acquisition and Data Exploration

The data understanding phase starts with initial data collection and proceeds with activities that enable to become familiar with the data. In this case acknowledged the good data quality, there will be no need for extra manipulation, just acquisition. Interesting subsets are derived by common locations. The only related problem is that the location string, returned by the API, is a location given by the author, often not corresponding to any real location, but, anyway, a much more common data than the location of the tweeter, because very few authors agree for geo-localization. This phase is the main data source for the lambda architecture pipeline. It uses the method "search" of the twitter streaming API to get the new entry, then it makes data exploration reading each tweet getting just the parameters necessary for the system objective. Finally it delivers each tweet to the Kafka producer to set down the information on the Kafka broker, to get ready for the Kafka Consumer with the offsets. It starts setting up the Twitter API, that has to be sign up as application on the server of Twitter, then finally gets the access key to make the authorization call for the twitter API, made by the credentials "Consumer Key" and the "Access Token". Then the streaming arriving by Twitter API is explored, and the parameters taken into account are deployed to the Kafka Producer, the parameters are: user id, created\_at, followers\_count, location, favorite\_count and re-tweet\_count. These are taken as a cumulative string separated by ";," among parameters, and each Tweet is sent to the Kafka Producer. In order to get Kafka up, the initial set up has to be done. Apache Zookeeper has been implemented, to control all the clusters, in this case the cluster is located on the port 9092 of the local-host. Finally the Tweets are requested to the Twitter API every minute and sent to the Kafka Producer. The id of the last taken tweet is saved, in such a way that it will be used in the next request with the Twitter API in order to obtain a reliable tweets sequence. The Tweets arrived to the Kafka Producer have to be taken and managed in an efficient way due to the Streaming process, this is performed by two Kafka applications, one take data in a streaming way and the second one in an historical-based way, so it has to prevent data loss, duplication or sequence loss. Therefore has been implemented a One-time readable offsets, which takes the offset tagged by a latest or earliest offset.

Figure 4: total model

	ABC id	ABC created_at	ABC followers_count	ABC location	ABC favorite_count	ABC retweet_count
1	738276139504963584	2018-09-18 14:13:06	644		0	58
2	2903059609	2018-09-18 14:13:06	302	deep in the heart of texas	0	34302
3	3261609530	2018-09-18 14:13:06	434	Nairobi, Kenya	0	465
4	1034014823594381313	2018-09-18 14:13:05	38		0	0
5	1716299822	2018-09-18 14:13:05	113		0	24592
6	20792010	2018-09-18 14:13:05	11187	Columbus, Ohio	0	0
7	807022147	2018-09-18 14:13:05	1020	Edinburgh, Scotland	0	0

### 3.3 Data Preparation and Modeling

Data preparation phase usually covers all activities needed to construct the final data-set, data that will be fed into the modeling phase, from the initial raw data. In this case no attribute selection needs to be done, since just required attributes are returned by the API. Different tables are created. Since data quality dimensions:

- Accuracy
- Completeness
- Consistency
- Time related

are optimal for the project's purposes, as stated above, the system is robust respect to garbage, no cleaning phase is necessary.

To get the Tweets from the Kafka broker to begin the data preparation, has been defined the Kafka consumer, where this delivers the data to the speed layer and to the batch layer to after get the data joined and delivered to the serving layer. Then the scope of get the Tweets are to take each location that the user define and make it each group to the respective aggregation, counting and the sum of the tweets parameters. To get finally the count of tweets took by each location, the sum of the user followers, user favorite, and retweets by each location too. So the serving layer consist on this parameters for the respective visualization.

So to treat the Tweets by each location and get the total of quantity of tweets, the total of Re-Tweets and the amount of followers of the user of the tweet, is denoted by  $Z_i$  as each of the sums that has to be process. Model 1: Each group of different locations

$Z_i$  where  $i$  : unique location

$$Z_i = \sum_n (x_n) \quad (1)$$

where :  $n$  is the quantity of elements inside the group  $i$

and  $x$  : are the  $n$  value inside the  $i$  group

Model 2: The ratio between the above quantities is used to measure the condition/support/approval of the searched word.

$$Approval = \frac{\sum_{Tweets}(Retweets)}{\sum_{Tweets}(Followers)} \quad (2)$$

Range from 0 to 1



where 0 = no retweet  
1 = all followers retweeted






This derived attribute, ratio calculation, plays the most fundamental role in the presentation of acceptance of a certain word. After the data acquisition the system is composed by two different ways to treat the streaming data, one is on the batch layer and the second one is on the speed layer. On the batch layer the data is requested to the Kafka consumer and then delivered to the on-line container Amazon S3 and stored as a CSV file. Each CSV file on S3 has the messages that hasn't been read by the other partitions of the Kafka consumer (speed layer) at that moment when the request was made. Every request is made every 10 minutes, with the goal is to take the offsets that the speed layer hasn't been able to read. Finally the information of S3 is delivered to the Database of Amazon Redshift. S3 and Amazon Redshift are static storage in which the query makes each request after that another query has finished. This transition is made by a query requested to Amazon Redshift in which take the information of the S3 container and storage it in a PostgreSQL database. SO that takes us that the speed layer manages the tweets in a different way than the batch layer. On the speed layer the Kafka consumer is taken by an incremental query as a streaming data frame, making a well known as a unbounded table, where the query remains open for the data that is being uploaded to the Kafka consumer. The Tweets that have been stored, have to be processed to get the goal of the Business and to be ready for the final deployment, so the tweets are Finally the information that is been captured from Kafka Consumer gets pushed to a static table of Amazon Redshift every minute while the streaming is being stored every 30 seconds. The model has been applied on the batch layer and the speed layer of the lambda architecture pipeline. The Model 1 is the one that makes to get the data regrouped by location, saving a lot of data storage. On the figure 5 and 6 are the resulting tables of the deployment of the speed layer and batch layer respectively.

Figure 5: Format of the speed layer table

	ABC location	123 count_id	123 sum_followers_count	123 sum_favorite_count	123 sum_retweet_count
1	Norway	1	1,112	0	0
2	FL -- AL	4	2,848	0	141,311
3	황마미	1	230	0	1,794
4	Asgard	4	16,264	0	8,791
5	Hudson, FL	1	154	0	0
6	Malaysia	7	722	0	27,650
7	London	2	7,759	0	3
8	Wichita, KS	3	1,635	0	11,115
9	East Lansing, MI	1	45	0	0
10	Philippines	1	67	0	2
11	Houston, TX	5	1,016	0	2,133

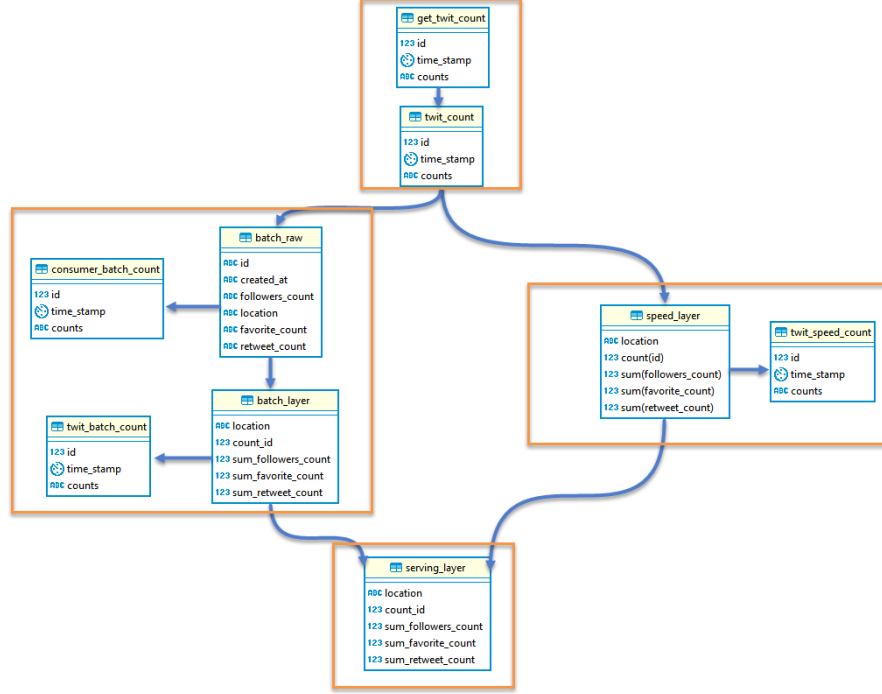
On lambda architecture pipeline different tables have been used to make the storage of temporal and/or checkpoint tables (diagram implemented: Figure 7).

Figure 6: Format of the Batch layer table

	ABC location 	123 count_id 	123 sum_followers_count 	123 sum_favorite_count 	123 sum_retweet_count 
1	North Jutland, Denmark	1	18	0	73
2	Worcester, England	1	29	0	0
3		580	413,474	6	6,736,930
4	Pretoria, South Africa	1	273	0	35,312
5	황미미	1	230	0	1,794
6	Hudson, FL	1	154	0	0
7	FL -- AL	4	2,848	0	141,311
8	Asgard	4	16,264	0	8,791
9	Malaysia	10	2,622	0	28,755
10	London	2	7,759	0	3
11	Philippines	2	2,112	0	5
12	Houston, TX	9	3,452	0	15,983

and this has the goal to get all the information pipeline of the tweet, and to get stored some measures to evaluate the performance of the model implemented.

Figure 7: Amazon Redshift data base diagram



### 3.4 Evaluation

At this stage in the implementation of the system and the built model is ready to be deployed. Real-time data will generate visualizations for immediate user evaluation. And that's why before proceeding to final deployment of the model, it is important to thoroughly evaluate it, to be certain the model properly achieves the business objectives. So is been analyzed the life cycle of the data and the final deployment is on the Figure 7, getting the data grouped by the location, where the speed layer and the batch layer get joined into.

Figure 8: Serving Layer Result

	ABC location	123 count(id)	123 sum(followers_count)	123 sum(favorite_count)	123 sum(retweet_count)
1	South East, England	4	1,456	0	0
2	Worcester, South Africa	3	1,092	0	0
3	Tangerang, Indonesia	4	20	0	4,836
4	Belfast, Northern Ireland	4	184	0	0
5	hunhan ♥	4	1,908	0	4
6	Sivakasi, India	4	520	0	256
7	GG™  TT™	4	4,100	0	205,330
8	Galway, Ireland	4	448	0	0
9	Western Visayas	4	3,768	0	244

To continue with the analysis of the life cycle of the data, we pass through

the quantity of the data was entering on the lambda architecture pipeline, to do that has been measured the quantity of tweets that are incoming every second, knowing that the search() method search for historical data, we take the last id given on each pull and is carried to the next iteration, so the data delivered by the search method can be only the knew data. And turns out that the data aren't been continuous, and that was the refreshing time of the servers of Twitter that the deliver the data to the search() method, And the result was:

$$\text{Refreshing of the server period: } \frac{1}{20} \frac{1}{\text{sec}} \quad (3)$$

and the other fact that comes out was the continuous amount of 15 tweet each pull that is make how it seen on the figure 9.

Figure 9: Tweet Count Table

	123 id ↕	🕒 time_stamp ↕	ABC counts ↕
1	46	2018-09-22 21:33:44	15
2	47	2018-09-22 21:33:46	15
3	51	2018-09-22 21:33:54	15
4	55	2018-09-22 21:34:02	15
5	58	2018-09-22 23:04:52	10
6	60	2018-09-23 04:32:22	20

Then the next analysis that has been made, was the research for new parameters of the search() method, and the parameters that delivers more data was the "count=100" parameter this takes:

$$\text{Maximum Quantity of tweets in a request: } 100 \text{ tweets} \quad (4)$$

Another test that has made was a 1 minute iteration getting 100 tweets in each iteration, to get all the Tweets that are new in that time, getting results of the figure9.

Figure 10: Tweets throughput related to the word "happy"

	Tweet Count
<b>*1 Minute (morning)</b>	70
<b>*1 Minute (night)</b>	284

\*average of 5 throughput values

Due to the fact that the initial limitation was that it only can carry over 15 request on 15 minutes, the iterations per request, to get all the tweets until there weren't more tweets to request, was more than 15 request to get all the queue empty, has left on execution for more than 15 minutes, getting all the tweets every minute, and the exception hasn't raised to the error "Rate limit exceeded", so that means that the request steel keeps giving the data.

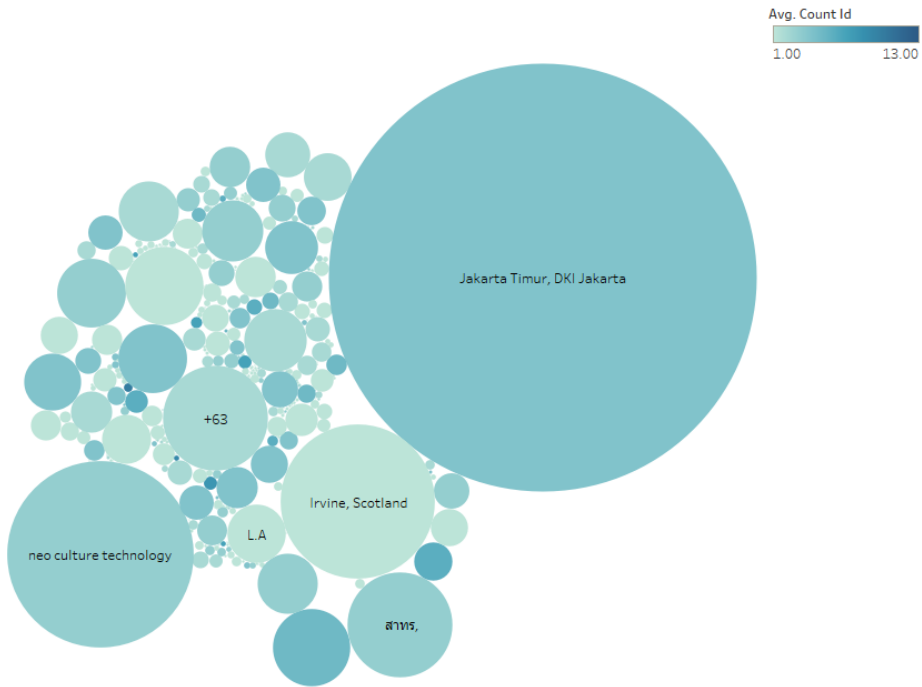
### 3.5 Presentation and Deployment

Data tables are directly set into Tableau, for a ready to be seen visualization request. In this way the knowledge gained is based on real-time data, organized in such a way that the final customer/user can use it exploiting the power of images. Connecting the serving layer, on Amazon Redshift, to Tableau on-live connection, so the deployment can be refreshed and get the data that is taken in that moment. The final deployment, of the Model 2 implementation of the equation 2, it can be visualized on the Figure 10.

q

Figure 11: Tweet Approval Index

Popularity index Dimention - Count(Id) Color



## 4 Discussion

Project's objectives have been reached, a table with latest data updated every 1 minute is provided for an on-line visualization. During the evaluation phase a much slower than expected throughput has been detected, even with the lambda-architecture implementation. With testing and appropriate checkpoints the reason has been discovered: the *tweetpy search()* function gives, at maximum speed: 15 requests each minute.

The analysis made by the limited throughput of the `search()` method is, compare the streaming filter and the `search()` method, both contained on the Tweepster API. The `search()` method gives the benefit of Tweepster processors of data semantics that brings computational power analyzing words, in this case the word "happy", giving high capability to search in different languages, geolocalization and more characters to search than the streaming method, and with the streaming filtering provides a similar computational power of data semantics but with a less matches into the search, providing essential matching of high and lower case, URL, and others. On the other hand, we got that the `search()` method with a limited streaming rate and the streaming filter provides unlimited throughput. Indeed the throughput of the `search()` method has been tested and is limited until 100 tweets for a request, that force to make iterations on every request all the times that want to take the tweets, but fortunately the throughput necessary by the word "happy" aren't meaningful due that it can be supported by the method `search()`. Another point that has to be considered is the way how the streaming filter and search method makes the query to the data, the `search()` method takes the historical data (backward) and the streaming push the actual data (forward), so the `search()` method data is a more controllable than the streaming filter, but with a rate limit makes it harder to make a reliable streaming. But the `search()` method, won't lose any data due to the fact that the tweets can be queried since a punctual tweet ID. Actually, data arrive through the system to the visualization tool clean and ready to be used, but the rate is much lower than expected Therefore if the objective of the lambda architecture implementation was manage high throughput of data this system is not effective. And, even efficiency is not so good since a powerful lambda-architecture has been implemented, but slowed down by the quantity of tweets queried and also can be slowed down by the Rate Limit of the search function in case we take a word with high matches on the search.

## 5 Conclusions

The implemented system represents a kind of visual stethoscope, thanks to real time visualization, is able to give the public opinion sensibility about a certain topic.

Is more desirable the search() method than the streaming filtered, due to the capability to search in different languages, if the business hasn't their own data semantics system.

The lambda architecture pipeline is being wasted, due to the implementation of a system with a throughput of 1KB per minute and with a data acquisition of historical data.

The Business goal makes that the desired Kafka streaming processing has to be "at-most-once" implementation, making no replication, but with a probability of losing data.

Further steps could be the use of data wrangling systems like Google Fusion Tables, OpenRefine and others, to only keep track of meaningful locations; and the implementation of Twitter streaming filter with a system of data semantics, if the desired word, to look for, exceeds the Rate limit, of request per minute, that the search() method offer.

## References

- [1] *Crisp-DM Reference Model*, <https://www.the-modeling-agency.com/crisp-dm.pdf>
  - [2] *Spark*, <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html>
  - [3] *Kafka*, <https://kafka.apache.org/0110/documentation/streams/developer-guide>
  - [4] *Offset Management*, <http://blog.cloudera.com/blog/2017/06/offset-management-for-apache-kafka-with-apache-spark-streaming/>
  - [5] *Twitter API Tutorial*, <http://140dev.com/twitter-api-programming-tutorials/aggregating-tweets-search-api-vs-streaming-api/>
  - [6] *Amazon Redshift*, <https://docs.aws.amazon.com/redshift/latest/mgmt/welcome.html>
  - [7] *Amazon S3*, <https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html>
- [references ]