Instituto Tecnológico de Buenos Aires - ITBA

Escuela de Ingeniería y Gestión

Calidad de Datos en un Contexto de Big Data

- Sobre la Calidad de Datos de Twitter -

Autor: Arolfo, Franco A. (Legajo 51.408)
Tutor de Tesis: Ph.D. Vaisman, Alejandro A.

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN INFORMÁTICA

Lugar: Lavardén 315, 1437 CABA, Buenos Aires, Argentina
Fecha: 4 / 4 / 2018

Instituto Tecnológico de Buenos Aires - ITBA

Escuela de Ingeniería y Gestión

Data Quality in a Big Data Context

- About Twitter's Data Quality -

Author: Arolfo, Franco A. (Legajo 51.408)
Thesis Tutor: Ph.D. Vaisman, Alejandro A.

FINAL PROJECT SUBMITTED IN ORDER TO GET THE DEGREE OF

INFORMATICS ENGINEER

Address: Lavardén 315, 1437 CABA, Buenos Aires, Argentina
Date: 4 / 4 / 2018

Abstract

In each of the phases of a Big Data analysis process, Data Quality (DQ) plays a key role. Given the particular characteristics of the data at hand, the traditional DQ methods, based on quality dimensions and metrics, must be adapted and extended, in order to capture the new characteristics that Big Data introduces. This paper dives into this problem, re-defining the DQ dimensions and metrics for a Big Data scenario, where the data arrives, in this particular case, as unstructured documents in real time, such as JSON objects. This general scenario is instantiated to study the concrete case of Twitter feeds. Further, the paper also describes the implementation of a system that acquires tweets in real time, and computes the quality of each tweet, applying the quality metrics that are defined formally in the paper. The implementation includes a web user interface that allows filtering the tweets, for example, by keywords, and visualizing the quality of a data stream in many different ways. Experiments are performed and their results discussed.

Table of contents

Chapter 1

Introduction and Motivation

The relevance of so-called Big Data has been acknowledged by researchers and practitioners even before the concept became widely popular through media coverage [9]. Although there is no precise and formal definition, it is accepted that Big Data refers to huge volumes of heterogeneous data that must be ingested at a speed that cannot be handled by traditional database systems tools. Big Data is characterized by the well-known "4 V's" (volume, variety, velocity, and veracity). implying that not only the data volume is relevant, but also the different kinds of structured, semistructured and unstructured data, the speed at which data arrives (e.g., real time, near real time), and the reliability and usefulness of such data. However, it is also acknowledged that most of the promises and potential of Big Data are far from being realized [20]. This gap between promise and reality is due to the many technical problems and challenges that are usually overlooked, although the database research community has warned about them, namely heterogeneity, scale, timeliness, complexity, and privacy, among other ones [12].

In each of the phases in a Big Data scenario, Data Quality (DQ) plays a key role, as the very nature of the "4 V's" suggest. The largely studied concepts of DQ must be revisited and re-studied in a Big Data context, since, as it will be discussed in this paper, many new problems appear, which are not present in traditional relational databases scenarios [19]. Intuitively, each of the "V's" define a different context for data analysis, and therefore, for DQ. Thus, there is a strong relationship between the work about contexts in DQ (e.g., [13, 16, 15]) and the problems of DQ in Big Data, since different notions of quality must be used for different types of Big Data. In particular, this paper deals with DQ in a real-time scenario, specifically Twitter feeds. This is a typical scenario where data come at high speed, highly unstructured, and with very volatile reliability and usefulness. All of these characteristics are the complete opposite of a relational database analytics scenario, where data are highly

structured, and cleaned, transformed and analyzed offline. Therefore, DQ must be addressed considering these differences.

In spite of the relevance of the topic, there has been not much work so far, in particular regarding the implementation of quality processes over Big Data sources. This paper tackles this issue. More concretely, the contributions of this work are:

- The definition of DQ dimensions and metrics in a Big Data scenario where data arrive as unstructured documents and in real time. Traditional DQ dimensions are redefined, to address those particular characteristics. This general scenario is instantiated to study the concrete case of Twitter feeds.
- The implementation of a system that acquires tweets in real time, and computes the quality of each tweet, applying the quality metrics defined formally in the paper. The implementation includes a web user interface that allows filtering the tweets e.g., by keywords, computing their data quality, and visualizing the DQ, not only the overall one, but also along each dimension.
- An experimental study of the quality of the feeds, using the tool described above. This study is aimed at showing how DQ can be used to determine the attributes that characterize the different quality of the tweets, filter out bad quality data, or validate the conclusions drawn in the data analysis phase.

1.1 Related Work

Ensuring the quality of data in databases has long been a research topic in the database community. Research in DQ has resulted in the definition of dimensions, metrics, and methods to assess the quality of a database [1]. In spite of this, classic research considers DQ as a concept independent of the context in which data are produced and used, which is clearly not enough to solve complex problems, particularly in current times, when, among other facts, ubiquitous computing requires accounting for space and time when a query is being answered. Strong et al. [26] realized this problem, and claimed that data quality is highly dependent on the context, which became an accepted fact thereon. The rationale for this conclusion was based on the fact that, similarly to quality in general, DQ cannot be assessed independently of the consumers who choose and use the products [16, 15]. The former proposes a system where contextual information allows evaluating the quality of blood pressure data. The latter proposes a framework that allows context-sensitive assessment of DQ, through the selection of dimensions for each particular decision-maker context and her information requirements.

There is a large corpus of work regarding data context management with a wide variety of uses. It is widely accepted that most modern applications, particularly over the web, are required to be context-aware. Bolchini et al. [7] presented a survey of context models, with a well-defined structure, that identified some important aspects of context models. In particular, they remark that models must account for *space*, *time*, *context history*, *subject*, and *user profile*. Preferences in databases have also been extensively studied [13, 14]. In the multidimensional databases domain, [21] proposes to define the context through the use of logic rules, representing the database as a first-order logic theory.

Recently, [18, 19] study the particularities of data quality in the context of Big data, that is, how the "4 V's" mentioned in the previous paragraphs impact on well-known DQ dimensions and metrics used in traditional structured databases [6]. The main message in [19] is that Big Data quality should be defined in source-specific terms and according to the specific dimension(s) under investigation. In some sense, this means that the context is again present in the Big Data scenario when quality is addressed. The present paper builds from these studies, as will be clear in the remaining sections.

Chapter 2

A Short Background

2.1 What is Data Quality?

Data Quality (DQ) is a multi-faceted concept, represented by different dimensions, each one referring to a different quality aspect [6, 26]. A *DQ dimension* captures a facet of DQ, while a *DQ metric* is a quantifiable instrument that defines the way in which a dimension is measured. Since a DQ dimension is in general a wide concept, an associated metric allows specifying a concrete meaning for the dimension. As a consequence, many different metrics can be associated to the same DQ dimension, and their application will measure several different aspects of the dimension.

In a broader sense, the *quality of an object or service* represents how much this object or service fits the needs to solve a given problem. That is, quality is not absolute to the object or service *per se*, but relative to the problem to be solved. This is the approach followed in this work.

As mentioned before, the report on Data Quality of the Data Warehousing Institute estimates that Data Quality problems cost U.S. businesses more than 600 billion dollars a year [4], and some companies in the tech industry are rising their voices on this issue. IBM's acquisition of Ascential Software (2005), a data integration tools company, highlights the role of Data Quality in one of their reports: it stays Data Quality and security issues as the leading inhibitors (55% of respondents in a multi-response survey) to successful data integration projects [5]. SAP has set up a project for testing in the area of Data Quality with important savings in several internal business processes [25]. Informatica, another leading company on data integration, and Oracle are also relying on Data Quality tools in their products [22] [24].

And this is not only taking place in private initiatives but also in public ones. In 2001, the US Government signed into law a new Data Quality legislation called the *Information Quality Act*, which concerns agencies to report periodically the number and nature of Data

Quality complaints received, and how those are being handled [2]. In 2010, the United States Environmental Protection Agency (EPA) also added a Data Quality legislation [11]. On the other hand, the ISO 8000, the new international standard for Data Quality, published in late 2008 their first sections and is still today under development [10].

2.1.1 Dimensions and Metrics

While a large number of DQ dimensions were proposed in the literature, there is a basic set of them, which are generally acknowledged to be representative of the quality of data [6, 3]. This set includes accuracy, completeness, consistency, freshness (or timeliness), among other ones.

- *Accuracy:* Specifies how accurate data are, and involves the concepts of *semantic accuracy* and and *syntactic accuracy*. The former refers to how close is a real-world value to its representation in the database. The latter indicates if a value belongs to a valid domain. In other words, it describes the closeness between a value v and a value v', considered as the correct representation of the real-life phenomenon that v aims at representing. For example, if someone wants to type the name "John" but typed "Jhn", there is an accuracy issue.

- *Completeness:* Represents the extent to which data are of sufficient breadth, depth, and scope for the task at hand. When dealing with relational databases, this can be characterized as the presence/absence and meaning of null values, assuming that the schema is complete.

- *Redundancy:* Refers to the representation an aspect of the world with the minimal use of information resources. For example, in the table of Figure 2.1 (center and right), the nodes that compose the different clusters in an architecture are shown, together with their status. It can be seen that cluster 3 has a "RUNNING" status, but its nodes are "STOPPED". redundancy here caused an inconsistency issue (see below).

- *Consistency:* Refers to the capability of the information to comply without contradictions with all the rules defined in a system. For example, in a relational database constraints are defined to guarantee consistency. As commented above, Figure 2.1 (right) shows a *consistency* issue.

- *Readability:* Refers to the ease of understanding of information. For example, suppose a hand-written paragraph was scanned, and some of the characters are not well defined, as depicted in Figure 2.2.

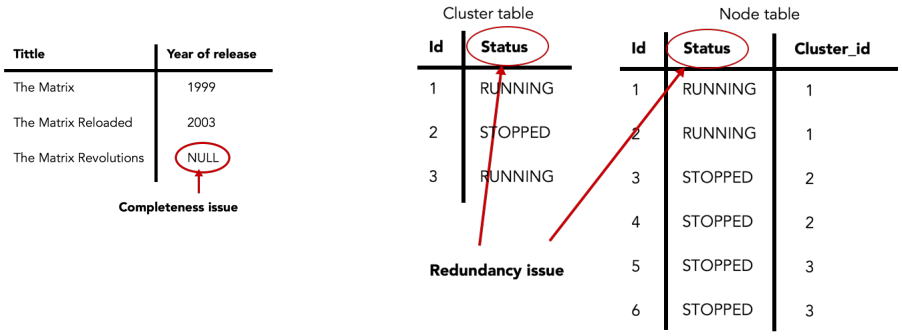

Fig. 2.1 (a) Completeness issue (left); (b) Redundancy issue (center & right).

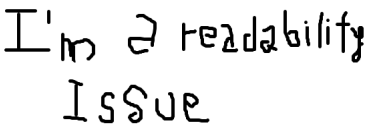

Fig. 2.2 Readability issue example.

- *Accessibility:* Also called *availability*, is related to the ability of the user to access the information.
- *Trust:* Refers to how much the information source can be trusted, and therefore to what extent data are reliable. For example, people may rely on Facebook or Twitter posts to find out the quality of a movie, or check the IMDB site at http://www.imdb.com, which might provide more reliable data.
- *Usefulness (cf. [19]):* This is related to the benefits a user can obtain when using the data to produce information. For example, Figure 2.3 shows scans taken from Bosch's *The Garden of Earthly Delights*. to observe technical details present in a picture of a painting, a user would choose the image with the highest contrast. Again, this is also a contextual quality dimension: a lower-quality picture may be enough for some users or for some kinds of requirements, while clearly not enough when the details are needed.

To quantify these dimensions and to be able to assess DQ according to them, the concept of *metrics* must be introduced. Mathematically, a DQ *metric* for a dimension D is a function that maps an entity to a value, such that this value, typically between 0 and 1, indicates the quality of a piece of data regarding the dimension D. For a given dimension, more than one metric could be defined and combined to obtain a concrete quality value. Note that metrics are highly context-dependent. For example, the readability of a hand-written text may be influenced not only by the text content, but also by the way the user writes (see Figure 2.2). The same occurs with metrics for other DQ dimensions.

Fig. 2.3 Usefulness issue example

2.2 Big Data Quality

In a Big Data context, datasets are too large to store, analyze, handle or process, for the traditional database tools. As explained above, Big Data are characterized by the well-known "4 V's", namely *Volume* (size of the datasets), *Velocity* (speed of incoming data, e.g., the number of tweets per second (TPS)), *Variety* (refers to the type and nature of the data), and *Veracity* (the reliability of the data, which, in this context, is greatly volatile, even within the same data stream). In the literature, many other "V's" can be found, but only these four will be considered in the present work. According to the stucture of data, they can be classified in: (a) *Stuctured*, where each piece of information has an associated fixed and formal structure, like in traditional relational databases; *Semi Structured*, where the structure of the data has some degree of flexibility (e.g., an XML file with no assiociated schema, or a JSON response from an API, whose structure is not completely defined); (c) *Unstructured*, where no specific structure is defined. Further, the United Nations Economic Commission for Europe (UNECE) classifies Big Data according to the data sources in: *Human sourced*; *Process mediated*; and *Machine generated* [8]. These are explained next.

- *Human-sourced data:* Information people provide via text, photos or videos. Usually, this information lacks of a fixed structure, like the texts written in natural language. Therefore, the information streamed here is *loosely structured* and often ungoverned. Examples are social networks posts (Facebook, Twitter), YouTube videos or e-mails, and, in general, data coming from social networks.

- *Process-mediated data:* This is the information that concerns some business events of interest, like the purchase of a camera in an e-commerce site or the sign-up of clients in a system. This information is *highly structured*, such as relational databases, coming from traditional Business systems.

- *Machine-generated data:* Refers to the data resulting of the tracking of sensors of the physical world (e.g., temperature sensors, human health sensors, GPS coordinates, etc.). This type of source is associated with very large amounts of data, given the constant tracking of the sensors. In general, these are data coming from the so-called Internet of Things.

Given these characteristics of Big Data, the DQ along the dimensions explained in Section 2.1.1 must be quantified using metrics specific to such a context, therefore the typical quality metrics used for structured, process-mediated data must be adapted to this new situation. This is studied in the next section.

Chapter 3

Data Quality in a Big Data Context

3.1 Data Quality Dimensions and Metrics in a Big Data Context

This section will study how the DQ dimensions can be used in a Big Data scenario. The study will focus in *human-sourced generated data*. The next sections will describe how these dimensions can be applied to address the quality of Twitter[1] streams. Metrics for the dimensions defined here will be presented later.

- *Readability (r)* Given a dictionary D, and a collection of words considered as valid in a document x, the *Readability* of x, denoted $r(x)$ is defined as the quotient between the valid words in x and all the words in x, if any, otherwise it is zero. That is, given a set W of the words (valid and non-valid) that are present in the document x, the readability of x is

$$r(x) = \begin{cases} \frac{\#\{w \in W \wedge w \in D\}}{\#\{w \in W\}} & if \quad W \neq \emptyset \\ 0 & if \quad W = \emptyset \end{cases}$$

  In the remainder, the problem to be addressed will refer to tweets in a Twitter stream, thus x will represent a tweet.

- *Completeness (c)* Consider an object x in domain, and an array $props_p$ that contains the names of the properties required to describe x for a given problem p; assume that x is represented as a collection of $(property, value)$ pairs of the form $\{(p_1, v_1), \ldots, (p_n, v_n)\}$, such that v_i is a value for p_i. If a property $p_i \in x$ has associated

[1] http://www.twitter.com

a non-null value v_i, it is called well-defined. There is also a function $validPropsOf$ that, given an object x retrieves the set of well-defined properties in it. The *Completeness* of x, denoted $c(x)$ tells if all the properties in a $props_p$ are well-defined in x, and it is computed as:

$$c(x) = \begin{cases} 1 & if \quad props_p \subset validPropsOf(x) \\ \\ 0 & otherwise \end{cases}$$

Example

Given an object (a tweet) x, such that x = *{ text: "I like Bitcoin", user: null }*, and an array $props_p = [text]$, it follows that $c(x) = 1$.

Consider now $props_p = [text, user]$. In this case, $c(x) = 0$, since the user property is not well defined, because it has a null value.

- *Usefulness (u)* Since this paper is dealing with human-sourced datasets, it will be assumed that this property is directly related to the possibility of (among others):
  - (a) Detecting a sentiment, whether positive or negative, in an object x, say a tweet or post. Therefore, if x reflects a positive or negative feeling about a certain topic or person, x will be considered useful. If the sentiment is neutral, or no sentiment could be computed by a Natural Language Processing (NLP) tool, x will be considered not useful.
  - (b) Detecting the domain or topic of x, for example, politics, marketing, sports, and so on.

  Many other ways of assessing usefulness could be considered, but this is outside the scope of this paper. In the remainder, Usefulness is defined as follows.

$$u(x) = \begin{cases} 1 & if \quad (sentiment(x) = P \ \vee \ sentiment(x) = N) \\ \\ 0 & otherwise \end{cases}$$

- **Trustworthy (*t*)** In a social network (or, in general, for human-sourced datasets) anyone in general can publish any kind of information anywhere, whether truthful or not. Although this is mentioned here for completeness, validating the trustfulness of a post is outside the scope of this paper.

3.2 Computing Big Data Quality

Quality, in general, reflects how much value is delivered in order to solve a particular issue, and how much an object or service fits into a given problem. As discussed above, the definition of DQ is not the same for all contexts and problems, but normally it depends on them. That is, DQ depends on the problem and the domain model at hand. This section provides a wide and general definition of DQ, that can be instantiated as needed.

Problem (p)

A problem p is defined as a string or sequence of characters that defines the problem to be solved in a human-readable way.

Example

A problem can be defined as: *Given this Twitter feeds stream, what are the best quality tweets for the hashtag #2020Elections?*.

Domain Model (X)

The set of objects such that their quality will be measured.

Example

For the case that will be studied in the next section, the domain model is defined as: *A set of Twitter feeds*.

Data Quality Metric (m_{Xp})

A Data Quality Metric is a function $m_{Xp} : X \to [0 \dots 1]$, such that, given $x \in X$, and a problem p, then $m_{Xp}(x) = 0$ if x contains data of very poor quality for the given problem p, and $m_{Xp}(x) = 1$ if x contains data of very good quality to fit the problem.

Data Quality Metric's Weight ($m_{Xp}.weight$)

Each DQ metric has an associated *weight*, which is a scalar value between 0 and 1, that measures the relevance of the metric for solving the problem p.

$$m_{Xp}.weight \in [0,1]$$

Data Quality (Q_{Xp})

Consider a problem p and a domain X, consider a set of metrics $M_{Xp} = \{m_1, m_2, ..., m_n\}$. Each m_i is a DQ metric function. Note that n is an integer number greater than zero and the set M_{Xp} is finite. *Data Quality (Q_{Xp})* is a function $Q_{Xp} : X \to [0 \dots 1]$ such that

$$Q_{XP}(x) = g_{(m_1,m_2,\ldots,m_n)}(x)$$

where g is a function $g : (X \to [0,1])^n \to (X \to [0 \ldots 1])$. The function g is called the *aggregation function*, because aggregates the metrics and weights to provide the final Quality value.

In this paper, the quality of a tweet x will be calculated as

$$Q(x) = g_{(r,c,u)}(x) = \sum_{m=\{r,c,u\}} m(x) * m.weight$$

where r, c and u are the metrics for *Readability*, *Completeness* and *Usefulness* respectively, defined in Section 3.1.

The reader may ask why there is a g function, when Quality could be just defined as the linear combination mentioned above. This is because the aggregation function g may vary depending the case; for example, think of the case *"Quality of my object MUST be zero if any of the given metrics returns a zero value"*. This could be easily achieved by using the product of the metrics instead of the linear combination, just by changing how the g function is defined.

A broader example

The Quality value of the following tweet x, using the weights values $r.weight = 0.5$, $c.weight = 0.25$ and $u.weight = 0.25$, is computed as follows.

```
- text:‘‘I love Big Data Quality m#a!sc["
- id:  1
- coordinates:  [48.864716, 2.349014]
```

- *Readability (r)*

$$r(x) = \frac{\#\{I, love, Big, Data, Quality\}}{\#\{I, love, Big, Data, Quality, m\#a!sc[\}}$$

$$r(x) = \frac{5}{6} = 0.833$$

- *Completeness (c)* Consider that $props_p = \{text, id\}$. Then:

$$\{text, id\} \subset \{text, id, coordinates\},\ and\ c(x) = 1.$$

- *Usefulness (u)* The text provided expresses positive sentiment, thus:

$$sentiment(x) = P, \text{ and } u(x) = 1.$$

Finally, the quality value for x is $Q(x) = 0.83 * 0.5 + 1 * 0.25 + 1 * 0.25 = 0.915$.

Chapter 4

Implementation

This section presents and describes the implementation of the concepts explained in previous sections, applying them to analyze the quality of Twitter feeds streams. The architecture is described first, detailing the technological components and how they interact with each other for capturing, filtering, and displaying the results. Finally, the user interface is described. The goal of the implementation is to develop a system that can let users to analyze a stream of Twitter feeds, based on a particular keyword-led search, and visualize the results to gain insight on the quality of the requested data.

4.1 Architecture

The core of the system is an Apache Kafka[1] cluster. Kafka is a distributed streaming platform for capturing, processing and storing data streams. In a Production-like environment, the implemented cluster should have three nodes running Kafka. A Zookeeper[2] service coordinates the cluster and manages the message topics. Besides the Kafka core, there are three components: one to produce data, one to consume data, and one to process and display data. The Figure 4.1 illustrates these components and their orchestration. The components are briefly described next.

- *Kafka Producer Service:* A Java 8 program exposing a REST API using the Spring Boot[3] framework. This API starts searches over the Twitter API by instantiating a Kafka Producer, fetching the feeds from The Twitter API and publishing the data to a particular Kafka *topic*. A Kafka *topic* is the identifier of the queue in use, where the

[1] https://kafka.apache.org/
[2] https://zookeeper.apache.org/
[3] https://projects.spring.io/spring-boot/

data is stored and waiting to be consumed. There is one *topic* for each different search, identified with a UUID as postfix.

- *User Interface (UI) Proxy:* In order to show the results, the UI needs a proxy that consumes the data from the producer and sends it to the UI via a persistent web socket connection, using the socket.io[4] framework. Also, this Node.js service uses express.js[5] framework in order to expose a REST API, through which the UI can request data in a new search.

- *Web UI:* The web UI is built on top of the dc.js[6] library, which uses the Big Data processing framework crossfilter.js[7] and the data visualization library d3.js[8].

4.1.1 Implementation Details

The data flow works as follows. First, the UI performs a request to the UI Proxy via the REST API in order to start a search using some query, indicating the *topic* ID to use (just a randomly-generated UUID), and some optional advanced parameters (a list of *completeness* properties to analyze, the list of keywords and the weights of each DQ metric). Then, the Proxy performs a REST API call to a Kafka Producer Service instance in order to start the ingestion of the feeds using those parameters. At this point, the Kafka Producer Service creates the topic involved and starts a Kafka producer that fetches the feeds from the Twitter API and publishes its to the topic. After this initialization, the UI proxy starts a Kafka consumer peeking the feeds from the respective topic, and forwards the information to the UI. Finally, the UI processes the records using crossfilter.js and shows the data in real time using the dc.js library.

The Twitter API endpoints are `https://api.twitter.com/1.1/statuses/filter.json` when searching for a particular hashtag or keyword, and `https://api.twitter.com/1.1/statuses/sample.json` when searching for a stream of random feeds without any constraint.

[4] https://socket.io/
[5] http://expressjs.com/
[6] https://dc-js.github.io/dc.js/
[7] http://crossfilter.github.io/crossfilter/
[8] https://d3js.org/

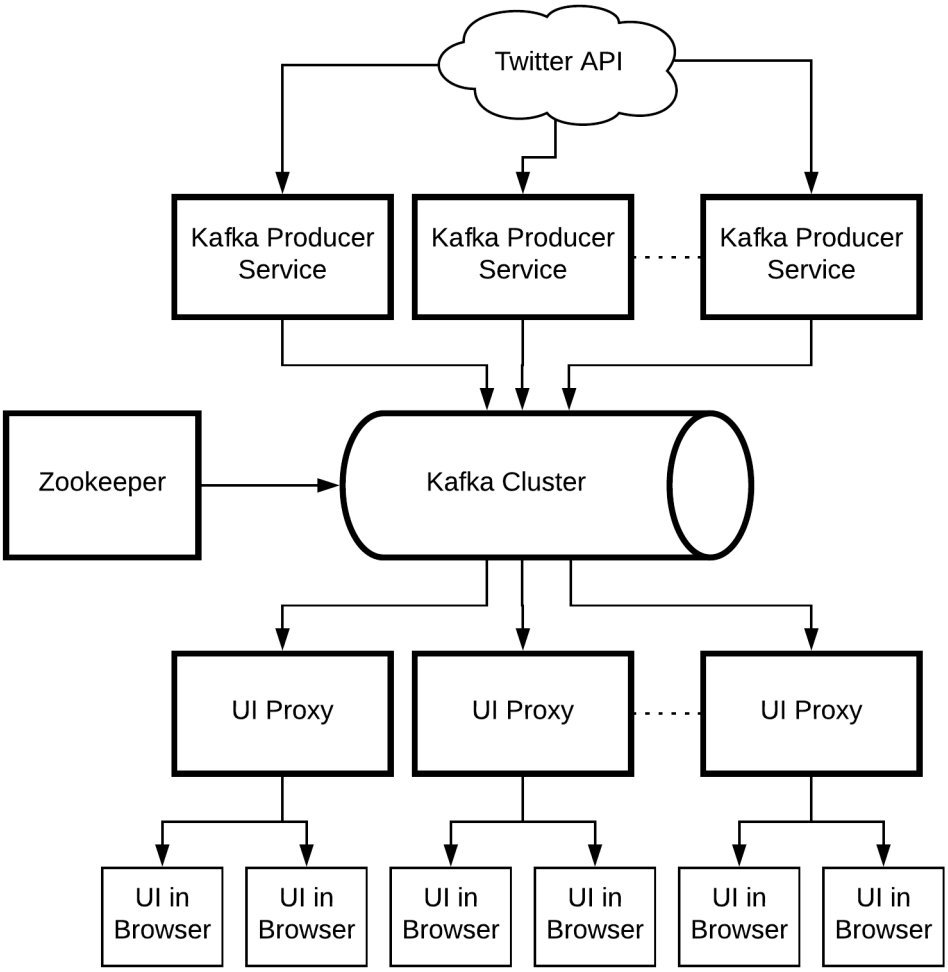

Fig. 4.1 Architecture diagram.

4.1.2 About Performance

The system may scale horizontally as needed, just adding more Kafka producer services. To scale the Kafka Cluster more workers can be added, and Zookeeper will take care of their coordination. The same happens with the UI Proxy, just by scaling horizontally the services.

In order to mitigate the risk of being throttled by the Twitter API, there are 4 sets of Twitter API's access keys. This way, the Twitter client uses a a randomly selected key from this set. We are selecting the key randomly and not in a round-robin policy because the Producer Service may scale if needed and, in that case, we don't want to repeat the sequence of access key selection in all the services.

In order to avoid CPU performance issues when rendering the graphics with dc.js, the UI redraws the charts based on the new provided data every 4 seconds.

For demo purposes, this project was hosted at AWS using EC2 machines.

4.2 User Interface (UI) Details

As mentioned above, the UI makes usage of the crossfilter.js library to process the records, among with the dc.js library in order to render the graphics.

At first sight, the UI shows a form where the user enters the necessary arguments for his search. Then, after hitting the Search button, the numbers of processed tweets will start rising, and also how much of these tweets are Re-Tweets will be displayed, this is, tweets which origin is another tweet, content that is not original from the current user, but shared within the network from another account.

Then, this UI is composed of five major parts: (a) The general DQ results; the DQ final value has been split in 10 clusters, from 0 to 1 in steps of 0.1, and size of the feeds that belong to each cluster are shown in a bar chart. (b) The DQ results per dimension; Readability values are shown in a box plot as their values go from 0 to 1 in a continues matter, while Completeness and Usefulness are shown in two bar charts with the 0 and 1 values in the x axis and the percentage of tweets of each value in the y axis. (c) A deeper analysis on the Completeness dimension, showing a pie chart per each Twitter feed property and how much percentage of the total sample validates that field. (d) The Tweets vs. Re-tweets part, comparing DQ values considering or leaving out re-tweets, respectively; and (e) The verified vs. unverified users part, indicating the DQ for tweets coming from verified or unverified users. All this information may be seen in the Figures 4.2 and 4.3, a screen shot of the UI with the search of the word "trump".

The user may also stop the current search by hitting a Stop button, or just stop it by starting a new search.

Of course, this UI can be easily extended according to the analysis needs, to gain insight on the DQ of the data streams.

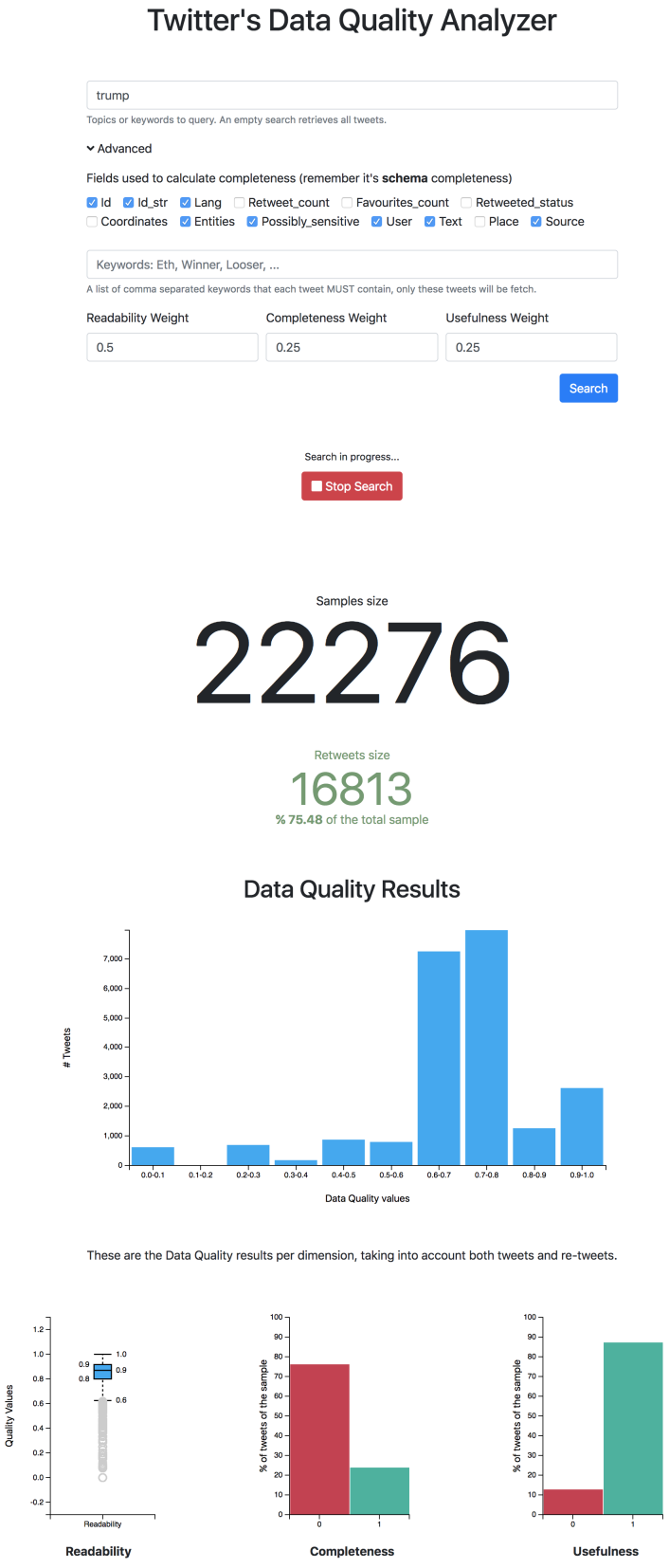

Fig. 4.2 User Interface (UI) - Top.

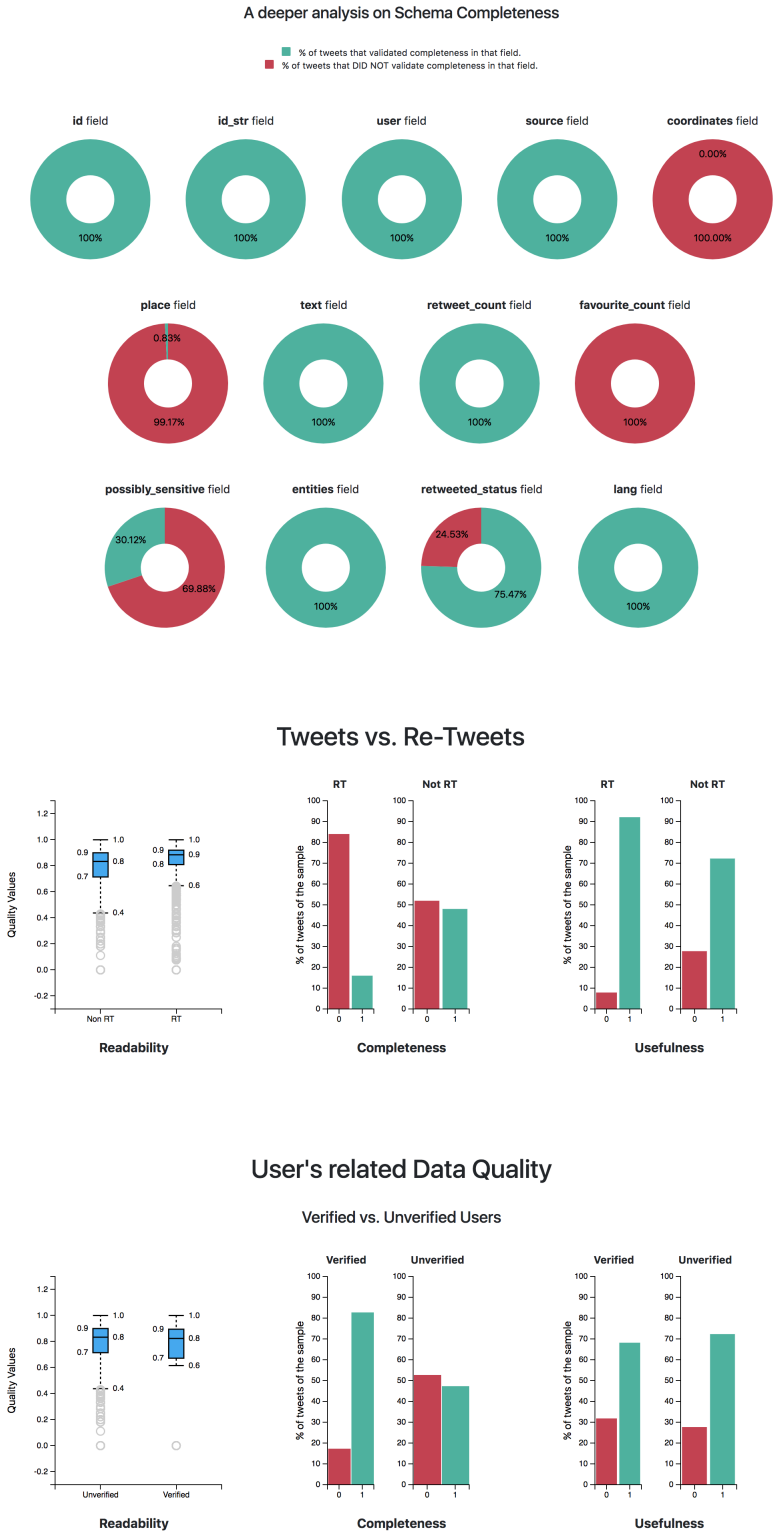

Fig. 4.3 User Interface (UI) - Bottom.

Chapter 5

Experimentation and Discussion

This section describes the experiments performed over the implementation presented in the previous chapter, reports the results, and discusses them.

5.1 Experiment Use-cases Description

The idea of the experiments is to illustrate how the quality of a Twitter stream can be measured using the dimensions and metrics presented above, in order to assess their quality. Of course, there are countless ways in which the quality of data in tweets can be analyzed. These experiments are just aimed at showing how the concepts discussed in this paper can be studied using the tool presented in the previous chapter. The quality dimensions considered in all cases are: readability, completeness, and usefulness, with the metrics described in Section 3.1, and with the following weights: 0.5 for readability, and 0.25 for completeness and usefulness. Of course, the user can modify the weights according to the analysis needs. The dictionary used to check readability, is given in [23] and contains 479,000 english words, including acronyms, abbreviations and even Internet slang. To compute sentiment (for usefulness), the Stanford CoreNLP software was used [17]. In all cases, the overall DQ of the stream is computed, as well as each DQ dimension individually, and the comparisons allowed by the UI are displayed.

The experiments are aimed at:

- (a) Comparing the DQ of the whole stream of tweets, against the DQ of a stream filtered by a set of keywords related to some topic. The hypothesis is that the latter are more likely to have better quality than the former.
- (b) Performing the same comparisons above, but requesting the presence of different sets of properties (that is, changing the requested schema). This will give insight

on which are the properties more likely to be present in a stream of tweets, and investigating the impact on this of the keyword filtering.

- (c) Determining if there is a correlation between the DQ of a stream, and the percentage of re-tweeted tweets that it contains. The hypothesis here is that a tweet with a high number of re-tweets is likely to be of high quality.
- (d) Comparing the quality of tweets from verified and not-verified users.

Next, the problems that will be used to address the goals above, are described.

Problem 1 The first problem p_1 consists in analyzing the tweets in a stream, with no keyword filtering, i.e., all tweets provided by the Twitter API. The UI allows to indicate the set of properties considered for schema completeness. In this case, $props_{p_1} = \{\texttt{id}, \texttt{id\_str}, \texttt{lang}, \texttt{entities}, \texttt{possibly\_sensitive}, \texttt{usr}, \texttt{text}, \texttt{source}\}$.

Problem 2 For the second problem $p2$, the same set $props_{p_1}$ is used, but the stream is filtered using the keyword *Trump*, referring to the U.S. President Donald Trump.

Problem 3 The third problem p_3 consists in analyzing the tweets in a stream like in the previous problem, but considering a larger set of properties, namely all the ones supported by the UI. This allows to study how are the properties distributed in Twitter streams, that is, how many tweets contain the space coordinates or the language, for example.

Again, it must be clear that it is not intended here to draw conclusions on the DQ of Twitter feeds, but to suggest how the concepts and tools presented in this paper can be used to analyze such feeds.

5.2 Results and Discussion

Regarding performance, results were quite satisfactory. Tweets were captured and displayed at a rate of 2000 per minute (for non-filtered tweets), and at about 600 per minute, for filtered tweets, depending on how many tweets pass the filters.

The results obtained for the problems above are commented next. The figures show the status of some runs, such that after several thousands of consumed tweets, the results become stable, that is, the graphs do not change significantly.

The left-hand side of Figure 5.1 shows a portion of the UI, displaying the results obtained from running the system with the conditions of *Problem 1*. The sample size is of 20891 feeds,

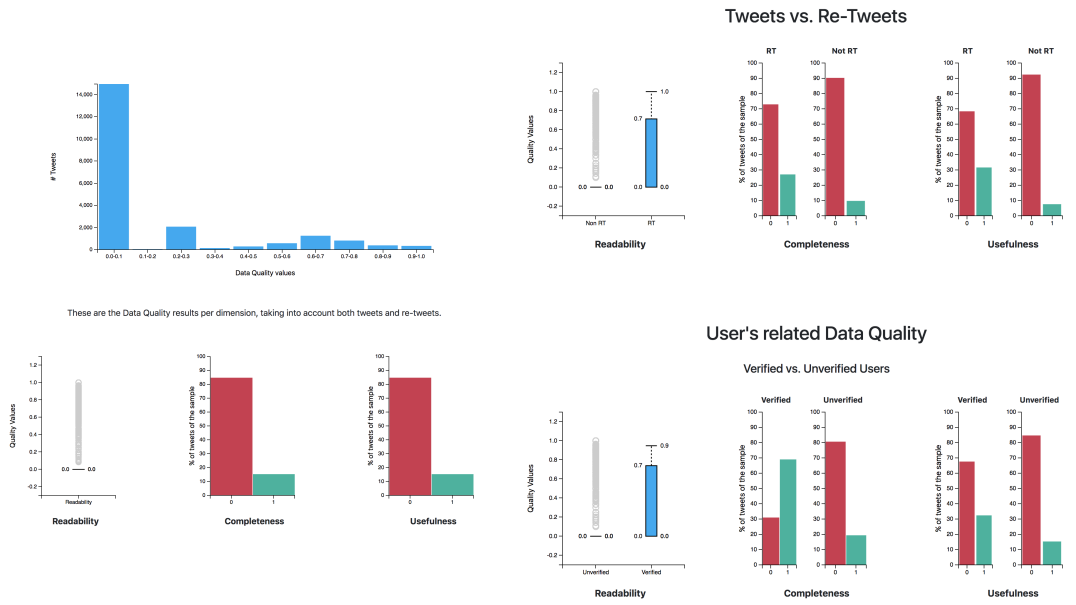

Fig. 5.1 Results for Problem 1.

where 6685 of them are RT (%32 of the total sample). In the graph in the upper-left part, the Y-axis shows the number of tweets, and the X-axis shows the DQ values, in intervals of 0.1. It can be seen that the general DQ is low (most of the tweets fall on the left part of the graph). Readability, completeness and usefulness are shown in the lower-left part. The first one is displayed as a box plot, while the other two are displayed as bar charts. Readability is low, some of the values in the box plot are outliers but most of it has zero Readability. About 16% of the tweets have values in all the fields in $props_{p_1}$, and more that 80% of the tweets have usefulness$=0$. On the right-hand side, it can be seen that all DQ values are better when only re-tweeted tweets are considered, and are also better for tweets posted by verified users.

Figure 5.2 displays (in a way similar to Figure 5.1) the results obtained running the system with the conditions of *Problem 2*, using the keyword Trump. The sample size is of 22276 feeds, where 16813 are RT (%75 of the total sample). The intention is to capture tweets with political content, based on the hypothesis that their quality should be better than for non-filtered tweets. It can be seen that the general DQ is much better, and on the upper part of the right-hand side, most of the tweets fall on the right part of the graph. Readability, completeness and usefulness are much better than in Figure 5.1, in particular the Readability and Usefulness dimensions. All DQ values (like in Problem 1) are better when only re-tweeted tweets are considered (except for Completeness), and are also better for tweets posted by verified users (except for Usefulness, which is very similar for both verified and unverified).

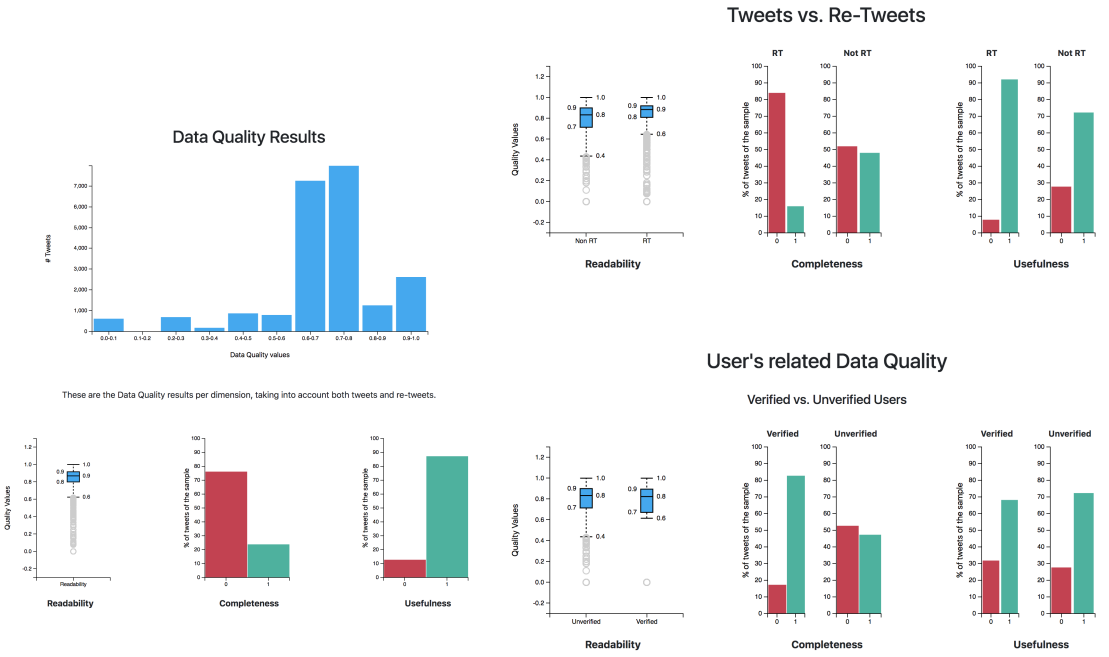

Fig. 5.2 Results for Problem 2.

Figure 5.3 displays, for every property supported by the UI, the percentage of tweets that contains values for that property. On the left-hand side, the values for Problem 1 are displayed, and the right-hand side shows the values for Problem 2. Note that this is shown independently of the set of properties that were checked by the user (i.e., $props_{p_1}$ for Problems 1 and 2). The darker portion of the circles indicates the percentage of tweets containing no valid value for the property. It can be seen that the `coordinates` field (at the top-right corner) has not value for any tweets, that is, no tweet is geo-referenced. Therefore, if this field would have been in the set $props_{p_1}$, the DQ would have been worse, since completeness would have been lower.

Figure 5.4, shows the results for Problem 3. Recall that in this problem, tweets are captured like in Problem 2, but the property set includes all properties supported by the system, for example, the spatial coordinates of the tweets. In Figure 5.3 is is shown that this property is not satisfied by any tweet. However, this does not affect the quality for Problem 2, since `coordinates` is not included in $props_{p_1}$, which is not the case for Problem 3. Therefore, Figure 5.4 shows that the completeness dimension has value 0 (given that all properties are required to be present), which lowers the overall quality.

There has also been noted an increment in the CPU value of the Producer Service while searching for tweets and analyzing them. After digging on this issue, it has been concluded that the sentiment analysis with the Stanford CoreNLP library was the root cause of this issue:

Fig. 5.3 Completeness for Problem 1 (left); Completeness for Problem 2 (right).

when processing the sentiment of about 2000 tweets' text per minute, the CPU increases about 30%. Several solutions might be found for this problem, but due the complexity of this task it is out of the scope of this project. Some possible solutions are: batch this analysis so is not calculated on each message that comes, although this would cause the tweets to be pushed to the Kafka cluster with the delay respective to the batch process. Another solution that the user may do with the current infrastructure is to scale the Producer Service, both horizontally and vertically.

It is also worth to mention the bad completeness value that the Twitter API provides on the `coordinates` and `place` fields. This could be caused by the devices that is creating the tweet not exposing its geographical data.

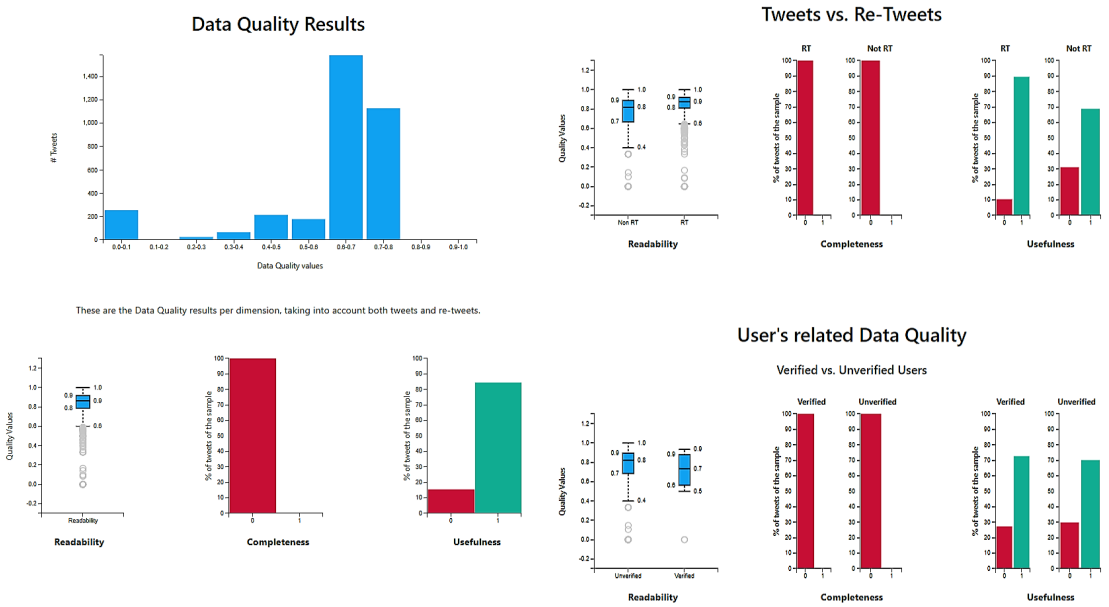

Fig. 5.4 Results for Problem 3.

Chapter 6

Conclusion and Future Work

This paper studied the particularities of assessing data quality in a Big Data context, and presented a system that allows analyzing such quality over Twitter streams in real time. Experiments performed over many different Twitter streams, showed how the concepts presented and tools developed could be applied in a real-world Big Data scenario.

When doing a search by a political query, like "Trump", the Twitter services provide very good feeds in terms of the Readability and Usefulness Quality dimensions. But, in general, it has serious problems on the Completeness Quality dimension in the fields `coordinates` and `place`.

Also, re-tweets tend to have better Quality than non re-tweeted feeds, and these cover more than 70% of the total sample on a political search. This may explain why the good Quality value of the total sample, as we have shown, users tend to share content with a good Quality value.

Still, there is plenty of room for further work. One line of research could be oriented to define more DQ dimensions and metrics for this or other settings (along the lines of [19]), since, as explained, this is typical context-dependent DQ. Also, new and more sophisticated visualization tools could extend and enhance the implemented framework. All of these will be part of future work.

References

[1] (1996). Wang, R.Y., Strong, D.M. Beyond accuracy: What data quality means to data consumers. J. of Management Information Systems.

[2] (2001). U.S. Fish and Wildlife Service. *Information Quality Act.* Section 515 of the Consolidated Appropriations Act, 2001. https://www.fws.gov/informationquality/section515.html.

[3] (2002). Scannapieco, M., Catarci, T. Data quality under a computer science perspective. Archivi & Computer 2 https://www.fing.edu.uy/inco/cursos/caldatos/articulos/ArchiviComputer2002.pdf.

[4] (2002). Wayne W. Eckerson. Data Quality and the Bottom Line. The Data Warehousing Institute, TDWI Report Series.

[5] (2005). White C. Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise. http://ibm.ascential.com.

[6] (2006). Batini, C., Scannapieco, M. Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications, Springer.

[7] (2007). Bolchini, C., Curino, C.A., Quintarelli, E., Schreiber, F.A., Tanca, L. A data-oriented survey of context models. SIGMOD Rec. http://doi.acm.org/10.1145/1361348.1361353.

[8] (2007). Task Team on Big Data: Classification of types of big data https://statswiki.unece.org/display/bigdata/Classification+of+Types+of+Big+Data.

[9] (2008). Data, data everywhere. https://www.economist.com/node/15557443.

[10] (2008). Peter Benson. *ISO 8000 - A New International Standard for Data Quality*. https://www.dataqualitypro.com/iso-8000-new-international-standard-data-quality/.

[11] (2010). U.S. Environmental Protection Agency (EPA). *Open Government Data Quality Plan 1.0*. https://www.epa.gov/sites/production/files/documents/opengov_data_quality_plan.pdf.

[12] (2011). Agrawal D., Bernstein P., Bertino E., Davidson S., Dayal U. Challenges and opportunities with big data.
https://docs.lib.purdue.edu/cgi/viewcontent.cgi?referer=https://www.google.com.ar/&httpsredir=1&article=1000&context=cctech.

[13] (2011). Ciaccia, P., Torlone, R. Modeling the Propagation of User Preferences. In: Proceedings of Conceptual Modeling – ER. pp. 304–317. Springer Berlin / Heidelberg.

[14] (2011). Stefanidis, K., Pitoura, E., Vassiliadis, P. Managing contextual preferences. Inf. Syst.

[15] (2011). Wagner, S., Toftegaard, T.S., Bertelsen, O.W. Increased data quality in home blood pressure monitoring through context awareness. In: 5th International Conference on Pervasive Computing Technologies for Healthcare. Dublin, Ireland. .

[16] (2012). Poeppelmann, D., Schultewolter, C. Towards a Data Quality Framework for Decision Support in a Multidimensional Context. IJBIR.

[17] (2014). Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D. The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations.
http://www.aclweb.org/anthology/P/P14/P14-5010.

[18] (2015). Batini, C., Rula, A., Scannapieco, M., Viscusi, G. From data quality to big data quality. Database Management.

[19] (2015). Firmani, D., Mecella, M., Scannapieco, M., Batini, C. On the meaningfulness of "big data quality" (invited paper). Data Science and Engineering.

[20] (2016). Bean, R. For Big Data, It's 'Show Me The Money' Time.
https://www.forbes.com/sites/ciocentral/2016/03/29/for-big-data-its-show-me-the-money-time/#2e24788527d8.

[21] (2016). Marotta, A., Vaisman, A.A. Rule-based multidimensional data quality assessment using contexts. In: 18th International Conference, DaWaK 2016, Porto, Portugal, September 6-8.

[22] (2017). Informatica.
https://www.informatica.com/products/data-quality/informatica-data-quality.html.

[23] (2018). English-words project.
https://github.com/dwyl/english-words.

[24] (2018). Oracle.
http://www.oracle.com/us/products/middleware/data-integration/enterprise-data-quality/overview/index.html.

[25] (June 2007). SAP. Best practices for high data quality in an SAP environment.

[26] (May 1997). Strong, D.M., Lee, Y.W., Wang, R.Y. Data quality in context. Commun. ACM
http://doi.acm.org/10.1145/253769.253804.