# BigTexts
## *A Framework for the Analysis of Electronic Health Record Narrative Texts based on Big Data Technologies*

Keywords: Electronic Health Record, Big Data, Natural Language Processing, Text Mining, Framework

Abstract: In the healthcare domain the analysis of electronic medical records can be classified as a Big Data problem since it has the three fundamental characteristics: Volume (lots of medical records including exams, treatments, diagnostics, etc.), Variety (structured and unstructured data) and Speed (need for high speed processing to perform certain analyzes). A major drawback is that most of the information contained in medical records is in narrative text fields, being natural language processing and text mining key technologies to enhance the utility of medical records when they are used to perform research, analysis, and in general to improve decision support. Within the tasks performed for processing natural language the most critical, in terms of time consumption, are the pre-processing tasks that allow to give some structure to the original non structured text. Increasing the performance of these tasks is a great opportunity for boosting the analysis of medical narrative texts in such a way that they can contribute to the detection of patterns and rules in the health domain. This paper presents BigTexts, a framework based on Big Data technologies that provides pre-built functionalities for the execution of pre-processing taks over narrative texts favoring the performance of electronic medical records analysis.

## 1 INTRODUCTION

Nowadays, the amount of information stored in the world is estimated around of 1,200 exabytes ($10^{18}$ bytes) (Mayer-Schonberger and Cukier, 2013), which are produced by multiple sources; for example, Google processes over 24 petabytes ($10^{15}$ bytes) of data per-day and Facebook gets more than 10 million photos every hour (Mayer-Schonberger and Cukier, 2013). In the business world, 80% of the data exist in an unstructured format (Murdoch, 2013) and if this amount was only printed books, they would cover the entire surface of the United States in 52 layers thick (Mayer-Schonberger and Cukier, 2013). Given these large amounts of data that are currently being generated, the paradigm of Big Data arises, and is often used to describe vast amounts of diverse data, both structured and unstructured that organizations need to access fast using innovative tools, which together help to accurately determine opportunities for improvement in the management and value generation (Moore et al., 2013). Big Data problem is one that complies with the three Vs: Volume (scale data), Variety (different forms of data) and Speed (on data flow analysis) (McAfee and Brynjolfsson, 2012). As in various sectors, the healthcare domain is generating large amounts of routine data, being the most relevant those contained in Electronic Medical Records (EMR). These become an invaluable source for analysis and research to address public health issues, improved service quality and generally increase the coverage and efficiency of health services (Sengupta, 2013).

Considering the large volume of information contained in EMR, the variety between structured and unstructured data, and the speed to which certain studies are required, the analysis of EMR using computational techniques becomes a problem of Big Data (Moore et al., 2013). In particular, the existence of narrative text in fields such as nursing notes, treatment plans, lab tests, among others; generates a major challenge that requires to be addressed using natural language processing technologies, text mining and Big Data.

To face this requirement from the health domain the aim of this paper is to present Big Texts a Big Data *framework* for the pre-processing of narrative texts contained in Electronic Medical Records (EMRs). Big texts provides pre-built functions for the execution of pre-processing tasks required to applied text mining techniques like tokenization and Part of Speech recognition over the text. In order to improve the performance of pre-processing tasks, Big Texts uses big data technologies like Map Reduce, Yarn and HDFS. It can be used by running a standalone application (with a graphical user interface -

GUI) or using its Application Programming Interface (API) from Java code.

In order to present Big Texts, this paper is organized as follows: Section 2 presents an overview of Big Data technologies and some previous research. Section 3, specifies the characteristics and arquitecture of Big Texts. Then Section 4, describes the evaluation of the *framework* and finally, Section 5 describes conclusions and future work.

# 2 RELATED WORKS

Big data technologies have already been used by different projects. This section describes first the most relevant technologies related to Big Data; then, it presents a set of related proposals for using these technologies to improve the analysis of data in the health domain as well as those created to enhance the execution of pre-processing tasks and text mining techniques in any context.

## 2.1 Big Data Technologies

As mentioned above, any problem of Big Data has three fundamental characteristics such as velocity, variety and volume. However, Big Data is not as such a technology, it is rather a paradigm that has promoted the creation of several products and frameworks; Table 1 presents a subset of them including their classification according to the V property they are intended for.

### 2.1.1 MapReduce

It is a computing style that can be used to manage many large-scale calculations in a way that is hardware fault tolerant (Rajaraman and Ullman, 2012). It consists of two tasks, the first is the Map that takes a dataset and becomes it another one, where the individual elements are divided into tuples, and the Reduce task that takes the output of the Map as input and combines those data in a smaller set of tuples (Zikopoulos and Eaton, 2011).

### 2.1.2 NoSQL

Also called Not Only SQL, is a data management approach and database design that is useful for very large datasets that are distributed. Its purpose is to solve scalability and performance problems that Big Data brings and for which relational databases were not designed. NoSQL is especially useful when a company needs to access and analyze massive amounts of unstructured data or remotely stored on multiple virtual servers in the cloud. A NoSQL database can organize data into objects, pairs key / value or tuples. Contrary to what its name means, NoSQL does not forbid the use of structured (SQL) query language. The most popular NoSQL database is Apache Cassandra, followed by others such as SimpleDB, Google BigTable, MongoDB, HBase, MemcacheDB and Voldemort (WhatIs.com, b).

### 2.1.3 Analytics

It is the science of examining raw data with the purpose of obtaining conclusions and convert them to information. Analytics is used in many industries enabling better strategic decision-making and in science to verify or disprove existing models or theories. Analytics is distinguished from data mining by the scope, purpose and analysis focus. Data miners work with large datasets using sophisticated software to identify undiscovered patterns and establish hidden relationships, where conversely, Analytics focus is on inference, the process of reaching a conclusion, based solely on what is already known by the researcher (WhatIs.com, a).

This science is usually divided into Exploratory Data Analysis (EDA), where new features are discovered in data, and Confirmatory Data Analysis (CDA), where existing hypotheses are tested as true or false. There is also the Qualitative Data Analysis (QDA), which is used in the social sciences to reach conclusions based on non-numerical data as words, pictures or videos (WhatIs.com, a).

### 2.1.4 Hadoop

It is a framework that allows distributed processing of large sets of data across computers clusters using simple programming models. It is designed to scale from a single server to thousands of machines, each offering computational and storage capacity. Instead of relying on hardware to deliver high availability, the library itself is designed to detect and handle failures in the application layer, thus providing a highly available service above computers clusters, where each may be prone to failure (Apache Software Foundation, ).

The framework includes the following modules:

- Hadoop Common: The common utilities that support the other Hadoop modules.
- Hadoop Distributed File System (HDFS): A distributed file system that provides high performance access to application data.
- Hadoop YARN: A framework for task planning and resource management in clusters.

Table 1: Big Data Technologies Classification.

| Tecnology | Sub-technology | Velocity | Variety | Volume |
|---|---|---|---|---|
| NoSQL | | | X | X |
| Cloud computing | Infraestructure as a Service (IaaS) | X | | X |
| | Software as a Service (SaaS) | X | | X |
| | Platform as a Service (PaaS) | X | | X |
| | Analytics as a Service (AaaS) | X | X | X |
| Analytics | Exploratory Data Analysis (EDA) | | X | X |
| | Confirmatory Data Analysis (CDA) | | X | X |
| | Qualitative Data Analysis (QDA) | X | X | X |
| MapReduce | | X | | X |
| Apache Hive | | X | | X |
| Apache Pig | | X | | X |

- Hadoop MapReduce: A YARN based system for the parallel processing of large datasets system.

#### 2.1.5 Apache Hive

Apache Hive provides a SQL-based dialect called Hive Query Language (HiveQL) to query data stored in a Hadoop cluster. Hive translates queries to MapReduce jobs. However, Hive is more appropriate for data warehouse applications, where relatively static data is analyzed, no fast response times are required and the data is not changing rapidly (Capriolo et al., 2012).

#### 2.1.6 Apache Pig

Apache Pig provides an engine for a parallel data flow execution in Hadoop and a language called Pig Latin for the expression of such data flows. This language includes operators for many of the traditional data operations (join, sort, filter, etc.), and also offers the possibility to develop their own functions for reading and writing data processing for users. Since pig runs on Hadoop, it makes a direct use of HDFS and MapReduce (Gates, 2011). Pig also offers the possibility of creating custom processing functions through the UDFs (User Defined Functions) that can be coded in Java and Python.

#### 2.1.7 Cloud Computing

It is a model that allows network access on demand and ubiquitous to a shared computing configurable resource set (eg, networks, servers, storage, applications and services) that can be rapidly provisioned and released with a minimal administrative effort or interaction of a service provider (Purkayastha and Braa, 2013).

Cloud Computing involves typically the following variants:

- IaaS (Infrastructure as a Service) It is based on the principle that investments in hardware become obsolete and organizations do not have to cover these costs in advance, instead of this, organizations rent clouds servers that can provide runtime, middleware, data and applications letting maintenance costs to the supplier. When applications require more resources, infrastructure that supports it can grow as needed (Purkayastha and Braa, 2013).
- PaaS (Platform as a Service) It is a wide infrastructure applications collection (middleware) and services (including application platform, integration, business process management and database services) (Gartner, a).
- SaaS (Software as a Service) Software that is proprietary distributed and remotely managed by one or more suppliers. The supplier delivers the software based on a common set of code and data definitions that are consumed in a model of one-to-many by all contracted customers at any time with a pay per-use base or a subscription-based use (Gartner, b).
- AAAS (Analytics as a Service) The analysis engine (composed by algorithms and query processor) is configured by the supplier, while the rest of transactional data, software required, platform and infrastructure is in the organization. AAAS provider usually houses the analysis engine on IaaS and analyzes it rapidly, possibly co-relating with external data in a public domain and delivers the results to the organization (Purkayastha and Braa, 2013).

#### 2.1.8 Big Data Reference Architecture

Taking into account the large set of technologies involved with the Big Data paradigm, we proposed the following reference architecture that classifies in a

layered model the technologies according to the role they play to enhance the management of Big Data.

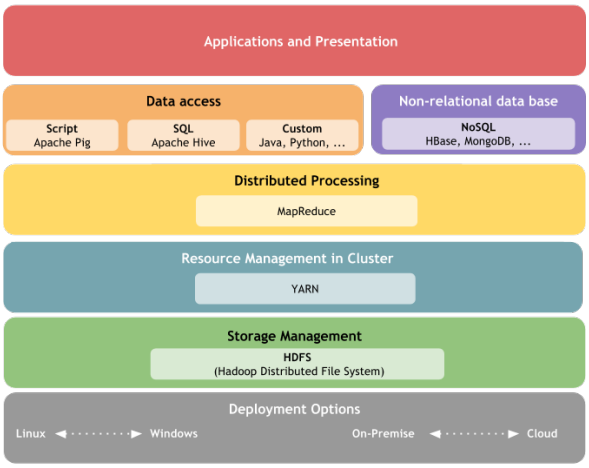


Figure 1: Big Data Reference Architecture.

1. Deployment Options. There are the different options to deploy the architecture of Big Data, having for example operating systems ranging from Linux distributions to Windows, as well as environment installations that can be On-Premise (on the premises of the organization) or in the cloud.
2. Storage Management. This layer stores and processes data in a distributed manner and it is exemplified by HDFS. It is responsible for dividing the files into small pieces (called blocks) through multiple computers (Cluster), allowing map and reduce functions being executed in small subsets of data. This provides the scalability to Big Data processing (Hortonworks Inc, ).
3. Resource Management in Cluster. Provides resource management to allow a wide variety of data access methods to use the information stored using the technology selected in the Storage Management level. This layer is represented by Apache Hadoop YARN, which function is to separate the resource management from the processing components (Hortonworks Inc, ).
4. Distributed Processing. It is the component that allows parallel processing through the phases Reduce and Map.
5. Data Access. Includes the mechanisms through which it can be accessed the set of data previously distributed. There are several options of solutions to access data as Apache Pig and Apache Hive.
6. Non-relational data base. It can also be seen as a way to access data, but with the particularity that it is not based on SQL.
7. Applications and Presentation. It is the layer in which existing or new applications are, that need access to the benefits of the Big Data architecture proposed through APIs and REST services.

## 2.2 Big Data applied to the Health Domain

Table 2 summarizes relevant projects related to the application of Big Data technologies in the healthcare domain. One of the main conclusion of this analysis is that they are mostly theoretical proposals of desirable uses of Big Data technologies in that domain. Similarly, there are few tangible practices and approaches of using this paradigm in products or prototypes that make the expected benefits a reality. Also it was evident that one of the recurring factors in papers is the high amount of data required to analyze by medical professionals and administrative personal to make better decisions, having in mind that one of the major sources of information is the narrative text in the EMRs. Therefore it is worth to review works done using the Big Data paradigm to enhance the management of textual data (Section 2.3).

## 2.3 Big Data Solutions to Improve Text Analysis

The literature review of solutions that provide natural language processing and/or text mining techniques threw that there is an important platform called GATECloud.net that is an open source platform for large-scale word processing in the cloud.

This platform provides several strategies to improve response time in natural language processing, among which had two options: the use of MapReduce and the Cloud Computing technologies management. From these two options the authors of these product recommend the second one, under a model PaaS, since it is not necessary to rewrite the processing algorithms to be executed. GATECloud.net is a Web platform where scientists can conduct experiments on text mining, and whose services are collected through payment for use. It is important to notice that this product does not provide libraries (APIs) that can be used from other systems (Tablan et al., 2012).

In addition to this proposal, there are other projects that create customized solutions that use Big Data technologies to analyze a specific set of narrative texts. It is the case of the project presented in (Das and Mohan Kumar, 2013) that proposes the use of Big Data technologies and text mining techniques to analyze public tweets. It uses technologies such as HBase, Java, REST and Hadoop to enhance

Table 2: Comparison of Big Data Projects in the HelathCare Domain.

| Project | Technology | Problem Solved | Type | Content |
|---|---|---|---|---|
| Use of Big Data and Cloud Computing for operational Health BI (Purkayastha and Braa, 2013) | Cloud Computing | Bridging the digital divide for Medical Information Systems in developing countries | Practical | Text |
| Big Data in personalized health care (Chawla and Davis, 2013) | Data mining | Risk Profile Custom diseases | Theoretical - Practical | Text |
| Big Data in Functional Echocardiography (Sengupta, 2013) | Cloud computing, Robots, Artificial Intelligence | To have more variables in the functional echocardiography | Theoretical | Images |
| Acquisition, compression, encryption and storage of Big Data in Health (Brinkmann et al., 2009) | File processing | Compressing data from electrophysiological recordings | Practical | Images |
| Big Data in medical devices (Meeker and Hong, 2014) | Sensors | Reliability and availability of medical devices | Theoretical | Text |
| Development of an ecosystem of healthcare using IPHIs (Liyanage et al., 2012) | Ontologies | Ecosystem health interoperability for heterogeneous systems | Theoretical | Text |
| Big Data to assess the impact of medical programs (Fox, 2011) | Analytics | Allocate and keep real patients | Theoretical | Texto |

the performance of the analysis. Even though these kind of projects are important to mature the conjunction of Big Data and Text Mining, they do not provide open products or frameworks that can be used by other projects, which is our need to analyze medical records.

As is evident, although there are approaches that try to unify the paradigm of Big Data to text processing, they are not still mature enough or do not provide APIs through which existing systems can run tasks (such as pre-processing) in large amounts of text, providing a great opportunity to attack it by BigTexts (Section 3) the proposal of this paper.

## 3 BIGTEXTS

BigTexts is a framework that allows the execution of pre-processing tasks over narrative texts required to apply text mining techniques. It was developed using Big Data technologies, which allows it to improve the performance when it is necessary to analyze large volumes of medical records and texts contained in them. BigTexts allows the pre-processing of hundreds, thousands or millions of narrative texts when a research required the analysis at institutional, regional or national level. BigTexts provides pre-built functionalities that can be used as an independent application (with a graphical user interface - GUI) or as a library (API) from an existing application.

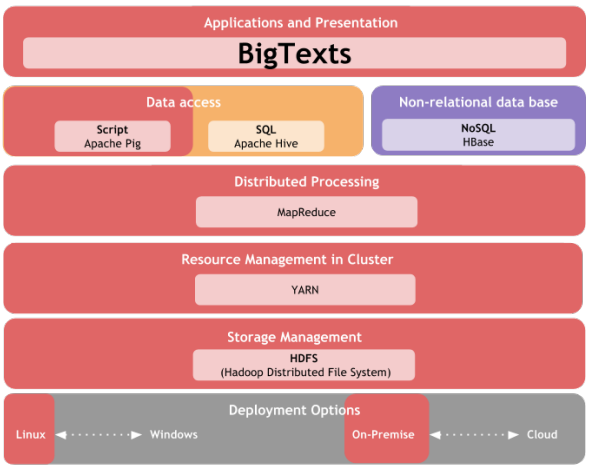


Figure 2: BigTexts in the Reference Architecture.

Figure 2 shows highlighted in red the BigData technologies used in BigTexts taking into account

the reference architecture exposed in Section 2.1.8. BigTexts uses Apache Pig at the data access level, MapReduce for distributed processing, YARN as a resource manager, HDFS for distributed storage and Linux Ubuntu as the operating system. BigTexts, takes the EMR texts and partition them in the Hadoop cluster using HDFS. Using MapReduce to execute the pre-processing tasks in parallel, in order to finally obtain the set of pre-processed documents.

As it is shown in Figure 3, typically the analysis of EMRs are executed sequentially (As-Is). This way of running generates a bottle neck when the number of EMR is large. With BigTexts the execution is parallelized across the set of machines available in the cluster where BigTexts is running. BigTexts takes the EMR texts and partitions them in the Hadoop cluster using HDFS. Using MapReduce it executes the pre-processing tasks in parallel. Finally, it obtains the set of pre-processed texts.

The following sections explain the functionalities provided by BigTexts and its architecture.

## 3.1 BigTexts Tasks

As it was mentioned, BigTexts provides pre-processing tasks required to give some structure to the originally unstructured texts. These tasks allow for example to divide the text in tokens, to identify sentences, to recognize the role of each word in a sentence, to recognize patterns in the text, etc. The following list explains the current tasks provided by BigTexts, which are based on the Stanford CoreNLP library (The Stanford NLP Group, b) and the Weka library(The University of Waikato, ). It is important to notice that these tasks can be extended according to the requirements of the analysis.

- Identification of co-reference: It identifies when two or more expressions in a text refer to the same person or thing. For instance, in the extract "The patient refers he is taking the medication.", "he" refers to the "Patient" (The Stanford NLP Group, a).
- Lemmatisation using Lovins Stemmer algorithm: It reduces a word to its lemma, root or canonical form using the Lovins Stemmer algorithm (Lancaster University, ).
- Named Entity Recognition in English and Spanish: It locates and gives a label to some elements in text taking into account pre-defined categories such as the names of persons, organizations, locations, dates, monetary values, etc.
- Part of Speech in English and Spanish: It recognizes and gives a label indicating the grammar role a word plays in a sentence (The Stanford NLP Group, c). This task also known as POS includes more than 20 labels to identify roles based on eight parts of speech: the verb, the noun, the pronoun, the adjective, the adverb, the preposition, the conjunction, and the interjection.
- Named Entity Recognition based on regular expressions (NE Transducer): It locates and gives a label to some elements in text taking into account rules described using regular expression. For instance, it is possible to identify medication dose using a regular expression that identifies a word recognized as a medication (using Named Entity Recognition in English and Spanish), followed by a number, followed by a word recognized as a unit volume (using Named Entity Recognition in English and Spanish).
- Lemmatisation using Snowball stemmer algorithm for English and Spanish: It reduces a word to its lemma, root or canonical form using the snowball stemmer algorithm.
- Partition text by phrases: It splits a text into its sentences using a set of configurable terminators like ., ?.
- Tokenization: It breaks down a text into tokens using a set of configurable separators like space, punctuation tokens, among others.
- Switch a text to upper case.

The addition of a new pre-processing task is designed to be a simple and flexible process. To do this, a new class is created in which the functionality is implemented and added to the preprocessing tasks catalog[1].

Each one of the previous tasks can be executed using an API provided by BigTexts or directly using its graphical user interface. Figure 4 describe the use cases of BigTexts:

1. CU001-Upload documents: The actor can load one document or a set of documents for further processing in BigTexts.
2. CU002-Select preprocessing tasks: The actor can select the pre-processing tasks he/she wants to execute in BigTexts.
3. CU003-Establish execution order: The actor can give the order in which the selected tasks must be executed and configured on the CU002.
4. CU004-Select destination result: The actor chooses the way the results are going to be de-

[1] An XML file with all the pre-processing tasks and their parameters.

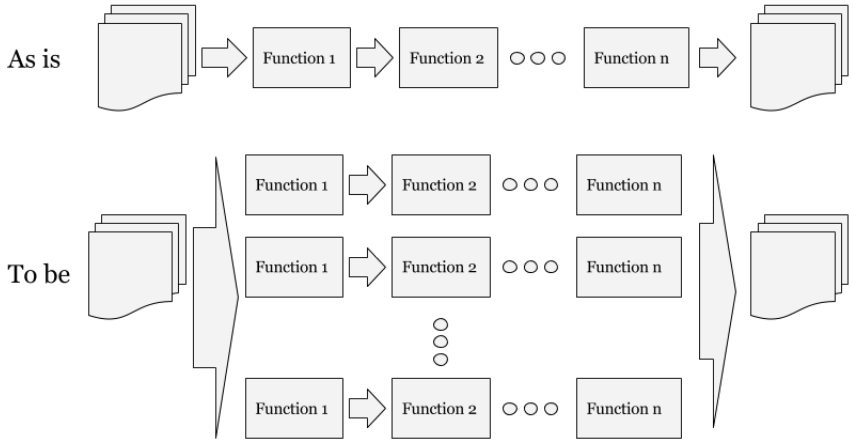


Figure 3: Current Situation vs. BigTexts situation.

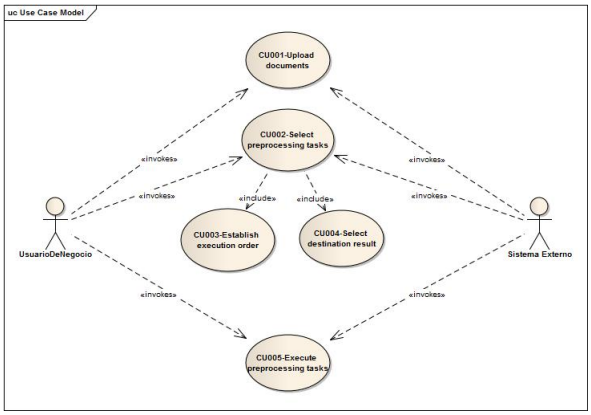


Figure 4: BigTexts Use Cases.

livered, with two options: a location on an FTP directory or in a HDFS directory.

5. CU005-Execute preprocessing tasks: The actor can launch the execution of the pre-processing task list over the uploaded documents.

## 3.2 BigText Process and Architecture

Figure 6 shows an overview of how BigTexts interacts with the BigData technology. As it can be seen it is composed by a Client component (BigTexts Client) and a Server component(BigTexts).

The Client offers a graphical user interface through which the user can select the files to be processed, the pre-processing tasks to run (and their settings) and the delivery method. This client application can also be used by an external application as a library. Once a user sends the files to the server using FTP and indicates the tasks he/she wanted to execute, the client component sends an XML format message to the server queue.

The BigTexts server component connects to the server queue (ActiveMQ) and gets the queued message, transforming it from XML (using the library JAXB[2]) to objects. Subsequently, the converted message to object is executed in the Hadoop cluster by an Apache Pig script that uses UDFs (User Defined Functions) with the pre-processing tasks. The Hadoop cluster is composed of a main machine (the master), which manages both storage and parallel processing, and a number of secondary machines (slaves) that stores data blocks and process them in parallel. Finally, a set of pre-processed documents are delivered in the HDFS or in the FTP directory, according to what the user has selected.

In addition, in order to trace all the executions in BigTexts it provides a PostgreSQL repository that stores the following audit information:

- Date and start time of processing.
- Date and end time processing.
- Date and time of uploading files.
- Name and weight of pre-processed files.
- Parametrized pre-processing tasks and
- The number of available secondary machines.

The internal architecture of BigTexts Server[3] is presented in Figure 6.

1. pig-client-0.13.0.jar: It is the library that allows the connection with the installation of Pig.
2. bigtexts-util-1.0.jar: BigTexts utility classes.
3. log4j-1.2.17.jar: The application log manager.

[2]Java Architecture for XML Binding

[3]bigtexts-1.0.jar: Server application, is responsible for executing the pre-processing tasks into the BigData infrastructure.

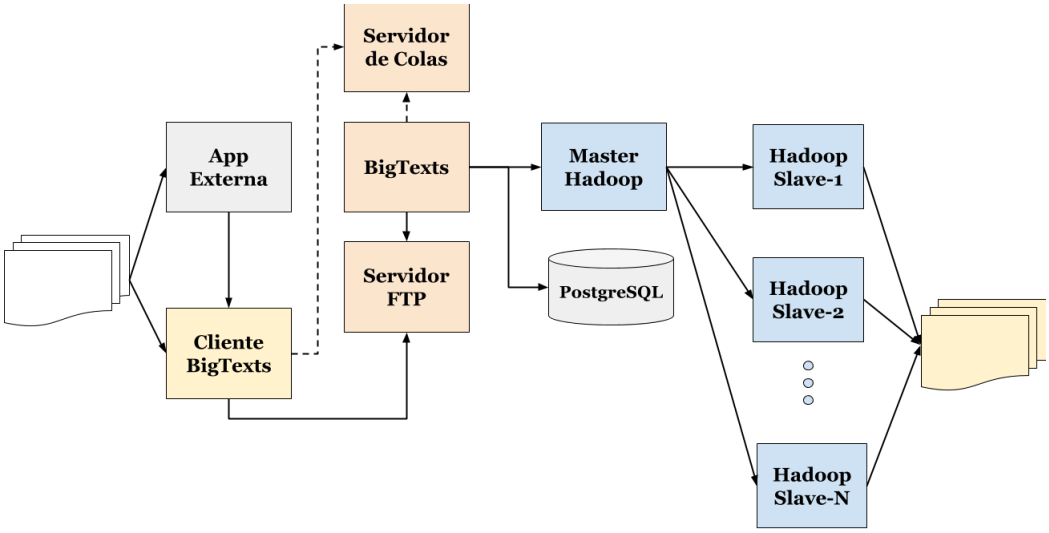


Figure 5: BigTexts.

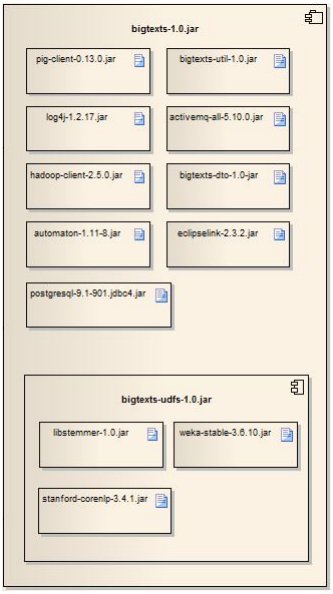


Figure 6: BigTexts Physical View.

4. hadoop-client-2.5.0.jar: Library that allows interaction with HDFS (Hadoop Distributed File System) and Hadoop.
5. bigtexts-dto-1.0-jar: Package with the classes that allow the communication between the BigTexts client and server.
6. automaton-1.11-8.jar: Finite state automata library, used by the Pig client.
7. eclipselink-2.3.2.jar: JPA (Java Persistence API) implementation created by The Eclipse Foundation
8. postgresql-9.1-901.jdbc4.jar: PostgreSQL client.
9. bigtexts-udfs-1.0.jar: Component with the implementations of the pre-processing tasks.
   (a) libstemmer-1.0.jar: Library with the Snowball Stemmer implementation.
   (b) stanford-corenlp-3.4.1.jar: Natural Language Processing library used for the implementation of pre-processing tasks.
   (c) weka-stable-3.6.10.jar: Library for the implementation of Lovins Stemmer.

# 4 VALIDATION

Below is presented both, the scenario in which the validation of the system was performed (Section 4.1) also its results (Section 4.2).

## 4.1 Validation Scenario

In order to validate BigTexts we applied it for the pre-processing of 10.000 narrative texts provided by a general hospital. These texts were previously anonymized and provided in 4 files. The machines used for the configuration of the Hadoop cluster (Table 3) are not high specification computers (server type), but rather common computers in order to simulate real hospital environments, where is not possible to have dedicated servers. A main machine (master) was used and a set of secondary machines (slaves). The client application is installed on the master machine as well as the BigTexts server. All computers were installed with the Ubuntu operating system 14.0.LTS. The machines were connected in a local

Table 3: Cluster Machines.

| Name | Hostname | IP Interna | Brand | Memory | Hard drive | Processor |
|---|---|---|---|---|---|---|
| bigtexts-1 | master | 192.168.0.101 | HP | 3,8 GiB | 155,3 GB | Intel Core 2 Duo CPU E7200 2.53GHz x 2 |
| bigtexts-2 | slave-2 | 192.168.0.102 | HP | 1.9 GiB | 155.3 GB | Intel Core 2 Duo CPU E7200 2.53GHz x 2 |
| bigtexts-3 | slave-3 | 192.168.0.103 | Lenovo | 1.9 GiB | 76.5 GB | Intel Core 2 CPU 4400 2.00GHz x 2 |
| bigtexts-4 | slave-4 | 192.168.0.104 | HP | 1.9 GiB | 155.3 GB | Intel Core 2 Duo CPU E4600 2.40GHz x 2 |
| bigtexts-5 | slave-5 | 192.168.0.105 | HP | 1.9 GiB | 155.3 GB | Intel Core 2 Duo CPU E7200 2.53GHz x 2 |

area network (LAN) establishing static IP addresses. All machines had Intel processors.

The size of the four provided files is the following:

- 1,945,886 bytes (1.945 Mb)
- 1,423,803 bytes (1.423 Mb)
- 437,643 bytes (437.643 Kb) and
- 761 bytes

The validation consisted of the execution of five iterations per file in each of the following preprocessing tasks:

- RegexNamedRecognition: Given a token list with a label for each of them, the system identifies using regular expressions if a word is in the token list and sets the MEDICINE label.
- Tokenizer: Partition files into tokens.
- Tokenizer + POS-Tagger: Two preprocessing tasks are performed. The first one breaks up the file into tokens and then applies Part of Speech task identifying whether each word is a verb, an adjective, an adverb, etc.
- Tokenizer + SnowballStemmer: It breaks up the file into tokens and then identifies the root of each one of them.

## 4.2 Validation Results

The analysis of the data, using figures of lines to identify trends is shown below. For proper display on the graph the file size was divided for 10.000, having all of them in the same scale. There was used a combination of $N$, $M$ and $L$ machines, where $N < M < L$.

Finding that the Regex Entity Recognition task (Figure 7) execution with the 1,945,886 bytes file and $N$ machines obtained an average time of 109.13 seconds, for $M$ machines of 92.78 seconds and for $L$ machines of 82.25 seconds. For the 1.423.803 bytes file, the times were 104.07, 98.33 and 80.45 seconds. Likewise, the file of 437,643 bytes had times of 70.54,65.13 and 58.71 and the file of 761 bytes had times of 61.57, 54 and 51 seconds for $N$, $M$ and $L$ active machines in the cluster. As it can be seen, there is a downward trend on time when the number of machines is increased.

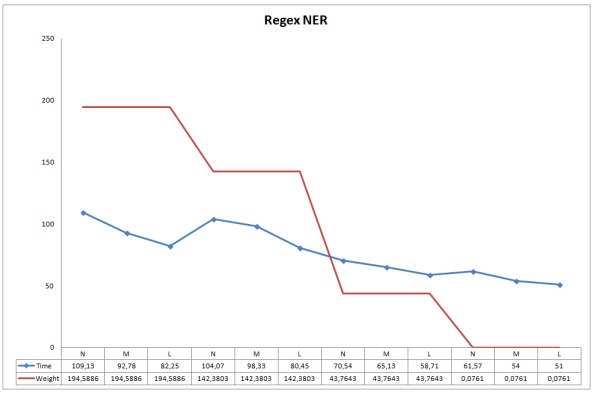


Figure 7: Validation - RegexNER.

For the tokenization task (Figure 8) for the file of 1,945,886 bytes, times obtained were 50.22, 46.63 and 36 seconds. For the file of 1,423,803, time average was 56.25, 41.33 and 33.6 seconds. Similarly, for the 437,643 files and 761 bytes, execution times improved when the number of machines in the cluster was increased.

Regarding the tokenization task added to the Part of Speech (Figure 9) task, for the 1,945,886 bytes files, times were 181, 113.75 and 85.67 seconds; for the 1.423.803 bytes the times were 118.33, 96.67 and 76.33 seconds; and for the 761 bytes the times were

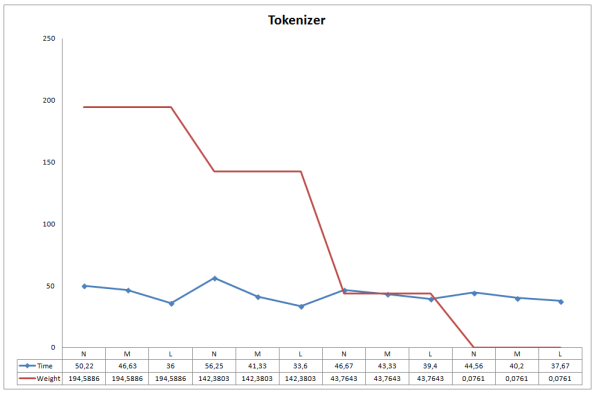


Figure 8: Validation - Tokenizer.

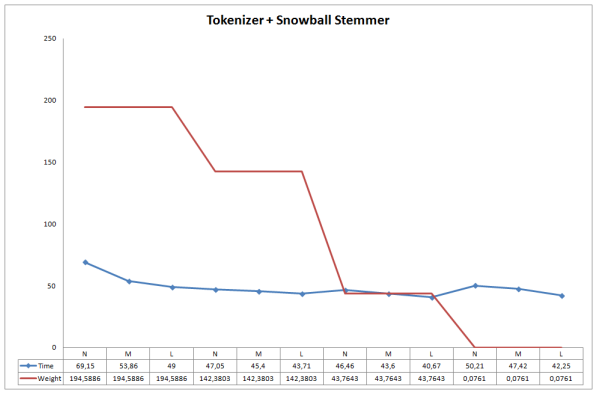


Figure 10: Validation - Tokenizer + Snowball.

69.67 , 65 and 52.5 seconds. These results show an improvement when the number of available machines in the cluster were increased. However, in the case of the 437,643 bytes file there was a not an improvement in time when the number of machines was increased slightly, but a substantial improvement was presented when the number of machines was changed radically.

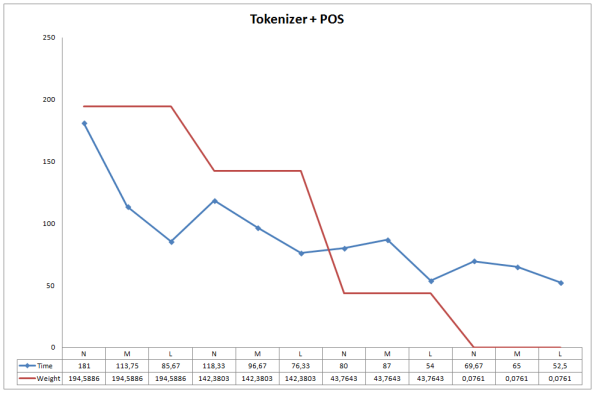


Figure 9: Validation - Tokenizer + POS.

For the tokenization task added to Stemming using Snowball algorithm (Figure 10) the following times for *N*, *M* and *L* machines were obtained, respectively:

- 1,945,886 bytes: 69.15, 53.86 and 49 seconds
- 1,423,803 bytes: 47.05, 45.4 and 43.71 seconds
- 437,643 bytes: 46.46, 43.6 and 40.67 seconds
- 761 bytes: 50.21, 47.42 and 42.25 seconds

Allowing identifying a downward trend as they were adding available machines to the cluster as a possible option.

As it can be seen the execution time for the same file in a same task, differs depending on the number of available slaves. It improves as new slaves were added to the cluster. Besides, it can be seen that the improvement is much more noticeable when complex tasks need to be executed like RegexNER and Part Of Speech. Additionally, the larger is the file, the greater the processing speed, for example, for the 761 bytes file the times for different scenarios (*N*, *M* and *L* machines) are very similar.

# 5 CONCLUSIONS AND FUTURE WORK

BigTexts demonstrated its utility to improve efficiency in narrative text analysis of electronic medical records, this will not only allow performing more analysis projects in the health sector but also allow generating new information products that can be constructed based on BigTexts.

Additionally, BigTexts showed that the addition of an abstraction layer to the entire ecosystem of technology of Big Data and text mining avoids installation, configuration and integration efforts of the Big Data components that, although powerful, are limited in accurate and reliable documentation.

The validation results let us to conclude that it is worth to use the entire infrastructure of Big Data when the analysis include large files or complex tasks, since it is found that for large files or complex tasks, the addition of nodes in the cluster markedly improves performance.

As a future work it is proposed to use cloud computing to address the posed problem in the present project, in order to evaluate the contrast of development effort, use of new technologies and an approach of scalability like that, compared to the one used in BigTexts. Additionally, another field of work is the development of more UDFs to increase the catalog of BigTexts pre-processing tasks.

# REFERENCES


Apache Software Foundation. Hadoop. http://hadoop.apache.org/.

Brinkmann, B. H., Bower, M. R., Stengel, K. A., Worrell, G. A., and Stead, M. (2009). Large-scale electrophysiology: Acquisition, compression, encryption, and storage of big data. *Journal of Neuroscience Methods*, 180(1):185–192.

Capriolo, E., Wampler, D., and Rutherglen, J. (2012). *Programming Hive*. O'Reilly Media, 1 edition edition.

Chawla, N. V. and Davis, D. A. (2013). Bringing big data to personalized healthcare: A patient-centered framework. *Journal of General Internal Medicine*, 28(S3):660–665.

Das, T. and Mohan Kumar, P. (2013). Big data analytics: A framework for unstructured data analysis. *School of Information Technology and Engineering, VIT University*.

Fox, B. (2011). Using big data for big impact. leveraging data and analytics provides the foundation for rethinking how to impact patient behavior. *Health management technology*, 32(11):16.

Gartner. Platform as a service (PaaS). http://www.gartner.com/it-glossary/platform-as-a-service-paas.

Gartner. Software as a service (SaaS). http://www.gartner.com/it-glossary/software-as-a-service-saas/.

Gates, A. (2011). *Programming Pig*. O'Reilly Media, 1 edition edition.

Hortonworks Inc. Hortonworks data platform delivers enterprise hadoop. http://hortonworks.com/hdp/.

Lancaster University. What is stemming? http://www.comp.lancs.ac.uk/computing/research/stemming/general/.

Liyanage, H., Liaw, S.-T., and de Lusignan, S. (2012). Accelerating the development of an information ecosystem in health care, by stimulating the growth of safe intermediate processing of health information (IPHI). *Informatics in primary care*, 20(2):81–86.

Mayer-Schonberger, V. and Cukier, K. (2013). *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt, Boston.

McAfee, A. and Brynjolfsson, E. (2012). Big data: The management revolution. *Harvard business review*, 90(10):p60–68.

Meeker, W. Q. and Hong, Y. (2014). Reliability meets big data: Opportunities and challenges. *Quality Engineering*, 26(1):102–116.

Moore, K. D., Eyestone, K., and Coddington, D. C. (2013). The big deal about big data. *Healthcare financial management: journal of the Healthcare Financial Management Association*, 67(8):60–66, 68. PMID: 23957187.

Murdoch, T. B. (2013). The inevitable application of big data to health care. *JAMA*, 309(13):1351.

Purkayastha, S. and Braa, J. (2013). Big data analytics for developing countries – using the cloud for operational BI in health. *The Electronic Journal of Information Systems in Developing Countries*, 59(0).

Rajaraman, A. and Ullman, J. D. (2012). *Mining of massive datasets*. Cambridge University Press, New York, N.Y.; Cambridge.

Sengupta, P. P. (2013). Intelligent platforms for disease assessment. *JACC: Cardiovascular Imaging*, 6(11):1206–1211.

Tablan, V., Roberts, I., Cunningham, H., and Bontcheva, K. (2012). GATECloud.net: a platform for large-scale, open-source text processing on the cloud. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983):20120071–20120071.

The Stanford NLP Group. Coreference resolution. http://nlp.stanford.edu/projects/coref.shtml.

The Stanford NLP Group. Stanford CoreNLP. http://nlp.stanford.edu/software/corenlp.shtml.

The Stanford NLP Group. Stanford log-linear part-of-speech tagger. http://nlp.stanford.edu/software/tagger.shtml.

The University of Waikato. Weka 3 - data mining with open source machine learning software in java. http://www.cs.waikato.ac.nz/ml/weka/.

WhatIs.com. What is data analytics (DA)? http://searchdatamanagement.techtarget.com/definition/data-analytics.

WhatIs.com. What is NoSQL (not only SQL)? http://searchdatamanagement.techtarget.com/definition/NoSQL-Not-Only-SQL.

Zikopoulos, P. and Eaton, C. (2011). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*. McGraw-Hill Osborne Media, 1 edition edition.