Big Data Discovery and Management

By

Sakhr Awadh Mohammed Saleh

A thesis submitted for the requirements of the degree

of Master of Science [Computer Science]

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

KING ABDULAZIZ UNIVERSITY

JEDDAH – SAUDI ARABIA

Rajab 1436 H – May 2015 G

Big Data Discovery and Management

By

Sakhr Awadh Mohammed Saleh

A thesis submitted for the requirements of the degree

of Master of Science [Computer Science]

Supervised By

Prof. Dr. Fathy ELBouraey Eassa

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

KING ABDULAZIZ UNIVERSITY

JEDDAH – SAUDI ARABIA

Rajab 1436 H – May 2015

Big Data Discovery and Management

By

Sakhr Awadh Mohammed Saleh

This thesis has been approved and accepted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

EXAMINATION COMMITTEE

	Name	Rank	Field	Signature
Internal Examiner	Hassanin M. AL-Barhamtoshy	Professor	Information Technology	
External Examiner	Ali M. Rushdi	Professor	Computer Engineering	
Advisor	Fathy ELBouraey Eassa	Professor	Software Engineering	

KING ABDULAZIZ UNIVERSITY

Rajab 1436 H – May 2015 G

Dedication

This thesis is dedicated to my father, my mother and my family

ACKNOWLEDGMENT

All praise be to ALLAH for giving me the knowledge and the strength to succeed in and complete this thesis. Also I would like to extend my sincere thanks, appreciation, and praise to my Prof. Dr. Fathy Elbouraey Eassa, the supervisor of this research, who has guided me with his effort and time to successfully overcome all difficulties and obstacles that I have faced and feared in my studies. I would like also to thank my parents, my wife, my family, and my friends for their gallant support.

Big Data Discovery and Management

Sakhr Awadh Mohammed Saleh

Abstract

In this research, we developed a technique and manager for collecting metadata of existing Big Data in an enterprise or organization. All collected metadata are stored in metadata storage. Also, in this research, we developed a technique for discovering simple knowledge from existing Big Data (Twitter & websites) and extracted the correlation between different Big Data.

Since Big Data are distributed across a large number of remote machines, we used mobile agents technology to build our manager. Using mobile agents and the metadata of Big Data will solve the Big Data transportation challenge in addition to the management challenge.

Table of Contents

LIST OF FIGURES

List of Tables

Chapter I

INTRODUCTION

In this chapter, we will discuss the rationale of the research and our goal, then present the definition of Big Data and their attributes. We will discuss Big Data techniques and when data become Big Data.

1.1 Rationale

When we work with Big Data, we face many challenges and issues, such as management and transportation challenges, as Big Data are distributed across a large number of remote machines and stored in large, rapidly increasing volume.

Managing massive volumes is difficult and if we want to analyze and extract information from Big Data using the traditional method, "bring the data to the code", it will take long time to transmit large volumes from the source to the processing point. Assuming a 1 gigabyte per second network transfer rate and bandwidth of about 100 megabytes, transferring one Exabyte would take about 2,800 hours [1], which is a long time. Therefore, we need to transmit only the resulting information from existing Big Data. Consequently, we introduced a manager for collecting the metadata of existing Big Data in an enterprise or organization. All collected metadata

are stored in metadata storage and we built a technique for discovering simple knowledge from the existing Big Data.

We used mobile agent technology to build our manager to "bring the code to the data." Using mobile agents and the metadata of Big Data will solve the Big Data transportation challenge in addition to the management challenge.

1.2 The Goal

To solve the challenges of Big Data distribution and transportation by building a manager for the collection of metadata and developing a way to extract knowledge from them.

1.3 Big Data definition

IDC defines Big Data technologies as:

"A new generation of technologies and architectures designed to extract value economically from very large volumes of a wide variety of data by enabling high velocity capture, discovery and analysis" [2]. The Journal of Science 2008 published "Big Data: science in the petabyte era", noting that Big Data "Represents the progress of the human cognitive processes, usually includes data sets with sizes beyond the ability of current technology, method and theory to capture, manage, and process the data within a tolerable elapsed time" [3]. Recently, Douglas and Laney noted: "Big Data are high-volume, high-velocity, and/or high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization" [4].

1.4 Big Data challenges

Big Data face many challenges. According to [1, 5], we can summarize Big Data challenges as:

- **Heterogeneity** (Variety): Different environments and types of data.
- **Inconsistency:** Big Data increasingly include information provided by increasingly diverse sources, of varying reliability. Uncertainty, errors, and missing values are endemic, and we must manage these.
- **Management:** Managing large, rapidly increasing volumes is difficult.
- **Transportation:** Assuming a 1 gigabyte per second network transfer rate and a bandwidth of about 100 megabytes, transferring one Exabyte would take about 2,800 hours [1].
- **Timeliness** (Velocity): As the data grow in volume, we need real-time techniques to summarize and filter what is to be stored in real time.
- **Privacy:** is another huge concern in the context of Big Data. For electronic health records, there are strict laws governing what can and what cannot be done.

1.5 When do data become Big Data?

Big Data's concept depends on the capabilities of the systems, and in the higher level it depends on the capabilities of the organization, which means that, if we have 1000 TB of variant data like text, audio, and video, and structure the data, then some companies will regard them as Big Data, and other companies will not.

There has been a data explosion in the world because of the many devices that can generate huge data, like smart phones, sensors and connected computers to the web. Other resources of data are social network sites like Facebook, LinkedIn and Twitter. By 2003, humans could create five Exabytes (2^{18} Bytes) of data, but this data now will be created in two days! [6] IDC predicted that the volume of digital content would grow to 2.7 ZB (1 ZB = 1 billion terabytes) in 2012, and to 8 ZB by 2015, and

that 90% of the data would be unstructured, like images, videos, MP3 music files, and other files based on social media and Web-enabled workloads [7].

IBM indicates that, every day, 2.5 exabytes of data are created, and 90% of data has been produced in the last two years [8].

A personal computer holds about 500 gigabytes ($10^9\ bytes$), so it would require about 20 billion PCs to store all of the world's data. In the past, the human genome decryption process took approximately ten years, but now it takes no more than a week. Only Google has more than a million servers around the world. There have 6 billion mobile subscriptions in the world and, every day, 10 billion text messages are sent [6].

Cisco IBSG predicts there will be 25 billion devices connected to the Internet by 2015 and 50 billion by 2020 [9].

According to "Big Data: A Review", "In 2012, The Human Face of Big Data accomplished as a global project, which is centering in real time collect, visualize and analyze large amounts of data. According to this media project, many statistics are derived. Facebook has 955 million monthly active accounts using 70 languages, 140 billion photos uploaded, 125 billion friend connections, every day 30 billion pieces of content and 2.7 billion likes and comments have been posted. Every minute, 48 hours of video are uploaded and every day, 4 billion views performed on YouTube. Google support many services as both monitories 7.2 billion pages per day and processes 20 petabytes ($10^{15}\ bytes$) of data daily also translates into 66 languages. 1 billion Tweets every 72 hours from more than 140 million active users on Twitter. 571 new websites are created every minute of the day" [6].

1.6 3Vs attributes

Big Data have the 3Vs attributes: Volume, Velocity and Variety, so Big Data depend not only on the size of the data but also on the data velocity and data variety.

- Volume – The very large size and traditional DBMSs cannot manage them; Volume is a relative word, that depends on the organizational capabilities, so some data appear for some organizations as Big Data and other data do not.
- Velocity – Data are generated in real time, with demands for usable information to be served up immediately [10]. Different resources generate data at high frequency, so we need to store, retrieve and process these Big Data in real time, which is what we call Velocity.
- Variety – the types of data vary, and there are three types of data: Structured, Semi-Structured and Unstructured data.

1. Structured data: Database engines like Oracle, DB2, MySQL, and MSSQL Server are relational database engines, which store data in tables using fixed schema and have a relationship with these tables.
2. Semi-structured data: These are a form of structured data that does not conform to the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. Therefore, they are also known as a schema-less or self-describing structure [11].
3. Unstructured data: documents, images, audio and video files.

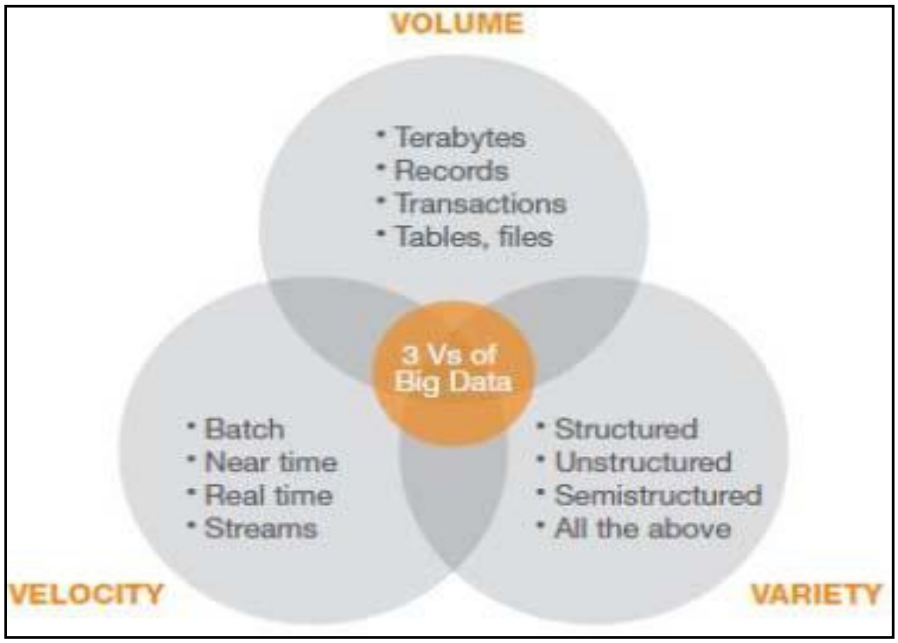

Figure 1.1 Three Vs of Big Data [8]

1.7 Big Data techniques or architecture

- **Hadoop**

Hadoop is an integrated environment for distributed data storage and parallel processing. It provides high availability by replicating data on distributed nodes, and also ensures high-performance by using the Map Reduce functions and executing tasks in parallel.

Traditional RDBMS cannot handle Big Data. Because of this, Google implemented the solution by "using a processing model called MapReduce. There are more solutions to handle Big Data, but the most widely used one is Hadoop, an open source project based on Google's MapReduce and Google File System. Hadoop was founded by the Apache Software Foundation." [12] Hadoop has two major components: the Hadoop distributed file system (HDFS) and Map-reduce methods.

- **HDFS**

"The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware", and HDFS is part of the Apache Hadoop Core project [13].

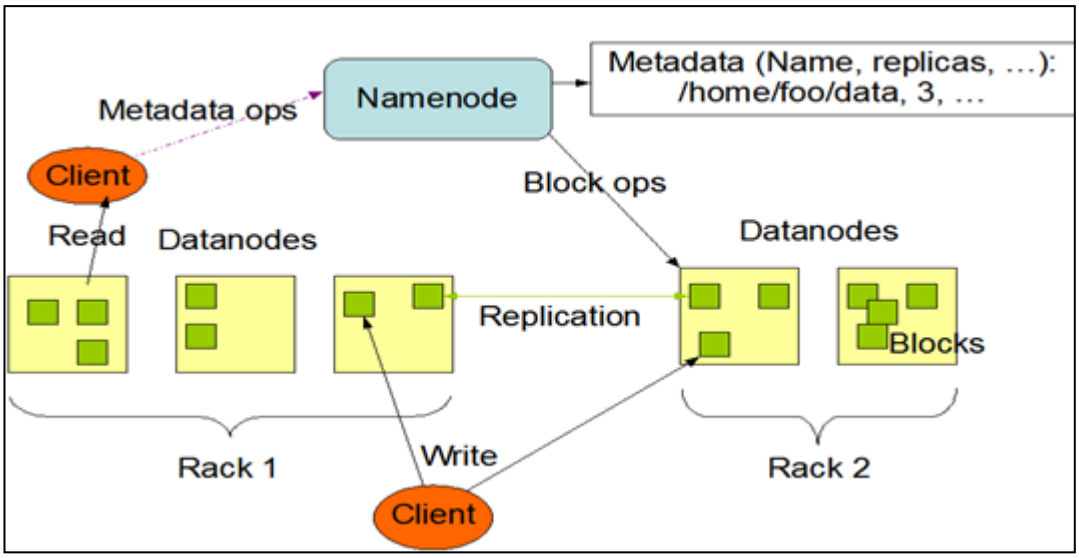

Figure 1.2 HDFS Architecture [13]

- **Map-Reduce**

MapReduce is a programming framework for distributed computing which was created by Google using the divide and conquer method to break down complex Big Data problems into small units of work and process them in parallel [14].

In a simple way, Map-Reduce is divided into two methods: the Map and the Reduce function. The Map method divides the problem into sub-problems and then executes it in parallel on hundreds or thousands of cheap servers or PCs, and then extracts partial results for each task.

The Reduce method assembles and integrates partial results, and then aggregates them into one result and returns it to users.

Google defines MapReduce as "a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key" [15].

- **NoSQL Database engine**

NoSQL is a new type of engine that does not support relations or SQL like RDBMS. It was designed to achieve performance or reliability features that are incompatible with the flexibility of SQL, JOIN, TRANSACTION, LIMIT and non-indexed WHERE usually are not supported by NoSQL engines. In addition, these engines are schema-less or have no fixed table schemas [16].

Martin Fowler and Pramod Sadalage [17] stated, "There is no standard definition of what NoSQL means" The term began with a workshop organized in 2009, but there is much argument about what databases can truly be called NoSQL.

However, while there is no formal definition, there are some common characteristics of NoSQL databases:

1. They do not use the relational data model, and thus do not use the SQL language
2. They tend to be designed to run on a cluster
3. They tend to be Open Source
4. They do not have a fixed schema, allowing you to store any data in any record [17].

Chapter II

LITERATURE REVIEW

2.1 Big Data's history

The first article to use the term Big Data [18] in October 1997 at ACM was by Cox and Ellsworth, entitled "Managing Big Data for Scientific Visualization" They stated that "Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of Big Data. When data sets do not fit in the main memory (in the core), or even on the local disk, the most common solution is to acquire more resources" [19].

However, there are many indicators of a data explosion or Big Data before that date.

In 1938, the Library of Yale University contained 2,748,000 volumes [20] and in 1944 Fremont Rider, Wesleyan University Librarian, published "The Scholar and the Future of the Research Library" He estimated that Yale Library will have approximately 200,000,000 volumes, which will occupy over 6,000 miles of shelves in the year 2040.

October 2000 - Peter Lyman and Hal R. Varian at UC Berkeley published "How Much Information?" in which they stated that "the world's total yearly production of print, film, optical, and magnetic content would require roughly 1.5 billion gigabytes of storage. This is the equivalent of 250 megabytes per person for each man, woman, and child on earth" [21].

October 2003 - Google published an academic paper in ACM to explain its own file system (Google file system). Google said "We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability". This file system " demonstrates the qualities essential for supporting large-scale data processing workloads on commodity hardware" [22].

In 2004, Google published "MapReduce: Simplified Data Processing on Large Clusters." In this paper, Google shows the world how it computes large data on hundreds or thousands of servers [15].

In 2006, Google also published "Bigtable: A Distributed Storage System for Structured Data" paper. Google states "Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance" [23].

IDC released a white paper entitled "The Expanding Digital Universe: A Forecast of Worldwide Information Growth through 2010" in March 2007. They stated that "In 2006, the amount of digital information created, captured, and replicated was $1{,}288 * 10^{18}$ bits. In computer parlance, that is 161 exabytes or 161 billion

gigabytes. This is about 3 million times the information contained in every book ever written.

Between 2006 and 2010, the information added annually to the digital universe will increase more than six fold from 161 exabytes to 988 exabytes" [24]. Figure 2.1 shows a 6 fold growth in four years.

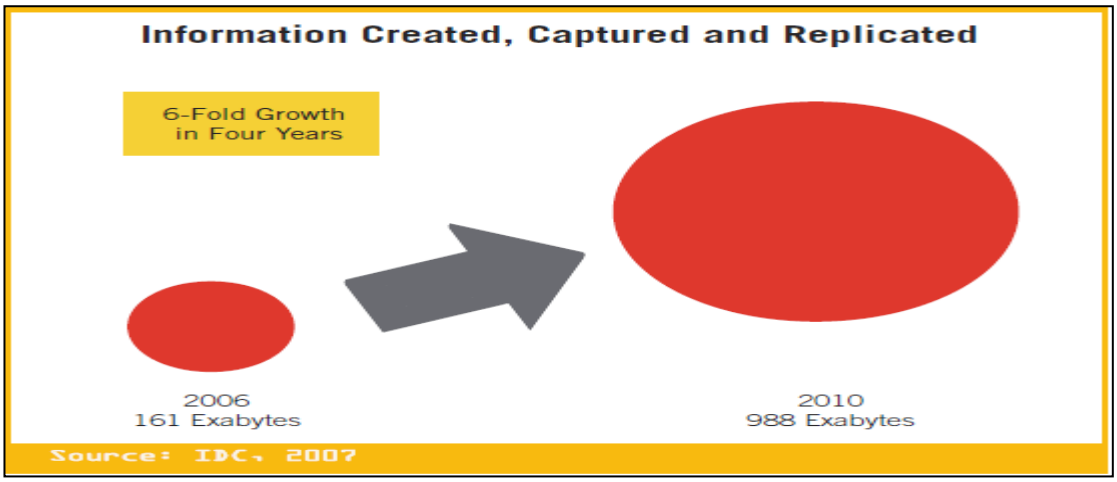

Figure 2.1 Information Created, Captured and Replicated

February 2011 - Martin Hilbert and Priscila Lopez published "The World's Technological Capacity to Store, Communicate, and Compute Information" in Science. "We estimated how much information could possibly have been stored by the 12 most widely-used families of analog storage technologies and the 13 most prominent families of digital memory, from paper-based advertisement to the memory chips installed on a credit card (Figure 2.2). The total amount of information grew from 2.6 optimally compressed exabytes in 1986 to 15.8 in 1993, over 54.5 in 2000, and to 295 optimally compressed exabytes in 2007" [25].

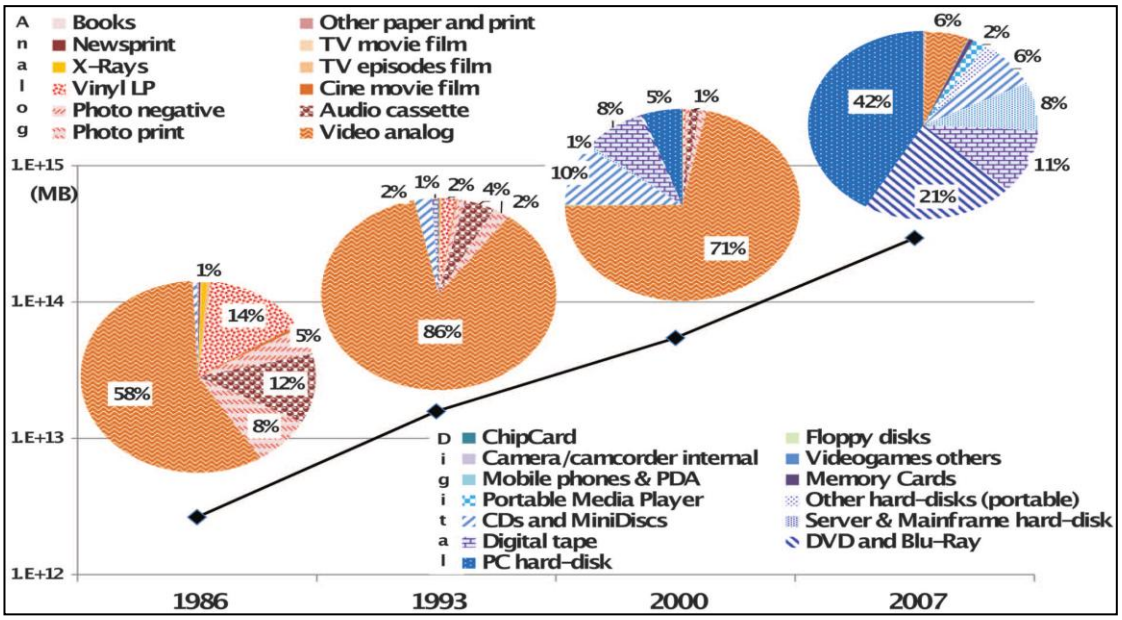

Figure 2.2 Four World's technological installed capacity to store information [25]

May 2012 - Danah Boyd and Kate Crawford published "Critical Questions for Big Data" in Information, Communications, and Society. They state that "Big Data is, in many ways, a poor term. As Manovich (2011) observes" and comment that "We define Big Data a cultural, technological, and scholarly phenomenon that rests on the interplay of:

(1) Technology: maximizing computation power and algorithmic accuracy to gather, analyze, link, and compare large data sets.

(2) Analysis: drawing on large data sets to identify patterns in order to make economic, social, technical, and legal claims.

(3) Mythology: the widespread belief that large data sets offer a higher form of intelligence and knowledge that can generate insights that were previously impossible, with the aura of truth, objectivity, and accuracy" [26].

Now, there are many database engines that can manage Big Data, the types of these engines are NoSQL engines like MangoDB, Bigtable, HBase, Cassandra, etc.

2.2 Background

In this section, we will define and explain a few of the concepts that will help us to understand this thesis.

- Generative model

The generative model is a model for randomly generating observable data, typically given some hidden parameters. It specifies a joint probability distribution over observation and label sequences [27].

- Topic models

Topic models are generative models which are a set of algorithms that provide methods for discovering and annotating a corpus of documents by analyzing the words in those documents to discover the themes running through them. Topic modeling describes a document as a mixture of topics, each of which is a collection of words that occur frequently with each other. Topic modeling is used to organize and summarize corpuses that we cannot annotate [28].

- LDA (Latent Dirichlet Allocation)

  LDA is a generative model that learns a set of latent topics for a document collection [29]. The input to LDA is a bag-of-words representation of the individual documents, and the output is a set of latent topics and an assignment of topics to every document in the collection [30].

- Mobile agents

  An agent is a computer program that performs actions on the user's behalf [31]. A mobile agent is an agent that can migrate from one computer to another to perform its tasks, even if the network is disconnected. Once the network comes up, it will send the results to the source computer.

- **Why topic modeling?**

We are using topic modeling to classify Micro-blog messages such as Twitter for the following reasons: as [32] paper,

- Topic models are bag-of-words models that means that the topic models do not make any assumptions about the ordering of the words [33]. Thus, topic models are suited to handling the irregular grammar of Twitter messages.
- Topic models are suitable for performing clustering and make comparisons easy, as they represent each document as a vector to describe its distribution over the topics.
- Training a topic model is easy since it uses unsupervised learning. It saves the effort required to create labeled data and training classifiers using such labeled data.
- We do not need to label our training dataset because the topic models use unsupervised learning.
- Topic models identify the hidden relationships between the elements in the data, so they can handle misspellings and other variant forms of messages.

2.3 Related work

In this section, we demonstrate the related work; we group them by type as social media and free text.

2.3.1 Free text

In the **"Metadata Management in Big Data"** paper, the author proposes five components: Metadata Discovery, Metadata collection, Metadata Governance, Metadata storage and Metadata distribution to be managed enterprise Big Data metadata, because Big Data introduce large volumes and different data formats. He adds that "Discovering metadata is critical in Big Data" [34].

The **GLEAN system** provides a data discovery environment for users of large scientific computing. GLEAN uses three types of metadata: fine-grained metadata, provenance metadata, and a summary of the datasets. The authors have used the Granules cloud runtime to orchestrate the MapReduce computations that extract metadata from the datasets

1. Access Interfaces:

Users can add new dataset entries, issue queries, and customize datasets through these interfaces.

2. Granules for data-driven MapReduce computations

Granules runtime orchestrates the execution of MapReduce computations that are responsible for extracting different types of metadata from the datasets.

3. Extracting Fine-Grained Metadata

This component extracts the metadata by using Granules runtime in several steps; they first split the original file to distribute the dataset over a cluster. Computations are then pushed to nodes that hold portions of the file. Each computation then extracts fine-grained metadata by parsing the XML document, and ensuring that the data are stored in the metadata storage. Reducers are activated as soon as the data are available from the mappers.

4. Generating Advanced Metadata

The metadata extracted from the datasets is processed to provide a summary, which is useful for data browsing and comparing large datasets.

5. Customizable Datasets

This component allows users to select or customize the dataset and they can download specific portions of the dataset that will be injected into their computation [35].

Oracle demonstrates a white paper entitled: **"Big Data and Natural Language: Extracting Insight From Text"**. In this paper, Oracle shows a combination of the Oracle Big Data Appliance and Digital Reasoning Synthesys software to analyze tens of millions of documents in a matter of hours. Synthesys is a software platform for automatically making sense of Big Data, and it is integrated with Cloudera's Distribution, including Apache Hadoop (CDH); Synthesys uses patented algorithms and machine learning methods in a three-phase process called "Read-Resolve-Reason" It analyzes scale horizontally to virtually any corpus size.

Synthesys uses a combination of model-building and unsupervised learning, and discovers the information as a human would – in context and without the need for a pre-defined ontology, it understands related terms and associations to improve entity recognition, and a contextual understanding of concepts across large sets of text. Extracted information is stored in a knowledge graph to process data continuously and more deeply refines it [36].

There are many resources for Big Data. Email is one of them, and "the use of email has grown, making it the most intensively used knowledge work tool" [37].

A paper entitled: **"Optimising the email knowledge extraction system to support knowledge work"** shows a tool that has been named EKE (Email Knowledge Extraction) which mines information contained in employees' emails. EKE automatically finds areas of interest by picking out key phrases from employees' email messages. EKE is designed in such a way that it is closely integrated into the employees' email client and fits well with the work they do in a natural manner.

There are abstract steps for EKE working [37]:

1. Email sent to remote server SMTP.

2. The email body is then passed to the EKE server, where it will be processed by the Key-phrase extraction web service.
3. Key-phrases Extracted are stored in a temporary buffer.
4. A ranking interface will be generated from the extracted key-phrases and sent to the employee.
5. The submitted interface will be stored on the server in the database
6. The result returned is a list of experts in the organization ranked by their suitability to answer the user's query.

2.3.2 Social media

In the social media world, in a paper entitled **"Detection and Extracting of Emergency Knowledge from Twitter Streams"**, authors propose a combined structural and content based analysis approach in a SABESS project. They use social network analysis to identify reliable tweets and content analysis techniques to summarize key emergency facts. They use ActiveMQ as a core massaging system, the crawler stores the tweet in the repository and then feeds the messaging system with this tweet. Tweets are formatted in the JSON format and encode various user and tweet related metadata. Analysis tools parse tweets' content and enhance the tweet with additional metadata descriptions before being stored in the outgoing queue. With the help of content analysis, additional facts about the emergency are obtained from the Tweet text. User data from Twitter, tweet metadata, credibility information from the social network analysis and emergency information obtained from the tweet content are used to construct emergency summaries through a matchmaking process [38].

In a paper entitled **"An Architecture for Metadata Extractor of Big Data in Cloud Systems"**, authors propose a framework for managing Big Data on cloud

distributed systems, and introduce an agent-based architecture for metadata extractor of Big Data. The architecture consists of many mobile and stationary agents. The mobile agents migrate to remote machines that include big data to process and discover the required knowledge and return back to the main server. The big data manager consists of two sub-managers: metadata extraction sub-manager and knowledge discovery sub-manager. The metadata extraction sub-manager extracts and retrieves metadata of the big-data on the cloud machines and stores them in metadata storage and the knowledge discovery sub-manager discovers the required knowledge or information that is needed by the user [39].

In a paper entitled **"A Data Analytic Framework for Unstructured Text"**, authors provide overview of unstructured data, challenges, technology, and data manager implementation. They describe a systematic flow of the unstructured data in industry, collected data, stored data, and the amount of data. This paper introduces an unstructured data framework for managing and discovering using the 3Vs of big data: variety, velocity, and volume including: service-based, metadata storage and data preparation. The development processes in this paper is implemented in Python, build up lexicon and calculated sentiment score [40].

In **"Using topic models for Twitter hashtag recommendation,"** the authors proposed an approach to recommending hashtags for tweets, which provides easy indexing and the search of tweets. They developed a binary language classifier for tweets based on the Naive Bayes method to discriminate between English and non-English language tweets. Then they applied a Latent Dirichlet Allocation (LDA) model in the context of tweet hashtag recommendations. They select keywords as hashtag recommendations by determined the topic distribution of a tweet then count

the number of words in the top five topics to determine the top words for every topic, so the final result is a set of keywords that is the general topic of the tweet [39].

In **"Real-Time Topic Modeling of Microblogs"** [40], the authors apply LDA to topic model tweets and use the Machine Learning for Language Toolkit (MALLET) API as the implementation of LDA in a Java environment to suggest a relevant topic being discussed in that tweet.

The actual workflow steps are as follows:

1. Filter tweets from the Twitter stream.
2. Save these tweets to a file. (One tweet per file per directory).
3. Apply MALLET (text-to-vector) conversion to each tweet file, and write to the vector file.
4. Apply MALLET (vector-to-topic) conversion to each vector file for topic extraction.

Chapter III

SYSTEM ARCHITECTURE & DESIGN

In this chapter, we will explain our proposed system, which includes a Rationale section and explains the high-level system architecture and system design.

3.1 Rationale

Big Data are distributed across large clusters, which consist of thousands of remote machines; it stores a variant data and there is no standard that describes these data. Big Data are generated by many sources like social networks, sensors, application logs, etc., and organizations want to collect, analyze and extract knowledge from Big Data to improve their business, but a traditional analysis tool cannot handle Big Data. Because of this, new methodologies and techniques for collecting, analyzing and extracting knowledge are required.

3.2 System Architecture

Figure 3.1 shows the System architecture of our system.

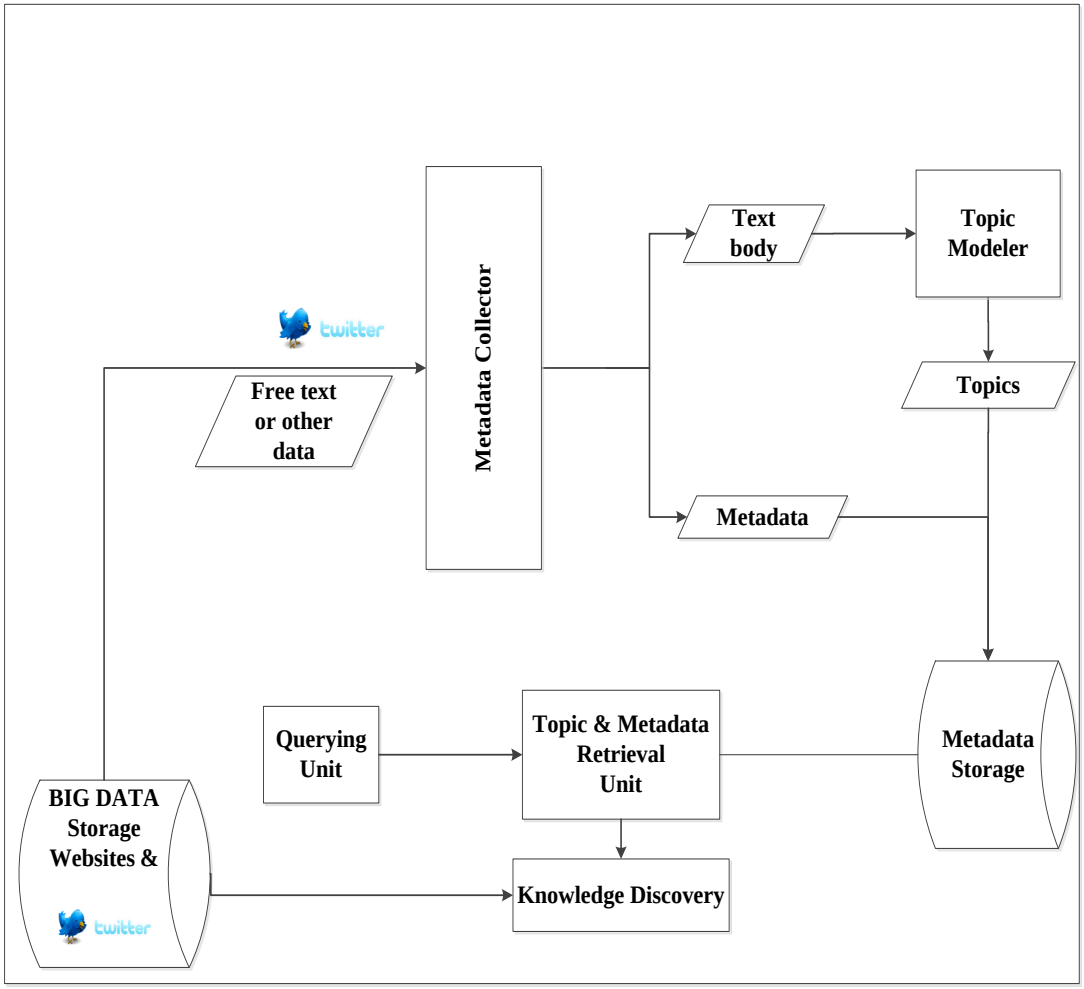

Figure 3.1 System architecture

3.2.1 Big Data Storage

This block represents the data source for our system, which are Twitter and the contents of web pages. These data sets will be explained in the system design section.

3.2.2 Metadata Collector

We collect metadata for Twitter massages and web page contents by Metadata collector block. We have two types of metadata collector. The first is the Twitter metadata collector and the second is the free text metadata collector. We are using

Twitter APIs to collect Twitter metadata and extract web pages metadata by using Google WebCrawler. These metadata will be stored in metadata storage, which will be discussed in the tweets and free texts metadata section.

3.2.3 Topic Modeler

The core of our system is a topic modeler, which trains topics from the input corpus of documents, and infers topics for new documents. In this block, we used the MALLET tool kit as an implementation for topic modeling. This tool kit is open source, so we used a number of classes, modified it and aggregated it into our system.

Topic Modeler takes its input from Metadata collector, which collects tweets or web pages metadata and extracts a message body (tweet) or a text (from the page content) and then sends it as input to the topic modeler. There are three steps in the topic modeler that train and infer topics:

1. Import dataset

   This step will import input dataset or documents to MALLET format and remove stop words, then generate a train file containing a features vector.

2. Train topics

   MALLET uses the LDA algorithm to train topics from the features vector file, and generates a setup file or inferencer file, that will be used to infer new documents.

3. Infer topics

   When new documents are available, the topic inferencer will extract the topics and calculate their proportions for each document.

3.2.4 Metadata Storage

We store tweets, free text metadata and their topics in a relational database, which is used by Topic & Metadata Retrieval to access topics and metadata for end user queries.

3.2.5 Metadata Retrieval

When the end user issues queries via the user interface, our system will find the topics for these queries by the topic modeler and send a request to Metadata Retrieval to retrieve all tweets or free text metadata on the same topics.

3.2.6 Querying Unit

This block is a user interface; it allows the end user to interact with our system.

3.2.7 Knowledge Discovery

Once we have obtained metadata for tweets or free text, the Knowledge Discovery block will extract some knowledge from the Big Data storage using mobile agents.

3.3 System Design

In this section, we will explain the detailed design of the most important parts of our system, and discuss the system, training and testing phases, and the training dataset. We will explain how we collect tweets and free text metadata and how can we find a correlation between them via the topic modeler. Finally, we will consider Metadata storage design.

3.3.1 System phases

Our system consists of three phases: the dataset collection, training and testing phases.

1. Dataset collection phase

   In this phase, we collect training datasets from Twitter and web pages. We will explain this phase in more detail in the training datasets section.

2. Training phase

   This phase builds a training model using MALLET. This phase consists of two stages: import data, and generate topic model file.

   The **Import Data** stage involves importing the training dataset to MALLET internal format by using the Text2Vector class, which converts the dataset from document files into Feature Vectors files. Also, in this stage, stop words will be removed.

   The **Generate Topics** stage uses the feature vectors file as input, which is generated from the previous stage, and then converts the feature vectors into topics using the Vectors2Topics class and generates a topic inference tool based on the current, trained model [41]. The inference file contains the top 'k' words associated with the topics [42]. In our system, we are using k=10, which is a default value. We can specify the number of topics to be generated by the num-

topics parameter, and in our case num-topics = 400. A number of topics between 200 and 400 is recommended by MALLET's document and will produce reasonably fine-grained results. Figure 3.2 illustrates the training phase.

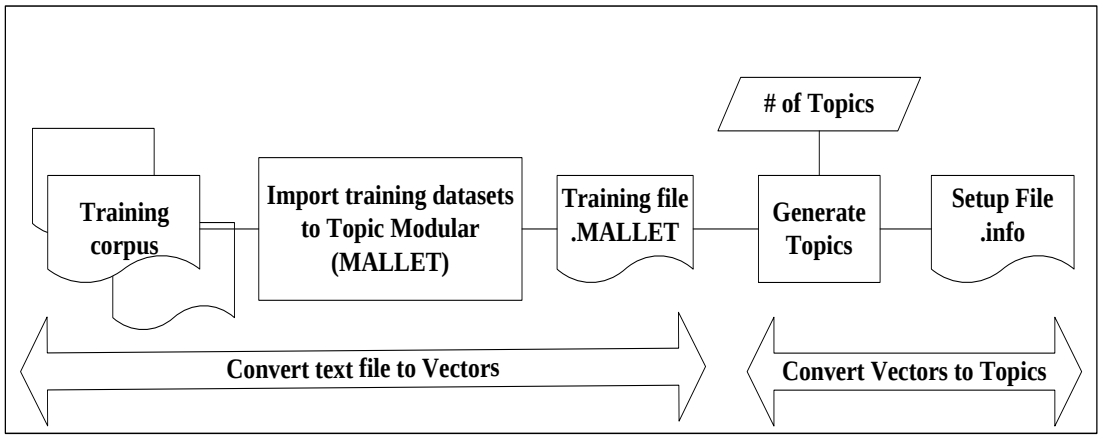

Figure 3.2 Training phase design

3. Testing phase

This phase allows users to interact with our system. Once the system has been trained, it is ready to use. The testing phase starts when the user issues a text to test on the system interface. Figure 3.3 shows the testing phase design. The user's text will be imported and converted to an internal MALLET format (Vectors Feature) to be used by the topic inferencer to infer topics from this text.

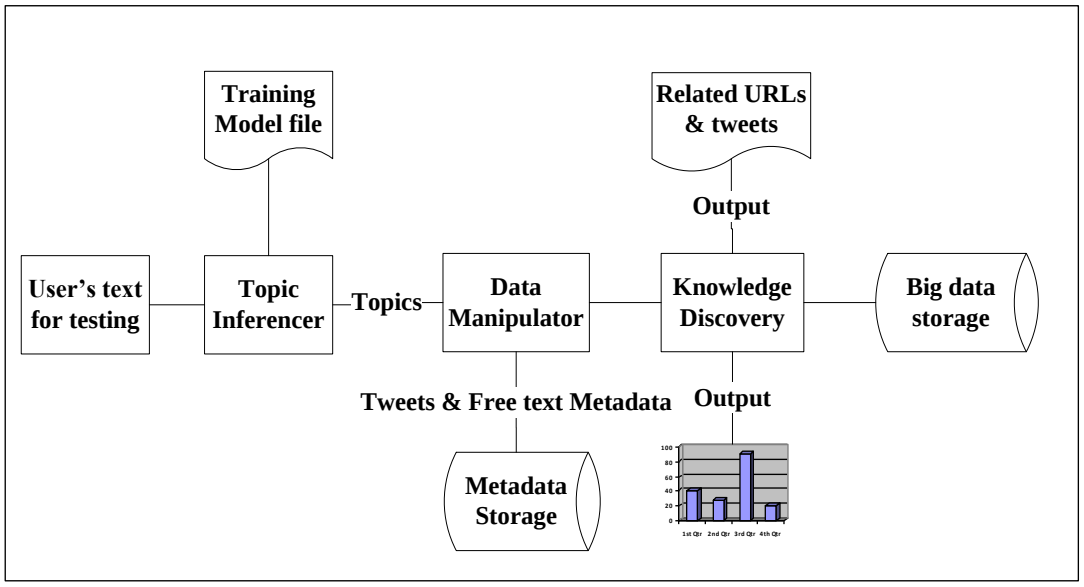

Figure 3.3 Testing phase design

The **topic inferencer** is part of the topic modeler, which is used in the testing phase. The topic inferencer uses the training model file which is generated from the training phase to infer topics for new documents by the MALLET InferTopics class, so the output of the Topic inferencer block are the topics for the user's text. After that, the **Data manipulator** will take these topics as input and then retrieve all metadata of tweets and free text related to these topics from the metadata storage. Once we have the metadata related to the user's text, **knowledge discovery** uses these metadata to collect specific tweets text and free text contents by using tweets ID and web page URLs respectively, then extracts knowledge from the texts that have been collected.

3.3.2 Training datasets

In this section, we will explain our training datasets, which are used by our system to generate the topics vectors model. These training datasets consist of two datasets, the tweets dataset and the free text dataset, which are in English, and all of them will be stored in file based storage. We will discuss the mechanism for collecting training datasets.

3.3.3 Training datasets collector

The system collects training data from Twitter by Twitter APIs which allow external applications to interact with the Twitter database. On the other hand, the system uses the contents of web pages as a free text resource and collects these contents by using Google crawler APIs, which allows us to crawl through URLs and read their contents. The Twitter dataset and free text will be saved in file based storage. Figure 3.4 shows the dataset collector design.

3.3.3.1 Twitter training dataset

A tweet is a very short message, up to 140 characters long, that "poses serious challenges to the efficacy of topic models on short, messy text and they are often less

coherent when applied to microblog content like Twitter" [43]. Because of this, we employ the Hashtag based Pooling technique [43] to create one document for each hashtag and append all tweets which are related to this hashtage to the document file. We created the No hashtag file and appended to it all tweets that have no hashtag. To date, we have collected about 8 million tweets, which consist of 316,313 hashtags document files with a total size on disk of 1.3 GB and 70000 files which were marked as No_Hashtag, of about 11 KB per file and with a total size of 780 MB.

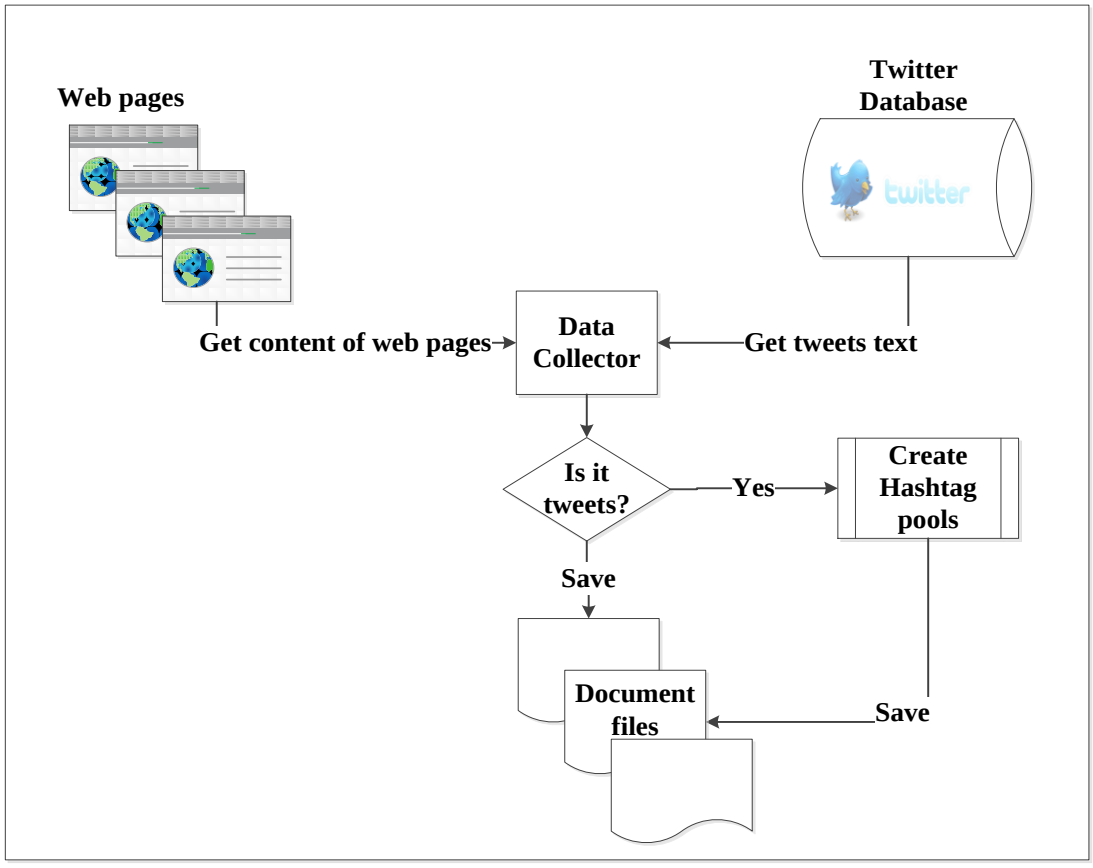

Figure 3.4 Datasets collector design

3.3.3.2 Free text training dataset

We are using unsupervised learning to create the training topics model. This type of training does not need to annotate the training datasets to make labels, so it takes less effort than supervised learning, saving us time and giving us the freedom to choose the free text datasets that can be used as the training dataset. We use two datasets to

create a system training dataset. The first dataset is the content of web pages, which are a large, reborn resource for free texts. We collected about 100,260 contents of web pages and stored them as text files, with a total size of 2.2 GB. This web pages dataset collection was performed by Datasets Collector using Google crawler APIs. The second dataset is the 20 Newsgroups, which is a collection of 18,846 newsgroup documents. We used 11,314 files from the newsgroup during the training phase and, as for the remaining dataset, we are using it in the testing phase. The 20 newsgroups collection has become a popular dataset for experimenting with text applications of machine learning techniques, which is available to researchers at http://qwone.com/~jason/20Newsgroups/.

3.3.4 Tweets & Free texts Metadata

Twitter has metadata for tweets and users, and allows external applications to read these metadata for public tweets by Twitter APIs. Raffi Krikorian shows in "Map of a Twitter Status Object" that up to 32 attributes for each tweet are stored on Twitter. These attributes describe the tweets. According to metadata's definition, "metadata are data about data" or, in other words, "metadata are data that describe data", so we can use tweets' attributes as metadata for tweets. Figure 3.5 shows a map of a Twitter status object including all of its metadata.

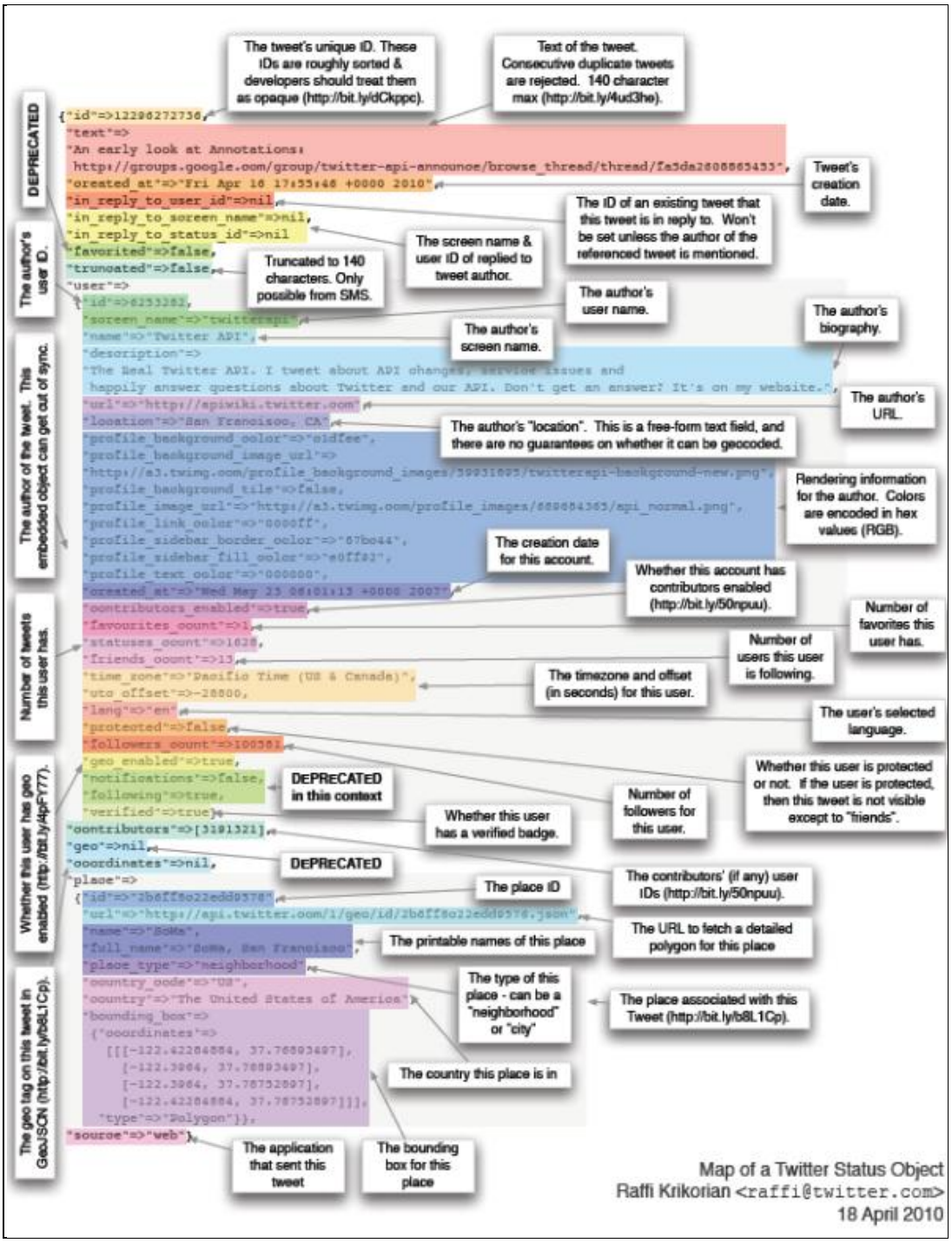

Figure 3.5 Tweets metadata

Table 3.1 shows the metadata we selected from Twitter Status Object to use in our system, which consisted of two groups - tweets and User Metadata. We collected other metadata, which is free texts metadata; the system will collect web pages metadata by Google web pages crawler APIs, which allows us to crawl and read web pages headers. Table 3.1 illustrates the metadata we used for web pages.

Table 3.1 Twitter and free text metadata

Twitter		Free Text
Tweets Metadata	**User Metadata**	**Web Pages Metadata**
Tweet ID	User ID	Page ID
Created at	Created at	Web page URL
Language	Name	Created date
Is rtweet?	Screen name	Last modified date
Retweet count	Location	Domain
Hashtag	Followers count	Sub domain
	Friends count	Title
	Tweets count	

3.3.5 Metadata Collector

Collecting metadata is an important task, because we want to collect as much metadata for Big Data as possible to find a result that satisfies the user's request. This process will be done via Metadata collector, which collects the metadata from Twitter and web pages and stores them in Metadata storage.

Metadata collector collects tweets metadata under the following conditions:

1. If (tweet's Language = English && Is Retweet? = false && Retweet count > 100).
2. If (Followers count > 1000 && User's tweets count > 1000)

We are using these conditions in Metadata collector to help us to find meaningful tweets and it informs us about significance of the tweets themselves. Figure 3.6 shows the metadata collector design.

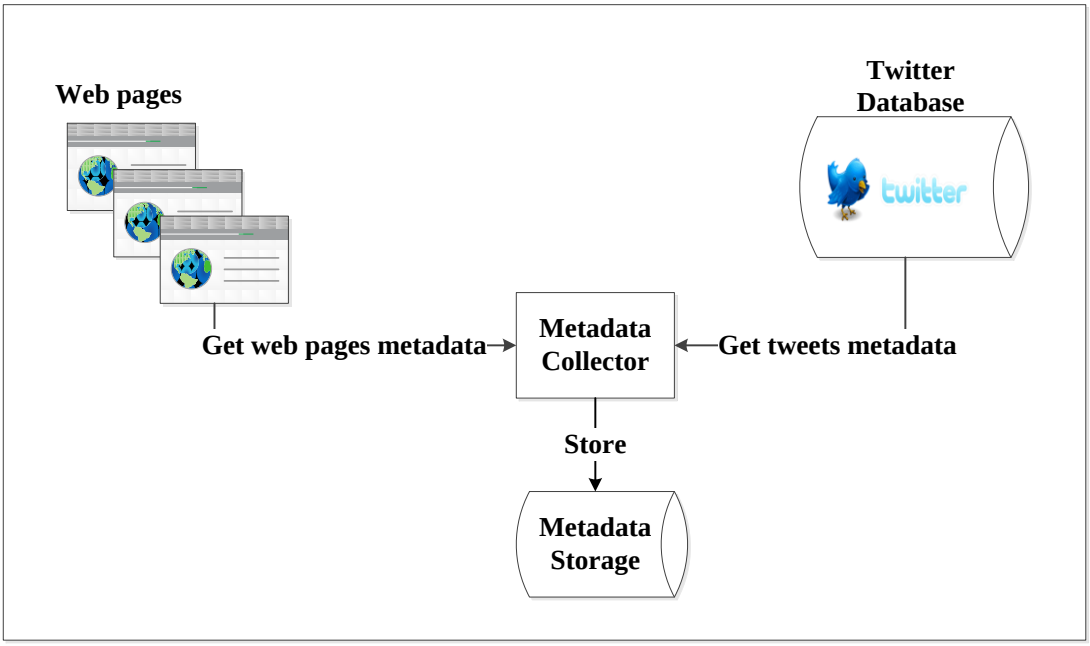

Figure 3.6 Metadata Collector Design

3.3.6 System approaches for classification

In this section, we explain two approaches used by our system to classify collected tweets and free text. We can distinguish between these approaches by classification methods, which are online classification or offline classification.

3.3.6.1 The online classification approach

In this approach the metadata collector will extract tweets' text or the content of a web page, then send it to the topic inferencer to classify it one by one. In other words, for each single tweet or web page, our system will run a full classification process to generate topics. This approach suffers from performance and accuracy issues, which we will discuss in the evaluation chapter. Figure 3.7 demonstrates the online classification approach.

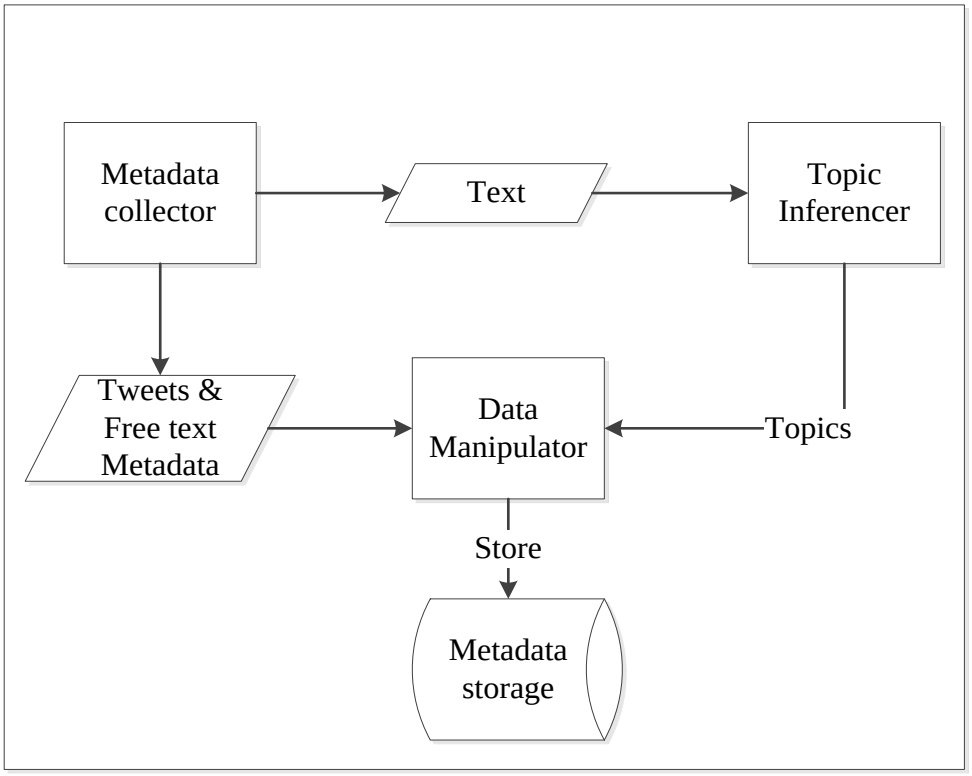

Figure 3.7 online classification approach

3.3.6.2 The offline classification approach

Figure 3.8 explains the offline classification approach, which is different from the previous approach, because the metadata collector will extract tweets' text or the content of web pages and does not send extracted texts one by one to topic inferencer to classify them, but sends these texts to a **pools generator** which creates a pool of tweets for each hashtag and stores them as a document file in the hashtags pool.

Other metadata are free text metadata, which are collected by metadata collector from web pages, which sends its content to the pools generator to store each web page content as a document file in the free text pool. After a period of time, our system will start to classify each file in the pools and generate its topics, which are stored in the metadata storage. The offline classification approach is shown in Figure 3.8.

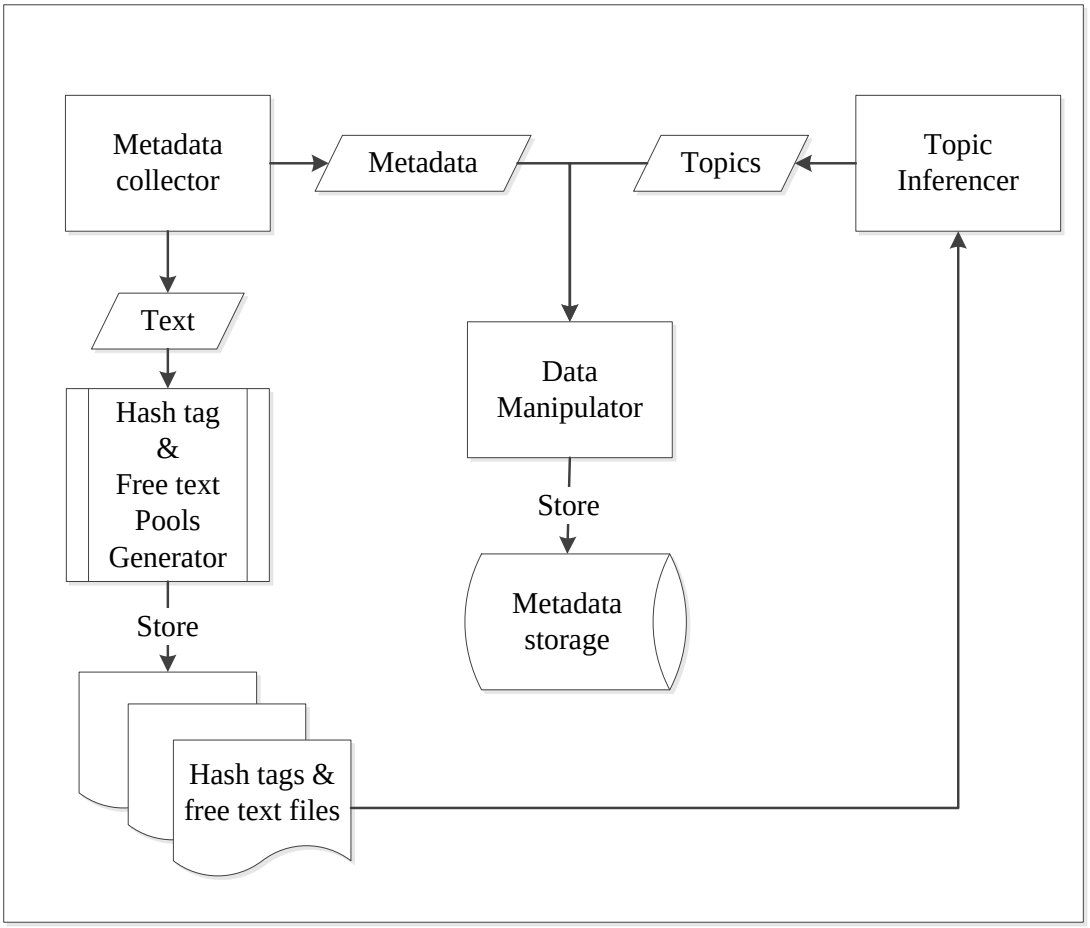

Figure 3.8 offline classification approach

3.3.7 Tweets & web pages' contents correlation

In our case, we have two different sources of Big Data (tweets & web pages) and want to search these Big Data to extract some information that satisfies a user's query so, in a typical scenario, we will search one Big Data set independently of the others and return results without any correlations between them. Because of this, we proposed a parameter which can find tweets and web pages correlations, so we employed topic modeling in our system to find this correlation by extracting the topics from the user's query and gathering tweets and free text correlated to these topics, so we find the correlations between tweets and free text by topic. This idea is shown in Figure 3.9.

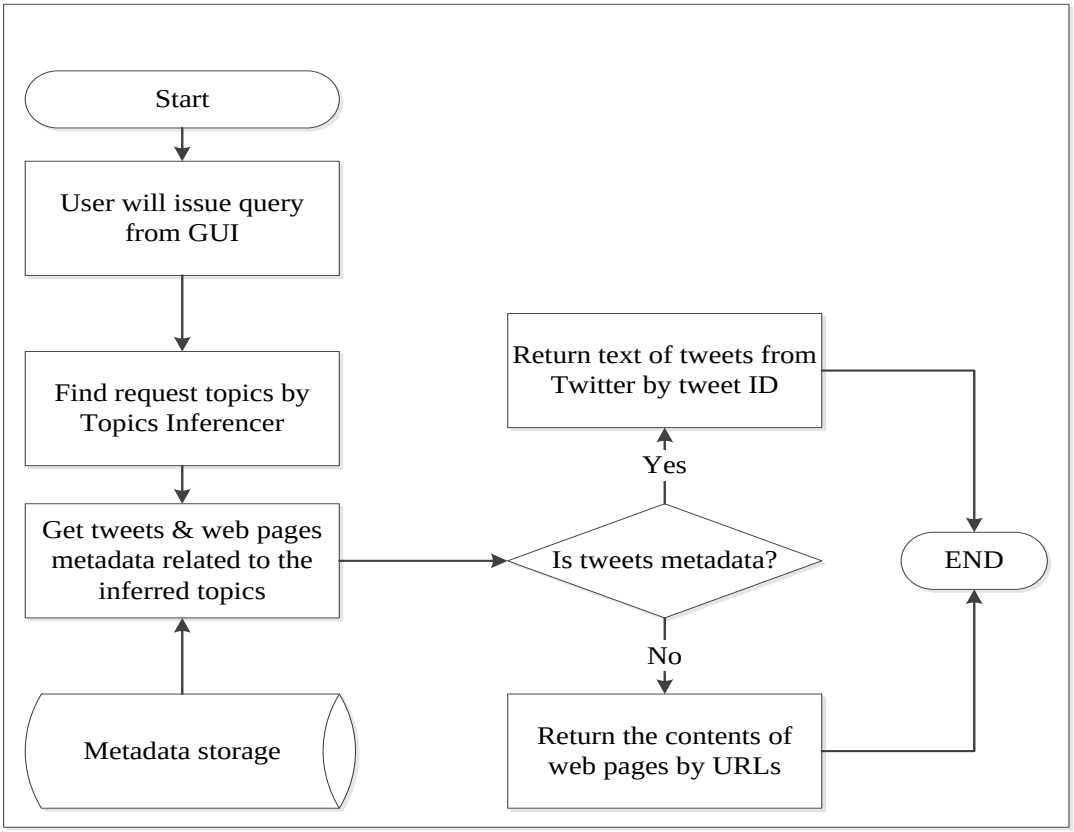

Figure 3.9 Correlation between different Big Data sets

3.3.8 Metadata storage design

In this section, we demonstrate the metadata storage design, which is designed to store tweets, free texts metadata and their topics in a relational database. We designed this database to store collected metadata and support the online and offline classification approaches. Figure 3.10 shows the metadata storage design.

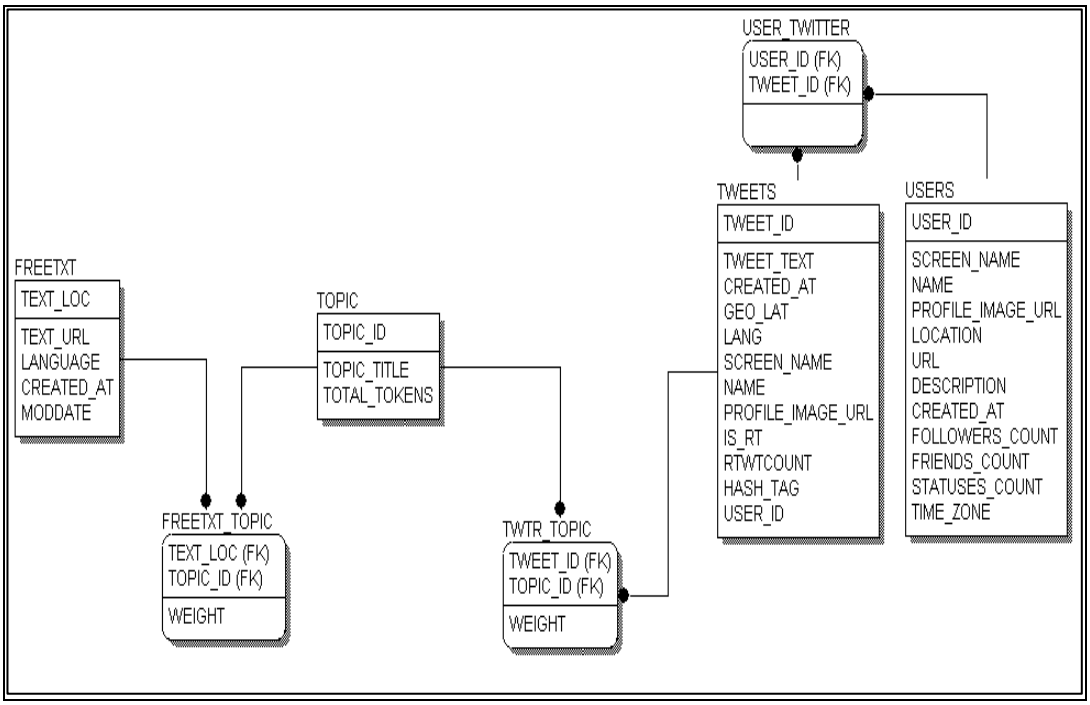

Figure 3.10 Metadata storage design

3.3.9 Mobile agent design

Since Big Data are distributed across a large number of remote machines, we use mobile agents technology to build our manager. Using mobile agents and metadata of Big Data will solve the Big Data transportation challenge.

We used the Mobile agent technique to retrieve data from Big Data and extract simple knowledge from these data to solve the transportation challenge. In this section, we will show the design of the mobile agent that has the ability to migrate (code & data) from the local host to a remote host to retrieve data and extract simple knowledge then return the results. We implement the mobile agent using JADE (Java Agent DEvelopment Framework), which is a framework for developing agents.

We created three agent containers that contain the mobile agents. All of them are on one platform. The first container is a main container, which maintains a central registry of all the others so that agents can discover and interact with each other. The other agent containers run on a remote machine and are connected to the main

container. We implement two mobile agents. The first is the Twitter mobile agent which retrieves tweets from the Twitter database. The second is the free text mobile agent that retrieves the content of websites. Our mobile agents have two behaviours (tasks, or intentions) which are retrieving data behavior and extracting knowledge behavior.

Retrieving data behavior will retrieve data that satisfies the user's request from existing Big Data while extracting knowledge behavior will extract simple knowledge from retrieved data and then return the results to the user. Figure 3.11 shows that the two database servers contain the Twitter database and websites documents and the third server will host our application and run as the main container on the platform.

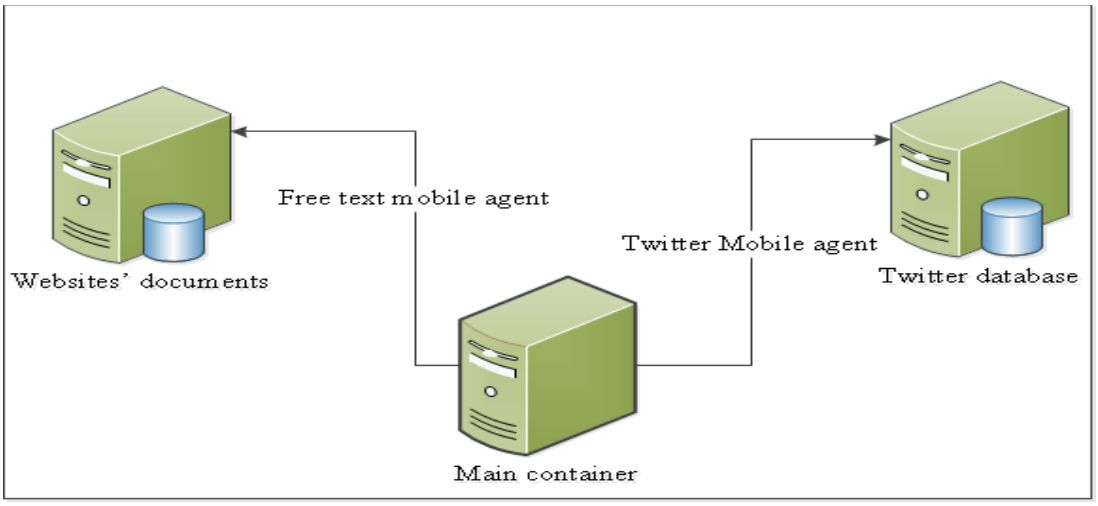

Figure 3.11 Mobile agent platform

3.3.10 Class & Sequence diagrams

In this section, we will display the important classes together with metadata collecting and classifying sequence diagrams for our system. We will explain the main task for some classes:

- TMC stands for Twitter Metadata Collector, which collects metadata of tweets, as explained in section 3.3.5 (Metadata collector).

- FMC stands for Free text Metadata Collector, which collects metadata of web pages; you can see these metadata in Table 3.1 on Twitter and free text metadata.
- Topic Inferencer is the most important class in our system, which is implemented in the MALLET tool kit. Topic Inferencer infers or extracts topics from a corpus of documents.
- Text_Importer transforms a corpus of documents or a single document into MALLET format.
- TWT_Outh uses Twitter APIs to obtain security keys and open a connection with Twitter.
- Knowledge Extractor will extract knowledge from the results, which is returned after the user issues his request. Figure 3.12 shows the classes diagram for our system. Figure 3.13 shows a sequence diagram for collecting metadata and classifying incoming texts from Twitter and free text and storing them in metadata storage. These operations are important, because our system wants to build a good metadata database.

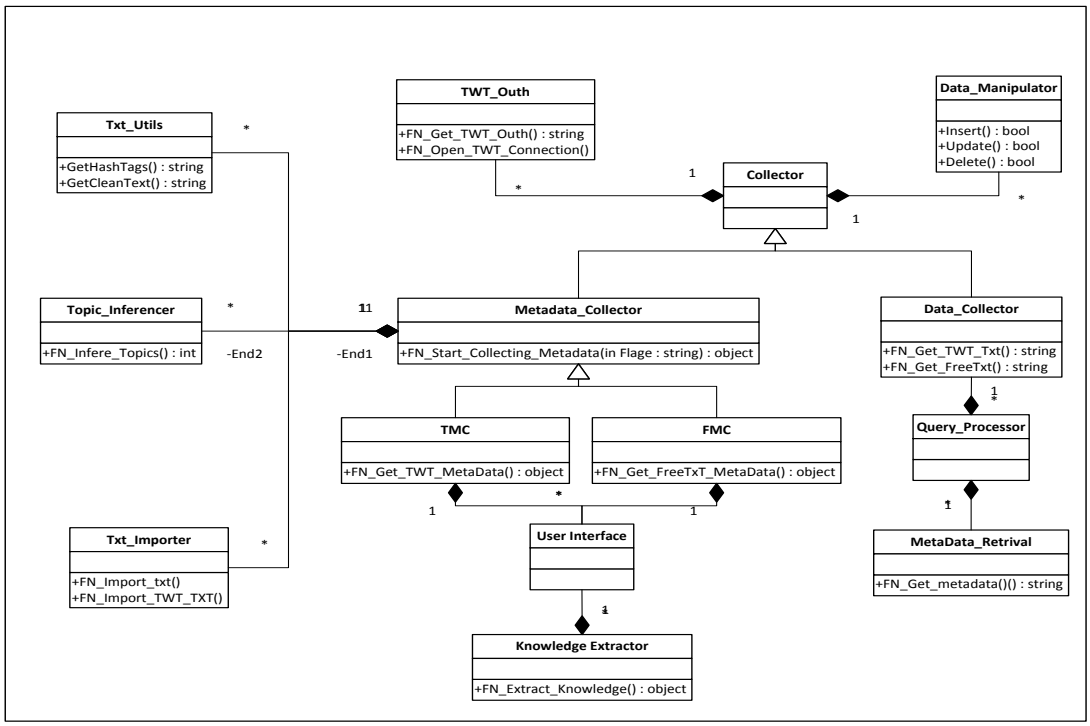

Figure 3.12 System Class Diagram

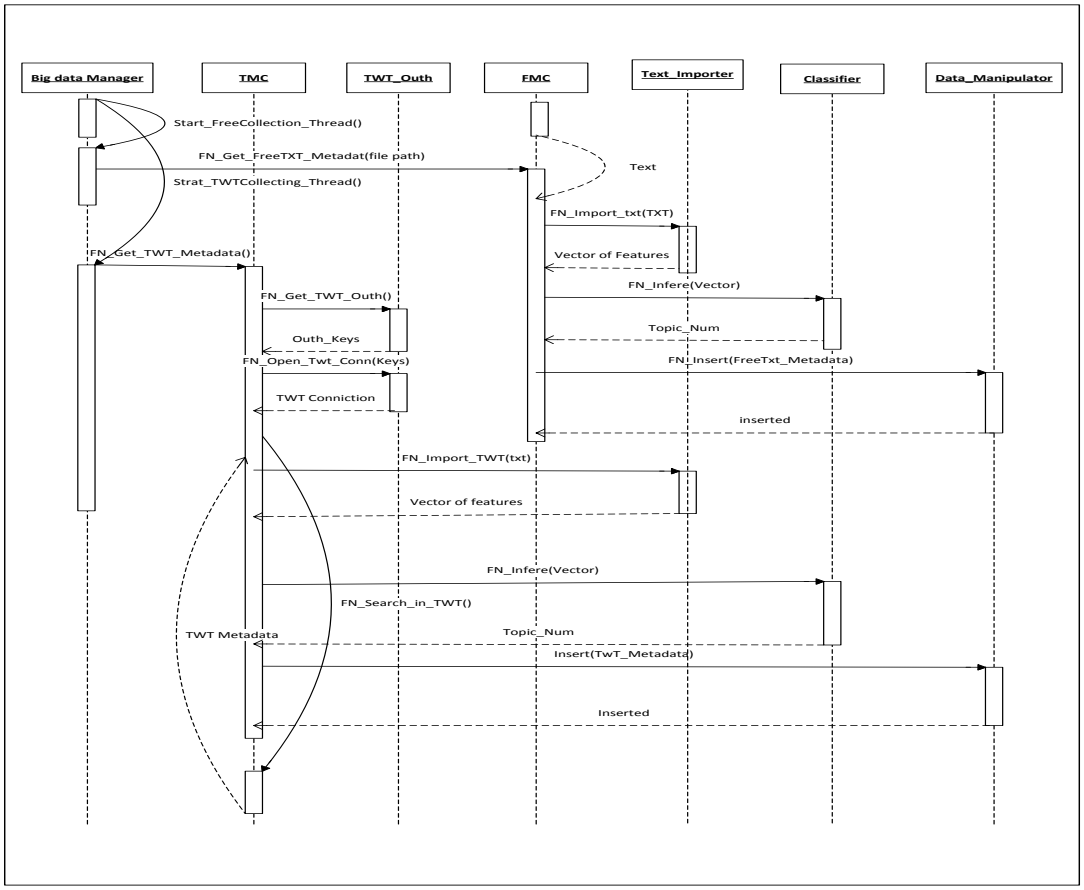

Figure 3.13 Metadata Collection and Text Classifying Sequence Diagram

Chapter IV

IMPLEMENTATION & TESTING

In this chapter, we will demonstrate the most important methods in our system and explain the GUI of the system. We also present the tools and libraries that are used to implement the system. Finally, we will test the scenario and explain the results.

4.1 Implementation

Our system is completely implemented in Java; we built a lot of classes and methods to implement it. In addition, we modified several open source tools and libraries to ensure their integration with our system and then embedded them in the system. In this section, we will show the most important methods, tools and libraries that we are using in the system.

4.1.1 Most important Methods used in the system

a. Training phase

1. Import training data

FN_Convert_text2vectors (Dataset Directory, Distention Training file)
{
This method uses MALLET to convert training dataset to features vectors and creates a training file .mallet extension, which is a MALLET format.
}

2. Train topic from text

FN_Convert_vectors2topics (Input training file, Topics Num, Distention Inferencer file)
{
This method uses MALLET to convert features vectors to topics and creates a configuration file which is used by the classifier.
}

b. Collecting metadata from Big Data

3. Collect Twitter metadata using Twitter APIs

Twitter FN_Open_TWTConnection (String [] Outh_Keys)
{
Open connection with Twitter by application authentication keys.
Return Twitter Instance.
}

List <users> searchUsers(String sp_characters)
{
This method collects ScreenName for Twitter's users.
Return list of ScreenName.
}

JSONArray Retrive_Statuses (User [] users)
{
For each user who has 1000 or more followers and his language is English, then collect the user's tweets.
Return JSONArray contains tweets.

```
for (User user : users) {

  if (user.getStatus() != null && user.getFollowersCount()>=1000 && "en.”equals(user.getLang()))
{
   JSONArray jsonarry= json_collector.GetStatuses (user.getScreenName());
}
                              }
return jsonarry;
}
```

FN_Extract_TWT_Metadata(JSONArray jsonarray)
{
This function extracts tweets and users' metadata from the JSON array.
}

4. Collect metadata from websites

URL_crawler(string [] URLs)
{
URL_crawler collects metadata of websites and their contents by using the Google crawler4j library.
}

5. Classify tweets and websites' contents

String path FN_Import_Txt(TXT_Path,Output_Vector,Training_file)
{
Import tweets or free text to MALLET format and convert it to vectors.
Return Vectors Path.
}

Number [][] FN_Classify_Txt(Vectors_Path, Inferencer_file)
{
Use the Inferencer_file to convert input features vectors to topics by using the MALLET vectors2topic function.
Return Array of topics.
}

---

6. Store metadata in metadata storage

FN_Save2Metadata_DB(Metadata, topic ID)
{
Store all metadata for tweets and free text in metadata storage with its topic ID, which is generated from the classification method.
}

c. Testing phase

7. Find related tweets or documents for user's request

String path FN_Import_Txt(TXT_Path,Output_Vector,Training_file)
{
Import request from user and convert it to features vectors.
Return Vectors Path.
}

Number [] FN_Classify_Txt(Vectors_Path, Inferencer_file)
{
Generate topics for user's request by using the inference file.
Return Topics ID.
}

8. Retrieve metadata from storage

String [] FN_Retrieve_metadata (Topic_ID, Threshold)
{
While (topic_weight >= ***Threshold***)
{
Retrieve all tweets ID and URLs from metadata storage with a greater than or equal threshold.
}
Return Array of Twts_ID and URLs
}

---

9. Retrieving related tweets and websites' contents (Mobile agents)

String [] FN_Tweets_IR(location Machine_location, long [] Tweets_ID)
{
Migrate tweets_mobile_agent to the right machine that contains tweets to retrieve specific tweets by using its tweets ID.
Return set of tweets.
}

String [] FN_Freetxt_ IR(location Machine location, String [] URLs)
{
Migrate Freetxt_mobile_agent to the right machine that contains the websites' contents to retrieve specific websites' contents.
Return sets of websites' contents by using URLs.
}

---

10. Extracting knowledge from retrieved data (Mobile agents)

Chart FN_Tweets_Knowledge_Extractor(location Machine_location, long [] Tweets_ID)
{
Migrate tweets_mobile_agent to the right machine which contains the tweets to extract knowledge.

Return knowledge from tweets as bar chart.
}

Chart FN_Freetxt_ Knowledge_Extractor(location Machine location, String [] URLs)
Migrate Freetxt_mobile_agent to right machine which contains the websites' contents to extract knowledge.
Return knowledge for URLs as bar chart.
}

4.1.2 Tools and Libraries

In the implementation phase, we used, modified and embedded several open source tools and libraries in the system, to collect and classify tweets and websites' contents.

4.1.2.1 MALLET (MAchine Learning for LanguagE Toolkit)

MALLET is open source software, and it provides a Java-based package for statistical natural language processing, topic modeling, document classification and clustering. We used MALLET as an implementation for topic modeling, which uses the LDA algorithm to train topics.

4.1.2.2 Twitter4j

Twitter4j is a Java library for Twitter APIs which is open source software. We used it to collect public tweets from Twitter by using search and get/statuses/user_timeline and other REST APIs.

4.1.2.3 JADE

JADE is middleware that facilitates the development of multi-agent systems and it is an Open Source library. It includes a runtime environment where agents can live, libraries to build the agents and it also has administration and monitoring tools.

4.1.2.4 Crawler4J

Crawler4J is a Java Open Source library that provides Java packages to collect data from websites.

4.2 Testing and results

In this section, we show the GUI of our application and apply a testing scenario on the application, then discuss the results.

4.2.2 Application GUI

There are six parts to the GUI, which are the training, testing, data collector, discover knowledge, user's request and results parts.

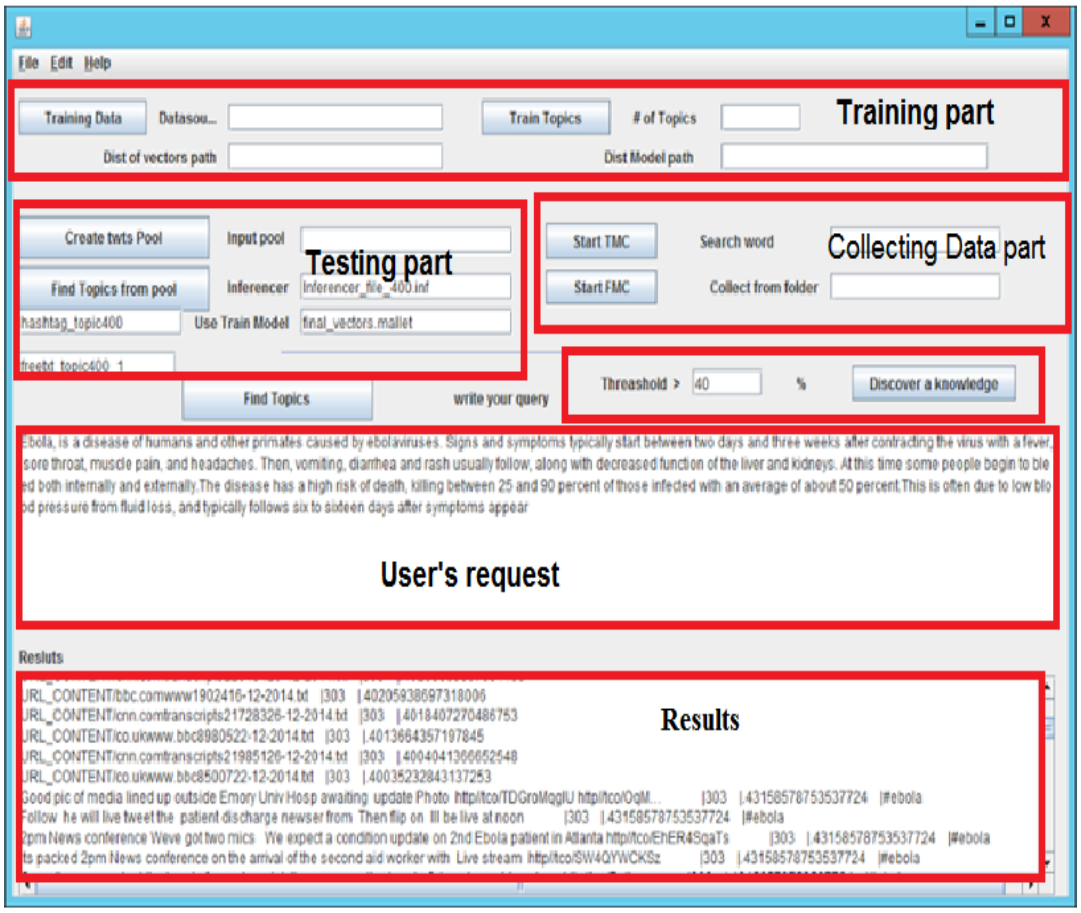

Figure 4.1 System GUI

In the training part, we can write the path of the training dataset and the path of the features vectors, which is provided by the import dataset process. Also, we can specify the number of topics and the path of the training model or inferencer file. This training model is generated by the train topics process. Tweets and websites metadata are collected by metadata collectors which are presented in the data collector part TMC (Twitter Metadata Collector) and FMC (Free Text metadata

Collector). In the test and request parts, the user can write his request then click on the find topics button to find related topics. The results will appear in the results part which are tweets and websites' documents related to the user's request. Then, using the knowledge discovery part, the user can present charts to visualize the retrieved data.

4.3 Testing Scenario

In this scenario, we used a training model that trains 400 topics from the training corpus, and we want to find related tweets and websites that talk about Ebola so, for this scenario, we put a text that describes Ebola and its signs into the User's request like Figure 4.1, then press find related topics to see the results. This is a request or a text for which we want to retrieve related tweets and websites. *"Ebola is a disease of humans and other primates caused by ebolaviruses. Signs and symptoms typically start between two days and three weeks after contracting the virus with a fever, sore throat, muscle pain, and headaches. Then, vomiting, diarrhea and rash usually follow, along with decreased function of the liver and kidneys. At this time some people begin to bleed both internally and externally. The disease has a high risk of death, killing between 25 and 90 percent of those infected with an average of about 50 percent. This is often due to low blood pressure from fluid loss, and typically follows six to sixteen days after symptoms appear"* [44]. The system retrieves only tweets and websites' documents that have a related probability greater than the threshold. In this scenario, we set the threshold as greater than 40%. There are two mobile agents that start working when a user runs his request, which are the tweets mobile agent and the free text mobile agent. These mobile agents have two tasks. The first one is retrieving related data from the destination machines, and the second is extracting knowledge from these data. Figure 4.2 shows the JADE Remote

Management Agent (**RMA**) which handles the GUI interface and shows all of the participating agents and containers.

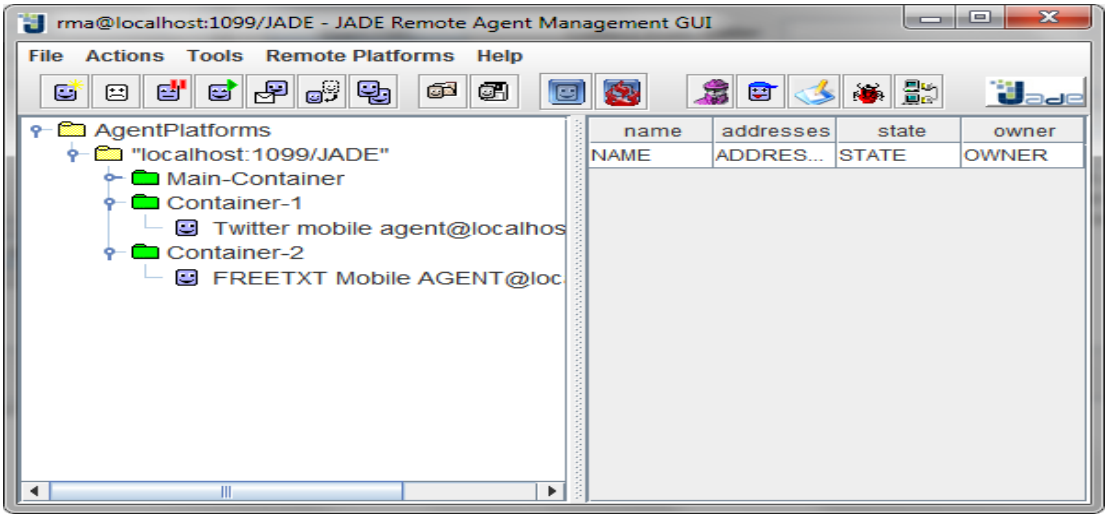

Figure 4.2 JADE Remote Management Agent (RMA)

4.4 Results

In this section, we will show sample results for the testing scenario in section 4.3. These results consist of retrieved data and charts that help us to visualize the result and extract knowledge from it.

- **Retrieved data**

  Table 4.1 and 4.2 show the results of the related tweets and websites that are related to the user's request about Ebola.

Table 4.1 Documents and URLs related to the Ebola request

Documents	Topic	Weight	URLs
cnn.comus11565823-12-2014.txt	303	0.82	http://us.cnn.com/2014/04/11/health/ebola-fast facts/index.html?hpt=wo_r1
bbc.comwww11990223-12-2014.txt	303	0.69	http://www.bbc.com/news/world-us-canada-29628622
co.ukwww.bbc8960022-12-2014.txt	303	0.67	http://www.bbc.co.uk/news/world-africa-26835233
cnn.comtranscripts21996426-12-2014.txt	303	0.4	http://www.nejm.org/doi/full/10.1056/NEJMoa1411100

Table 4.2 Tweets related to the Ebola request

Tweets	Topic	Hashtag weight	Hashtag name
Ebola virus scans slowing down the process at passport control Expect delays	303	0.43	#ebola
Health Minister Aaron Motsoaledi says Cabinet decided on total ban on traveling to countries affected by Ebola			
WATCH US flight held as passenger jokes 'I have Ebola' http//tco/H8tTjcZ4s0 http//tco/2xNF7Swlh6			
My thoughts and prayers to the family of Nigerian Nurse Who Treated Ebola Patients http//tco/O9hPN70pfI via			
Plane on lockdown in Dublin Airport after man claims to have he does not have Ebola http//tco/wVXnNtK7CY http//tco/VRW8RWbqnw			
Ebola Fact A person infected with is not contagious until symptoms appear http//tco/1zZJaP6HSa http//tco/SB…			
For information on, including FAQ and travel advice, check out the Ebola information page on website htt…			
This Ebola outbreak is like no other - from the hot zone http//tco/F1YChI2b5x http//tco/4Uc5QZDgom			
while US panics over one case, a single clinic in Liberia quietly produces 236 Ebola survivors http//tco/JWthyrd…			
Ebola has brought my country to a standstill Letter from Liberia http//tco/nUqCma6Jc0 http//tco/JNFEQYzhLc			
The highest virus level is found in a dead body, hence currently the highest risk of Ebola transmission is during burial ce…			
Air travel, even from -affected countries, is low-risk for Ebola transmission			
Good pic of media lined up outside Emory Univ Hosp awaiting update Photo http//tco/TDGroMqglU http//tco/OqMâ€¦			
Follow he will live tweet the patient discharge newser from Then flip on Ill be live at noon			
2pm News conference Weve got two mics We expect a condition update on 2nd Ebola patient in Atlanta http//tco/EhER4SqaTs			
Its packed 2pm News conference on the arrival of the second aid worker with Live stream http//tco/SW4QYWCKSz			
Canadian research at the heart of experimental therapy recently given to 2 American aid workers http//tco/Bslhâ€¦			
- Patient Nancy Writebol arrives at Hospital - http//tco/ZL92Q0hSlj http//tco/JHU49dD3Fn			
A copy of the news release from Emory Healthcare http//tco/6CziZesmfz			
outbreak worsens as the says theres been a 40% increase of cases in just 3 weeks http//tco/xqfMTTFrZ5 httâ€¦			
Heading to interview a doctor who survived Check out this road! Its the rainy season in Liberia http//tco/â€¦			

- **Knowledge discovery**

  Our application creates charts to visualize the data; these charts will help us to extract simple knowledge from the results. We will explain these charts in detail and present a value of these charts.

  Figure 4.3 shows the total tweets per year for the Ebola trend. In this chart, we have more than 1000 tweets in 2014 for the Ebola trend, so we can say, for example, 2014 is an Ebola year.

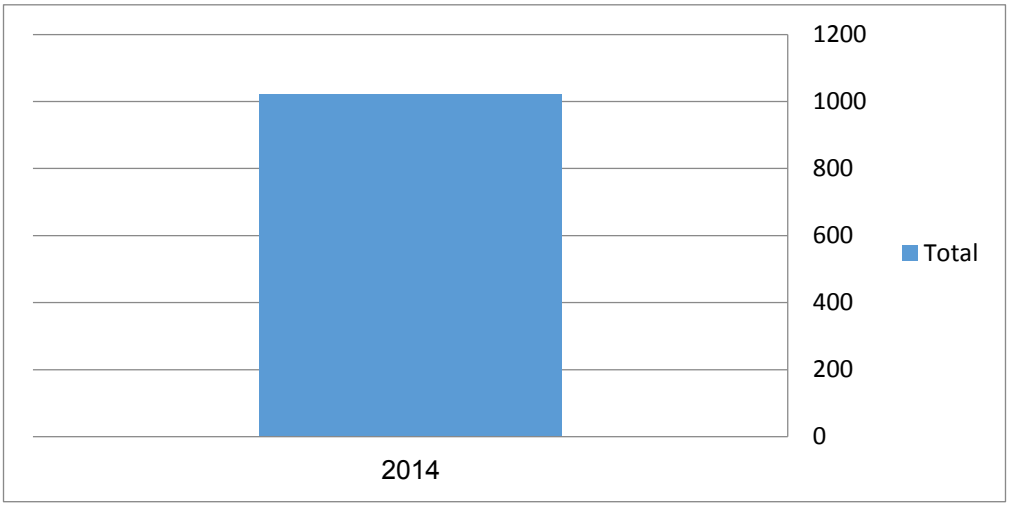

Figure 4.3 Total tweets per year for the Ebola trend

The next chart is more detailed and it shows the total tweets per month for the Ebola trend. In this chart, we note that the discussion about Ebola does not start at the beginning of 2014 but in April, and the maximum value of this trend is in August, with more than 700 tweets, so we can obtain more details and create a chart for only August to discover more knowledge.

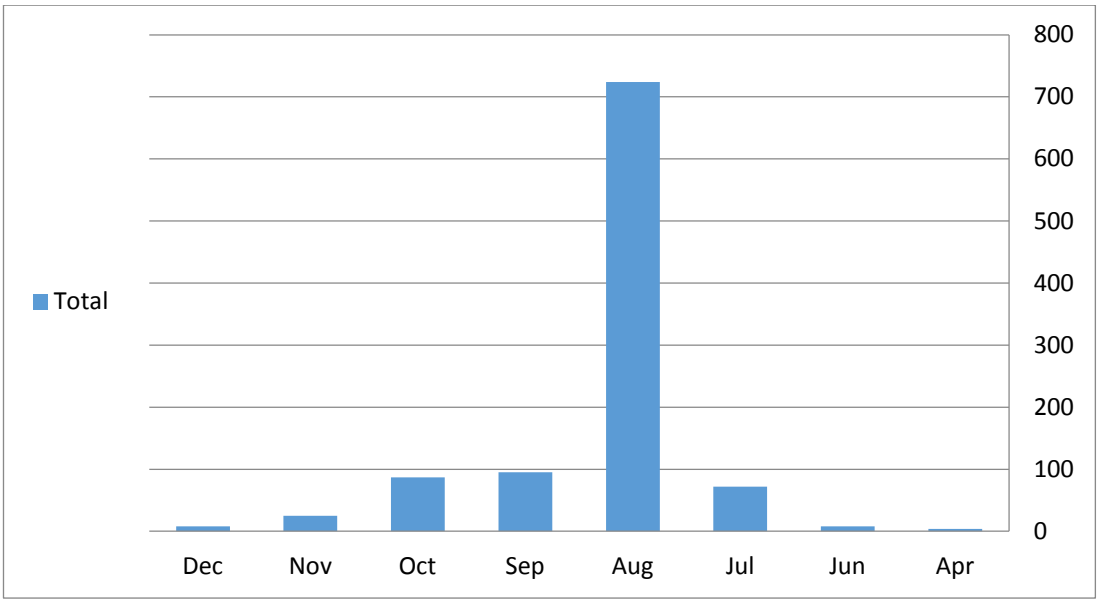

Figure 4.4 Total tweets per month for Ebola trend

The total tweets for August per day in chart 4.5 gives us more details about Ebola. Clearly, from this chart, we note that the tweets curve increases on some days and decreases on others and it reaches a maximum every 7 days. The first day of August in 2014 is a Friday, so we can extract important information from this chart, which says; users' interest in the Ebola trends mid-week; in other words, this trend is less important at weekends.

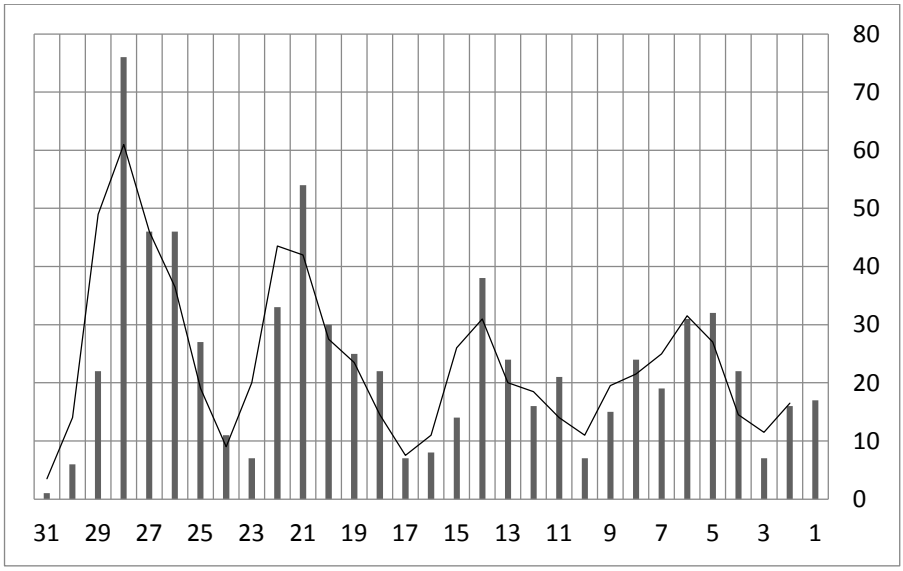

Figure 4.5 total tweets for Aug per day

Lastly, chart 4.6 shows the total tweets for 2014 per day of the week. This chart gives us an overview of the Ebola trend and proves that the users are active with regard to this trend mid-week, then we can make a decision. We can add programs, topics related to same trend on Tuesday, Wednesday and Thursday to discuss it and introduce solutions.

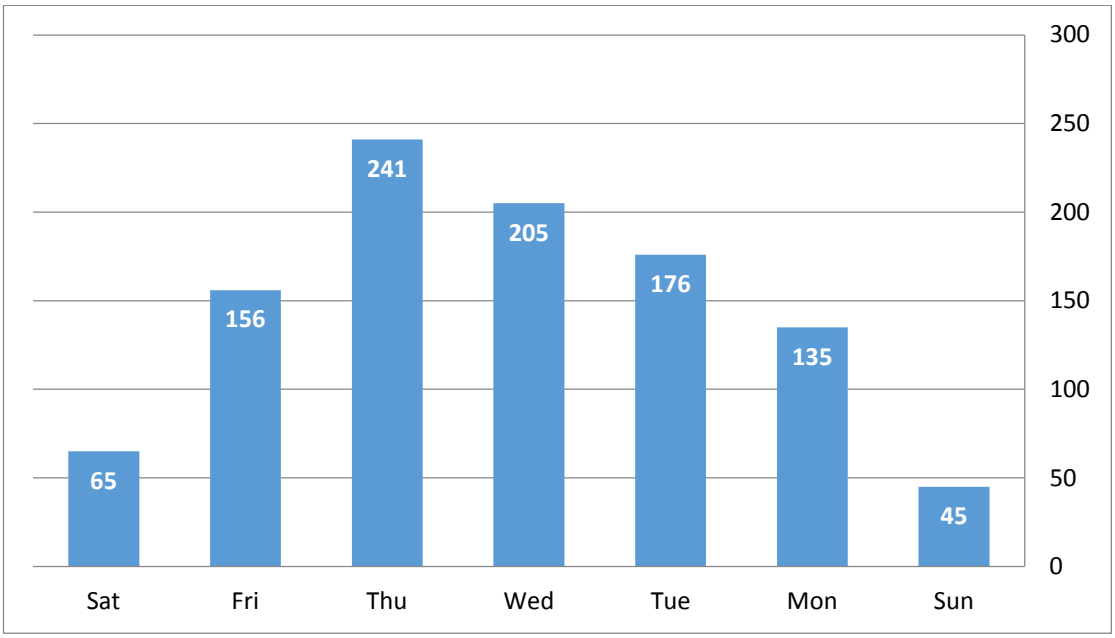

Figure 4.6 Total tweets for 2014 per day of week

Chapter V

EVALUATION

In this chapter, we will show the results of the experiments to present the application performance when we change the training size and number of topics, and compare the size of the retrieved data and extracted information when using our technique and when not using it. Also we will compare two classification methods, online and offline classification. All test experiments were done on Windows 7 OS, 64 bit with 12 GB RAM and an SSD.

5.1 Application performance

In this section, two experiments will be presented to identify the effect on application performance of changing the training data size and the number of topics.

5.1.1 Effect of changing the training data size

In this experiment, we used the size of the training data set as a parameter to determine the effect on the application performance. For this experiment, we used five text files to test it and selected the 400 topics training model to infer the topics from these texts. Then, we started calculating the time until the application gets results. When we performed this experiment, we get the results as shown in Table 5.1 below.

Table 5.1 Effect of training data size

Total Training Size	Text 2 test	Time of 400 topics (sec)
4.385 GB	Ebola	38
	Sql Server	36
	Oil	37
	Football	37
	Malaysian plane	37
1.82 GB	Ebola	25
	Sql Server	24
	Oil	24
	Football	25
	Malaysian plane	24

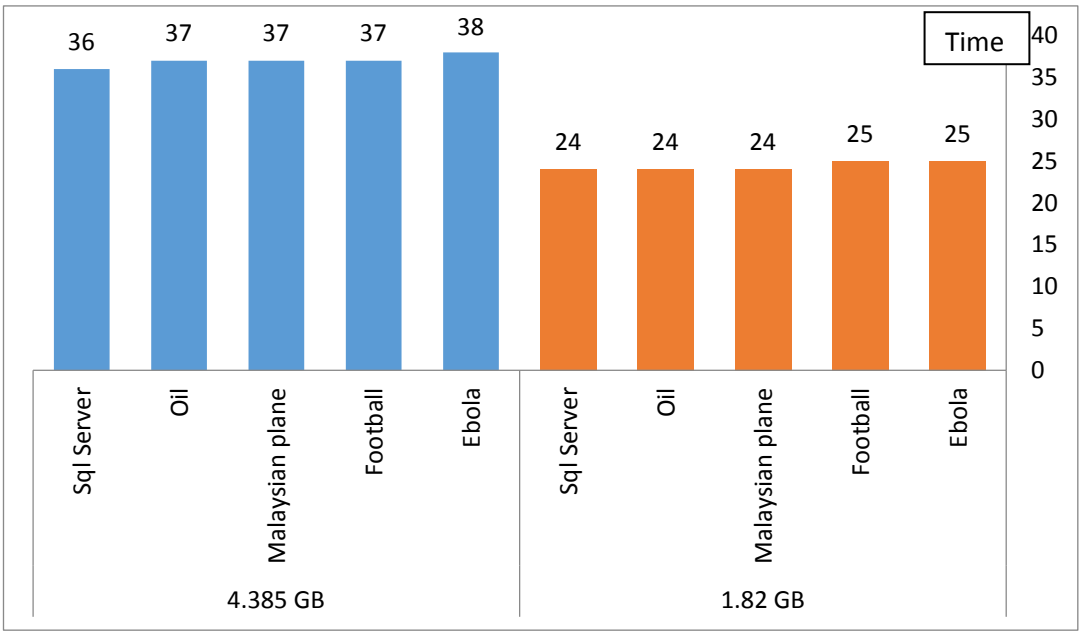

Figure 5.1 Effect of training data size

From this experiment, we note that, when we use the 1.82 GB training data set as the application training model, we get better performance compared with using the 4.385 GB data set, so increasing the size of the training data set will decrease the application performance. This decreases of the performance because we are using the pipe and alphabets from a previously created vectors file to allow the creation, for example, of a test set of vectors that are compatible with a previously created set of training vectors [45].

5.1.2 Effect of changing the number of topics

A. Time of inferring new text

We want to study the application performance when we change the number of topics. In this experiment, we created three topics models, which are 100, 250 and 400 topics inferencers by changing the topics number parameter in the training phase as follows:

We create two tables in the database for each topic model to store the topics, their weights and the training data set size = 4.385 GB. This experiment was performed on VM Oracle Linux OS, which has 6 CPUs and 20 GB of RAM. The first step imports the training data set to the MALLET and converts this data set to the MALLET format (text to vectors). This step takes 30 minutes. The second step will train a topic model from Mallet data files or vectors file and generate three inferences files: inference_100.inf, inference_250.inf, and inference_400.inf.

Table 5.2 Training time for different topic numbers

Total Training Size	# of topics	Inference file generated	Training time
4.385 GB	100	inference_100.inf	14 H
	250	inference_250.inf	17.5 H
	400	Inference_400.inf	24 H

Finally, the testing step uses a trained topic model to infer topics for new documents by using each inference file. For example, we will take the Ebola text file as a test file and then we will infer its topics from inference_100.inf, then repeat this action for inference_250.inf and inference_400.inf. Table 5.3 shows us the execution time for each topic model.

Table 5.3 Inferring time for testing text for different topic models

Total Training Size	Text file	Text size (Bytes)	100 topics time (sec)	250 topic time (sec)	400 topic time (sec)
4.385 GB	Ebola	649	36	37	38
	MSSQL	198	36	36	36
	Oil	2757	37	37	37
	Football	281	37	36	37
	Malaysian plane	1480	36	37	37

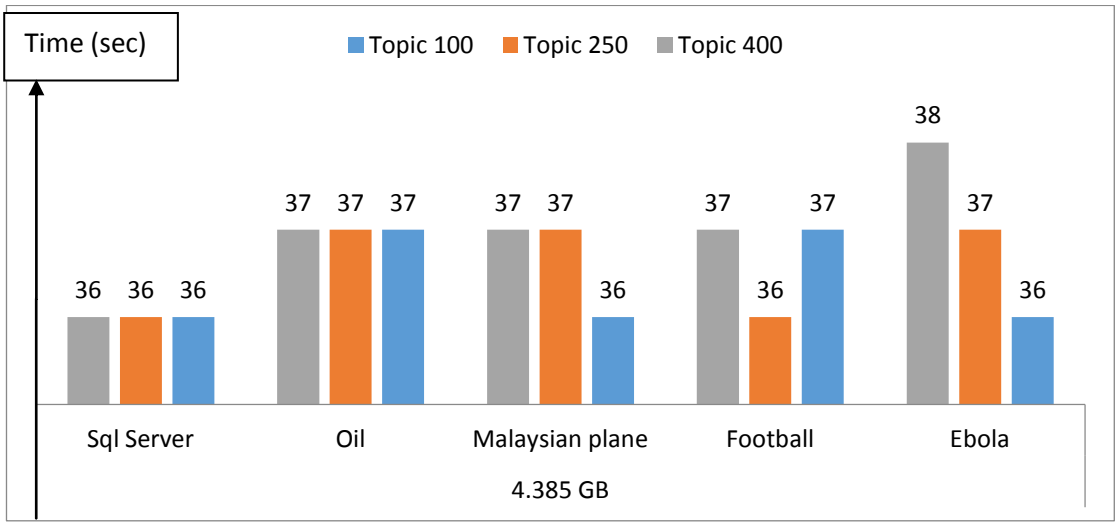

Figure 5.2 Inferring time for each topic model

This experiment shows us that changing the number of topics does not impact on performance. Figure 5.2 shows the difference between the inferring time for each topic, which is (in some cases) one sec.

B. Number of retrieved documents

Depending on the previous experiment which provides results as in Table 5.3, the number of retrieved documents will be affected when we change the number of topics. As we see in Figure 5.3 (total documents for each topic for all testing text), T_100 or 100 topics has 47589 (URLs + Tweets) which is the largest number of documents, while T_250 has 13566 and T_400 has 4878.

That means that increasing the number of topics will decrease the amount of retrieved data, on the other hand, and when we increase the number of topics, we will search efficiently.

Table 5.4 Total retrieved documents for each topic

Testing text	# of topics	Total retrieved URLs & Tweets
Sql server text	400	555
	250	6214
	100	6628
Oil	400	705
	250	390
	100	1906
Malaysian plane	400	1261
	250	1251
	100	3355
Football	400	1151
	250	4367
	100	32354
Ebola	400	1206
	250	1344
	100	3346

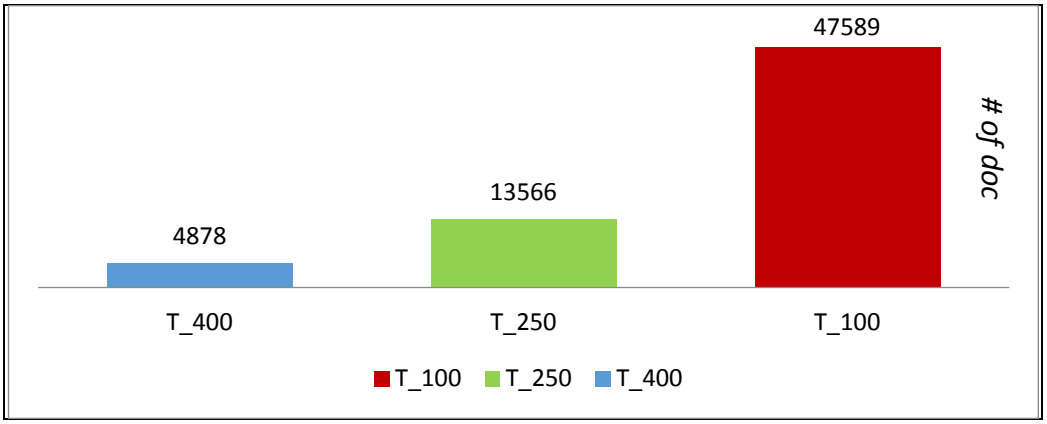

Figure 5.3 Total documents for each topic for all testing text

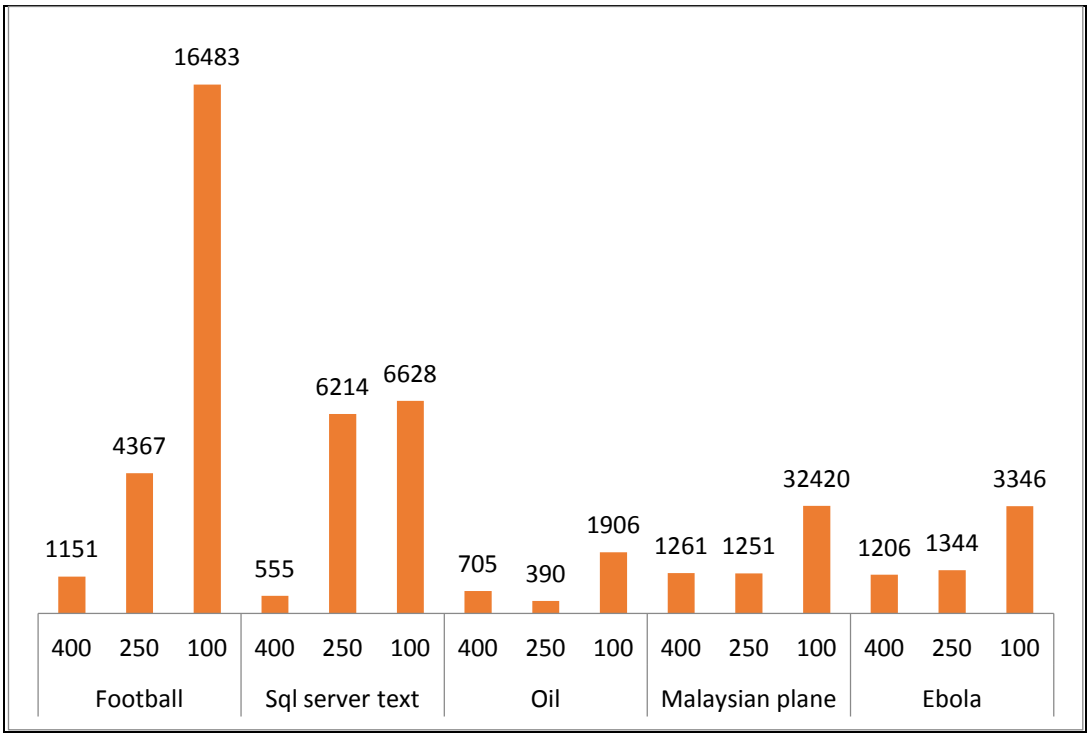

Figure 5.4 Details of total documents for each topic

5.2 Big Data Transportation

Transporting Big Data on networks with a limited bandwidth is a major challenge. In this section, we will evaluate our techniques, which are proposed to solve this challenge.

5.2.1 Transport code and data on the network using a mobile agent

We employ the mobile agent technique to retrieve data and extract knowledge from Big Data; this technique allows us to send the agent with its code and data to the destination (in our case Twitter and web sites) and migrate them from one Big Data set to another to perform their tasks then return the results.

In the mobile agent approach, a computer sends a massage containing procedures and data to another computer. Once the agent is transported, a user agent can interact with a server without using the network. On the other hand, the Client server consists of the client part and the server part. The client sends during the data processing one or more requests to the servers to perform specified tasks and the server part will

provide a service for the clients. This approach enables one computer to call procedures in another, and it has two acts of communication (request and acknowledge) for each interaction, so it needs a reliable network to complete its tasks. Figure 5.5 shows the client server and mobile agent approaches and the interaction messages between the client and server.

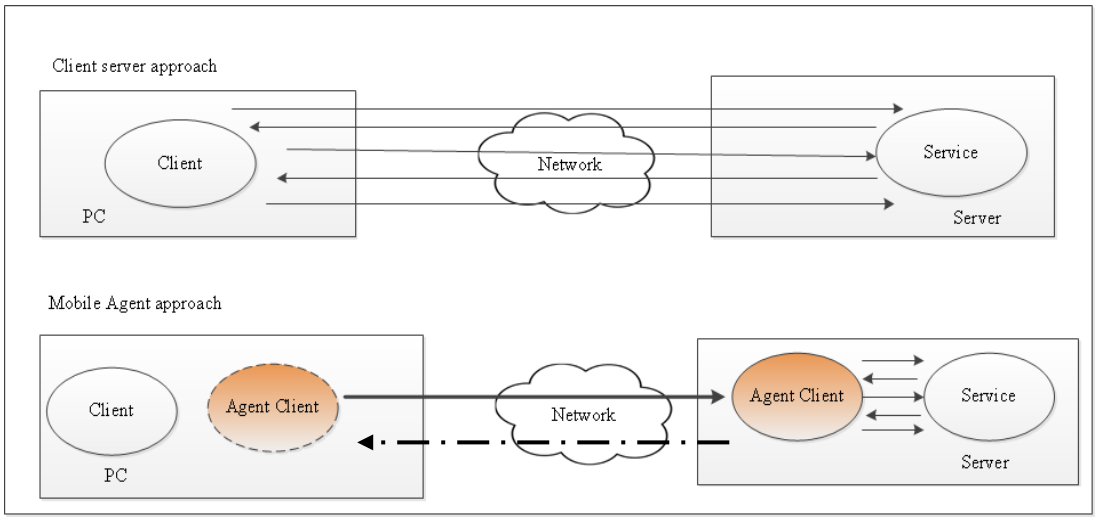

Figure 5.5 Mobile agent and client server approaches

Mobile agents can continue the retrieval task even when the network link goes down. Once the network comes up, the agent will send its results. Mobile agents save network bandwidth by moving data and source code to the destination and interact locally with the destination services [46]. We present a comparison between the two approaches in Table 5.5.

Table 5.5 Mobile agent and client server techniques comparison

Technique	Network traffics	Migration	Computing	Continuity
Mobile agent	Only when transport the agent	Code + data	At Server side	Continue if link goes down
Client server	N request ➔(2N+1) massages	No	Fat client : at client side Lite client: at server side	No

5.2.2 Size of retrieved data and extracted information

Since Big Data transportation is an issue, we proposed metadata storage to store metadata about existing Big Data and employ the mobile agent technique to transport processing data to the destination side and transmit only the results. In other words, we "bring code to data" instead of "bring data to code" [1]. Assuming that a 1 gigabyte per second network bandwidth, thus transferring one Exabyte would take 11574 days, as in the equation below.

$$Transfer\ Time\ =\ (Size\ of\ data)/Bandwidth \quad [47]$$

$$Transfer\ Time\ for\ 1\ Exabyte\ = \frac{10^{18}\ Byte}{10^{9}\ Byte/Sec} =\ 10^{9} Sec$$

$$Transfer\ Time\ for\ 1\ Exabyte = 10^{9} sec * \frac{1\ min}{60\ sec} * \frac{1\ hr}{60\ min} * \frac{1\ day}{24\ hr}$$

$$= 11574\ days$$

In this section, we compare two approaches for retrieving data from Big Data to distinguish the size of the retrieved data from them. The first approach is using the client server without metadata storage and the second approach (which is our technique) is using a mobile agent with metadata storage for Big Data (Twitter and web sites).

1. **Client server without using metadata of Big Data**

   This approach is a simple search process. Assuming that we have an application to search Twitter and web sites, a user will issue a request on the application interface then wait to see the results. In our case, the search process will search all tweets in Twitter and web sites without prior knowledge about the search scope or domain. This approach will retrieve a huge amount of data in an unknown time and does not solve the Big Data transportation issue.

2. Mobile agent using the metadata of Big Data (our approach)

In this approach, the user's request is processed by a topic modeler to find its topic ID. Then, we identify tweets and URLs metadata related to this topic ID from metadata storage. In addition, we employ the mobile agent technique to retrieve specific tweets and URL content by using tweets' ID and URLs from Twitter and websites respectively and/or extract knowledge then return the result. The approach will transport the computing to destination site and decreases the search time because we have prior knowledge about the topic or domain that we are searching. Also, this approach will retrieve the needed data only, which is related to the user's request. Table 5.6 shows a comparison between the two approaches.

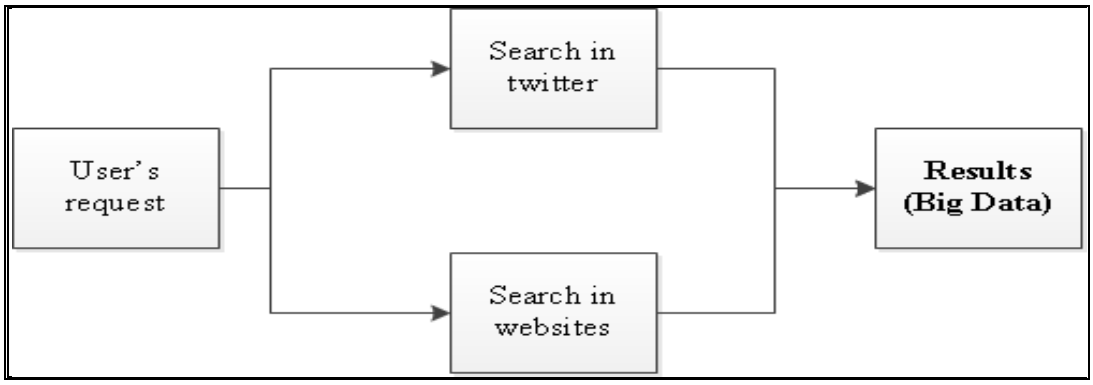

Figure 5.6 Search process without using metadata of Big Data

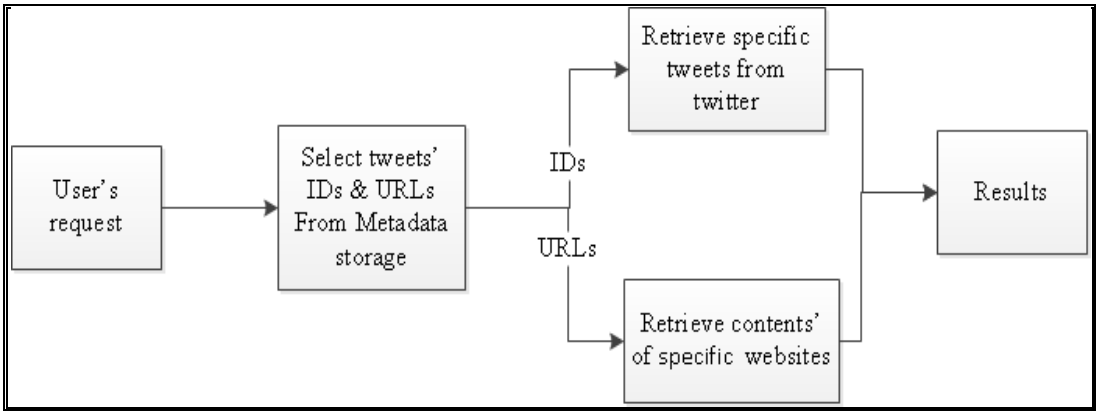

Figure 5.7 Search process using metadata of Big Data

Table 5.6 Comparison between two approaches for retrieved and extracted data

Approach	Size of retrieved data	Processing time	Transportation issue
Without metadata and mobile agent	Big Data	Unknown	Still
With metadata and mobile agent	In MB	Short time	Solved

5.3 Comparing the online and offline classification approaches

In this section, we evaluate two approaches (online and offline classification), which classifies retrieved tweets and the contents of websites, as discussed in section 3.3.6 (**System approaches for Classification)**. We evaluate these approaches from two points of view: Metadata collection and classification time and accuracy of results.

5.3.1 Metadata collection and classification time

The Metadata collection process is the first process in our system, which collects tweets and web sites metadata. This process is followed by the classification process, which is implemented by the online or offline classification approaches.

We performed an experiment to evaluate a classification time for collected tweets, and showed which approach has the best performance. In this experiment, we collected 47 tweets related to one user from Twitter, so in the online classification approach each tweet will be classified immediately after being collected. This means that we import and convert each tweet to the features vector by MALLET then infer its topics by the topic inferencer.

All of these steps will be applied for each tweet. On the other hand, the offline approach offers a different scenario. All related tweets which belong to one user will be collected and pooled in a single file. Once the collection is finished, we import and convert this file to a features vector by MALLET, and then infer its topics by the topic inferencer. All of these steps will be applied to a single file containing all

tweets related to a single user. Figure 5.8 shows the tweets classification time using the online classification approach.

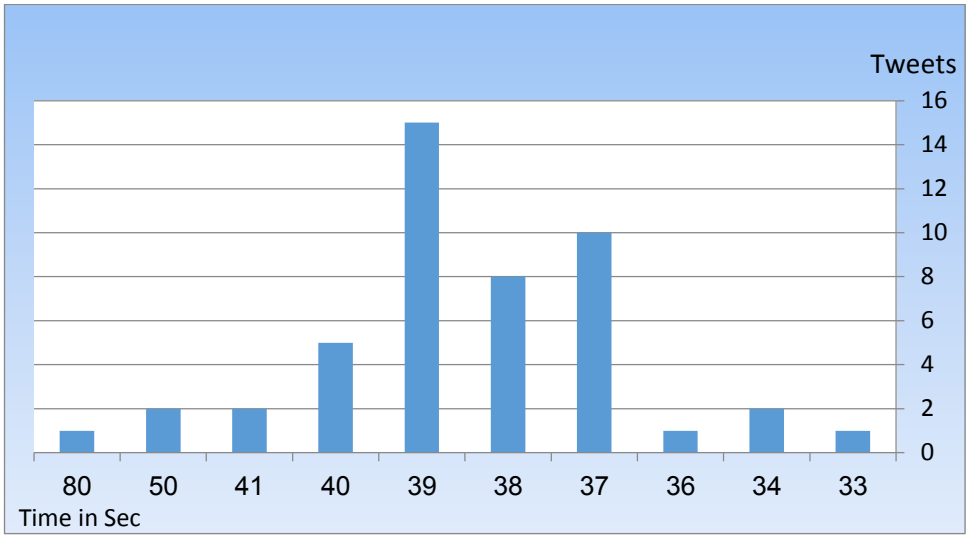

Figure 5.8 Classification time for tweets per sec

In this experiment, we noted that the offline approach is faster than the online one and has good performance. As we see in Table 5.7, the online classification time for 47 tweets takes 1858s while the offline approach takes 38s. In the online approach, the topic modeler converts each tweet to features vector and infers it individually, but in the offline approach, we build a document for each author which contains all tweets related to this author. After that, the topic modeler converts text to the features vector and infers all tweets as a single document.

Table 5.7 Comparison between online and offline classification approaches

Classification Approach	Total tweets	Pooled	Classification time	AVG time	Overall performance
Online	37 tweets, single file for each tweet	No	1858s	39.5s	Poor
Offline	Single file containing all tweets	Yes (pooled by author's tweets)	38s	38s	Good

5.3.2 Accuracy of results

Tweets can be up to 140 characters. This is one of the challenges to the efficiency of the topic models related to short text. LDA as a topic model has less coherence on Twitter [43] so, to improve our training module, we use the hashtags pool technique by aggregating all tweets related to a hashtag into a single document and then we train the topics based on these documents' corpus.

In the classification process, we have two approaches to classifying tweets, which are the online and offline classification approaches; they differ from each other. The offline approach aggregates all tweets by hashtag and classifies these into a single document, then assigns topics with the highest probability to the hashtag. All tweets in this hashtag will have the same topic ID. In the online approach, each tweet will be classified as a single document and assigned to a topic with the highest probability.

We conducted an experiment to find the accuracy of the results for the online and offline classification approaches. In this experiment, we collected tweets for 4 hash-tags which are #ebola, #sqlserver, #gaza and #mh370, as shown in Table 5.8.

All tweets are classified by two approaches (online and offline classification). We present the top topic for each hashtag and the number of tweets related to this topic.

We use purity to calculate the accuracy of the topic modeler, which is a simple, transparent evaluation measure and an external evaluation criterion for cluster quality [48], this means that, if true class labels are known, the validity of a clustering can be verified by comparing the class labels and clustering labels.

In an attempt to cluster with LDA, we suppose that a topic represents each cluster, assign each tweet to its topic of highest probability, and label the cluster by the topic number most frequent in the cluster [43].

To compute purity, each cluster is assigned to the topic number which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned tweets and dividing this by the total tweets in the cluster [48].

Let T_i be the total tweets in cluster i, Q_j the total tweets with label j and N the total tweets in all clusters. Formally:

$$purity(T, Q) = \frac{1}{N} \sum_i \max_j \left| T_i \cap Q_j \right| \qquad (6.1)$$

Table 5.8 shows the highest topic in the hashtags and computes the purity for each hashtag.

Table 5.8 The purity for each hashtag

Classification Approach	Hashtag	Total Tweets	No. of files	Max proportion (ALL tweets)	AVG proportion (ALL tweets)	Top topic	AVG proportion	No. of correct tweets	Purity
Online	***#ebola***	1023	1023	0.139	0.038	303	0.052	524	51.2%
	#sqlserver	1198	1198	0.173	0.058	200	0.076	608	50.8%
	#gaza	5436	5435	0.177	0.047	13	0.062	3554	65.4%
	#mh370	353	353	0.138	0.037	388	0.053	169	47.9%

The purity for the online approach is calculated by using equation 6.1 as follows:

$$purity(T, Q) = \left(\frac{524 + 608 + 3554 + 169}{1023+1198+5436+353}\right) * 100 = \left(\frac{4855}{8009}\right) * 100 = 61\%$$

On the other hand, the offline classification approach aggregates tweets by its hashtag, so we classify the hashtag as one document. All tweets in one hashtag will take the same topic ID with the highest probability. In this approach, we suppose that all tweets in one hashtag are related to it so, if we request the hastag by its topic ID, then all tweets contained in that hashtag must be retrieved.

We show in Table 5.9 the top topic (the highest probability) and its proportion that calculated by the topic modeler for each hashtag in our experiment. From Table 5.9, we note that, the No. of retrieved tweets is equal to the total tweets in the hashtag.

Table 5.9 The proportion for each hashtag

Classification Approach	Hashtag	Total Tweets	Pooled by Hashtag	No. of files	Top topic	No. of retrieved tweets	proportion
Offline	***#ebola***	1023	Yes	One file	T_303	1023	0.434
	#sqlserver	1198	Yes	One file	T_200	1198	0.416
	#gaza	5436	Yes	One file	T_13	5436	0.494
	#mh370	353	Yes	One file	T_388	353	0.512

In the offline classification approach, we are dealing with hashtags pools like free text (websites' contents), so to compute the purity of the offline approach we performed another experiment.

The dataset for this experiment is the 20_Newsgroups, which is a collection of 18,846 newsgroup documents; we are using 7532 files from the newsgroup in our experiment, which is available to researchers at http://qwone.com/~jason/20Newsgroups/.

The dataset contain 20 annotation folders. Each folder represents each newsgroup. Within the folders, there are separate documents representing posts to that newsgroup. Now, to compute the accuracy of the topic modeler, we compute the purity as in the previous experiment. We supposed that a topic represents each cluster, assigned each document to its topic of highest probability, and labeled the cluster by the topic number most frequent in the cluster. The accuracy of this assignment is then measured by counting the number of correctly assigned documents and dividing this by the total number of documents in the cluster. Table 5.10 shows the topic modeler purity for each newsgroup.

Table 5.10 Purity for each newsgroup

Classification Approach	Folder name	Total DOC	Max proportion (ALL files)	AVG proportion (ALL files)	Top topic	AVG proportion	No. of correct files	Purity %
Offline	rec.sport.hockey	399	0.9	0.31	13	0.32	343	86.0%
	rec.motorcycles	398	0.47	0.25	0	0.26	339	85.2%
	soc.religion.christian	398	0.65	0.29	3	0.31	332	83.4%
	rec.sport.baseball	397	0.75	0.25	13	0.27	326	82.1%
	sci.space	394	0.59	0.26	4	0.27	323	82.0%
	talk.politics.guns	364	0.58	0.24	14	0.26	292	80.2%
	sci.crypt	396	0.65	0.26	10	0.28	306	77.3%
	rec.autos	396	0.56	0.22	0	0.24	304	76.8%
	sci.med	396	0.66	0.22	6	0.26	257	64.9%
	comp.windows.x	395	0.9	0.24	7	0.27	246	62.3%
	comp.sys.mac.hardware	385	0.46	0.2	16	0.21	218	56.6%
	comp.os.ms-windows.misc	394	0.92	0.2	16	0.22	219	55.6%
	alt.atheism	319	0.58	0.26	8	0.3	168	52.7%
	comp.sys.ibm.pc.hardware	392	0.55	0.22	16	0.23	200	51.0%
	talk.politics.mideast	376	0.8	0.31	12	0.28	191	50.8%
	talk.religion.misc	251	0.71	0.27	3	0.3	124	49.4%
	comp.graphics	389	0.7	0.21	17	0.23	186	47.8%
	sci.electronics	393	0.45	0.19	19	0.21	184	46.8%
	misc.forsale	390	0.71	0.2	11	0.19	118	30.3%
	talk.politics.misc	310	0.58	0.23	14	0.24	93	30.0%
	$\sum$	7532					4769	63 %

To compute the accuracy of the offline approach, we use equation 6.1 as follows:

$$purity(T, Q) = \left(\frac{4769}{7532}\right) * 100 = 63\%$$

From the above experiments, we note that the offline classification approach is more accurate than the online approach and it enhances the collection and classification time.

Chapter VI

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this research, we built a technique and manager for collecting metadata of existing Big Data in an enterprise or organization. All collected metadata are stored in metadata storage. We built a technique to discover simple knowledge from the existing Big Data (Twitter & websites) and extracted the correlations between different Big Data by using the topic ID.

Since Big Data are distributed across a large number of remote machines, we use mobile agents technology to build our managers. We used the MALLET (Machine learning for language toolkit) tool kit (open source) as an implementation for topic modeling, which uses the LDA algorithm to train topics. We used mobile agents and metadata of Big Data to solve the Big Data transportation challenge in addition to the management challenge.

6.2 Future work

In future work, we will focus on improving the overall system performance and database scalability by building our system on a Hadoop cluster and creating metadata storage on an NoSQL database. Another concern is that we want to build a real time data collector that uses Twitter stream APIs to collect data from Twitter. This approach allows us to extract the topics and refresh the metadata storage in real time.

List of References

[1] S. Kaisler, F. Armour, J. A. Espinosa and W. Money, "Big Data: issues and challenges moving forward," *in Proceedings of the IEEE 46th Annual Hawaii International Conference on System Sciences (HICSS '13)*, p. 995–1004, January 2013.

[2] L. V. Richard, W. O. Carl and M. Eastwood, "Big Data: What It Is and Why You Should Care," IDC, 2011.

[3] "Big Data: science in the petabyte era," 2008.

[4] Douglas and Laney, "The importance of 'Big Data': A definition," *Gartner,* 2008.

[5] H. S. Bhosale and P. D. P. Gadekar, "A Review Paper on Big Data and Hadoop," *International Journal of Scientific and Research Publications,* pp. Volume 4, Issue 10, October 2014.

[6] S. SAGIROGLU and D. SINANC, "Big Data: A Review," *IEEE,* 2013.

[7] F. Gens, "IDC Predictions 2012: Computing for 2020," *IDC #231720,* p. Volume: 1, December 2011.

[8] S. Singh and N. Singh, "Big Data Analytics," *2012 International Conference on Communication, Information & Computing Technology Mumbai India, IEEE,* October 2011.

[9] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Eeverything," *Cisco IBSG,* April 2011.

[10] Intel IT Center, "Planning Guide Getting Started with Big Data Steps IT Managers Can Take to Move Forward with Apache Hadoop Software," FEBRUARY 2013.

[11] "http://en.wikipedia.org/wiki/Semi-structured_data," [Online]. [Accessed 12 10 2013].

[12] D. Garlasu, V. Sandulescu, I. Halcu, G. Neculoiu, O. Grigoriu, M.Marinescu and V. Marinescu, "A Big Data implementation based on Grid Computing".

[13] "http://hadoop.apache.org/docs/r0.23.9/hadoop-project-dist/hadoop hdfs/HdfsDesign.html," [Online]. [Accessed 09 10 2013].

[14] R. Schneider, Hadoop for Dummies Special Edition, John Wiley&Sons Canada, 2012.

[15] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *In 6th OSDI*, 2004.

[16] A. Boicea, F. Radulescu and L. I. Agapin, "MongoDB vs Oracle - database comparison," *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, 2012.

[17] M. Fowler and P. Sadalage, "NoSQL Databases Polyglot Persistence," ThoughtWorks, 08 February 2012. [Online]. Available: http://martinfowler.com/nosql.html. [Accessed 11 10 2013].

[18] G. Press, "A Very Short History of Big Data," [Online]. Available: http://www.forbes.com/sites/gilpress/2013/05/09/a-very-short-history-of-big-data/. [Accessed 09 09 2013].

[19] M. Cox and D. Ellsworth, "Managing Big Data for Scientific Visualization," *ACM*, October 1997.

[20] Hardcover, Indicators of Social Change: Concepts and Measurements, 1968.

[21] P. Lyman and H. R. Varian, "How match information," 2000. [Online]. Available: http://www2.sims.berkeley.edu/research/projects/how-much-info/how-much-info.pdf. [Accessed 22 04 2015].

[22] S. Ghemawat, H. Gobioff and S. Leung, The Google file system, New York, USA: SOSP'03, Bolton Landing, Copyright 2003 ACM 1-58113-757-5/03/0010, October, 19-22, 2003.

[23] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes and R. Gruber, "Bigtable: A distributed structured data storage system," *in 7th OSDI*, p. 305–314, 2006.

[24] J. F. Gantz, D. Reinsel, C. Chute, W. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva and A. Manfrediz, "The Expanding Digital Universe:

A Forecast of Worldwide Information Growth through 2010," *IDC,* Marsh 2007.

[25] M. Hilbert and P. Lopez, "The World's Technological Capacity to Store, Communicate, and Compute Information," *in Science,* 2011.

[26] K. C. d. boyd, "Critical Questions for Big Data," *in Information Communication & Society, Vol. 15, No. 5,* p. 662–679, June 2012.

[27] "http://en.wikipedia.org/wiki/Generative_model," [Online]. [Accessed 22 04 2015].

[28] D. Blei, "Probabilistic topic models," *Communications of the ACM, 55(4),* p. 77–84, 2012.

[29] D. Blei, A. Ng and M. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research 3,* pp. 993-1022, 2003.

[30] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan and X. Li, "Comparing Twitter and traditional media using topic models," *In ECIR'11,* p. 338–349, 2011.

[31] D. Horvat, C. D., M. V., K. P. and K. V., "Mobile Agents and Java Mobile Agents Toolkits," *Proceedings of the 33rd Hawaii IEEE International conference on System Sciences (HICSS)-2000, Maui, Hawaii, USA,* January 2000.

[32] K. Kireyev, L. Palen and A. Anderson, "Applications of topics models to analysis of disaster-related Twitter data," *NIPS Workshop,* 2009.

[33] M. Steyvers, T. H. Griffiths and P. Smyth, "Probabilistic author-topic models for information discovery," *In Proceedings in 10th ACM SigKDD conference knowledge discovery and data mining,* 2004.

[34] G. Vemuganti, "Metadata Management in Big Data," *Infosys labs Briefings, VOL 11 NO 1,* 2013.

[35] S. L. Pallickara, S. Pallickara, M. Zupanski and S. Sullivan, "Efficient Metadata Generation to Enable Interactive Data Discovery over Large-scale Scientific Data Collections," *2nd IEEE International Conference on Cloud Computing Technology and Science,* 2009.

[36] Oracle, "Big Data and Natural Language: Extracting Insight From Text," September 2012. [Online]. Available: http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/18

63164.pdf. [Accessed 22 04 2015].

[37] S. Tedmori, T. W. Jackson and D. Bouchlagem, "Optimising the Email Knowledge Extraction System to Support Knowledge Work," *ECIS,* pp. 681-691. University of St. Gallen, 2007.

[38] B. Klein, X. Laiseca, D. Casado-Mansilla, D. Lopez-de-Ipina and A. P. Nespral, "Detection and Extracting of Emergency Knowledge from Twitter Streams," *UCAmI 2012, LNCS 7656,* p. 462–469, 2012.

[39] F. E. Eassa, H. Al-Barhamtoshy, A. Almenbri, O. H. Younis and K. Jambi, "An Architecture for Metadata Extractor of Big Data in Cloud Systems," *International Journal of Scientific & Engineering Research, Volume 5, Issue 1,* January 2014.

[40] H. M. Al-Barhamtoshy and F. E. Eassa, "A Data Analytic Framework for Unstructured Text," *Life Science Journal,* pp. 339-350, 2014;11(10).

[41] F. Godin, V. Slavkovikj, W. D. Neve, B. Schrauwen and R. V. d. Walle, "Using topic models for Twitter hashtag recommendation," *International World Wide Web Conference, Brazil,ACM,* pp. 593-596, 2013.

[42] Y. Tewari and R. Kawad, "Real-Time Topic Modeling of Microblogs," Infosys Limited Labs in Bangalore, India, March 2013. [Online]. Available: http://www.oracle.com/technetwork/articles/java/micro-1925135.html. [Accessed 11 12 2014].

[43] "Topic Modeling," MALLET, [Online]. Available: http://mallet.cs.umass.edu/topics.php. [Accessed 22 04 2015].

[44] A. Karandikar, Clustering short status messages: A topic model based approach, University of Maryland, 2010.

[45] R. Mehrotra, S. Sanner, W. Buntine and L. Xie, "Improving LDA topic models for microblogs via tweet pooling and automatic labeling," *In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, ACM,* pp. 889-892, 2013.

[46] wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Ebola_virus_disease. [Accessed 22 04 2015].

[47] "MALLET homepage," [Online]. Available: http://mallet.cs.umass.edu. [Accessed 22 04 2015].

[48] R. S. Gray, D. Kotz and J. Ronald A. Peterson, "Mobile-Agentversus Client/Server Performance: Scalability in an Information-Retrieval Task,"

Proceedings of the Fifth IEEE International Conferenceon Mobile Agents, Atlanta, Georgia, pp. 229-243, December 2001.

[49] Cisco Systems, "CCNA 1: Networking Basics v3.1 Instructor Guide," 2004. [Online]. Available: http://www.mtee.net/cisco/ccna1/en_CCNA1_IG_v31.pdf. [Accessed 05 2015].

[50] C. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

Appendices

Appendix A: Glossary

Agent - is a computer program that performs actions on the user's behalf

API - Application Program Interface

Billion - $billion = 10^9$

Exabyte - 2^{18} Bytes

FMC - **F**ree **T**ext **M**etadata Collector, which collects metadata of web pages

GUI - **G**raphical **U**ser **I**nterface

Hadoop - is an integrated environment for distributed data storage and parallel processing

HDFS - **H**adoop **D**istributed **F**ile **S**ystem is a distributed file system designed to run on commodity hardware

JADE - **J**ava **A**gent **DE**velopment Framework is middleware that facilitates the development of multi-agent systems

JSON - **J**ava **S**cript **O**bject **N**otation

LDA - **L**atent **D**irichlet **A**llocation is a generative model that learns a set of latent topics for a document collection

MALLET - **MA**chine **L**earning for **L**anguagE **T**oolkit is open source software, and it provides a Java

Map-Reduce - is a programming framework for distributed computing

Metadata - are data that describe data

NoSQL - **N**ot **O**nly **S**QL is a new type of engine that does not support relations or SQL like RDBMS

OS - **O**perating **S**ystem

Purity - is a simple, transparent evaluation measure and an external evaluation criterion for cluster quality

Screen name - the user name of author in Twitter

SSD - **S**olid **S**tate **D**isk

TMC - **T**witter **M**etadata **C**ollector, which collects metadata of tweets.

Topic – is a collection of words that occur frequently with each other.

Tweet - is a very short Twitter message, up to 140 characters long.

URL - **U**niform **R**esource **L**ocator

ZB - Zettabyte = 10^{21}Bytes

إدارة واستكشاف قواعد البيانات الكبيرة

صخر عوض محمد صالح

بحث مقدم لنيل درجة الماجستير في العلوم [علوم الحاسبات]

كلية الحاسبات وتقنية المعلومات

جامعة الملك عبدالعزيز

جدة – المملكة العربية السعودية

رجب 1436 هـ - مايو 2015 م

إدارة واستكشاف قواعد البيانات الكبيرة

صخر عوض محمد صالح

بحث مقدم لنيل درجة الماجستير في العلوم [علوم الحاسبات]

إشراف

الأستاذ الدكتور / فتحي البرعي عيسى

كلية الحاسبات وتقنية المعلومات

جامعة الملك عبدالعزيز

جدة – المملكة العربية السعودية

رجب 1436 هـ - مايو 2015 م

إهداء

أهدِي هذه الأطروحة إلى والدي العزيز الذي ما فتئ يَحُضُّنِي على طلبِ العلمِ

ويدعمني قولاً وفعلاً

وأهديها أيضا للحنون التي لايزال لسانُها يلهجُ بالدعاءِ لي إلى أمي

وأهديها لمن آزرتني ووقفت إلى جانبي في أصعبِ الظروفِ إلى زوجتي

وهي مهداةٌ لأحبابِ قلبي إلى أولادي

إدارة واستكشاف قواعد البيانات الكبيرة

صخر عوض محمد صالح

المستخلص

تعتبر إدارة البيانات ذات الأحجام الضخمة والتي يزيد حجمها بوتيرة عالية مع اختلاف أنواع البيانات التي يتم تخزينها من أهم التحديات التي تواجه الشركات والمنظمات الكبيرة. علاوة على ذلك فإن استكشاف المعرفة واستقراء الواقع من هذه البيانات الضخمة لدعم نظم اتخاذ القرار يعد عملية بالغة الصعوبة، وذلك نتيجة لاحتياجنا إلى وقت طويل لنقل أحجام كبيرة جدا من البيانات عبر الشبكات ذات السعات المحدودة إلى نقاط المعالجة ليتم بعد ذلك تهيئتها ومعالجتها واستخراج المعلومات المفيدة منها. في هذه الأطروحة قمنا ببناء طريقة تقنية ومديراً للبيانات يقوم بجمع توصيف للبيانات الموجودة في قواعد البيانات الضخمة المتواجدة في المؤسسات والمنظمات الكبيرة، وكل هذه البيانات الوصفية يتم تخزينها في قاعدة بيانات. وفي هذه الأطروحة قمنا ببناء طريقة تقنية لاستكشاف المعرفة من قواعد البيانات الكبيرة المتوفرة.

وبما أن البيانات الكبيرة موزعة على عدد كبير من أجهزة الحاسب البعيدة لذلك سوف نستخدم تقنية الوكيل المتحرك لبناء مدير البيانات الخاص بنا. وباستخدام تقنية الوكيل المتحرك وقاعدة البيانات الخاصة بتوصيف البيانات تم حل مشكلة نقل البيانات وإدارتها.

إدارة واستكشاف قواعد البيانات الكبيرة

صخر عوض محمد صالح

الملخص

إن البيانات التي تتصف بأحجامها الكبيرة جداً والتي تحتوي على أنواع مختلفة من البيانات كالصوتيات والمرئيات والنصوص، وتحتاج في الوقت ذاته إلى سرعة معالجة وكذلك سرعة تخزين، وتعجز عن إدارتها أنظمة إدارة البيانات العلاقية وأدوات التحليل ذات الأنماط القديمة محدودة القدرة، هذا النوع من البيانات يدعى بالبيانات الكبيرة أو البيانات الضخمة. هذه الصفات أو الخواص التي تتصف بها هذه البيانات وما تخفيه من معلومات هائلة في طياتها، ومع زيادة الطلب لاستنباط واستخراج هذه المعلومات من البيانات الضخمة للاستفادة منها في كثير من المجالات والأنظمة المتعددة مثل نظم دعم اتخاذ القرار والمجالات التجارية والعلمية، جعلت الشركات العملاقة والجامعات العريقة تهتم وتتسابق لاقتراح الحلول والعمل في هذا التوجه الجديد.

إن بوادر الانفجار المعلوماتي الذي ظهر في الآونة الأخيرة سبقه تلميحات وتنبؤات كثيرة توحي بزيادة هائلة في كم البيانات التي ستضخ في العالم، ففي عام 1938م احتوت مكتبة جامعة يالي على 2,748,000 كتاب وفي عام 1944م تم التوقع والتنبؤ بعدد ما ستحتويه الجامعة من كتب في عام 2040م بـحوالي 200,000,000 كتاب وأنها ستحتل 6000 ميل من أرفف المكتبة. ولحق ذلك كثير من الإحصاءات والتقديرات للبيانات في العالم، حتى وصلت تقديرات البيانات الرقمية لعام 2006م بحوالي 161 إكسابايت (10^{18}بايت) ثم ما لبثت أن ارتفعت إلى 988 إكسابايت في عام 2010م.

ونلاحظ أن مصطلح البيانات الكبيرة أو ما يعرف بـ (Big Data) مصطلح جديد، وأول ورقة علمية استخدمت هذا المصطلح للدلالة على البيانات الكبيرة كانت في عام 1997م وتضمنت إظهاراً للمشكلة التي ستواجه الأنظمة وعتاد الحاسبات وأنه لن يتناسب مع البيانات الكبيرة.

وأظهرت السنوات الأخيرة حلولاً وأعمالاً لشركات كبيرة لتلبية حاجة البحث في البيانات الكبيرة وتحليلها ومن أمثال هذه الشركات شركة جوجل التي أخرجت ورقة علمية عام 2003م

تظهر فيها نظام الملفات الخاص بها والذي يلائم الزيادة الكبيرة للبيانات وتستخدمه جوجل في عملياتها اليومية كالبحث وغيرها، وهو نظام ملفات موزع على عدد كبير جداً من أجهزة الحاسب، وفي عام 2004م نشرت جوجل للعالم كيف أنها تعالج الكم الكبير من المعلومات على مئات أو الآلاف من الخوادم، وكذلك قامت بنشر ورقة علمية في عام 2006م تشرح فيها نظام التخزين الموزع والذي يستخدم كمستودع للبيانات الخاصة لبعض التطبيقات مثل جوجل إرث وفهارس صفحات الويب، كل هذه الحلول تصب في إطار العمل على التلاؤم والتكيف مع طبيعة البيانات الكبيرة ومعالجتها.

إن العمل في مجال البيانات الكبيرة يجعلنا نواجه الكثير من التحديات والعديد من الصعوبات والتي هي نتاج لطبيعة هذه البيانات، فمن ضمن التحديات التي ستواجهنا إدارة البيانات الكبيرة، يرجع ذلك لتّسارع الكبير في زيادة أحجام البيانات، كذلك نقل البيانات الكبيرة إلى نقاط المعالجة لتحليلها والاستفادة منها يعتبر من التحديات والصعوبات التي تواجهنا بسبب محدودية نقل البيانات على شبكات الحاسب والذي من الممكن أن يستغرق وقتاً قد لا يكون مقبولاً في أحيانٍ كثيرة.

مما سبق يتضح لنا ضرورة بناء أنظمة قادرة على إدارة ونقل ومعالجة البيانات الكبيرة في زمن مقبول وتتلاءم مع المصادر الحاسوبية المتوفرة. ففي هذه الأطروحة قمنا ببناء طريقة تقنية ومدير للبيانات يقوم بجمع توصيفٍ للبيانات الموجودة في قواعد البيانات الضخمة المتواجدة في المؤسسات والمنظمات الكبيرة، وكل هذه البيانات الوصفية يتم تخزينها في قاعدة بيانات. وفي هذه الأطروحة قمنا ببناء طريقة تقنية لاستكشاف المعرفة من قواعد البيانات الكبيرة المتوفرة. وبما أن البيانات الكبيرة موزعة على عدد كبير من أجهزة الحاسب البعيدة لذلك استخدمنا تقنية الوكيل المتحرك لبناء مدير البيانات الخاص بنا. وباستخدام تقنية الوكيل المتحرك وقاعدة البيانات الخاصة بتوصيف البيانات تم حل مشكلة نقل البيانات وإدارتها.

أ- بحوث ذات علاقة

هناك بحوث مختلفة في مجال استكشاف وإدارة البيانات الكبيرة، ويمكن تقسيمها بحسب نوعها الى قسمين رئيسيين:

1. شبكات التواصل الاجتماعية.

يعتبر تطبيق تويتر أحد تطبيقات شبكات التواصل الاجتماعية، والذي يمّكن المستفيدين منه بالتواصل فيما بينهم بواسطة رسائل تبلغ كحد أقصى 140 حرف. ومجمل الابحاث والأوراق العلمية في هذا المجال تنضوي تحت استخراج واستنباط المعرفة من تغريدات مستخدمي تويتر، أيضا محاولة استخراج المواضيع التي تُطرق في فترات زمنية مختلفة والمنثورة في

التغريدات، كذلك جاءت بعض الأبحاث لاستنتاج وسم معين للتغريدات التي لا تندرج تحت وسمٍ ما.

2. الكتابة الحرة.

تنتشر الكتابة الحرة في مواقع شبكة الشبكات (الإنترنت) والبريد الإلكتروني والمعاملات الإلكترونية وغيرها. وركزت الأبحاث في هذا المجال على أهمية إدارة هذه البيانات، وذلك بتوصيف البيانات المتواجدة في المصادر المذكورة سابقا لضمان السهولة والسرعة في الوصول للبيانات والاستفادة منها.

ب- استعراض ملخص لفصول الأطروحة:

تم تقسيم هذه الأطروحة إلى ستة فصول تحتوي على أقسام وهي كالتالي:

1. الفصل الأول: المقدمة

في هذا الفصل تم استعراض المشكلة الرئيسية التي قامت عليها هذه الأطروحة والهدف المراد الوصول إليه. كما تم العروجُ على تعريفات البيانات الكبيرة والتحديات التي تواجه الباحثين في هذا المجال، وبما أن لهذه البيانات صفات تختص بها دون غيرها من كبر حجم وزيادة متسارعة لتلك الأحجام في ظل اختلاف أنواع البيانات التي يتم تخزينها، تم استعراض هذه الخواص أو الصفات وبيان أنواع البيانات التي يمكن لقواعد البيانات الكبيرة تخزينها.

أيضا تم في هذا الفصل إيجاز سريع لبعض التقنيات التي تستخدم في التعامل مع البيانات الكبيرة مثل Hadoop Cluster والذي يعتبر بيئة متكاملة توفر نظام الملفات الموزعة لضمان إتاحة البيانات في أكثر من مكان، وتوفر هذه البيئة طريقة لمعالجة البيانات على التوازي لرفع كفاءة النظام وتنفيذ المهام في أقل وقت ممكن. كما يحتوي Hadoop على قاعدة بيانات تدعم أنواع البيانات المختلفة المتواجدة في البيانات الكبيرة.

2. الفصل الثاني: أدبيات البحث

هذا الفصل يناقش عرضاً تأريخياً للبيانات الكبيرة، فيستعرض العلامات الأولية التي ظهرت في عالم البيانات والذي يوحي بانفجارٍ معلوماتي ضخم لا يمكن التنبؤ بمدى تسارعه، وكذلك تم التعرض لبدايات ظهور مصطلح Big Data والإحصاءات التي أظهرت حجم البيانات الموجودة على مستوى العالم. وفيه تم استعراض وشرح بعض المفاهيم التي لابد للقارئ من الإلمام بها كي تكون مساعدة له في فهم هذه الأطروحة. وأخيراً قمنا بعرض الأبحاث ذات الصلة والتي تناقش مختلف جوانب الموضوع.

3. الفصل الثالث: بنية وتصميم النظام

يناقش هذا الفصل البنية العامة للنظام ويشرح تفاصيل تصميمه ويستعرض أهم مكونات النظام.

- **أهم مكونات النظام**

1) جامع توصيف البيانات (Metadata Collector)

يقوم هذا المكون بجمع توصيف البيانات الخاصة بتويتر ومواقع شبكة الشبكات (الإنترنت). مثل رقم التغريدة واسم صاحبها ومتى أُنشئت ... الخ، وبالنسبة لمواقع شبكة الشبكات فتوصيفها يختلف عن توصيف التغريدات في تويتر، فلها توصيف برقم الصفحة وتأريخ إنشائها ومحدد موقع المعلومات (URL) وغيرها.

2) مصنف المواضيع (Topic Modeler)

يعتبر هذا المكون أهم مكون من مكونات النظام، ومهمته الرئيسية هي تصنيف النصوص والتغريدات التي تم جمعها بواسطة جامع توصيف البيانات. ولبناء هذا المكون تم استخدام أداة تدعى MALLET والتي تستخدم عدّة خوارزميات في عمليات تصنيف النصوص.

3) مستودع توصيف البيانات (Metadata Storage)

هي قاعدة بيانات النظام الخاصة بتخزين توصيف البيانات بعد تجميعها وتصنيفها.

4) مسترجع توصيف البيانات (Metadata Retrieval)

هذا المكون يقوم بجلب توصيف البيانات من قواعد البيانات عند طلبها من قبل النظام.

5) مستكشف المعرفة (Knowledge Discovery)

بعد جلب توصيف البيانات يقوم مستكشف المعرفة بإرسال الوكيل المتحرك الذي أشرنا إليه سابقاً لجلب النتائج من البيانات الكبيرة، والوكيل المتحرك هو عبارة عن جزء برمجي ينتقل عبر الشبكة إلى أجهزة حاسوبية بعيدة ويقوم بتنفيذ المهام المنوطة به ثم يقوم بإرجاع النتائج للمصدر الذي انطلق منه.

- **تصميم النظام**

تحدثنا عن أهم المكونات التي يرتكز عليها النظام، ولكل مكون تصميم خاص به، ولهذا جاء هذا القسم ليناقش وبشكل تفصيلي تصميم هذه المكونات وعلاقة كلٍ منها بالآخر. كما أنه يستعرض مجموعة البيانات (Dataset) التي يتعلم منها النظام استخراج واستنباط المواضيع. هذه البيانات تتكون من 8 مليون تغريدة جمعت من التغريدات العامة لمستخدمي تويتر وخزنت على شكل ملفات بناءاً على الوسم الذي تحمله، وذلك بواسطة دوال يوفرها تويتر لمطوري

البرامج للتعامل مع قواعد البيانات الخاصة به، أيضا تم جمع البيانات من 100,260 صفحة ويب، وكان حجم مجموعة بيانات التعّلم حوالي 4.385 جيجابايت.

وينقسم النظام إلى ثلاث مراحل، المرحلة الأولى هي مرحلة جمع بيانات التعّلم، والمرحلة الثانية مرحلة تعليم النظام (Training phase) لاستخراج واستنباط المواضيع من النصوص وآخرها المرحلة الاختبارية (Testing phase) والتي يتم فيها اختبار النظام والتحقق من صحة النتائج. وخلال هذا الفصل تم التعرض لما اعتمد في هذه الأطروحة من توصيف للبيانات، والتى تم استخراجها من تويتر ومواقع شبكة الشبكات (الإنترنت).

إن عملية تحديث مستودع توصيف بيانات النظام بما تم جمعه من توصيف خاص بالتغريدات ومواقع شبكة الشبكات ، هي من أهم العمليات التي تستمر على الدوام لضمان الحصول على نتائج مرضية، وفي نفس الوقت فإن التغريدات ومحتويات صفحات شبكة الشبكات تصنف بواسطة مصنف النصوص الذي تم تعديله ودمجه في النظام ليقوم بعملية التصنيف بطريقتين، أولها طريقة التصنيف الفورية والتي تعتمد على التصنيف الفوري للتغريدات كلاً على حده، وهي طريقة مكلفة في الوقت، ولكن يمكن بواسطتها تصنيف التغريدات مفردة بدون ربطها بالوسم الذي تندرج تحته. الطريقة الثانية هي الطريقة غير الفورية (offline) وتعتمد على تجميع التغريدات من تويتر بحسب الوسم الذي تحمله وتخزينها في ملفات تحمل نفس اسم الوسم الذي تندرج تحته التغريدات، وفي نفس الوقت يتم تجميع توصيف التغريدات وتخزينها في مستودع البيانات وربطها بالوسم الخاص بها، ثم يتم بعدها تصنيف الملفات والتي تحتوي على تغريدات كل ملف على حدة. هذه الطريقة تزيد من كفاءة النظام وتحسن مستوى دقة النتائج.

ويقوم هذا الفصل أيضاً بعرضٍ لتصميم وكيل متحرك والتعرض لكيفية عمله المتمثلة بجلب النتائج من تويتر ومن مواقع شبكة الشبكات، وباستخدامنا لتقنية الوكيل المتحرك استطعنا نقل برامج البحث والتحليل مع بياناتها إلى أماكن تواجد البيانات الكبيرة لمعالجتها واستخراج المعرفة منها، بدلاً من نقل البيانات إلى نقاط المعالجة والتي تستغرق وقتاً طويلاً، وبهذا نكون قد حللنا مشكلة نقل البيانات الكبيرة على شبكات الحاسب.

- **الفصل الرابع: تنفيذ واختبار النظام**

الفصل الرابع من الأطروحة يستعرض كيفية برمجة النظام، وشرح لأهم الدوال والمكتبات والأدوات المستخدمة في بناء النظام، كما يقوم بعرض مبسط لواجهة المستخدم التي من خلالها يمكن لمستخدمي النظام البحث في البيانات الكبيرة واستكشاف المعرفة منها.

وقد قمنا بعمل تجربة متكاملة على النظام، بإدخال نص معين للبحث عن المواضيع المشابهة له من تويتر ومن مواقع شبكة الشبكات، ومن ثم تم استعراض النتائج والمكونة من التغريدات وصفحات الويب ذات الصلة. كما تم استخراج واستنباط المعرفة من هذه النتائج بواسطة إظهارها بشكل رسوم بيانية يمكن للمستخدم الاستفادة منها.

- **الفصل الخامس: تقييم النظام**

هذا الفصل يناقش التجارب التي تمت على النظام لتقييم كفاءته ودقة نتائجه والحلول التي اقترحتها وقامت ببنائها هذه الأطروحة، وينقسم التقييم إلى ثلاثة محاور، المحور الأول أداء النظام، والمحور الثاني نقل البيانات الكبيرة عبر شبكات الحاسب، والمحور الثالث دقة النتائج.

1) المحور الأول : أداء النظام

وتنقسم التجارب فيه على قسمين

- مدى تأثير التغير في حجم مجموعة بيانات التعّلم (Training Dataset) على أداء النظام، وتم في هذه التجربة تشغيل واختبار النظام على نموذجين مختلفين من نماذج التعّلم بأحجام مختلفة (1.82) جيجابايت و(4.385) جيجابايت. وتبيّن لنا من هذه التجربة أنه كلما زاد حجم بيانات التعّلم كلما أثر ذلك سلباً على أداء النظام والعكس صحيح.
- مدى تأثير التغير في عدد المواضيع المستخرجة من بيانات التعّلم على أداء النظام، ففي هذه التجربة تم تغيير عدد المواضيع التي يجب على النظام استخراجها في مرحلة التعّلم لبناء ثلاثة نماذج تعّلم، وكان عدد المواضيع لكل نموذج كالتالي، (100)، (250) ، (400) موضوع. وتم عمل اختبار للنظام لاحتساب زمن تصنيف النصوص الجديدة لكل نموذج تعّلم، واتضح لنا من النتائج أن التغيير في عدد المواضيع المستخرجة من بيانات التعّلم لا يؤثر في أداء النظام ولكنه يؤثر في دقة النتائج وأنه كلما زادت أعداد المواضيع في مرحلة التعّلم كلما زادت دقة النظام، ولكن مع مراعاة توفر حجم مناسب من بيانات التعّلم.

2) المحور الثاني : نقل البيانات الكبيرة على الشبكة

في هذا المحور تم مقارنة ثلاثة أمور:

- طريقة نقل البرامج وبياناتها عبر الشبكة إلى وجهتها بواسطة الوكيل المتحرك ومقارنتها بطريقة عمل العميل/الخادم:

اتضح لنا في هذه المقارنة بأن طريقة الوكيل المتحرك تنقل عملية معالجة البيانات من أجهزة العملاء التي قد يكون عتادها ضعيفاً إلى الوجهه التي تنتقل إليها، كذلك هذه الطريقة تخفف من العبء والضغط على الشبكة كثيراً، وتضمن هذه الطريقة استمرارية عمل الوكيل المتحرك وإنجاز مهامه على الأجهزة البعيدة حتى لو قطع الاتصال بين العميل والخادم وإرجاع النتائج بعد عودة الاتصال.

- حجم البيانات والمعلومات التي يتم جمعها من البيانات الكبيرة:

هذه التجربة تبين لنا الفرق بين طريقة البحث البسيطة (البحث بدون استخدام الوكيل المتحرك وتوصيف البيانات) ومقارنتها بالطريقة المقترحة في هذه الأطروحة (البحث باستخدام الوكيل المتحرك وتوصيف البيانات)، ومن هذه التجربة استنتجنا أن الطريقة البسيطة سوف تقوم بجمع حجم كبير جداً من البيانات والتي من الممكن جدا أن المستخدم لن يستفيد من أكثرها لأنها ستقوم بالبحث في البيانات الكبيرة بدون أي سابق معرفة بالموضوع العام للبحث، وكذلك نجد أن هذه الطريقة ستستخدم النمط الأول والذي يقوم بنقل البيانات إلى نقاط المعالجة، وهذا النوع من الطرق - كما ذكرنا آنفاً- لن يحل مشكلة نقل البيانات الكبيرة على شبكات الحاسب محدودة القدرة لأنه سيستغرق وقتاً طويلاً جداً في نقلها، ويرجع هذا إلى كبر حجم البيانات الناتجة من عملية البحث. وعلى العكس تماماً، فإن استخدام توصيف البيانات وتقنية الوكيل المتحرك سوف يوفر لنا الوقت المستهلك في عملية البحث، لأن البحث سيتم على قاعدة البيانات المقترحة الخاصة بتوصيف البيانات - بدلاً من البحث المباشر في البيانات الكبيرة - ومن ثم يقوم الوكيل المتحرك بجلب البيانات المحددة فقط من البيانات الكبيرة واستخراج المعلومات منها، وتقوم هذه الطريقة بنقل عمليات المعالجة (البرامج) من أجهزة العملاء إلى أجهزة الخوادم، وبهذا تكون قد وفرت الوقت في نقل النتائج الكبيرة جداً وخففت الضغط على الشبكة بشكل كبير جداً.

- **الفصل السادس الاستنتاج والعمل المستقبلي**

1. الاستنتاج:

في هذه الاطروحة تم بناء مدير للبيانات وتقنيات لتجميع توصيف البيانات الكبيرة (تويتر ومواقع شبكة الشبكات(الإنترنت)) المتواجدة في الشركات والمنشآت الكبيرة. كل هذه التوصيفات تم تخزينها في قاعدة بيانات. كذلك قمنا ببناء طريقة لاستخراج واستنباط المعلومات

من البيانات الكبيرة، واستكشاف علاقة تربط بين البيانات الكبيرة المختلفة وذلك عن طريق ربط المواضيع المتشابهة في البيانات الكبيرة ببعضها البعض. ولأن البيانات الكبيرة موزعة على عدد كبير من الخوادم الحاسوبية، فقد استخدمنا تقنية الوكيل المتحرك إضافة لتوصيف البيانات الكبيرة في بناء مدير البيانات المقترح لحل مشكلتي إدارة ونقل البيانات الكبيرة. وقمنا باستخدام خوارزميات تصنيف المواضيع والتي تصنف المستندات والوثائق النصية ذات الأعداد الكبيرة وتدعى (Latent Dirichlet Allocation) ولتطبيق هذه الخوارزميات تم استخدام وتعديل ودمج أداة برمجية تسمى (MALLET) في النظام. وأظهرت التجارب التي تم تنفيذها على نظام مدير البيانات بأن الوقت اللازم لتصنيف المواضيع في مرحلة الاختبار يزيد مع زيادة حجم مجموعة بيانات التعّلم كذلك أظهرت التجارب عدم تأثر أداء النظام بزيادة عدد المواضيع التي تم استخراجها في مرحلة التعّلم ولكنها تؤثر في الوقت نفسه على دقة وحجم النتائج المسترجعة من البيانات الكبيرة فكلما زاد عدد المواضيع المصنفة في نموذج التعّلم كلما كانت البيانات المسترجعة أدق وأقل حجماً.

2. العمل المستقبلي:

في عملنا المستقبلي سنركز على رفع الأداء العام للنظام وقابلية التوسع لقواعد البيانات وذلك ببناء النظام على (Hadoop cluster) وبناء قواعد البيانات على (NoSQL Engine). ولدينا اهتمام آخر ألا وهو تحديث قاعدة البيانات الخاصة بتوصيف البيانات الكبيرة (تويتر) بطريقة آنية باستخدام دوال (Twitter Stream APIs) بدل من (REST APIs)، هذه الطريقة ستسمح لنا باستخراج المواضيع من تويتر وتحديث توصيف البيانات الكبيرة في الوقت الحقيقي (Real Time).