Towards a Cloud-based Data Analysis and Visualization System

by

Zhongli Li

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In recent years, increasing attentions are paid on developing exceptional technologies for efficiently processing massive collection of heterogeneous data generated by different kinds of sensors. While we have observed great successes of utilizing big data in many innovative applications, the need on integrating information poses new challenges caused by the heterogeneity of the data. In this thesis, we target at geo-tagged data, and propose a cloud based platform named City Digital Pulse (CDP), where a unified mechanism and extensible architecture are provided to facilitate the various aspects in big data analysis, ranging from data acquisition to data visualization. We instantiate the proposed system using multi-model data collected from two social platforms, Twitter and Instagram, which include plenty of geo-tagged messages. Data analysis is performed to detect human affections from the user uploaded content. The emotional information in big social data can be uncovered by using a multi-dimension visualization interface, based on which users can easily grasp the evolving of human affective status within a given geographical area, and interact with the system. This offers costless opportunities to improve the decision making in many critical areas. Both the proposed architecture and algorithm are empirically demonstrated to be able to achieve real-time big data analysis.

Index Terms: Cloud, Geo-tagged Data, Sentiment Analysis, Data Visualization

Acknowledgments

I would like to give my sincerest gratitude and appreciation to my supervisor, Prof. Abdulmotaleb El Saddik, for his continuous guidance and support not only in academic domain but also in my personal life. Thanks for him bring me into MCRLab and let me do the things I like.

Unique and sincere thanks go to Dr. Shiai Zhu for the precious assistance, invaluable guidance, and feedback he supplied going through my research, as well as his review and revision of this thesis. I also need to thank to Huiwen Hong and Yuanyuan Li for helping me on the implementation of the project.

I would also like to thank all my colleagues in the MCRLab for their suggestions and contributions throughout the research, and all my friends for their help on my campus life.

Finally, I am grateful to my family. Their love, understanding and support made me grow up these years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Social Media Big Data

We have entered the era of big data [4] generated by the increasing number of cheap sensors aggregated in the facilities that are used in our daily life. One of the most important application scenarios is the smart city [30], which includes many smart objects equipped with various sensors. For example, smart car is equipped with the sensors that can be used to inform other neighboring cars about the traffic jam [53]. In addition to the hard sensors, Web data (e.g., social networks) is a rich knowledge source that can be utilized to facilitate many innovative applications. In this thesis, I refer the websites or social networks on the Web as the soft sensors. By using the collected big data covering different aspects of our life, high quality of citizens' life and efficient management of the city would be provided with the help of various data analysis techniques.

There's a lot of social media website and Apps today (figure 1.1). Social website provides a convenient platform, on which users can share information with their friends

Figure 1.1: Social Media Web and Apps

and post timely status or attitude. There exist the most up-to-date and heterogeneous data of users. Different social websites provide various functions that users can upload, comment and repost messages with different media types. In addition, friendship and community can also be built on the Web. Plenty of knowledge can be obtained from the users who contributed data and social activities. Many applications benefit from the exploration of such rich resource, ranging from media data understanding to big data visualization. For example, tagged images and videos can be utilized as weakly (inaccurate) labeled training instances for classifier learning. Recommendation can be performed by exploring the common patterns embedded in the crowd-sourcing knowledge of Web users. In addition, geo-location based services can benefit from the plenty of geo-tagged social data. For each user, personal interests can be identified from the history of user social activities. This is especially important for providing personalized services, such as image tagging, search and

recommendation.

Compared to the advantages, there are several challenges. Firstly, the data contributed by general users tends to be diverse, inaccurate and unstructured. Mining knowledge from these data needs careful designs. Secondly, with the huge daily updated data, all the applications and designed algorithms should consider the problem of scalability. Efficient solution is extremely important for analyzing social media. Thirdly, the data on the Web has quite different attributes. It is difficult to represent them in a unified way. For example, image can be represented as visual features, while a user is usually represented with some statistical features such as the number of uploaded images or comments. In addition, the social entities are connected with each other, finally a huge network can be extracted. This further introduces more difficulties on social media analysis. Other interesting issues in social media include the message diffusion on the Web, data organization and visualization, etc. The various facets of the information existing in the massive pool of data, and provide the needed services for general users. The difficulties, raised by the heterogeneity of sensors, data and employed algorithms, lie in two major aspects. Firstly, as it involves many facilities, the increasing complexity poses challenges in the design and implementation of the system. For example, different sensors may provide different interfaces implemented with different computer languages, and deployed on different operation systems. This may result in different naming of the metadata corresponding to a same concept. Furthermore, as the sensors are not independent, the decision making in many applications needs the knowledge from multiple sensors. Data transition and communication would be difficult too. Secondly, increasing size of data poses challenges on the data storage, retrieval and analysis algorithms which are usually computationally expensive. Thus well-designed architecture is needed.

1.1.2 Sentiment Analysis and Predicting

One of the most important aspects in social data is the conveyed human affection, which is the focus of this thesis. Companies can investigate the consumers' opinions on their products to design new marketing strategies based on the discussions on the products in social media. For politicians, the attitude of voters exploited from social media is helpful for predicting the result of an election campaign. A fundamental issue behind these prevalent and anxious needs is to accurately analyze sentiment carried in the user generated data, which has recently enjoyed eruption of research activities.

Sentiment analysis aims to computationally identify people's opinion or attitude towards entities such as events, topics or product features. This work is also known as opinion mining or subjectivity analysis which are used interchangeably in literature. In [45], the differences between sentiment analysis and other researches on affective computing, such as emotion detection and mood analysis, are discussed. In general, the computational representation of sentiment can be either categorical states (i.e., two opposing sentiment polarities) or the level of positivity (i.e., continuous value between two polarities). In this thesis, we will concentrate on the problem of filling data into positive or negative sentiment as categorical states are easier for people to understand and justify. This is essential for many applications such as opinion-oriented summarization and item retrieval.

Sentiment analysis can be performed on different kinds of media types, such as text, image or video. Sentiment analysis of textual document has been a longstanding research field. We can roughly categorize existing works into two groups: lexicon-based approaches and statistic learning approaches. The former leverages a set of pre-defined opinion words or phrases, each of which is assigned with a score representing its positive or negative level of sentiment. Sentiment polarity is the aggregation of opinion values of terms within a piece of text. Two widely used sentiment lexicons are SentiStrength[1] and SentiWordnet[2].

[1] http://sentistrength.wlv.ac.uk/

[2] http://sentiwordnet.isti.cnr.it/

On the other hand, statistic learning approaches treat sentiment analysis as a standard classification problem. A variety of supervised learning approaches are utilized with some dedicated textual features. In addition, sophisticated nature language processing (NLP) techniques such as dependency parsing are developed to address the problems of syntax, negation and irony. These techniques can be found in two comprehensive surveys [45, 33]. On the other hand, visual sentiment analysis attracted extensive attentions recently, as visual instances is exponentially increasing in social media. The basic idea on visual sentiment analysis follows the same way of automatic visual content understanding.

Previous efforts concentrate mostly on opinionated textual documents from review-aggregation resources such as Internet-based retailers and forum Websites, where the fetched texts are usually in stereotyped format and substantially different from the unstructured data on social Websites. As a platform for quick and instant information sharing, Twitter messages are short, and thus difficult to gather sufficient statistics for sophisticated sentiment analysis. In addition, the desired material is regarded as "dirty", since users' presentation may be quite different in style, content and vocabulary (e.g., abbreviations or acronym). Thus, it is much harder for machines to accurately analyze these free-form texts [74]. To address this problem, recent works exploit additional information in social media, such as user relationship [26, 57] or Hashtags [32]. However, this is a subtask for boosting the performance rather than a standalone problem. In this thesis, we will study the fundamental issues of sentiment analysis using traditional techniques, whose performances on Twitter messages are still unclear. This is different from previous studies [46, 45], which investigate this problem on well-structured documents.

As a new and active research area, visual sentiment analysis adopts a similar framework with general concept detection. Supervised learning techniques are utilized on extracted low-level (e.g., GIST and SIFT) or middle-level visual features [59, 31]. In [70, 65], robust feature learning using deep neural networks is introduced into sentiment analysis. Similar to the semantic gap in concept detection, there is also an affective gap in sentiment

analysis. To narrow down the gap, middle-level features defined on a set of affective atom concepts [71] and emotional Adjective Noun Pairs [2, 10] are investigated.

To this end, sentiment analysis is performed purely on textual or visual data. However, Web users tend to post messages containing different media types. For example, Twitter messages are usually attached with images or videos, which are more attractive than texts [69]. The statistical analysis in [69] shows that there is positive correlation between the sentiment detected from texts and images. Thus more accurate results are expected by taking different views into consideration. The challenge coming up with this advantage is to analyze the data types with quite different characteristics. Early or late fusion, which simply combines the results generated from different views, is the most straightforward way [28, 2]. In [68], emotional images are selected by integrating the rank lists produced using visual content and corresponding texts respectively. However, the interaction between different views is ignored. While little effort has been devoted on sentiment analysis of multi-view data, many researches on cross-view or multi-view learning [64, 29] may be helpful to handle this problem. For example, a joint representation of multi-view data is developed using Deep Boltzmann Machine (DBM) in [55]. In this thesis, the different ways for combining the information from multiple views will be introduced and evaluated.

Since supervised learning needs plenty of clean training instances, one important obstacle limiting the progress of sentiment analysis in social media is the insufficient manually justified training data, especially for visual and multi-view analysis. In addition, to promote the research on this problem, a challenging benchmark dataset is needed. In [2], a few hundreds of labeled multi-view Twitter messages are provided for sentiment analysis. This small dataset is further used in [65] for transferring Convolutional Neural Networks (CNN) trained on object images into the domain of sentiment classification. In [70], textual analysis results are utilized as weak labels for pre-training CNN, which is further tuned using annotated 1,269 Twitter images. Other efforts on dataset construction for affective computing include video emotion detection [28] and aesthetics analysis [58, 42]. These

datasets are either too small or no being directly used for sentiment analysis.

1.1.3 Data Visualization

Figure 1.2: Alibaba 2015.11.11 Global Shopping Festival (Beijing)

How to let people, especially the decision-makers to see the data or analysis results clearly is important. We should make the result easy to understand, that makes the data visualization technique become more and more important. Data visualization combine computer graphics and image processing technology, convert data to graphics or images displayed on the screen. Data visualization can help people know better about the data and know the world better. For example, figure 1.2 is the city analysis screen of Beijing in the Alibaba 2015.11.11 Global Shopping Festival and we can read a lot of information on it. In the TED talk, *The Beauty of Data Visualization*, David McCandless said that data visualization gives us a second language of the eye for communicating information. When we combine the language of the eye with the language of the mind (words, numbers and

concepts), we start speaking two languages simultaneously, each one enhancing the other. Not only does data visualization help us communicate information better, it also allows us to detect patterns, trends and correlations in our businesses that might otherwise go undetected.

In this thesis, we build a web system to visualize the data. It can display the data in many different ways to tell the user the meaning of the data. More details will be show in Chapter 3.

1.1.4 Cloud Service

With the emerging of cloud-computing technology [36], it is convenient to deploy and operate a cloud-based system. Many applications [53, 61, 67] are developed under the paradigm of cloud-computing. While developing of a powerful cloud-computing platform is too expensive for many research institutes, some commercial cloud services (e.g., Amazon Cloud) can be used to promote the researches on cloud-based systems.

Cloud computing is a computing style where scalable and flexible IT functionalities are delivered as a service to external customers using Internet technologies. Cloud computing is an evolutionary concept that integrates various existing technologies to offer a useful new IT provisioning tool.

Cloud applications extend their accessibility through the Internet by using large data centers and powerful servers that host web applications and services. Anyone with a suitable Internet connection and a standard web browser can access a cloud application. Rapid evolution of cloud computing technologies can easily blur its definition perceived by the public. In [15], there are five key attributes to distinguish cloud computing from its conventional counterpart:

- **Flexibility** - Cloud computing allows universities to expand or contract computing power as required and allows "bursts" of computing power to be utilised on an "on-

demand" basis. This flexibility helps ensure resource-intensive processes will not slow down other business processes and computing services are always operating at optimal cost.

- **Scalability** - Cloud computing enables universities to quickly scale up their IT operations as provisioning of new computing resources and software applications can be delivered at a desired pace. Furthermore, constraints on pre-purchasing of resources to meet peak requirement in traditional IT no longer exist.

- **Economics** - Traditional IT has multiple fixed and variable cost elements. In order to fulfill business requirements and sustain day-to-day business operations, universities must invest a large fixed amount for initial IT infrastructure establishment and continue to spend variably for software and hardware maintenance. By outsourcing IT functions to the cloud, universities can leverage the features of a lean IT structure to reduce the overall IT expenditures involved in software licensing, infrastructure development, on-going support and upgrades.

- **Inherited resiliency** - Cloud computing removes single points of failure since the Internet is a highly resilient computing environment. Some competitive service providers also add extra functionalities to enhance resiliency. For example, the "Availability Zones" and "Elastic IP Address" features of Amazon.com EC2 allow multi-location of application software and dynamic IP address re-mapping mechanism in an event of service interruption.

- **Highly automated** - Cloud computing services are maintained by dedicated IT professionals of cloud service providers. As a result, universities' IT staff no longer need to worry about complex details behind the delivered computing services, such as hardware maintenance, constant software update, etc

1.2 System Requirements

1.2.1 Functional requirements

We want to build a system that can achieve these requirements:

- **User access control:** There will have multiple users in our system, and different user have different data. In order to protect our data we need a user management module that allows users to register, login and reset password. Furthermore we need create a role-based access control (RBAC) [18] to prevent the data get accessed by unauthorized people.

- **Continuous data collection:** We use the social media as the soft sensor [73] to collect the data from users in the city. If data is not big enough, we cannot have a good result on the city emotion analysis so we need to collect the data all the time. That's means we need a very reliable collecting system.

- **Data mapping:** LBS (Location Based Service) become more and more popular, and we want combine the geo-location and the social media data together and let user browse the spatial information on the city map. At the same time, geographical information should come as textual description [21]. So we need a geo-based visualization module that allows user to browse filtered real-time message posts of multiple SNS on the map.

- **Data analysis and visualization** the data itself is not enough, and we want to mine the knowledge behind the data. Data analysis helps to make sense of our data otherwise they will remain a pile of unwieldy information. After get the result of analysis, we need to show the result in an easy way, which help people understand better. Thats why we need to visualize the data.

- **Building training data set:** The machine learning algorithm always need the training data set. The bigger the train set the better result we will get [11]. So we need to have an annotation function in our system to get the training data from user to modify the machine learning algorithm.

1.2.2 Non-Functional requirements

Our goal is to design a complete runnable system that can be used widely in various applications in the context of smart city. In this section, we will illustrate our design principles by considering the challenging issues raised in the CDP system.

- **Data security:** To ensure the security of collected and processed data, we sacrifice certain storage to backup the important data. While this may cause redundancy, the data can be easily recovered in case of system crashed. In addition, an access control function is needed to make sure our data can only be accessed by the authorized users.

- **User friendly:** User interface should be designed user-friendly. It should provide the most useful functions with the simplest operations to get away from unnecessary problems. Nowadays, a variety of devices with different screen size show up, and the user interface should provide an optimal view and operation experience regardless of the device.

- **High availability:** Encountering runtime errors as well as updating and maintaining system are common and inevitable issues. However, we hope that the state of a single component does not affect the entire system. Therefore, during the design process, each module should be split up to ensure that it is independent of others, decreasing the impact of a module failure to minimum level. The system should also be able to automatically detect settings or data changes and make the appropriate changes without interrupting the ongoing tasks.

- **High scalability:** The system will run on the cloud, so we need to design a system with high scalability. Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth. [3] Methods of adding more resources for a particular application fall into two broad categories: vertical and horizontal scaling. Vertical scaling increases the performance of a single machine, while horizontal scaling increases the number of the machines. Horizontal scaling allows us to use more normal PCs to increase the performance of the system once it is necessary. In order to do that, the service has to be stateless. Service statelessness is a design principle that is applied within the service-orientation design paradigm, in order to design scalable services by separating them from their state data whenever possible. [49]

- **High extensibility:** Future growth needs to be considered in the design process. Extensibility is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. At the same time, it needs to be designed carefully to avoid other traditional software development issues, such as low cohesion and high coupling.

- **High efficiency:** The designed system can be used on different kinds of end devices (e.g., laptop and mobile phone). Since some devises are inherently resource limited and the wireless transition is expensive, resource management strategy should achieve high efficiency in computational cost, transition cost and storage. Basically, most of the computing processes are performed on the cloud. End devices may keep some needed information and light-weight running for visualization and interaction. Thus the data transition between serves and client can be reduced.

1.3 Contribution

In this work, we are faced with more complicated scenarios in the context of smart city. The main contribution:

- Design and develop a cloud based system based on Jersey framework
- Design sentiment analysis algorithm using bag of words and SVM
- Design and development of annotation system
- Design and development of social media (twitter and instagram crawler)

The objective is to build an end-to-end system, namely City Digital Pulse (CDP). It takes the various aspects into consideration, ranging from the data collected from many sensors to the various information visualization techniques. For the system design, we carefully divide the architecture into several functionally independent components. The communications between components are achieved by APIs developed within each component. In this way, the system is more reliable and can be easily extended to include other applications by adding corresponding APIs.

In order to demonstrate the feasibility of our proposed architecture and design requirements, we implement the platform by using the timely social data, which is easily accessed, with large scale and heterogeneous. Thus, it is a suitable choice to evaluate our system performance. In specific, we focus on the application of detecting sentiments in the geo-tagged messages posted in social media. Both the system design and adopted solutions for each part are illustrated. Different from the works in [75, 5], which only focus on one aspect (i.e., visualization [75] or data collecting [5]), we introduce a integrated architecture with optimized solutions on data acquisition, storage, retrieval, process and visualization.

The system is deployed on both our local server[3] and commercial cloud server[4], that can be accessed by the public.

1.4 Thesis Overview

The remain of this thesis is organized as follows:

- Chapter 2 reviews the related works on sentiment analysis and other data visualization system.
- Chapter 3 introduces each component in the framework of our system.
- Chapter 4 shows the implementation details.
- Chapter 5 concludes this thesis. The potential future works on data collection and some interesting remaining issues for further study are included.

[3] http://citypulse1.site.uottawa.ca

[4] http://citydigitalpulse.us-west-2.elasticbeanstalk.com

Chapter 2

Related Work

In this chapter, we first introduce related datasets for sentiment analysis, and then briefly discuss some representative approaches on single-view and multi-view data.

2.1 Sentiment Analysis

The existing works on text sentiment analysis can be roughly categorized into two groups: lexicon-based approaches and statistic learning approaches. The former utilizes prior knowledge of opinion words or phrases, which can be either derived from manually constructed knowledge sources or induced from a document corpus. Two widely used sentiment lexicons are SentiStrength and SentiWordnet, where each word or phrase is assigned with a value to indicate its positive or negative strength. Sentiment polarity of a given text can be determined by aggregating the values of opinion phases within it. However, the size of lexicon is relatively small due to the expensive annotation process. In [25], starting with a small set of justified opinion words, the lexicon is enriched by their synonyms and antonyms extracted from WordNet[1]. However, utilizing common knowledge sources (e.g.,

[1]http://wordnet.princeton.edu/

WordNet) is not able to find domain specific opinion words. Early work in [22] adopts a linguistic approach to extract opinion words which are linked with the available opinion words by conjunctions such as "but" or "and". In [60], the opinion words are identified if they are frequently co-occurred with "poor" or "excellent" in the documents. In [50], word polarity is determined according to the frequency of the word in positive or negative documents.

On the other hand, sentiment polarity classification is naturally a standard binary classification problem. A variety of supervised learning approaches can be leveraged with some unique designs. One of the most important aspects is to derive dedicated textual features. A widely used representation is the Bag-of-Words (BoW) model, where each entry in the feature vector corresponds to an individual term (e.g. word or phrase). Despite the various feature selection methods used to find informative terms in traditional text processing, such as Point-wise Mutual Information (PMI) and Chi-square (χ^2), specific approaches for sentiment analysis adopt adjectives and opinion words through part-of-speech tagging [62, 41]. In addition to unigrams which assume tokens are independent, we can define high-order n-grams, where terms are combination of tokens considering their positions in the textual units. Another important aspect is the choice of supervised learning techniques. The most effective approaches investigated in previous works [41, 46, 13] are Naive Bayes (NB), Support Vector Machines (SVM) and Maximum Entropy (ME). However, there is no conclusion on which one is the best, and the choice of feature and learning technique is variant across different domains. For example, unigrams perform better on movie reviews [46], while conversely, Dave et al. [13] reported a better result using bigrams and trigrams on product reviews.

Natural Language Processing (NLP) techniques are usually used as a pre-processing step in lexicon-based approach [39] or linguistic feature extraction. In [6], a deep NLP analysis of the sentences with a dependency parsing stage is used in their proposed sentiment analysis method. In [38], the NLP techniques are used to analyze time and tense

presentations. The defined two parameters related to the time of ordering products and the time of using the products. Then, the important opinion threads for the two parameters are extracted from the review data.

2.2 Data Visualization

Today's data visualization tools go beyond the standard charts and graphs used in Excel spreadsheets, displaying data in more sophisticated ways such as infographics, dials and gauges, geographic maps, sparklines, heat maps, and detailed bar, pie and fever charts. The images may include interactive capabilities, enabling users to manipulate them or drill into the data for querying and analysis. Indicators designed to alert users when data has been updated or predefined conditions occur can also be included. There's a lot of company release their data vis tools such as SPSS Clementine, SAS Enterprise Miner, IBM Intelligent Miner, Microsoft Analysis Service, DBMiner and so on.

The purpose of data exploration and visualization is to offer ways for information perception and manipulation, as well as knowledge extraction and inference [27, 23]. Data visualization provides users with an intuitive means to explore the content of the data, identify interesting patterns, infer correlations and causalities, and supports sense-making activities. Data exploration and visualization systems are of great importance in the Big Data era, in which the volume and heterogeneity of available information make it difficult for humans to manually explore and analysis data.

A large number of works studying issues related to data analysis and visualization have been proposed. But most traditional systems cannot handle the large size of many contemporary datasets. Exploring and visualizing large datasets has become a major research challenge [20, 63, 40, 23, 54]. In [1], the author classify these works into the following categories: Browsers and exploratory systems, Generic visualization systems, Domain, vocabulary & device-specific visualization systems, Graph-based visualization systems, Ontology

visualization systems, Visualization libraries.

2.3 Smart City

Our system is developed in the context of smart city, which is defined by IBM as the use of information and communication technology to sense, analyze and integrate the key information of core systems in running cities [16]. In order to achieve intelligent responses, various sensors are deployed in the city to collect different kinds of data related to our daily livelihood, environment, public safety and etc [34, 72]. Because of the great research progress on smart city, dynamic scalability can be supported in many applications [56, 44]. In [12], cities' sustainability is improved by using the data mining techniques in the context of smart city. In [7], the existing street lighting infrastructure is enhanced with additional sensors and control to achieve smart lighting. In addition, the rising of cloud-computing and cloud storage offers a great opportunity for data intensive applications, such as the smart video surveillance [14] and house data mining [66]. In the cloud-based system, smart objects are deployed in the bottom layer (hardware), and applications are defined at the top layer. A cloud-enabled middle layer (services) defines different interfaces (e.g., software-as-a-service) to deliver functional values [30, 35, 66, 14].

In the research of smart city, one important issue is the data analysis. In [4], the distinctive characteristics of data analysis in smart city compared to the general big data analysis are elaborated, such as the semi-structured or unstructured data, and data analysis is only performed on demand. Most of the works are devoted on the data from hard sensors, such as [24], where real-time event processing and clustering algorithms are proposed using the intelligent servers from the OpenIoT project. We argue that smart city should also involve the data generated by soft sensors (e.g., social media), which have received intensive attentions recently. The researches on social media differ from each other in the aspect of data type, applied social network, task and also adopted algorithm. For example,

sentiment analysis can be performed on text messages [41, 45], social images [9, 8] or even user networks [26, 57]. The proposed algorithms may employ pre-defined knowledge sources or supervised learning techniques. Furthermore, different approaches are proposed to handle the data from different kinds of resources, such as the stereotyped data from Internet-based retailers [17] and free-formed Twitter messages [26]. In this thesis, we will focus on a framework for smart city by considering the usability on both hard and soft sensors.

Chapter 3

System Architecture Design

3.1 Overview

The proposed City Digital Pulse (CDP) is an end-to-end architecture to facilitate the various aspects in big data analysis, ranging from the data acquisition to the data visualization. As showed in Figure 3.1, the CDP consists of three conceptional parts: sensors, cloud service provider and end devices. Each part may include several subsystems to meet the requirements of different tasks. Sensors can be either hard sensors (e.g. GPS) or soft sensors (e.g., social media), which are responsible for generating data. The collected raw data can be further processed and stored on the cloud server, where data transition and retrieval are also deployed. Most of the computational or storage intensive works were deployed on the cloud to reduce the cost of communication and computing resources in the end devices. Finally, filtered information is delivered to the clients. User inputs are collected by the web applications.

We further illustrate the design by instantiating the architecture using an implemented application, which aims to detect and visualize citizens' sentiment from the social data. Figure 3.2 shows the major components and their interactions in the architecture. Each

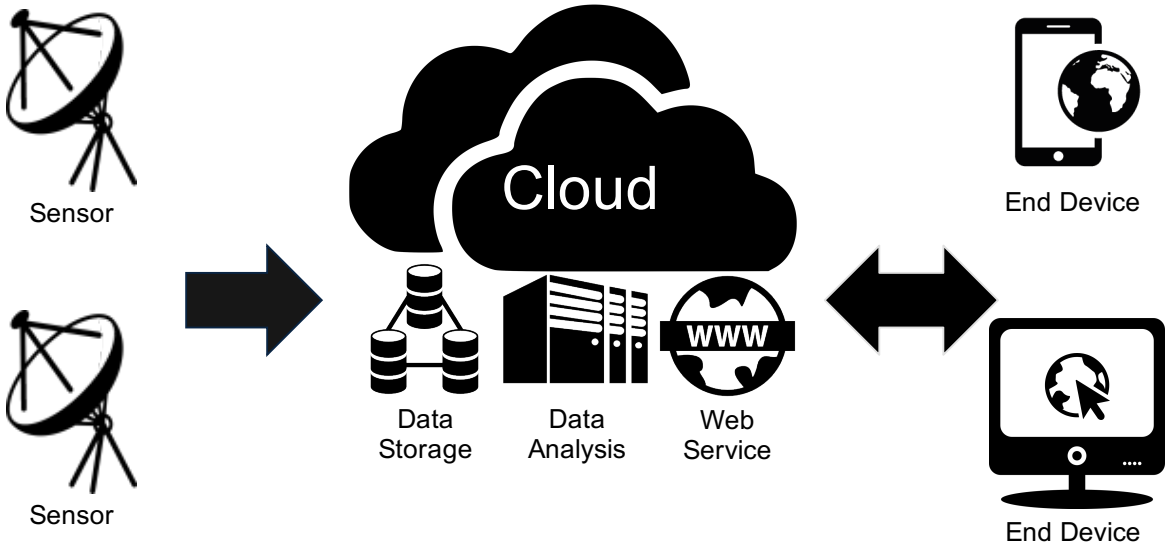


Figure 3.1: System architecture of the City Digital Pulse (CDP).

part is developed separately, and designed as APIs with a unified interface. Thus the overall system can be easily extended. In this example. There are 5 major functional components in the system:

- **Data collector** is used to acquire and upload the data. There can be multiple collectors for different data sources. In our CDP, two collectors were developed for gathering data from two social networks (i.e., Twitter and Instagram). More details of the data collector will be introduced in section 3.2.

- **Data storage** is responsible for structuring and storing various kinds of data in the CDP. To optimize the security, scalability and retrieval speed, two major databases were deployed to store different kinds of data: cache database and main database. The details will be further illustrated in the section 3.3.

- **Data analysis** consists of a set of algorithms, which are used for filtering data, representing data and mining useful knowledge from the data. The adopted solutions are determined by the data types and applications. In our designed system, sentiment

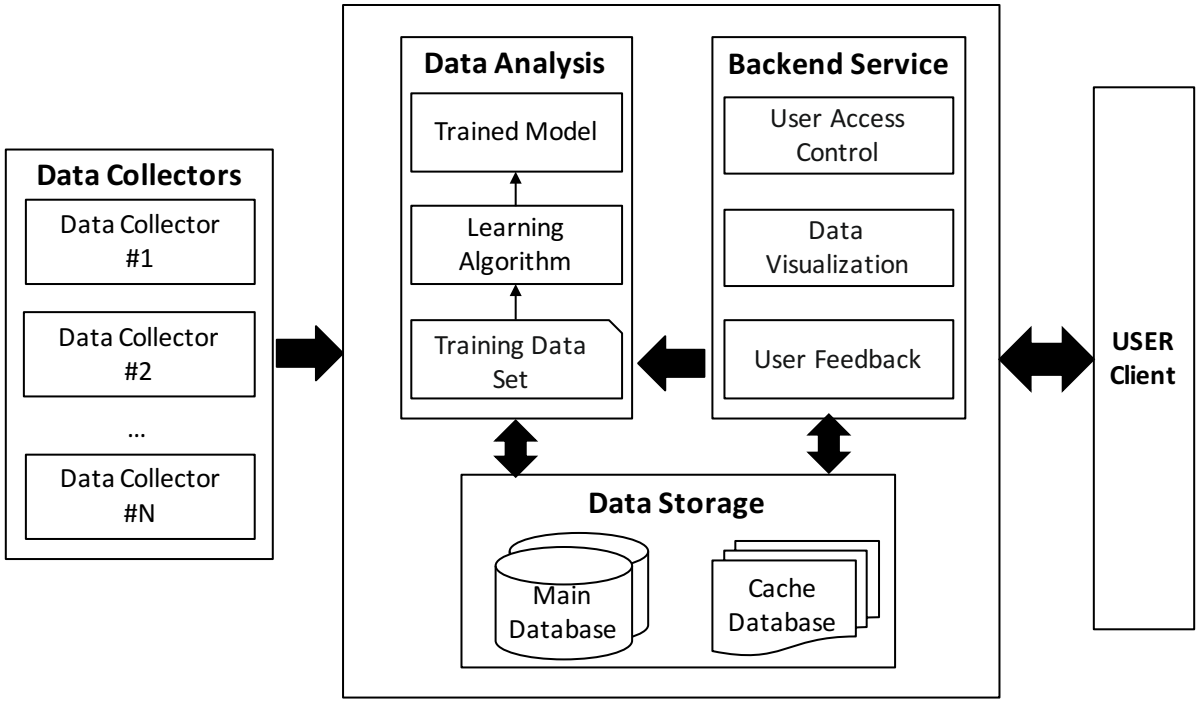


Figure 3.2: The major components and their functional relationships.

analysis is modeled as a classification issue, which involves training data collection, feature extraction, classifier learning, prediction and information aggregation. We will introduce each of them in section 3.4.

- **Backend service** is the bridge of the cloud service and the user-client, it provide the data and the control function through a well-designed RESTful API. The design will be introduced in section 3.5.

- **User-client** provides an interface to show the information explored from the data and receive user feedbacks, such as visualizing sentiment on the map and receive users' annotations on the training data. The bowser is used as the user-client to let the users use the system anywhere on any devices. In section 3.5 we will introduce more details.

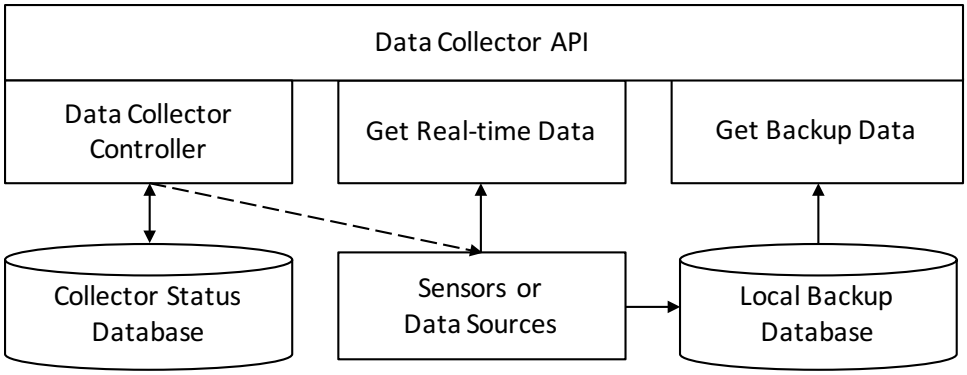


Figure 3.3: Structure of our designed collector component.

3.2 Data Collection

The collectors are responsible for collecting different kinds of data from different data sources, and sending the data to the cloud servers. As the data may be contributed by various sensors, we define a collector for each sensor separately. Figure 3.3 shows the structure of the collector. The intra-connections are also indicated. A data collector should provide three kinds of APIs: **Data Collector Controller API**, **Get Real-time Data API** and **Get History Data API**.

3.2.1 Data Collector Controller API

The collector should follow the software as a service (SaaS) model and can be controlled remotely. Data collector controller API can provide the Create, Read, Update and Delete (CRUD) functions.

- **Create:** Start a new collection task. After the collector is started, data collector controller will change the status in the collector status database and begin collecting data from sensors or other data sources.
- **Read:** Get the status of collector. It will let other part of the system knows the

collector is working or not. If the collector need to restart or crashed or has other problems, it can easily get the status from the status database without reset the Settings.

- **Update:** This allows to modify the setting of the task such as change the address for uploading the real-time data.
- **Delete:** Remove a task from the collector to stop collecting.

3.2.2 Get Real-time Data API

Get Real-time Data API will return the latest data from the sensors or data sources when the collector is running. It's the most important function in the system. In CDP, the collector send the real-time data to the address provided in the collector status database. The collector can be designed to send the real-time data to different addresses.

3.2.3 Get History Data API

The data from sensors or other data sources also goes into the local backup database. This database is designed to store the recent data. If the network has problems or other unusual situation, other part of the system can call the **Get History Data** API to get a lot data using one API call. This makes the system with more availability.

3.2.4 Sensors or Data Sources

There're so many different types of sensors so different sensor adapters are used for different sensors. In the CDP, the social network is used as a soft-sensors and three different types of adapters were designed to handle different situations. More information will be introduced in section 4.2 and section 4.3.

3.3 Data Storage

3.3.1 Data Restructuring

The formats of collected raw data variant across different data sources. For example, most of the data collected from the social media is in JSON (**J**ava**S**cript **O**bject **N**otation) format including very diverse data fields. We can utilize NoSQL database (e.g., MongoDB) to store the non-structured data. However, this will results in high storage cost and inefficient data retrieval, which are two significant issues in big data analysis. Thus we filter the raw data according to the requirements of the system and store the structured data in SQL database. In our example of social messages, the information stored in SQL database includes text, image URL, Geo-location and time stamp. Following this section, we will illustrate our designs regarding the data storage and optimized data transition.

Since different data sources have different formats, we adopt the NoSQL database such as MongoDB allowing us to store non-structured data directly in the database. For example, most of the data collected from the social media is in JSON (**J**ava**S**cript **O**bject **N**otation) format including very diverse data fields. It is difficult and inefficient to extract and store all the attributes in structured database.

However, query is not convenient using NoSQL. Thus we further consider filtering the raw data according to the requirements of system and store the structured data in SQL database. In this way, the data analysis can be performed efficiently, such as message tagging using Nature Language Processing (NLP) module. The basic information stored in SQL includes text, image URL, Geo-location and time stamp.

3.3.2 Replication and Read / Write Splitting

Data is the most valuable property, and Master Slave Replication is used to protect the data. Once the master database is changed by the client, the master database will send

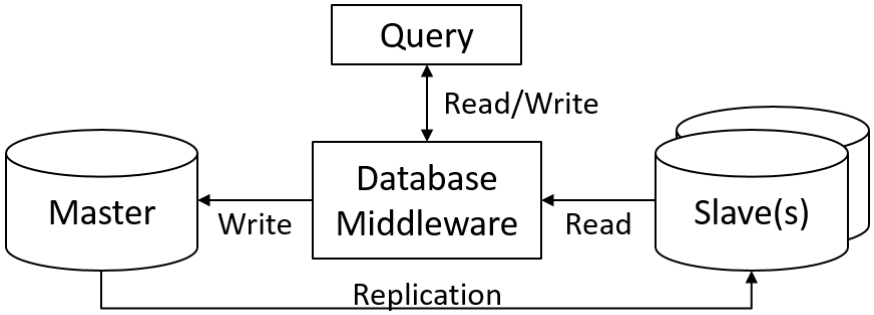


Figure 3.4: Master-Slave Replication and Read/Write Split.

a change log to every slave database. When the slave database receives the change log, it will update the data and make sure they have the same data as the master database. If the master database failed, one of the slave databases will become the master database and make the system working.

After we have the multiple databases we can split writing and reading. In the CDP system, we have the collector to collect the data all the time and users will get data from the database. Splitting the writing and reading will cause the database servers more stable. Figure 3.4 shows the basic idea of the Master-Slave Replication and Read/Write Split.

3.3.3 Indexing and Partitioning

When the system is running for some time, the data in the database will grows to a large number. For example, after continuous operation for two weeks, the data in the database will have over 3 million rows. We need to control the query time within an acceptable range. First we should set add the index into the data table appropriately. If not, when a query comes, the database will scan the whole database and get the results which will take a lot time. The indexes will cost more space so we cannot add the index to every columns. In CDP, we add the index of the timestamp, locations and the text.

The designed databases and corresponding operations are showed in Figure 3.5. As

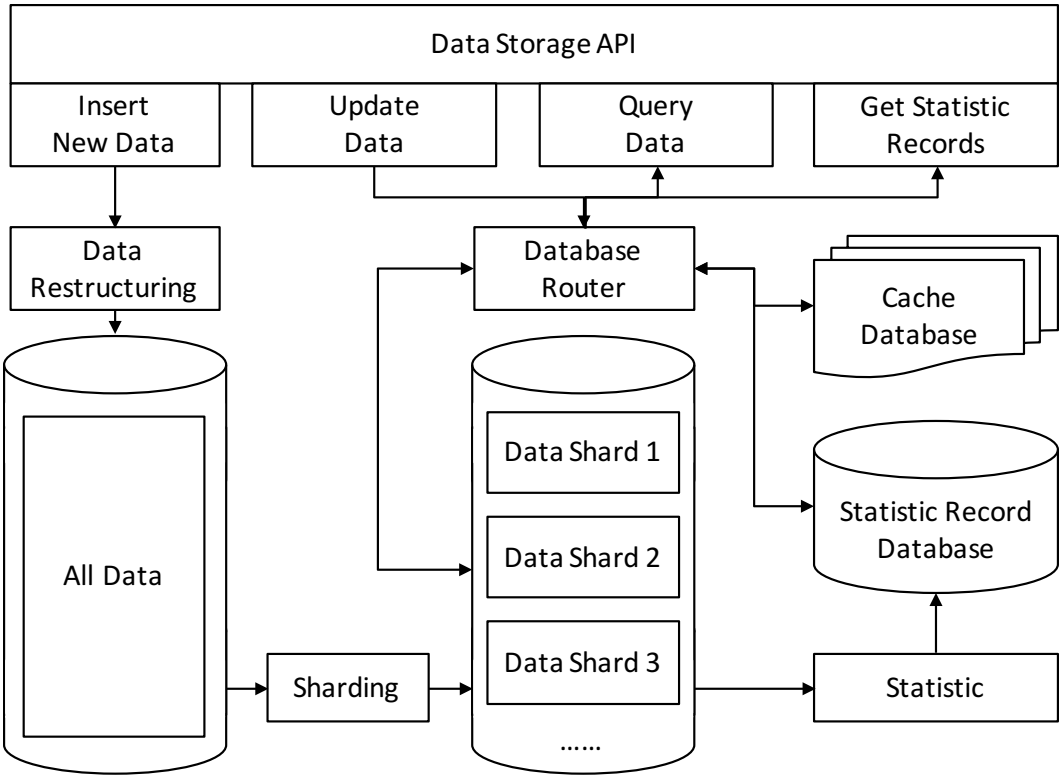


Figure 3.5: Design of the Data Storage part

data security is one of the most important aspects in our system, we use the concept of Master Slave Replication to protect the data. Once there is a change of data in the master database, a change log will be sent to every slave database to update the data accordingly. If the master database crashes, the function of master database will be replaced by one of the slave databases. As data collection is performed continuously, there will be frequent read and write operations in the system. In order to improve the response speed, we split the read and write operation by utilizing the master and slave databases.

When the amount of data generated from the sensors increases, after a certain point, keeping all data in a single database may result in single cant drop in access time. For example, collecting data from 30 cities in Canada, our database reached over 3 million rows just over 20 days after the launch of our system. To address this problem, we split the

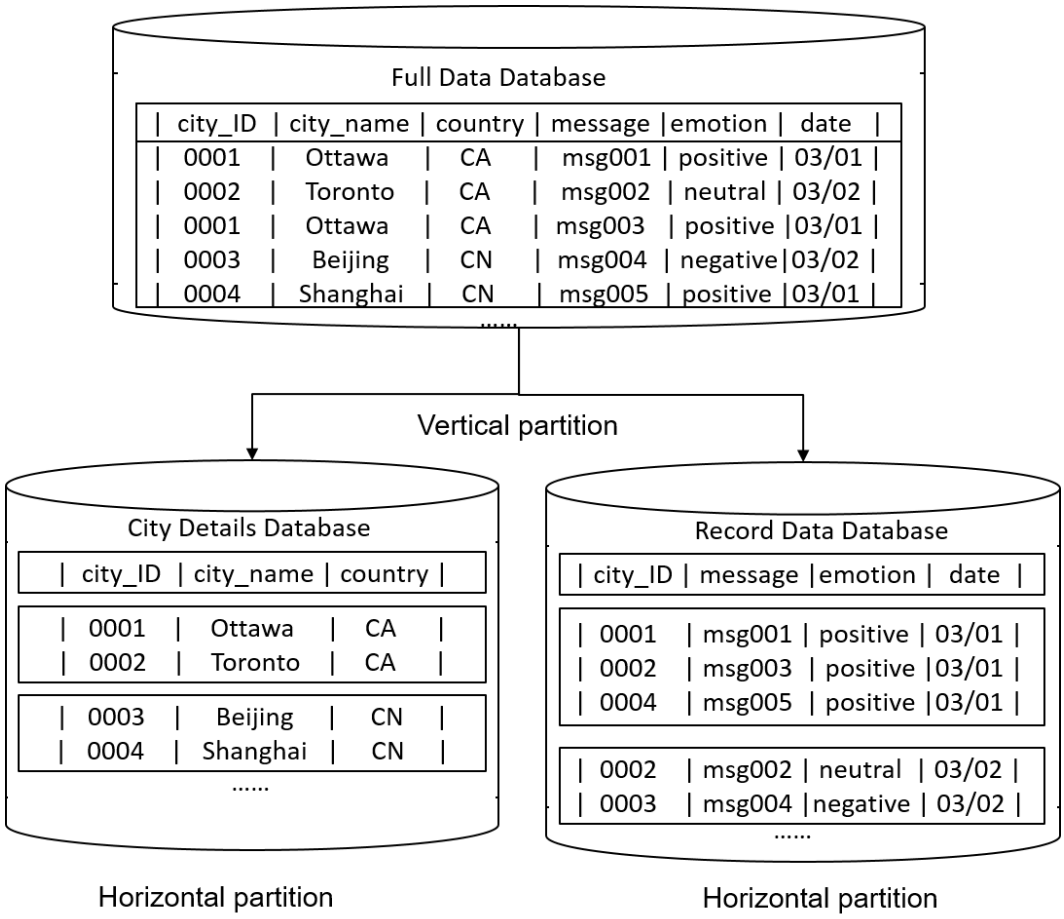


Figure 3.6: Design of the Data Storage part

large database into several small databases. There are two kinds of partitioning schemes. Vertical partitioning is used to split the column of the data table according to the meanings of different data fields. In this way, access to the needed data fields will be improved. Figure 3.6 shows an example of database partitioning. The original database is split into city detail database and record data database. The left one only keeps information related to cities, such as name and country. The right one only keep information related to messages. Many repeated records of city in the original database are removed, and thus the storage space can be saved. On the other hand, Horizontal partitioning (Sharding) splits the data by rows.

Vertical partitioning is the act of splitting up the data stored in one entity into multiple entities. For example, for a user with a lot of feedback history, we put the records into a separate table with a user id reference so that we have the flexibility to move the records into a separate database, or different security context, etc.

Horizontal partitioning (Sharding) involves putting different rows into different tables. When we shard a database, we create replicas of the schema, and then divide the data into different shard based on a shard key. For example, in CDP we use the date to split data into different tables, so the shard shard key is like partmessage 2016 01 01.

Each sub-database includes data having contiguous values in the key field. This is especially helpful for querying a range of data. For example, we split the items in the right database in Figure 3.6 by uploaded date, since it is a common practice to refine a query using a specified time period as filter condition. It is easier to locate the targeted messages in the filtered list. The target messages can be easily located according to the start and end timestamp. The query of different small databases is controlled by the database router which maintains a Shard key for each sub-database (Shard). Note that partitioned database is not effcient for handling complex queries, which require merging the results from different sub-databases. However, the effect on the overall query performance is marginal, since complex queries are rare and the number of corresponding messages is small. To improve query speed, we can add indices into the data table when appropriate. However, this will incur additional storage space especially when all elds are indexed. In our implementation, we only index three fields, namely timestamp, location and keyword, which are frequently used in queries generated by some applications.

3.3.4 Query Optimization and Caching

In CDP, we can analyze the data, allowing users to view statistics of any city in any time period in our database. In figure 4 we can see the shape of a city is not regular. In order

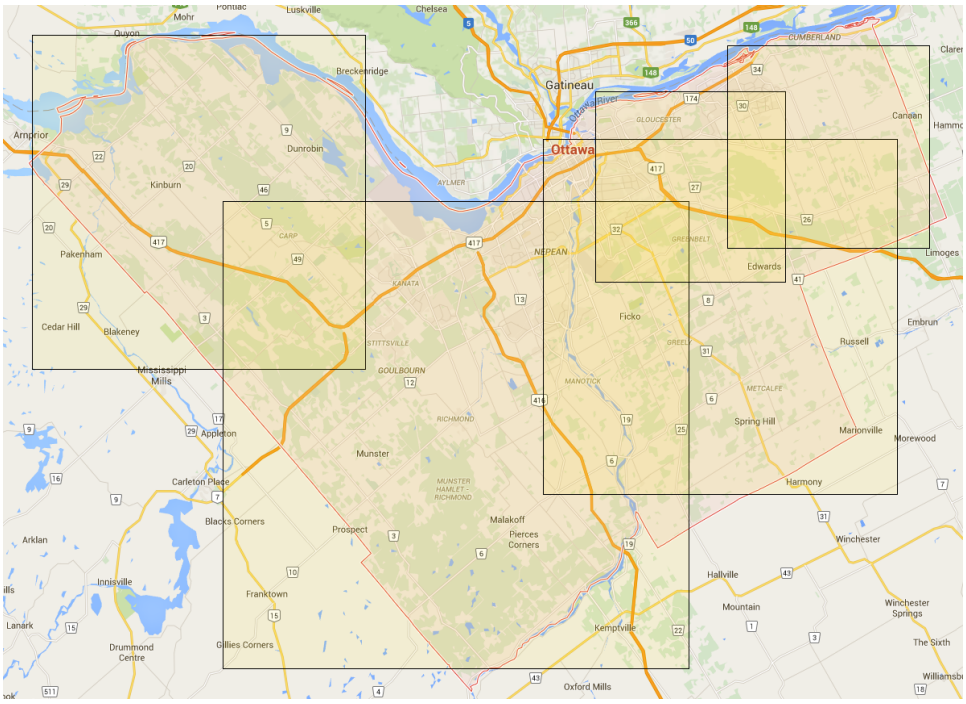


Figure 3.7: Multiple rectangles to illustrate Ottawa

to speed up queries, we use multiple rectangles to illustrate the shape of the city. Figure 3.7 is how we illustrate the boundary of Ottawa.

Well calculate emotional state per city per day and save the statistic record in the statistic record database. We use the unique record key to reference each record. When receiving the query request, the database router will create the record key base on the query parameter and get the data from the statistic record database without having to search all data.

In CDP, we have a lot of data, so we use the hard disk database as our main database. As we know, the speed of the hard disk is much slower than the memory. Caching the popular data into the memory can speed up the query and improve user experience. In CDP, if a user queries data for the first time, the database router will query the main database and return the data. At the same time, database router will store this data into the cache database. If this user or another user queries the same data again, the database

router will return the data from the cache database without querying the main database again.

3.4 Data Analysis

The system can be easily extended to handle many applications, and the techniques in data analysis are vary for different tasks. In this section, we utilize the sentiment analysis of social data as an example to explain the data analysis component in our designed platform. Both the approaches and experimental results are illustrated.

3.4.1 Sentiment Analysis Method

Extensive research has been devoted to sentiment analysis of different media types such as textual documents and visual images. In social media, plenty of opinion-rich textual data is posted to voice users opinion towards different topics. The state-of-the-art approaches on sentiment analysis can be roughly divided into two groups: Lexicon-based and statistical learning methods.

Lexicon-based approaches adopt the lexicons consisting of a set of emotional words, which are assigned with scores to indicate the strength level of the sentiment. Two widely used sentiment lexicons are SentiWordnet[1] and SentiStrength[2]. SentiWordnet is constructed based on the WordNet[3], where words with the same meaning are grouped into a synset. In SentiWordnet, each synset is associated with three numerical scores corresponding to its level of positive, negative and objective sentiment. As one word may belong to multiple synsets, SentiWordnet further defines which one is the first meaning,

[1] http://sentiwordnet.isti.cnr.it/
[2] http://sentistrength.wlv.ac.uk/
[3] https://wordnet.princeton.edu/

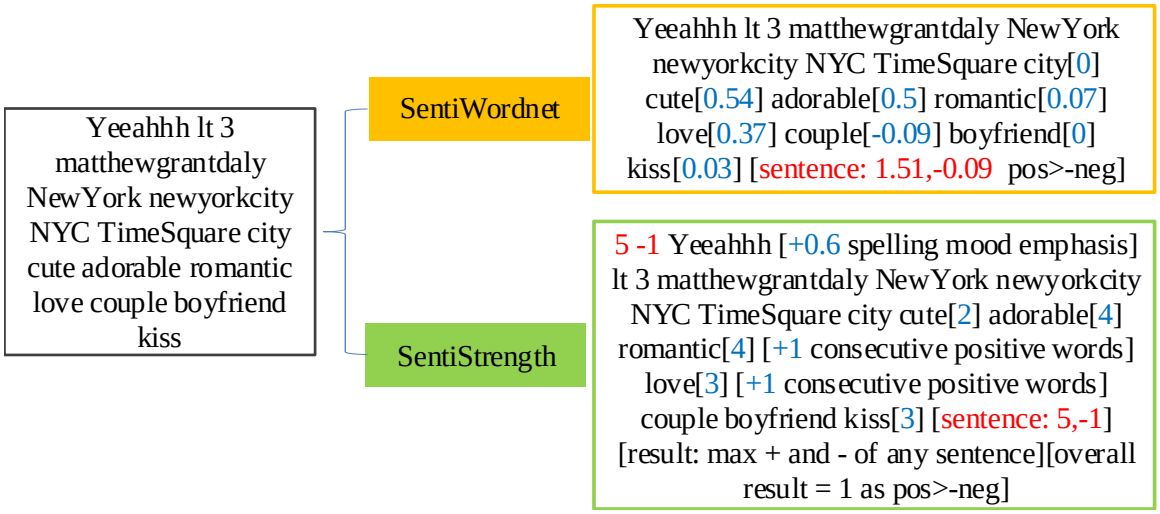


Figure 3.8: Sentiment analysis by using two Lexicons: SentiWordnet and SentiStrength. The contents within "[ ]" is the analysis results. Blue value means the score for each individual word, and red ones are the overall results of the sentence.

second meaning and so on. Given a word, the total sentiment score is computed by weighting its synset scores based on the rank of meanings. As showed in Figure 3.8, adjective word "cute" receives positive score 0.54 in SentiWordnet.

In SentiStrength, the positive strength of a word is defined as a number between 1 and 5. The meaning of 1 is not positive and 5 means extremely positive. Similarly, the negative strength score is between -1 and -5 indicating the sentiment ranging from "not negative" to "overly negative". In the example of Figure 3.8, positive words include cute, adorable and romantic etc. In addition, the scores of consecutive positive words (i.e., romantic) will be increased by one. Thus the maximum positive score in the sentence is 5.

Given a document including N individual words, the overall sentiment score using lexicons can be computed as:

$$Score = \frac{1}{N} \sum_{n=1}^{N} F_n \times score_n, \tag{3.1}$$

where F_n and $score_n$ are the term frequency and sentiment score of word n respectively.

Another way adopted by SentiStrength is defined as:

$$Score = \max_{n \in pos} score_n + \min_{n \in neg} score_n, \tag{3.2}$$

where *pos* and *neg* are the word sets received positive and negative scores respectively. Then the sentiment polarity of a document can be decided in the following way:

$$Sentiment = \begin{cases} Positive, & Score > 0 \\ Negative, & Score < 0 \\ Neutral, & Score = 0 \end{cases} \tag{3.3}$$

With the lexicon-based approaches, the message showed in Figure 3.8 is labeled as positive sentiment using either SentiWordnet or SentiStrength.

Sentiment analysis can be considered as a standard classification problem, which includes two major components: feature representation and model learning. In this thesis, we consider to use the standard textual feature (i.e., Bag-of-Words) and linguistic feature designed for affective computing. BoW feature represents the word distribution in the document. Each element in the feature vector corresponds to a word from a pre-defined vocabulary $\mathcal{V} = \{w_1, w_2, \ldots, w_n\}$. The value can either be a binary value indicating the appearance of the corresponding term or a counting value indicating its term frequency (TF). Other variants include informative vocabulary and weighted term. In this thesis, we use the most popular TF.

Different from the BoW feature, linguistic feature considers the sentimental symbols embedded in the documents. The linguistic feature used in this thesis is as follows [48]:

- The average and sum of positive and negative sentiment scores of all the words. The score of each word is derived from the sentiment lexicons (i.e., SentiWordnet and SentiStrength).
- The number of question marks "?" in the message.

- The number of exclamation marks "!" in the message.
- The number of combinations of exclamation and question marks "!?" in the message.
- The number of uppercases.

For the learning algorithm, the approaches widely used in previous works [41, 46, 13] on text sentiment analysis are Naive Bayes (NB), Support Vector Machines (SVM) and Maximum Entropy (ME). However, there is no conclusion on which one is the best, and the choice is variant across different domains. In this thesis, we adopt the SVM which has been demonstrated to be effective in different applications [19, 2, 28]. SVM aims at searching a hyperplane $f(x) = Wx - b$ that gives the maximized distance to the training data. The optimization function is defined as

$$\begin{aligned} &\arg\min_{W,b,\xi} \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \\ &y_i(Wx_i - b) \geq 1 - \xi_i, \xi_i \geq 0, i \in [1, n] \end{aligned} \tag{3.4}$$

where x_i and $y_i \in \{+1, -1\}$ are feature and label for instance i respectively. Parameter C controls the trade-off between the large margin and the classification error. In addition to LinearSVM defined in Equation 3.4, KernelSVM adopts the kernel trick to map the original feature x into a high dimensional feature space, where the training data is expected to be easily separated. Both the LinearSVM and KernelSVM using the polynomial kernel will be evaluated in the following section.

3.4.2 Empirical Study

In this section, we conduct experiments on SemEval [51] and MVSA [43] datasets to evaluate the different approaches. SemEval is a large dataset including manually labeled tweets constructed for the annually organized competition. MVSA is constructed for multi-view

Table 3.1: Statistics of the two datasets.

Dataset	#positive	#neutral	#negative	#all
MVSA	14,089	2,684	1,851	18,624
SemEval	2,279	3,049	851	6,179

sentiment analysis. Two subsets are provided in MVSA. We adopt the MVSA-multiple dataset, where each tweet is annotated by three annotators. Only the messages receiving at least two same labels are selected in our experiments. In this way, the annotations are expected to be more accurate. Table 3.1 lists the statistics of the two datasets.

The performance is evaluated by accuracy and F-score. Formally, accuracy is defined as

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.5}$$

where tp, tn, fp and fn indicate true positive, true negative, false positive and false negative respectively. F-score is defined as

$$F\text{-}score = \frac{2 \times precision \times recall}{precision + recall} \tag{3.6}$$

where precision and recall are calculated by

$$\begin{aligned} precision &= \frac{tp}{tp + fp} \\ recall &= \frac{tp}{tp + fn} \end{aligned} \tag{3.7}$$

In our experiment, F-score is computed on positive sentiment (F-positive), negative sentiment (F-negative) and neutral sentiment (F-neutral) respectively, and their average is denoted as F-average.

For each dataset, the data was randomly split into 10 groups, where 9 groups are used for model training and the last one used for testing. Table 3.2 lists the average performance of the results for each method. Generally, the statistical learning approaches

Table 3.2: Performance comparison of different approaches on SemEval and MVSA datasets. The best results are highlighted.

SemEval					
	Accuracy	F-positive	F-neutral	F-negative	F-average
LinearSVM+BoW	**0.680**	**0.684**	0.740	0.364	**0.596**
LinearSVM+Ling.	0.675	0.657	**0.742**	0.374	0.591
KernelSVM+BoW	0.655	0.662	0.718	0.369	0.583
KernelSVM+Ling.	0.668	0.650	0.735	0.351	0.578
SentiWordnet	0.551	0.541	0.587	0.371	0.499
SentiStrength	0.562	0.563	0.601	**0.384**	0.516
MVSA					
LinearSVM+BoW	0.752	0.861	0.076	0.265	**0.401**
LinearSVM+Ling.	**0.760**	**0.863**	0.000	0.204	0.356
KernelSVM+BoW	0.755	0.862	0.047	0.247	0.385
KernelSVM+Ling.	0.758	0.862	0.000	0.047	0.303
SentiWordnet	0.642	0.521	**0.127**	0.273	0.307
SentiStrength	0.664	0.564	0.106	**0.284**	0.318

perform better than the lexicon-based approaches. This is because that the messages contributed by uncontrolled users are short, noisy and incomplete. In other words, the sentiment-related clues are weak and difficult to be detected by using lexicons, which only index limited number of emotional words. Thus sentiment analysis on tweets needs more advanced approaches and dedicated designs. The F-positive of statistical learning approach is much higher than F-negative and F-neutral on the MVSA dataset. The reason is that statistical learning methods maximize the overall performance of the classification, and thus may perform poorly on the rare class. This is consistent with the observation in [52]. In contrast, lexicon-based approaches are employed on each tweet independently.

In some cases, it may be helpful to boost the performance of the rare class. Thus, the F-negative of SentiStrength is better than those of other statistical learning approaches on MVSA, where negative instances are much less than others. Comparing the two kinds of features, similar performance were observed on the two datasets. Traditional BoW indicates the word distribution, which may ignore the informative symbols in the tweets. On the contrary, linguistic feature explicitly gives the statistics of sentimental signals. However, some messages may not include the defined symbols. Thus a more advanced feature is needed for social data analysis. Finally, we observe that LinearSVM performs better than KernelSVM on the tweet sentiment analysis. This is not surprising that the representations of short Twitter messages are very sparse. Mapping the feature into a high dimensional space results in a sparser feature, which even hurts the performance. Since KernalSVM will introduce much more computational cost, the efficient LinearSVM with BoW feature was adopted for sentiment analysis. For each coming message, the sentiment score will be computed using the learned classifier and stored in the main database.

3.5 Back-end Service and Front-end Client

The overall system follows the front-end and back-end design principles. Front-end refers to the interface between users and the system. Back-end consists of the functional parts for responding the requests from Front-end, and accessing the data. We use the RESTful API let the front-end and back-end talk to each other. Our interface is developed as Web applications. In the system design, we designed two different frameworks to build our system.

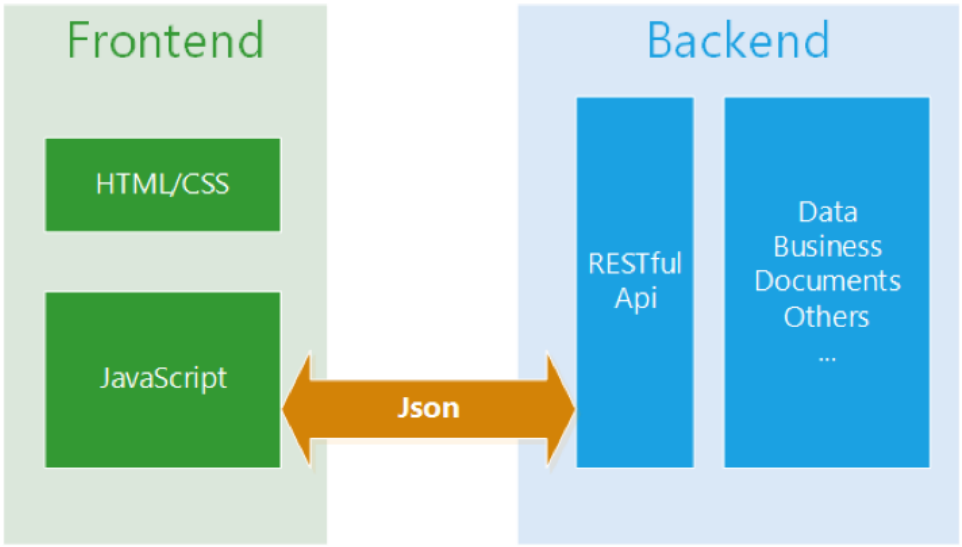


Figure 3.9: The relationship between front-end and back-end

3.5.1 Single Page Application Framework

We use the MVC framework at first because its easy to program.The programmer can do the back-end and the front-end at the same time. We developed our first version of system base on MVC and deployed this system on Amazon Web Service for two months. We successfully collected the training data for our system in the nature language processing part.

Different from MVC, SPA (Single Page Application) adopts a static web page at the front-end. Only the parts with new data will be updated. The back-ends job is to perform data querying, processing and handle business logic. Different function requires different service and the services should work independently. For example, if the user want to see the rank of statistic record of a city, the web service will receive a request from the front-end, then it will send the query request through the data storage API to the data storage part. After received the data, the web service will reassemble the data to the JSON format and send back to the front-end. Thus the reload of the whole web page is not needed.

This also allows the front-end and the back-end to be independently developed and tested; it makes the teamwork more efficient. The separation of the back-end and front-end also brings other benefits:

- **Staticize front-end:** The front-end will only has static content.with HTML and CSS. The content is from static resources and there is no need for the back-end to do construction job. The running environment and engine of the front content are based on the web browser. When to the front-end changes we just need modify the single file and dont need to re-deploy the whole project to the web server.

- **Datalization of back-end:** The back-end can be developed in any language, with any techniques and on any platforms as long as it follows one rule: only data will be provided; nothing on the display of data in the front will be provided. Therefore, the data provided by the back-end can be used in other clients such as on mobile phones and other clients.

- **Platform irrelevance:** The third techniques of front-end are platform irrelevance. And the essence communication with back-end is proper RESTful interface and interaction with JSON data. This can be done on any platforms using any techniques.

- **Framework decentralization:** The framework of the front-end is based on HTML/CSS and has nothing to do with back-end programming languages such as C#, Java and C++. As the front is pure static content, Content Delivery Network or content distribution network (CDN) is a way to work for in large-scale structure project.

CDNs serve a large fraction of the Internet content today, including web objects (text, graphics and scripts), downloadable objects (media files, software, and documents), applications (e-commerce, portals), live streaming media, on-demand streaming media, and social networks.

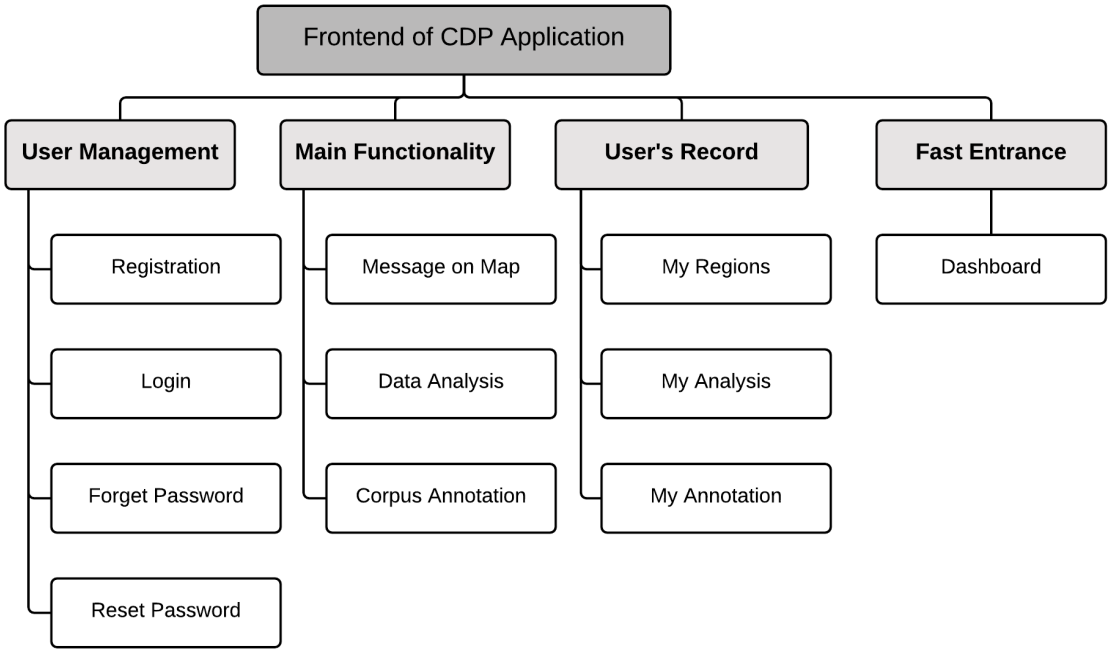


Figure 3.10: The Organization chart of the functionality modules

3.5.2 Front-end Design

To take a further step from the design to the implementation of the web application front-end, we specify and reallocate the functional requirements into eleven modules under four categories, respectively. The organization chart of the functionalities is shown in figure 3.10. And in the developmental process, each module will correspond with a HTML page.

The descriptions about each module (each HTML page)are listed as below:

- **Registration** - Guide user to register an account for the CDP system.
- **Login** - Guide user to login the CDP system with the account ID and the password.
- **Forget Password** - Instruct user to send an email to the registered email address as validation of resetting password when user forget the login password.

- **Reset Password** - Guide user to reset the login password.
- **Message on Map** - Display the message posts that satisfy the data filter options (sending place, social networking service platform, language, keywords etc.) with the sentiment analyzed by the CDP sentiment analysis algorithm on the map based on their geographic locations (latitude and longitudes).
- **Data Analysis** - Display the ranking information of all the public regions with regard to the sentiment analysis result produced by the CDP sentiment analysis algorithms. Public regions refer to the regions defined by the CDP system administrators. Provide the detail information (such as region boundary, hot topics, sentiment changes etc.) of every single region and the comparison information of several regions selected by user.
- **Corpus Annotation** - Let user annotate the sentiment of text and images of the messages by his/her intuition collected by the CDP data collectors.
- **My Regions** - Allow user to add public regions into his/her region list as well as define a new private region on the map and add it into his/her region list.
- **My Analysis** - Allow user to browse the ranking information, single region information and comparison information of only the regions in his/her region list.
- **My Annotation** - Display users annotation record (message text, message images, annotated emotion, annotation date etc.).
- **Dashboard** - A fast entrance to each main functionality.

Chapter 4

System Implementation

In this chapter, the hardware and software used for building the CDP system will be introduced. I will also introduce how we achieve the function and non-function requirements in this chapter.

4.1 Hardware and Software Environment

In this system, each part should work independently, and different parts are able to run on different devices. This system was built in our laboratory located in the University of Ottawa. Two PCs were used to simulate the environment, one for the cloud server and the other for the sensors. The PCs have same hardware and software settings listed as below:

- **CPU:** Intel Xeon Processor E5-2403 v2 (10M Cache, 1.80 GHz)
- **Memory:** 8G
- **Hard Drive:** 1T
- **Operating System:** Windows Server 2012 R2 (64bit)

- **Programming Language:** Java 1.8
- **Server Software:** Tomcat 8, Jersey 2.2
- **Database:** MySQL 5.7
- **Front-end Language & Framework:** HTML5, CSS3, JavaScript, Bootstrap3

4.2 Data Collecting using API

In order to collect data continuously, we built a collecting system to collect the data. In CDP, data is collect from Twitter and Instagram. Both of these social media websites are very friendly to the developers, and they provide the API for the developers to get the data. We developed the collector program using their API. Figure 4.1 shows the process of the Twitter API.

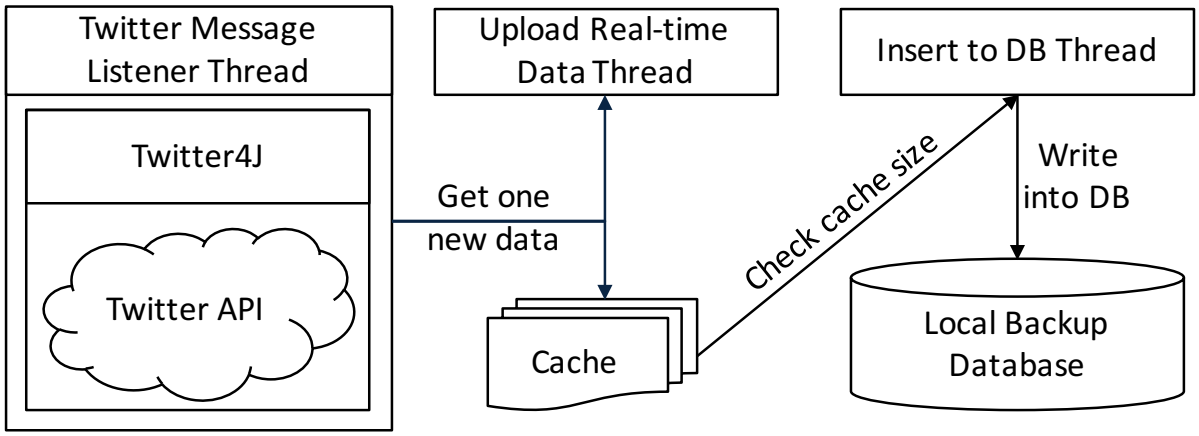


Figure 4.1: Details of Twitter Collector Program

In our system, Twitter4J is used as the client of the Twitter API. Twitter4J is an unofficial Java library for the Twitter API, and a multi-thread collector program was developed to receive the incoming data. When new data comes, Upload Real-time Thread will upload the data to the cloud server. To reduce the workload of the database system,

the new data was written to the cache memory first. At the same time, the Insert to DB Thread keep checking the memory and will upload the data from cache into database if theres enough data in cache. Two different kinds of APIs used in the system will be introduced below:

4.2.1 Streaming APIs

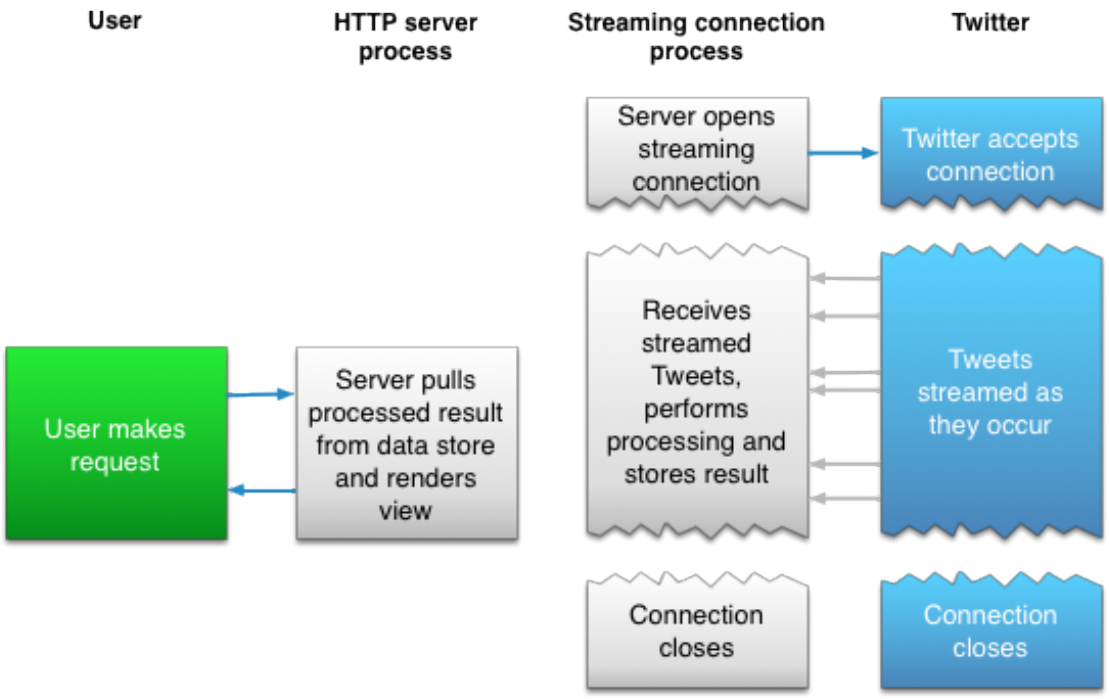


Figure 4.2: Twitter's Streaming API

Twitter's Streaming APIs continuously deliver new responses to REST API queries over a long-lived HTTP connection. Receiving updates on the latest Tweets matches a search query, stay in sync with user profile updates, and more. The streaming process gets the input from Tweets and performs any parsing, filtering, and/or aggregation as needed before storing the results to a data store. The HTTP handling process queries the data store for results in response to user requests. While this model is more complex

than the first example, the benefits from having a real-time stream of Tweet data makes the integration worthwhile for many types of applications. Figure 4.2 shows the process of the Streaming API[1]. The Twitter Streaming API have many parameters[2] but only the locations parameter is used in our system to get the data.

4.2.2 Search APIs

There's another API called Search API, and it will be explained how it was used in our system with an example of Instagram real-time collector.

The basic structure of the Instagram part is similar to that of the Twitter part. One of the difference is that the streaming API is not used in the Instagram so the search request needs to be sent continuously. Another difference is that Instagram only provides the RESTful API endpoint[3], , and the response data (JSON File) needs to be converted into Java objects. Figure 4.3 shows the process of the transformation.

There are two parameters in the Instagram Location Endpoints Search API. One is the latitude and longitude of the circle, and the other one is the start timestamp and end timestamp. This feature made it easy for to get the posts both in real time Instagram posts and to search for historical data. To get the real-time posts of a specified latitude and longitude pair, all it needs to be done is to set the start timestamp a few minutes later than the current timestamp, and the end timestamp to be set as the current timestamp. Both the end and start timestamp will be increased automatically to continually get the posts. This is not exactly a real-time but with a delay of about 5 minutes. However, this 5 minutes delay is acceptable in our system as the system is not time-dependent. To get Instagram posts for a specified time period of a latitude and longitude pair, the start and end timestamps can be used to send requests through this search API to get the posts.

[1]https://dev.twitter.com/streaming/overview

[2]https://dev.twitter.com/streaming/reference/post/statuses/filter

[3]https://www.instagram.com/developer/endpoints/

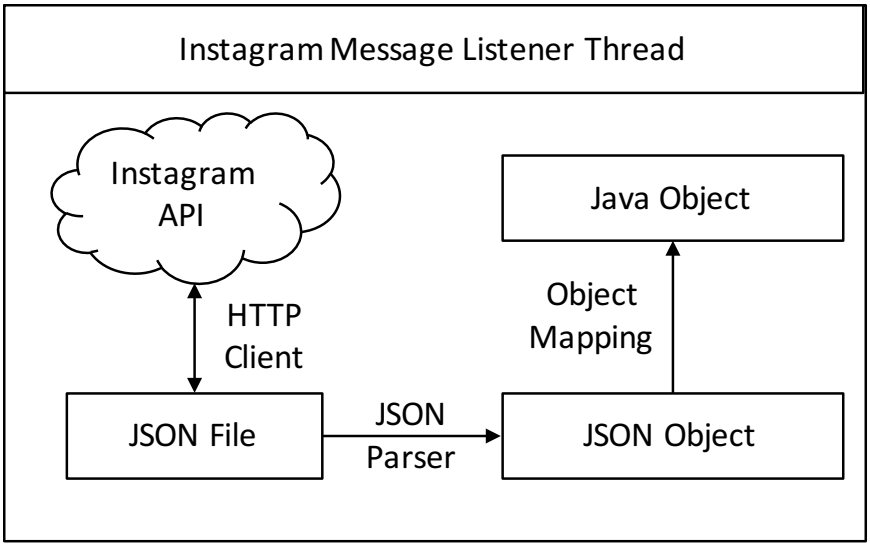


Figure 4.3: Transfer the JSON response into Java Object

4.3 Data Collecting using Web Crawler

Although the API is easy to implement but there still have some limitations. For example, in Twitter search API, data can only be searched for the last 7 days. If we want more data the web crawler needs to be used. This section will introduce how we use the web crawler to get the data from Twitter.

4.3.1 Why Web Crawler

Although using API is very easy but there's still some reasons that we should develop a web crawler for data collecting. Twitter has some limitations of the APIs. From the detail introduction of Twitter search API[4], it shows that the API can be called 450 times every 15 minutes by one token with maximum 100 tweets per call. This sounds good, but Twitter also says that: "Please note that Twitters search service and, by extension, the Search API is not meant to be an exhaustive source of Tweets". Not all Tweets will be

[4] https://dev.twitter.com/rest/reference/get/search/tweets

indexed or made available via the search interface. Sothe result we got using APIs may not be complete. That is not the result that we are looking for. The principle of web spider is to analyze known URL, extract new URLs then start a new process of analsis. That means the web crawler can get the expected data.

Another reason is we want to collect the data when the data is unstructured or does not have API for developer. How can we go fetching them? We could always hire people to manually log on and save the info into an Excel sheet but the process gets tedious and impractical. This is where Web Crawling comes into play. There is a good chance for automated web crawling to gather data which will be processed to make business decisions. Web Crawling technology was made popular by Google for its use in their search. They were the first to see the importance of immense amount of data on the web which was then not crawled and indexed. They capitalized on that sent out thousands of crawlers to the web and indexed everything they could possibly find.

4.3.2 Crawler for Twitter Advanced Search

Twitter have an Advanced Search site[5] and we can search the data that older than 7 days before. Once we knew that we only need to crawl the Twitter Advanced Search site then it will be much easier to get the tweets we want. We can assemble the request URL according to the rules and parse the response to get the information we need. The architecture of the system is shown in figure 4.4.

As if we use the Twitter Advanced Search site, the URLs we send can be formed in an ordered way so there is no need to check if the URL has been fetched or not. This will be discussed in next section. In this project, we use Java HTTP Client to send GET request and get the response. We use the Jackson to parse the JSON data and adopt JSOUP for the HTML document.

[5]https://twitter.com/search-advanced

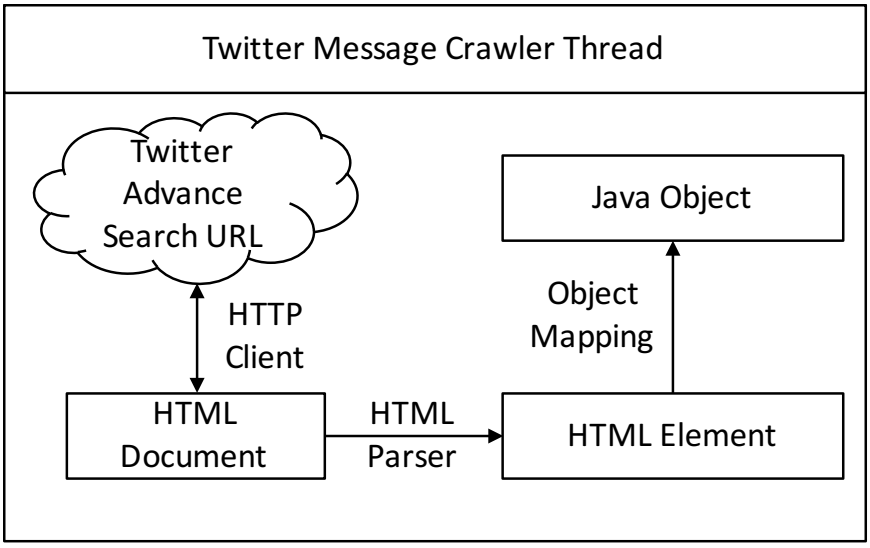


Figure 4.4: Parse HTML document to Java object

- **Jackson** is a suite of data processing tool in Java. Jackson can be used in extracting and generating JSON (JavaScript Object Notation) data. "JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language"[6]. JSON is independent from programming languages but users can easily extract or generate JSON format data no matter they use JAVA, C+, Python or Perl. In JAVA, Jackson is the so called "best JSON parser for Java".

- **JSOUP** is a JAVA HTML parser which allows developers to extract HTML document in the same way like using jQuery. It provides a very convenient API for extracting and manipulating data. JSOUP also can work on HTML 5 and parses HTML in the way that the browses do. JSOUP can parsers HTML from URL, string or files and it is designed to deal with all varieties of HTML can create a sensible parse tree.

The overall sequence diagram of the system is shown as figure 4.5.

[6] https://github.com/FasterXML/jackson

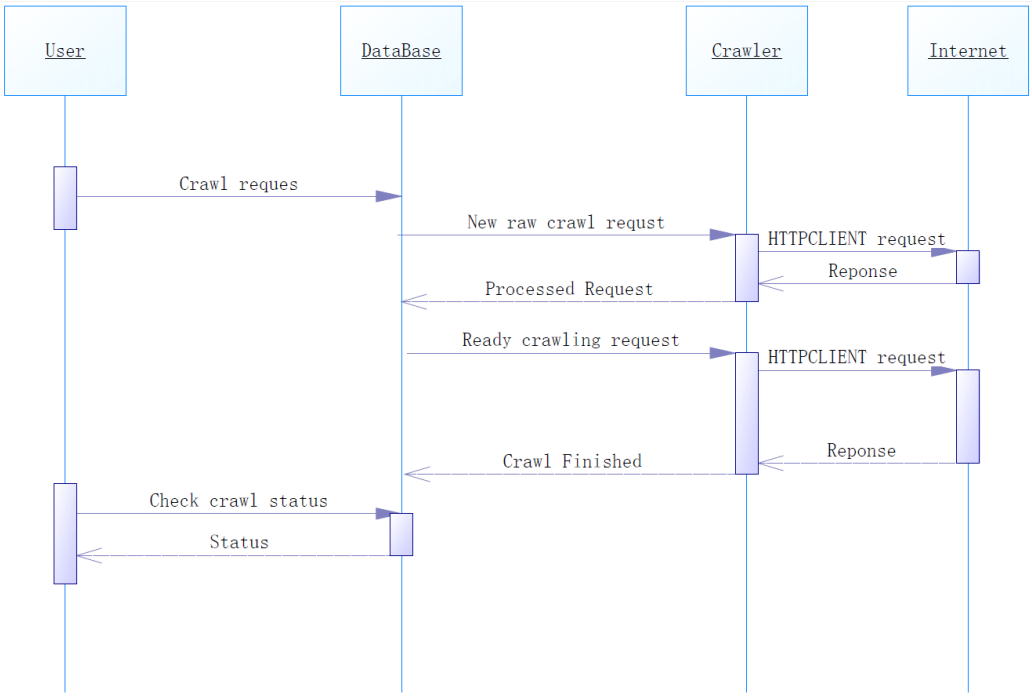


Figure 4.5: Sequence Diagram of Crawl system

When the request is sent to the Twitter Advanced Search for the first time, the URL looks like follow:“ https://twitter.com/search?f=tweets&vertical=default&q=place: place_id since: startDate until: endDate” Where place_id is the id in the Place object by Twitter of a city, startDate is the starting data of search and endDate is the ending day of search. Take the city of Toronto at Canada Day of the year of 2015 as an example. The URL to be sent at the first time is “*https://twitter.com/search?f=tweets&vertical=default &q=place%3A3797791ff9c0e4c6%20since %3A2015-06-30% 20 until%3A2015-07-02*”. This first URL will give us 20 tweets which can be found through the sour code. When a user scrolls down to the end of the page, another 20 tweets will be added to the page if it’s available.

This feature is done by the use of AJAX. Twitter Advanced Search site also uses AJAX to let data loaded asynchronously. It is easy to solve this if we can find out the meaning of the parameters. Through the study by using the ”Inspect” feature pro-

vided by Chrome, we found that a new request URL will be sent when use scrolls to the end. The new URL is: “https://twitter.com/i/search/timeline?f=tweets&vertical=default &q=place %3A3797791ff9c0e4c6 %20since% 3A2015-06-30 %20 until %3A 2015-07-02 &include_available_features=1 &include_entities=1 &max_position= TWEET - minTweetID - maxTweetID- &reset _error _state= false” . For each tweet, it has one unique ID which is called the ‘raw_id_str’.

When a user sends a search request, Twitter Advanced Search site will return the result in a decreasing order of the tweets ID. That is to say, when we search for Toronto of the day at Canada Day at 2015, the first tweet is the one that has the biggest tweet ID which is 616395845127274496 and this ID is also the parameter maxTweetID in the URL. For the minTweetID, it is the id of the last tweet in previous response. That is to say, if for the first 20 tweets, the id of the 20th is xxxxxxx, then in the second URL, the minTweetID parameter is the xxxxxxx. Thus, we know how to construct URLs continuously. By using the HTTP Client class discussed in previous section we can also get the responses and extract the tweet. If we extract no tweet in the html file, then we will stop sending HTTP Clients.

Once we got the html files, the next steps are to analysis the html file and get the tweets. Through the use of the inspect function provided by Chrome, we found that all the tweets are in an ol label with class name stream-items js-navigable-stream. If this parser finds out that the stream-items js-navigable-stream class is empty or this doesn’t exist then it will tell the web page downloader to stop sending HTTP Client request.

<li class="js-stream-item stream-item stream-item expanding-stream-item
" data-item-id="711365977842253825" id="stream-item-tweet-711365977842253825" data-item-type="tweet">
<div class="tweet original-tweet js-original-tweet js-stream-tweet js-actionable-tweet js-profile-popup-actionable
" data-tweet-id="711365977842253825" data-disclosure-type="" data-item-id="711365977842253825" data-permalink-
path="/Senators/status/711365977842253825" data-screen-name="Senators" data-name="Ottawa Senators" data-user-id="43885373" data-expanded-
footer="<div class="js-tweet-details-fixer tweet-details-fixer">

Figure 4.6: Tweet’s id and user information

For each tweet itself, it is in a li label with class ”js-stream-item stream-item stream-

item expanding-stream-item" and attribute "data-item-id="x" id="stream-item-tweet-x" data-item-type="x". For each tweet sent, it must have a user attached with it. Figure 4.6 shows how the information of a tweet was structured in the js-stream-item class. Inside this class we can find that for a tweet, witch must have a unique id called "data-item-id" and the display name of the user is "Senators" and the id of this tweet is "722365677842253825". This is part of the basic information of a tweet. If we want more information, we need to continue analysis the html file.

Figure 4.7: Information of a user and content of a tweet

Figure 4.7 gives us a few more information that we interested. In the following figure, there is a class named "stream-item-header". From the name we can know that this class is about the "hearder" of a tweet. A header will contains almost all the information that we want such as the user name, the date and place a tweet was posted and the caption of a tweet. From the header we can easily know that there is a class named "account-group" that has the user name that we are looking for. The following img class "avatar" has the profile image followed by the full name of a user. Inside this div class, there is also a small class with attributes time of which we can find the time information of a tweet includes the timestamp and the date of when the tweet was posted. To get the text of a tweet, we need to find the p class. We can find information like the text and in what language the text was written in this class.

After getting all the data we need, next step is the same as the collector that using the API.

4.4 Training Model and Predicting

One of our system important functions is to predict the emotion. In this section we will introduce what tools we use and how we training the model.

4.4.1 Weka: Data Mining Software in Java

Weka [7] is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Weka is open source software issued under the GNU General Public License. In CDP, Weka 3.7 was used as our machine learning tool. Section 3.4 already introduces the algorithm we use, so in this section, the main point is using the algorithm in Java.

- **Build Model with Weka**

  In CDP, we have corpus annotation function (Section 4.5.4) that can help us to collect a lot of training data. However, if we want use these data in Weka, we have to convert these data in to an ARFF Attribute-Relation File Format(ARFF) format which Weka required.

  An ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections. The first section is the **Header information**, which is followed the **Data information**.

  The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. The data is a list and Figure 4.8 is a example.

[7] http://www.cs.waikato.ac.nz/ml/weka/

```
@relation training_data

@attribute text string
@attribute emojis string
@attribute emotion {positive,neutral,negative}

@data
'canvassing for usr #yegfed #elxn42','',positive
'\" i think it\'s time for change \" - ana commit to vote : #generationtrudeau #sfu #lpc #elxn42','',positive
'the past and future of the refugee crisis - thomas sowell #elxn42 #polqc','',positive
'rdy to watch usr rock it tnight in the usr debate at usr café #ndp #cdnpoli #elxn42','',positive
'ca not wait to vote for usr and usr on october 19 . #cdnpoli #elxn42 #realchange go usr','',positive
'vote for ndp is vote for another harper . #elxn42 andrew thomson , mulcair\'s financial adviser','',positive
'the end of the road and destined for the scrap heap #harperbus #canada #cdnpoli #elxn42','',positive
'? ? who is who and does what in the #elxn42 conservative war room ? #cpc #cdnpoli #canpoli','',negative
```

Figure 4.8: Example of the ARFF File

Weka provides a good user interface which let us choose what algorithm we want to use to training the model. For example, as the arff file shown in Figure 4.8, we can first use the StringtoWordVector filter[8] converts String attributes into a set of attributes representing word occurrence (depending on the tokenizer) information from the text contained in the strings and then choose a classifier to build the model.

- **Use Model in Java** Weka provides the Java API that lets the developers use the trained model in their program. If we want to use model to predict a sectence, we need to preprocess this sectence as the same way as train the model. In CDP, we use some tools to preprocessing the text and these tools will be introduce in section 4.4.2. After the preprocessing, the new sectence should be reformatting into the arff format and then we can get the classified result.

4.4.2 Other Tools

Weka is for the machine learning algorithm and we need more tools to preprocess the text and let the algorithm work.

- **Apache OpenNLP**

[8] http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html

The Apache OpenNLP[9] library is a machine learning based toolkit for the processing of natural language text. In CDP, we use this tool as the Sentence Detector. OpenNLP supports multiple languages by using different model files. For now we only develop the English version and we download the model from the pre-trained models download page[10].

- **Ark-Tweet-NLP**

  Ark-Tweet-NLP is also a open-source Java Part-of-Speech Tagging tool[11]. It provide a fast and robust Java-based tokenizer and part-of-speech tagger for tweets, its training data of manually labeled POS annotated tweets, a web-based annotation tool, and hierarchical word clusters from unlabeled tweets. The advantage of this tool is it can recognize a lot of emoticons and make it more accurately in social content.

[9]https://opennlp.apache.org/index.html
[10]http://opennlp.sourceforge.net/models-1.5/
[11]http://www.cs.cmu.edu/ ark/TweetNLP/

4.5 Data visualization and user interaction

There are three functions in Digital City Impulse (shown in Figure 4.9). In this section, we will introduce the detail of each part and several algorithms.

Figure 4.9: Applications implemented in the CDP.

4.5.1 Visualization of region: Hexagons

In this project, requirement of displaying the region boundary exists in all functions with regard to maps. Compared with rectangle which is widely used for splitting the map, hexagonal grid can approximate the irregular boundary (e.g., city) more accurate; we choose to draw hexagons on Google Maps to illustrate the region boundary. As the projection from 3D earth to 2D map user by Google Maps will result in distortion, we define the hexagon directly on the plane coordinate of 2D map rather than the latitude/longitude coordinate. Figure 4.10 shows several defined hexagons on the map.

In addition, we further index the hexagons in a hexagonal coordinate system, where each hexagon can be represented as less than three values corresponding to the three principle directions.

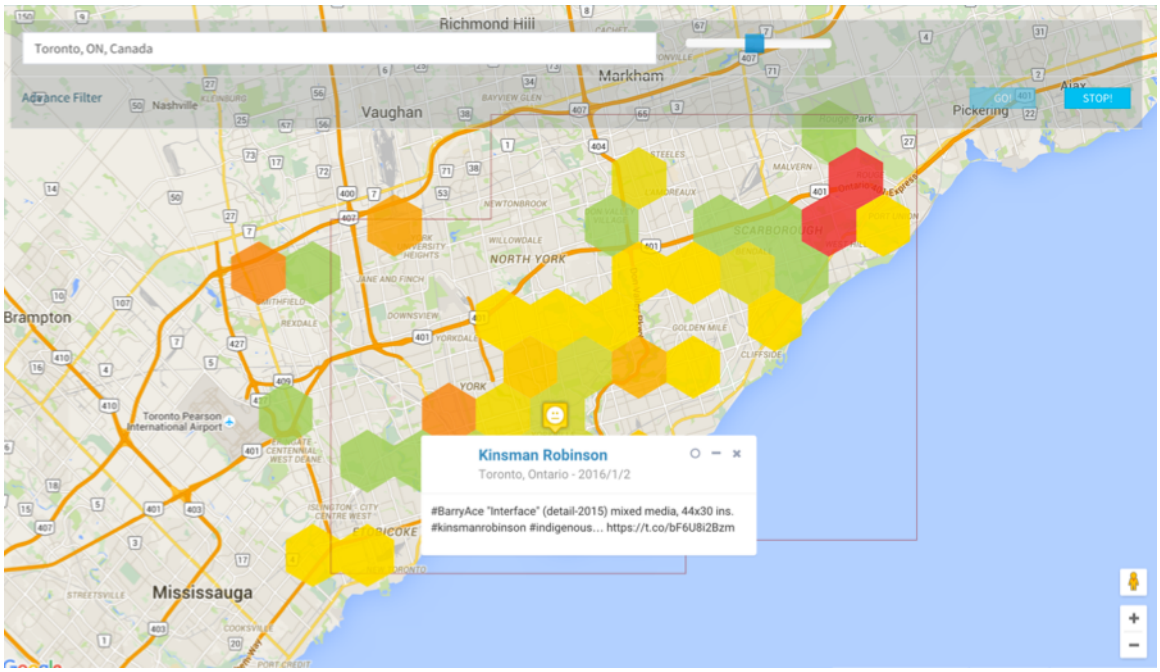


Figure 4.10: Data Visualization in Hex

The conversion between plane coordinate of 2D map and the hexagon coordinate can be easily computed. Thus, we can quickly locate the hexagon of the coming geo-tagged message using its plane coordinate, which is computed by the GoogleMap API.

The hexagonal coordinate system used here is called cube coordinates, which has there are three primary axes. The naming of cube coordinates is inspired by a cube grid sliced out a diagonal plane at $\mathbf{x} + \mathbf{y} + \mathbf{z} = \mathbf{0}$. Each direction on the cube grid corresponds to a line on the hex grid. And each direction on the hex grid is a combination of two directions on the cube grid. Therefore, the cube coordinates are a reasonable choice for a hex grid coordinate system. The algorithms must preserve the constraint of $\mathbf{x} + \mathbf{y} + \mathbf{z} = \mathbf{0}$ [47]. The corresponding algorithms with regard to the hexagonal coordinate system is inspired by [47].Based on these algorithms, three classes for the visualization of region boundary were defined: **Hex**, **HexOnMap** and **HexBoundary**. The code of defining the three classes shown in figure 4.11.

```
function Hex(q, r, s) {
    this.q = q;
    this.r = r;
    this.s = s;
};
```

```
function HexOnMap(map) {
    this.map = map;
    this.TILE_SIZE = 256;
    this.l_earth = 40075000;
    this.r_earth = this.l_earth / (2 * Math.PI);
    this.defaultZoomR = 10000;
    this.defaultZoomLevel = 9;
    this.minZoomLevel = 9;
    console.log("Create HexOnMap Object.");
};
```

```
function HexBoundary(hom, rec) {
    if (hom instanceof HexOnMap) {
        this.hom = hom;
        this.rec = rec;
        this.defaultColor = "#dbdbdb";
        this.defaultOpacity = 0.7;
        this.currentColor = null;
        this.currentOpacity = null;
        this.fadeOutColor = "#b2b1ad";
        this.hexBoundaryPolygon = new Array();
        this.colorAnimationInterval = null;
        console.log("Create HexBoundary Object.");
    } else {
        console.error("Error:Fail in HexBoundary Initialisation.");
    }
};
```

Figure 4.11: Code of Defining three classes with regard to the Hexagonal Grid

- **Hex:** The class of the hexagonal coordinate of a hex. The functions on this class are some basic algorithms of the hexagonal coordinate originated from the persuedo code in [47]

- **HexOnMap:** Binds a hexagonal coordinate to a map variable. This class is used to enable the use of multiple maps (map variables) in one webpage. The functions on this class are the conversion between plane coordinate of 2D map and the hexagon coordinate.

- **HexBoundary:** Binds a HexOnMap instance to a region boundary rectangle information received from the server.

4.5.2 Message on Map

The first function is the real-time visualization of messages. User can obtain what kind of data they want to see by setting the data filter options such as regions, keywords, languages, data platform etc. Once the data filter options were set, the message will be displayed in real time with emotion label. If no data filter option is set, any message that are sent within the thirty cities on our listened list in the back-end will be labeled with emotion and displayed on the map.

This function in the page **msgonmap.htmlmsgonmap.html**, as shown in figure 4.12. After loading the page, **GET collector/getlistenplacelist** is called to get the public region list, which carries the name of all regions that the system has predefined.

When the user enters a region name on the search box, a list of autocomplete place name the GoogleMap AutoComplete API provided is displayed under the search box to let the user select the desired one. Once the place name selected, a code block will first check if the place name is in the public region list. If yes, **GET collector/getplaceinfo** is called to get the region boundary information to display the region boundary on the map in the form of hexagons. If no, a draggable and resizable square is stick to the place marker according to the place location provided by GoogleMap AutoComplete API on the map.

So far we have listened top thirty cities in Canada ranked by population. If users select these cities, the city boundary that are already be defined in the system will be shown automatically on the map (Figure 4.13). Otherwise users will be informed to define the region boundary themselves.(Figure 4.14) Once users submit their defined region boundary, the system will inform the data collection modules to create new listen task for this user-defined region.

When the **Start** button is clicked, **GET message/getlatest** is called every three seconds to get a latest message posts that satisfies the message filter options from the

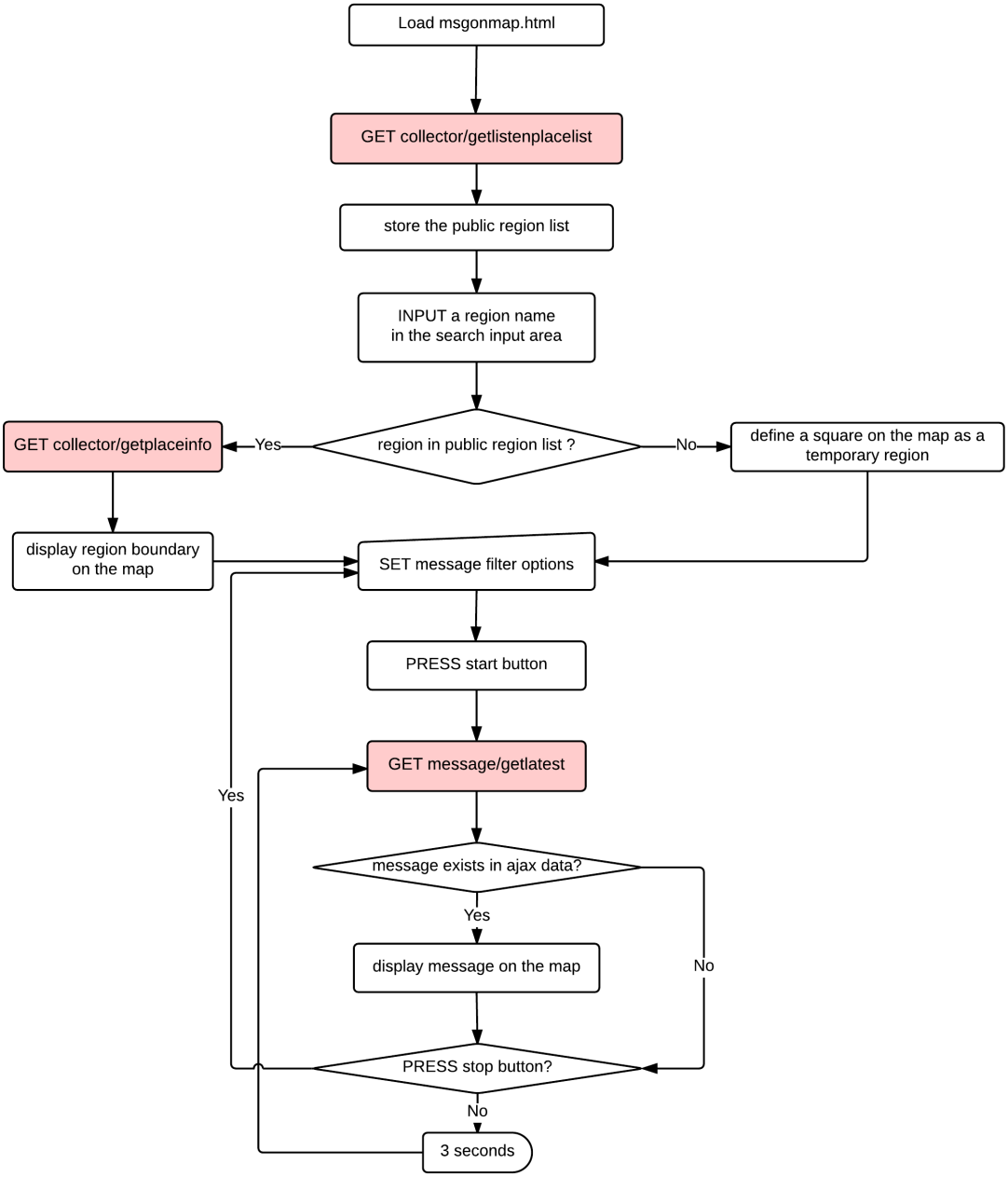


Figure 4.12: The flow chart of calling Ajax Request via the RESTful API used in msgonmap.html

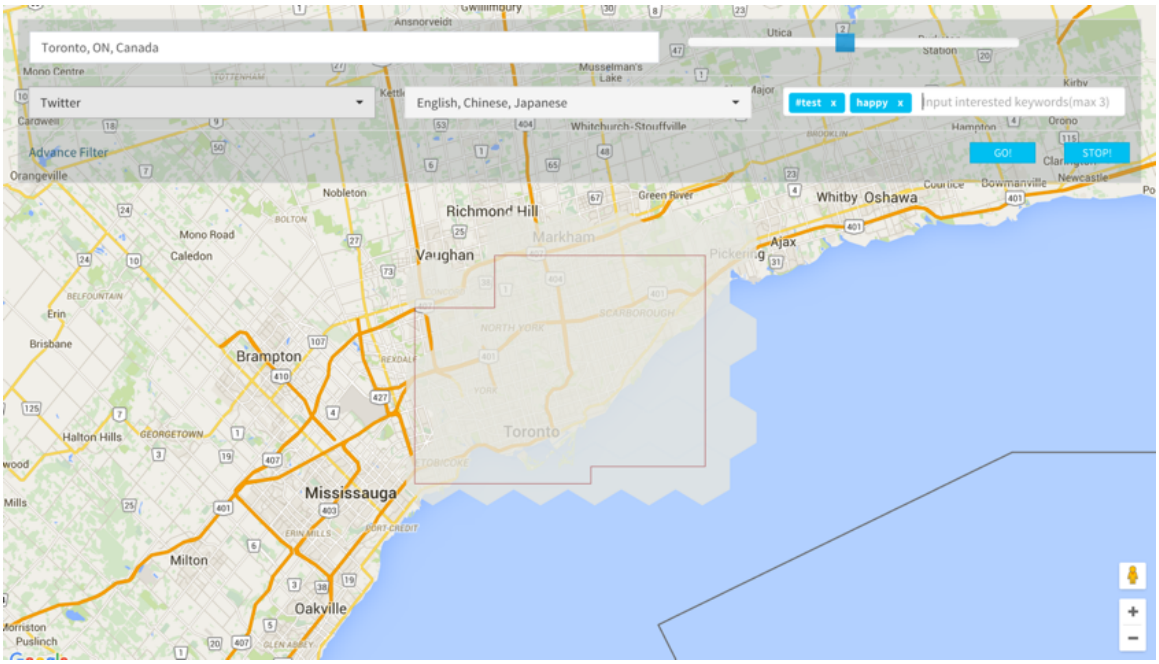


Figure 4.13: The area in the database

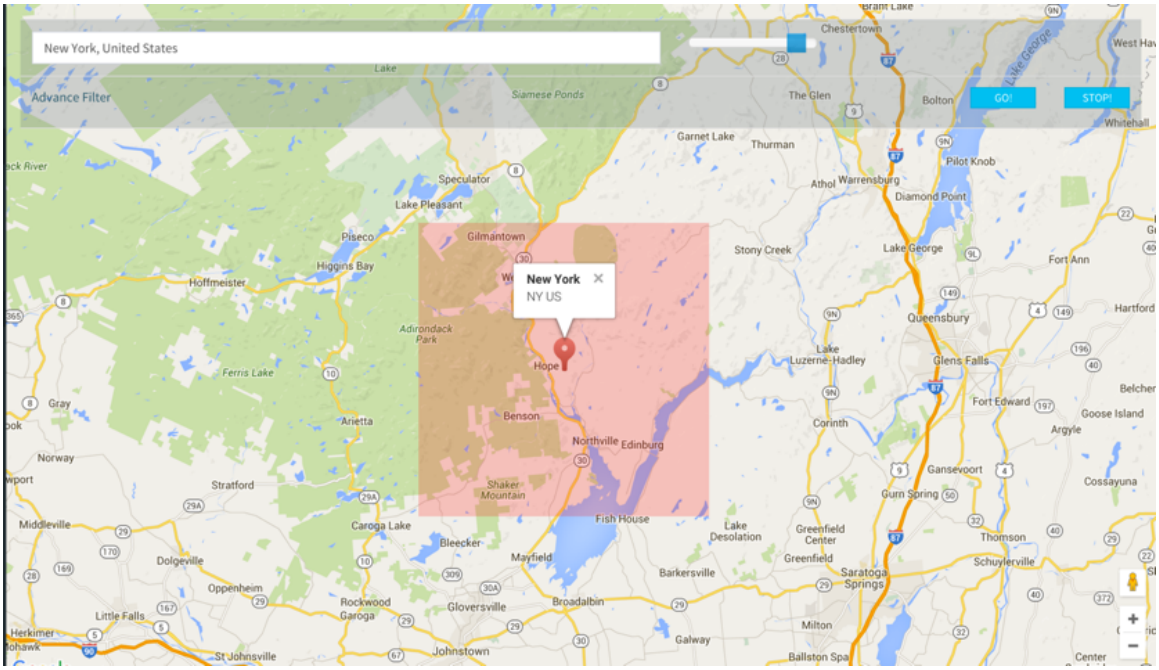


Figure 4.14: The area not in the database

backend. If the information of the message is included in the returned data, the message will be displayed on the map with its sentiment label analyzed by the algorithm of the system. During this time the hexagon that the new message dropped in will also change its color correspondingly as it is displayed with emotion label. Figure 4.10 shows the emotion of Toronto and Figure 4.15 shows the relationship of color and emotion. When the **Stop** button is clicked, the operation of calling **GET message/getlates** is stopped.

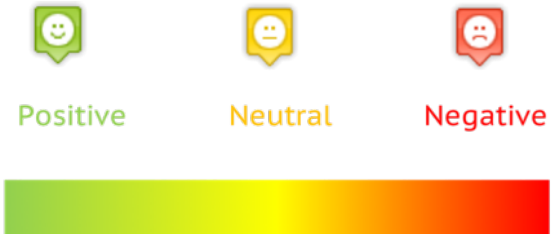


Figure 4.15: Emotion and Color

User can search a region, set message platform, language and keywords, and then press GO button to start browsing the real-time message posts. Sentiment color and involved messages are shown in the unit of hexagons. Figure 4.16 shows an example of using this functionality to filter real-time message posts including **#SOS**, which may be posted by citizens in danger.

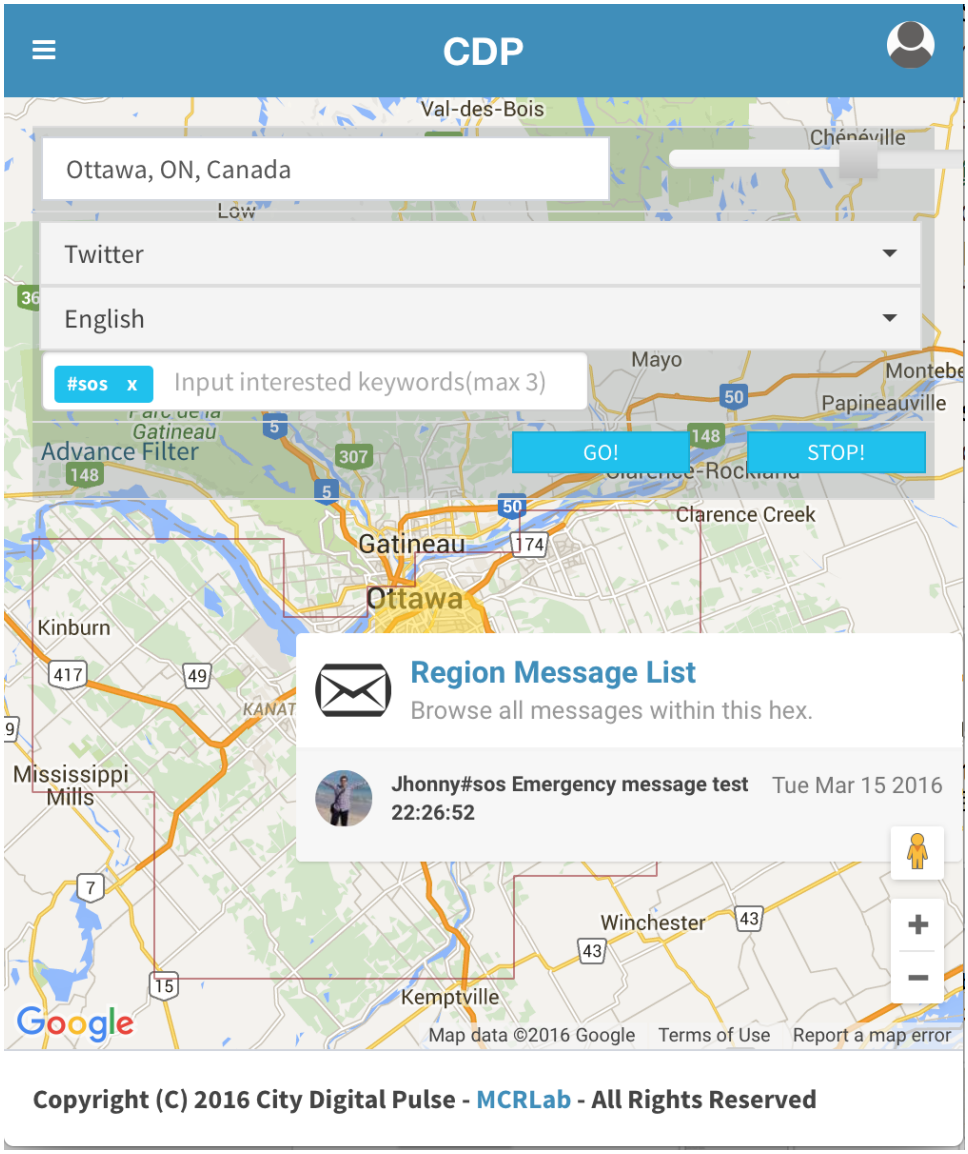


Figure 4.16: Visualization of original messages

4.5.3 Data Analysis

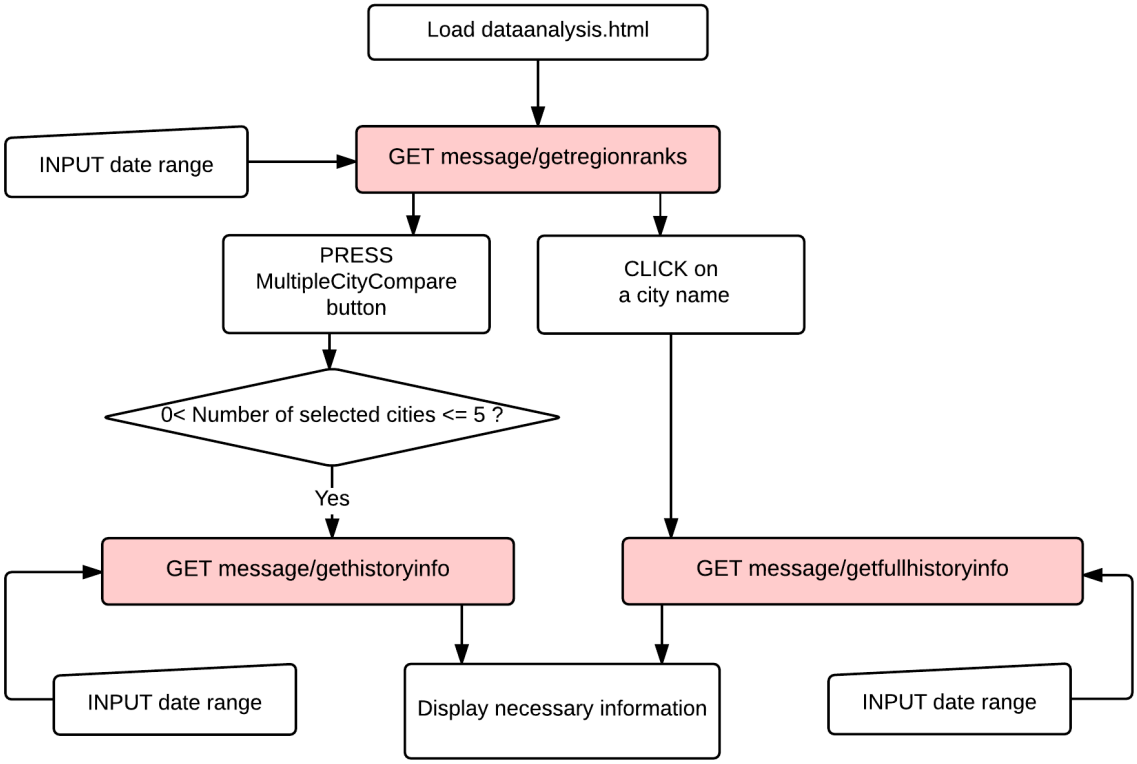


Figure 4.17: The flow chart of calling Ajax Request via the RESTful API used in data-analysis.html

As discussed in Section 3.4, the sentiment embedded in each message can be identified. However, many applications need the aggregated knowledge such as the overall attitude towards a topic, rather than the emotion of a particular message. In this section, we introduce our designs to show how the aggregated useful information is defined under different conditions of geo-location, timeline and topic.

This function in the page **dataanalysis.html**, as show in figure 4.17. After loading the page, **GET message/getregionranks** is first called to get the region rank of all public regions on the default date (now is set to the day before the current date). And

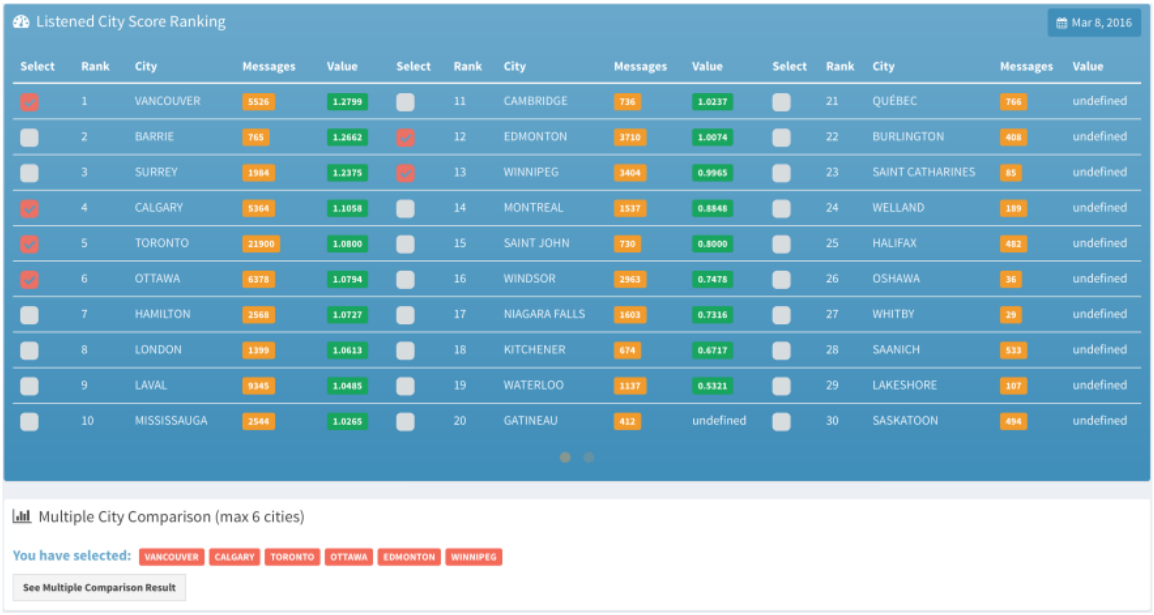
when the user selects another date and press the **Browse City Rank** button, **GET message/getregionranks** is called again to get the region rank of all public regions on the selected date. The rank is displayed as tables on the pages.

Figure 4.18 shows the overall sentiment of the 30 cities selected in Canada during a given period. In Figure 4.18(a), the cities are ranked based on their levels of positive sentiment. The number of messages from each city is also listed. The detailed daily information is further obtained by selecting the wanted cities in figure 4.18(a).

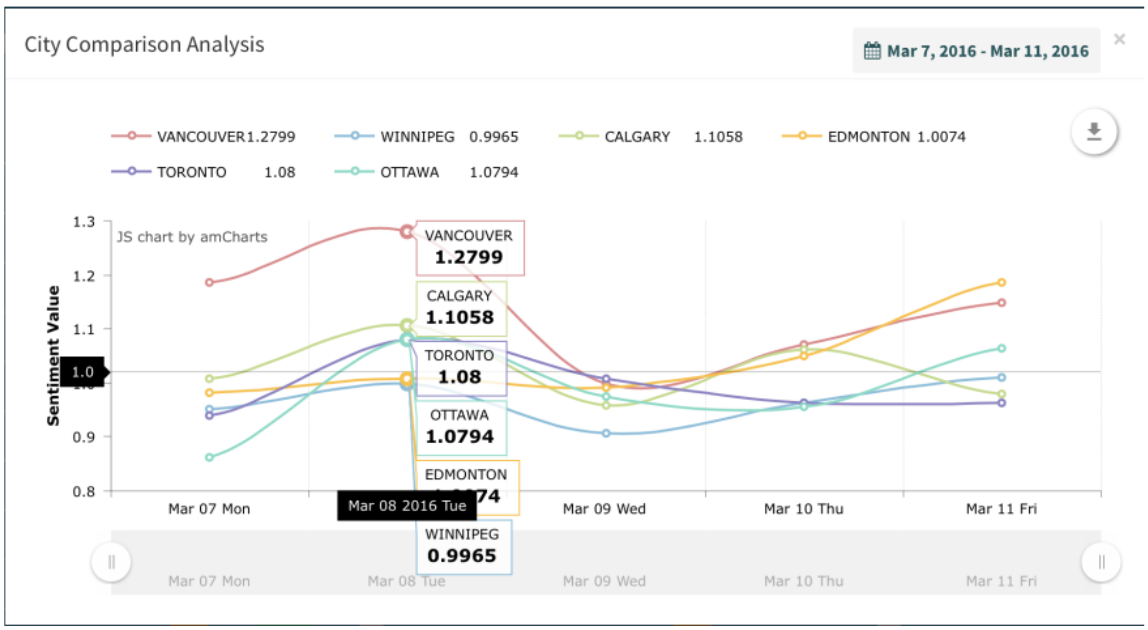
Another function is to allow users to browse the emotion changes in a certain region during a certain time period, and the data can also be filtered by keywords, languages and data platforms etc. In addition to the classic charts like bar chart and line graphs, the dynamic color changing animation within the hexagonal grid on the map will be used as well to illustrate the emotion change of the selected region. The comparative analysis between multiple regions is also supported in this function. If users select a region that are not listened in the system, request will be sent to the data collection module to get history data, and once data collection is finished, users will receive notification emails to browse the emotion changes result.

When the user selects several regions in the region rank tables and press the **See Multiple Comparison Result** button, **GET message/getfullhistoryinfo** is called to get the sentiment analysis result of selected regions. Returned data includes only the sentiment analysis result in days, which will be formatted and displayed as a chart. The sentiment evolves of multiple cities during the selected time period is showed in Figure 4.18(b). We can see that citizens in Vancouver are happier than other cities. There is a obvious peak for all the cities on March 08, which is the International Womens Day. With this dashboard, users can easily grasp the level of Web users' online activity and emotional status in different cities.

When the user clicks on a certain region name, **GET message/getfullhistoryinfo** is called to get all the information and sentiment analysis result of the selected city within a



(a) Rank of cities based their overall sentiment scores.



(b) Sentiment comparison of different cities during a given period.

Figure 4.18: Comparison of sentiment in different cities during a selected time period.

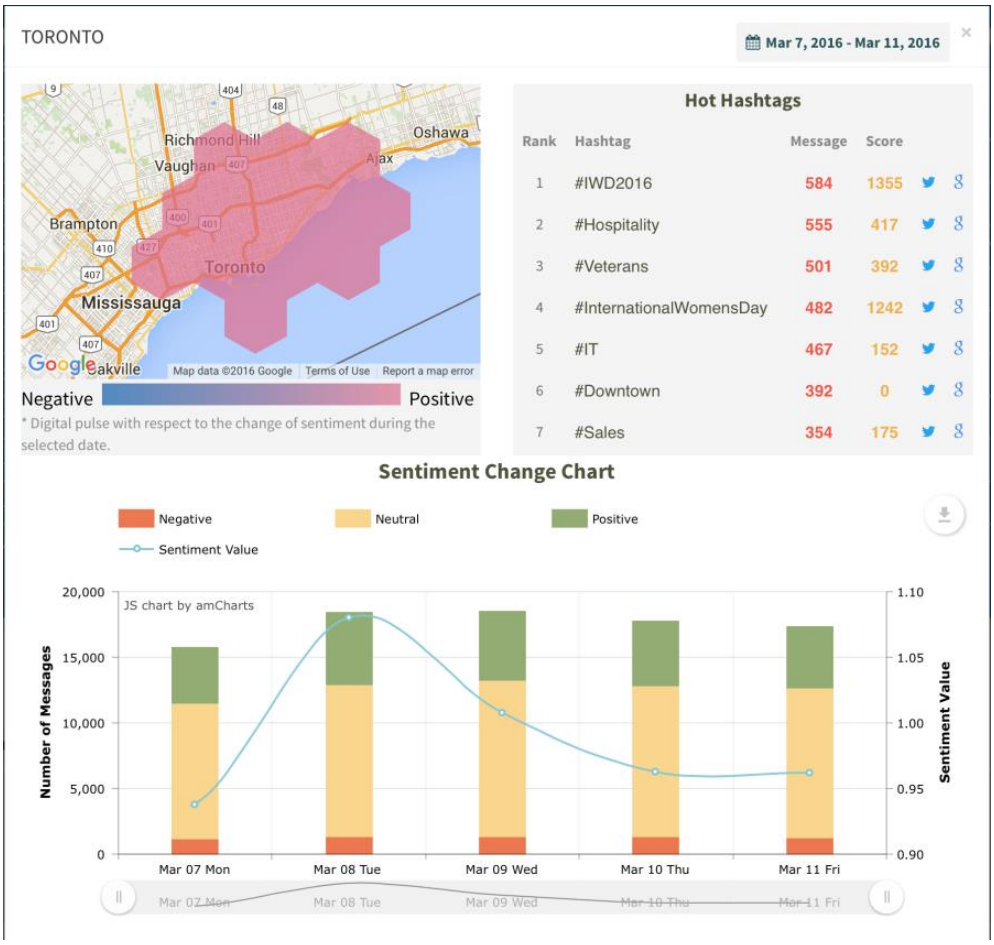
default date range. Returned data includes the region boundary information of the region, the hot hashtag list of the region and the sentiment analysis result in days. And when the user selects another date range, **GET message/getfullhistoryinfo** will be called again. We can get the city's detailed information as showed in Figure 4.19.

The top left image in Figure 4.19(a) adopts a vivid change of colors to indicate the sentiment evolving of the selected city. As showed in Figure 4.19(b), the four images correspond to the sentiments detected in four days. In the bottom image of Figure 4.19(a), the curve indicates the overall sentiment changes during the time period. Each bar represents the number of messages, as well as the percentage of positive, neutral and negative sentiments.

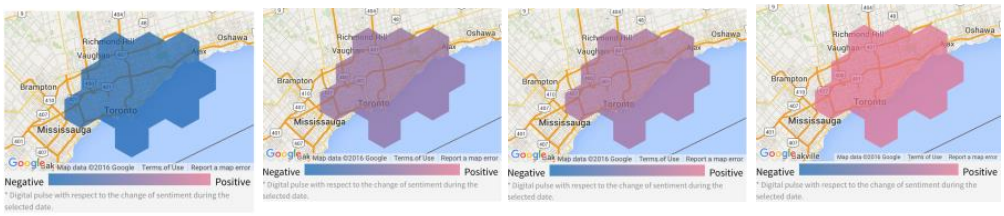
In addition, we also list the hot hashtags in the top right part of Figure 4.19(a). This can be used for users to understand the reason of emotion change. As some users may not know the story behind the hashtags. The search results on Google and Twitter can be obtained by clicking the icons after the hot hashtags. For example, the search results of "IWD2016" are showed in Figure 4.19(c) and Figure 4.19(d) respectively. By integrating the information from different platforms, the digital sentiment pulse and reasons behind the results of data analysis can be captured in a convenient way.

We can further narrow down the analysis by clicking the hashtag. As showed in Figure 4.20, the top left image will be updated to show the sentiment changes of messages containing the selected hashtag. Similarly, the curve and chart in the bottom image are also updated accordingly. In addition, we list the images attached in the tweets of the selected topic in the top right part of Figure 4.20.

In addition to the visualization of sentiment according to time, location and topic, we also provide other conditions, such as language and data source. Therefor, the identified information can be showed to users in various dimensions and granularities. Different user requirements can be satisfied using our system.

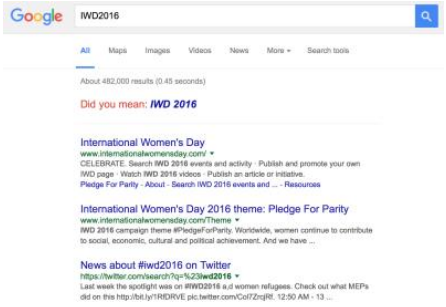


(a) Sentiment distribution on the map, sentiment scores and hot hashtags for a given city.



(b) Dynamic visualization of detected sentiment during the given period

(c) Search result on Twitter for hot hashtag "IWD2016"



(d) Search result on Google for hot hashtag "IWD2016"

Figure 4.19: The detailed information of each city.

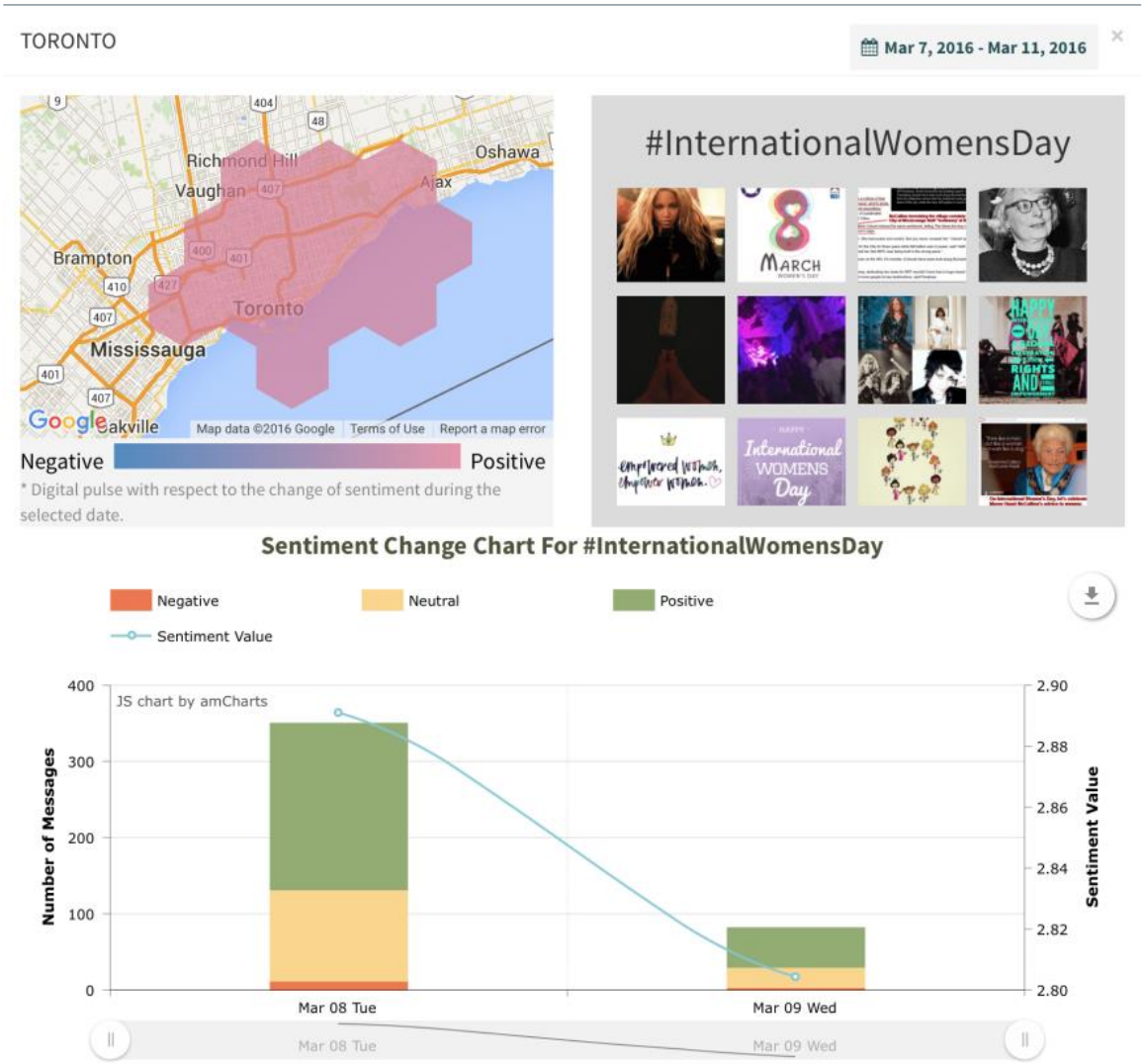


Figure 4.20: The information corresponding to a selected hot hashtag.

We also allow user add their regions using **My Regions** function in **myregions.html** (Figure 4.22), as shown in figure 4.21, after loading the page, **GET collector/getlistenplacelist** is called to get the public region list, which carries the name of all regions that the system has predefined. **GET user/getuserregs** is called to get the user's region list, which carries the name of all the private regions the user has defined by himself/herself under **myregions.html** and all the public regions the user has added to his/her list. When the user enters a region name on the search box, same as **msgonmap.html**, a list of autocompleted place name the GoogleMap AutoComplete API provided is displayed. Once a place name is selected, a code block will first check if the place name is in the public region list. If yes, **GET collector/getplaceinfo** is called to get the region boundary information to display the region boundary on the map in the form of hexagons. If no, another code block will check if the place name is in the user's region list. If yes, **GET collector/getplaceinfo** is called to get the region boundary information to display the region boundary on the map. If no, a marker showing the position of the region will be placed on the map.

When user press the **add into my region** button while browsing the region boundary of a certain defined region on the map, a code block will first check if this region is already in users region list. If not, **POST user/adduserreg** is called to send region information to the server. If the server responses with **OK**, the region is successfully added into user's region list. When user draws a new region on the map, type in a region name and press the **add a new region into my regions** button, **POST user/adduserreg** is called to send the name and the boundary information of the newly defined region to the server. If the server responses with **OK**, the region is successfully added into user's region list. The collector will receive the new task request and start collecting. Figure 4.23 shows the rank of one user's region list.

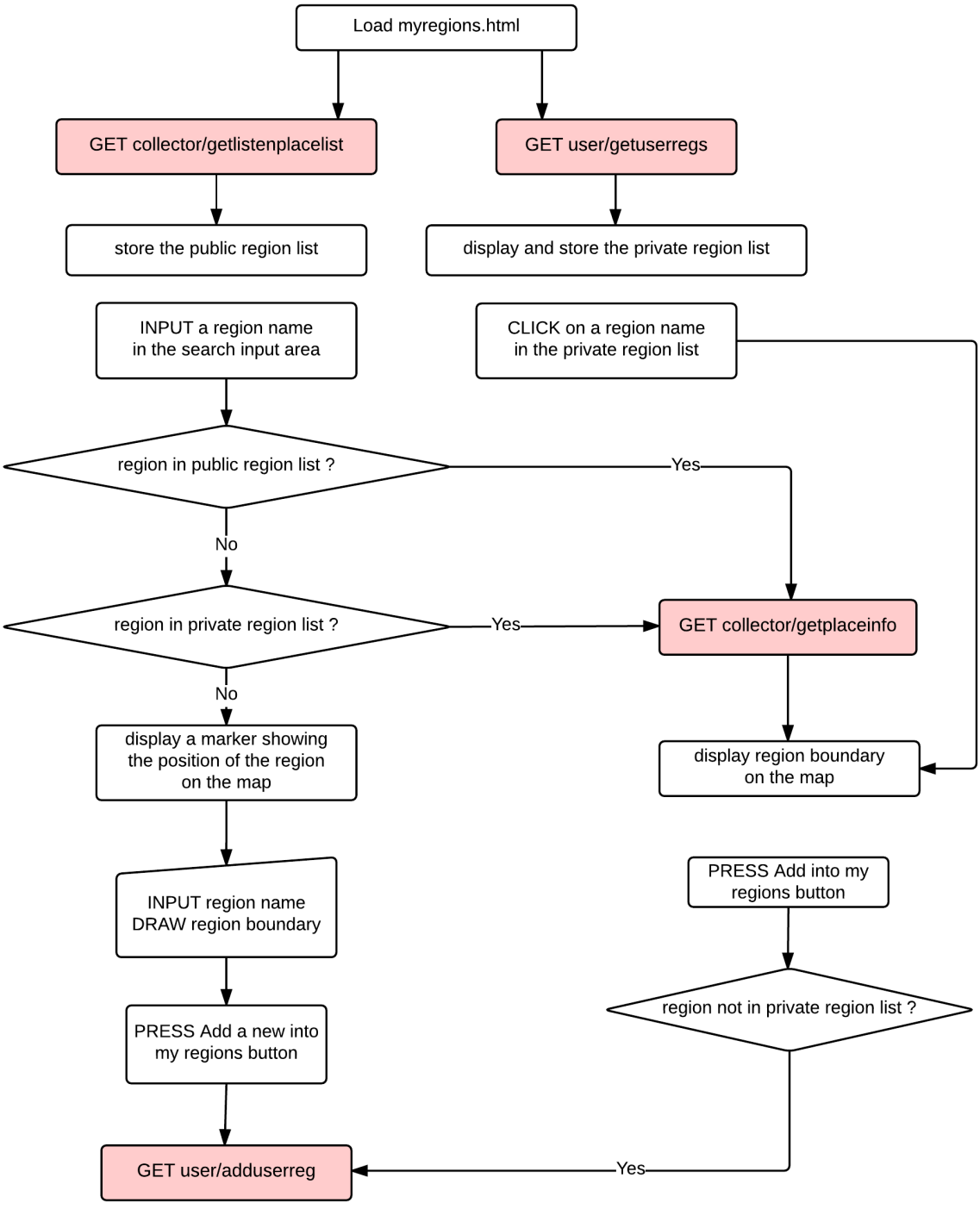


Figure 4.21: The flow chart of calling Ajax Request via the RESTful API used in myregions.html

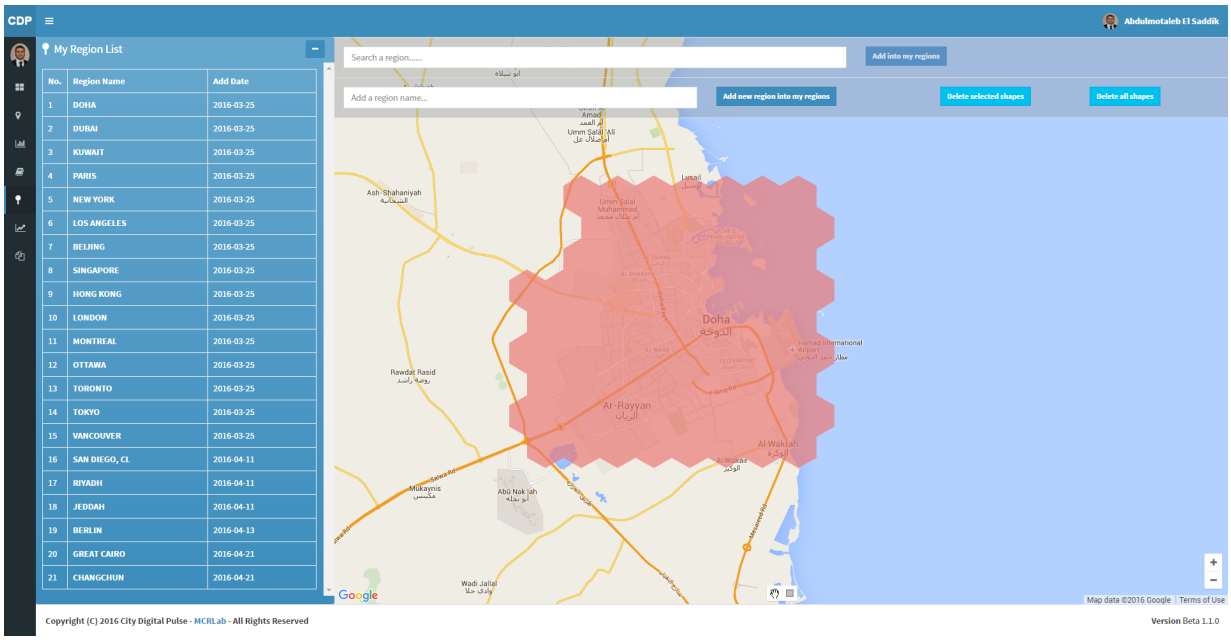


Figure 4.22: My Regions Page

Figure 4.23: My Regions Rank Page

4.5.4 Corpus Annotation

In CDP, the emotions of messages are decided through machine learning algorithms, thus a large amount of training data are necessary to help training a better learning model. Therefore, the third function is a corpus annotation module. Users can select the languages, data platform and the part (text or images) of the messages they want to help do the annotation. Figure 4.24 shows the APIs that this function used.

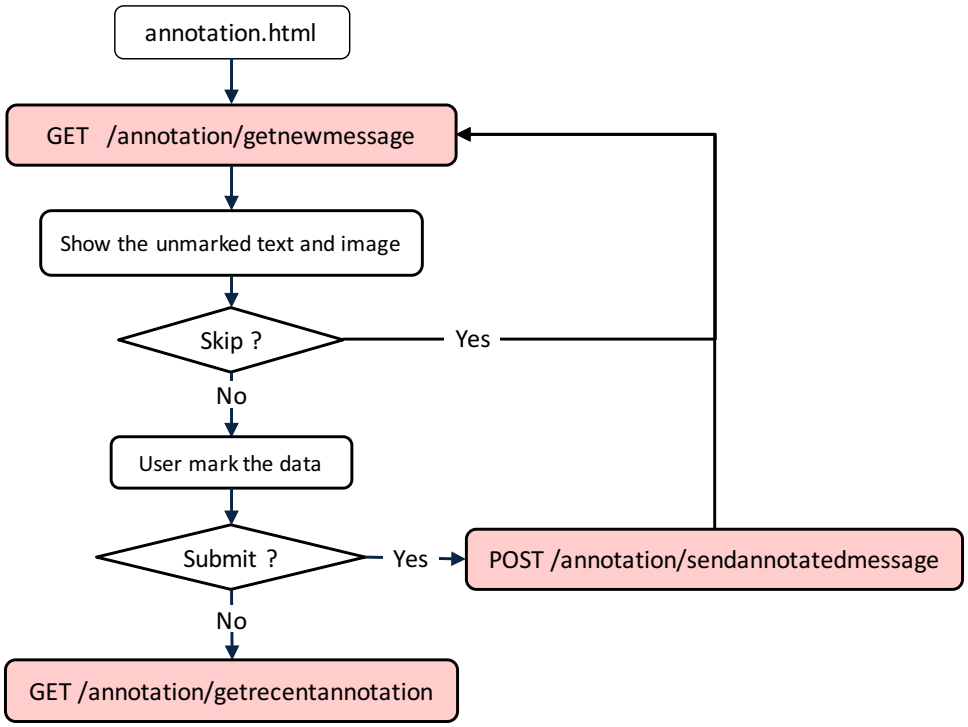


Figure 4.24: Corpus Annotation Page

Every message is set to be annotated three times by different users to dilute the influence of users' subjective emotions and maintain training data with higher quality. Users can browse their annotation history and download the messages they have annotated. Manual corpus annotation is a time-consuming effort and the annotation result is thus precious. Therefore, we will store these data in an individual database and open to other researchers. Figure 4.25 and 4.26 are example of the annotation part.

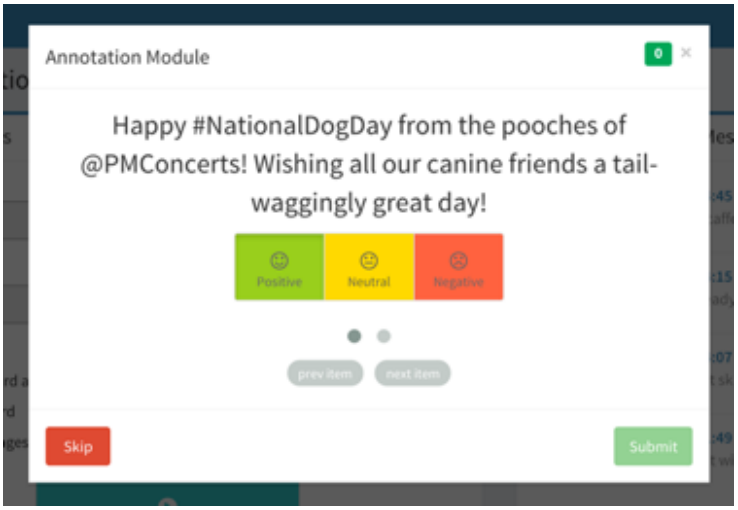


Figure 4.25: Mark the text

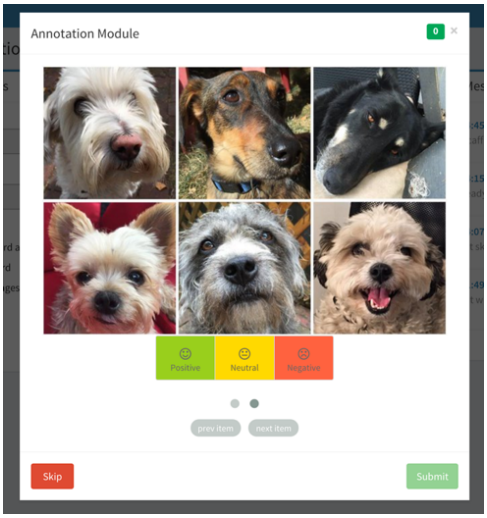


Figure 4.26: Mark the image

4.5.5 Front-end Client

We developed a web application for users to have easy access to our system. The frontend and the backend are separated, and data are transferred through the RESTful API, which allows different devices share the same API calls and thus enhances the scalability of our system. The visualization can be performed under certain conditions defined by users such as the geographical location, language and keyword etc. The data is geo-labeled, and therefore displaying data through space dimension (e.g., on a map) is a natural choice. We use different colors to illustrate different emotions of the messages and regions. As showed in Figure 3, Red is negative, yellow is neutral and green is positive. Besides, data visualization through other dimension, such as time line, is also provided to gain better illustration of the evolving of human sentiment. Users can also label the emotion of the data displayed to help improve the data analysis in our system. The development of the web application follows the principle of responsive web design to provide an optimal view and interaction experience through various devices.

The front-end follows the principle of responsive web design. It is an approach to design

web pages that provides an optimal viewing and interaction experience across a wide range of devices (phones, tablets, desktops) with a single code base. The front-end is developed based on the HTML5 and a popular responsive framework, Bootstrap, and modified by some CSS and javascript code. The components and elements will show, hide, shrink, resize, move as the size of the viewport changes to make webpages look delight and proper and allow user to have access to an optimal user interface.

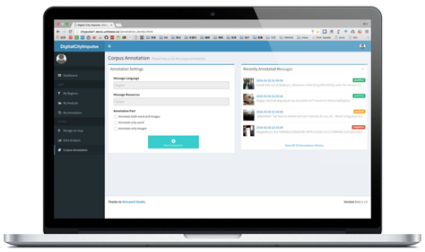


Figure 4.27: PC

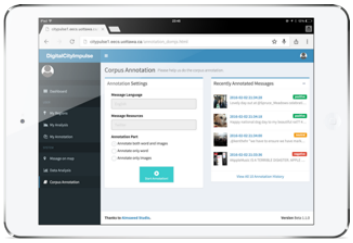


Figure 4.28: Pad

Figure 4.29: Phone

4.6 Back-end Web Service

In this section, we will introduce the back-end of our system and some important points.

4.6.1 Backend Software Platform

In section **??**, we introduce the RESTful API and we use Jersey as our RESTful service provider. Jersey RESTful Web Services framework is an open source framework for developing RESTful Web Services in Java. It provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation[12]. Jersey is very easy to use, Figure 4.30 is a code example.

[12]https://jersey.java.net/

```
@GET
@Path("/getlatest")
@Produces(MediaType.APPLICATION_JSON + ";charset=utf-8")
public ResMsg getLates(
                @QueryParam("token") @DefaultValue("") String token,
                @QueryParam("message_from") @DefaultValue("") String message_from,
                @QueryParam("keyword") @DefaultValue("") String keyword,
                @QueryParam("location_areas") @DefaultValue("") String location_area_json,
                @QueryParam("lang") @DefaultValue("") String lang,
                @QueryParam("limit") @DefaultValue("1") int limit,
                @QueryParam("skip_num_ids") @DefaultValue("") String skip_num_ids) {
```

Figure 4.30: The information corresponding to a selected hot hashtag.

The **@GET** annotation in figure 4.30 means this API is the GET function introduced in section **??**. **@Path** annotation shows the path of the API, in CDP this API's full path is http://citypulse1.site.uottawa.ca/api/message/getlatest. If you just put this URL into the bowser, it will return a JSON message like this:

{"code":400,"type":"BAD_REQUEST","message":"Token is wrong.","obj":null}

@Produces(MediaType .APPLICATION_JSON + ";charset=utf-8") make sure the return data type is in JSON format and we already shows the benefit of JSON format in section 3.3.1.

In JAX-RS, we can use **@QueryParam** annotation to inject URL query parameter into Java method. For example, we have to verify every API request make sure that's from the user of our system. For demonstration, we use the fix token for this API. We use **?token=ArashiArashiFordream** to send the token from bowser to back-end server and the full request URL is http://citypulse1.site.uottawa.ca/api/message/getlatest?token=ArashiArashiFordream and we can get the latest message in our system.

4.6.2 Deploy to Cloud

In the future, CDP platform will have more sensors and data sources, and the size of data will grows huge. Using the cloud server to host our system is a good choice. There is a crucial feature of cloud server which is Elastic Expansion. Elastic Expansion refers to dynamic increasing or decreasing of cloud servers to increase the capability and performance or to decrease the cost depends on how busy the operations are. For now, Amazon Web Service (AWS), Google App Engine (GAE) and aliYun all support this Elastic Expansion requirement.

In addition to the system deployed on our local server, we further deploy CDP on cloud server which can dynamically increase or decrease resources based on the system status. In our work, we select the AWS[13], which provides many services on various aspects of cloud-computing. For our system, the front-end employs the Amazon CloudFront service to distribute the content to end users with better performance. The back-end of our Web application is hosted on AWS Elastic Beanstalk. For our main SQL database, we adopt the Amazon Relational Database Service, which is used for deploying the hard disk database. In our system, we also design a cache database (memory database), which can be deployed on the Amazon ElastiCache service. As the commercial cloud service is expensive, our current system on cloud only keeps the timely statistical information explored from the collected data. The original data is kept on the local server. All the functions discussed in Section 4.5 are achieved on both local server[14] and Amazon cloud[15].

The web service is running on Java Virtual Machine (JVM) and in AWS we can deploy it very fast.

For storage, we use the Amazon Relational Database Service and Amazon ElastiCache. In section 3.3 we mentioned we use the hard drive database as the main database and the

[13] https://aws.amazon.com/

[14] http://citypulse1.site.uottawa.ca

[15] http://citydigitalpulse.us-west-2.elasticbeanstalk.com

in memory database as the cache. For the main database we use MySQL and we use Redis as the cache and both these two database can easily move to the Amazon Cloud.

- **MySQL** - an open-source relational database management system (RDBMS). MySQL is a popular choice of database for use in web applications.
- **Redis** a popular open-source in-memory key-value store that supports data structures such as sorted sets and lists.

Since the SPA structure is used, the front-end can be published into the CDN and increase the loading speed. CDN is a globally distributed network of proxy servers deployed in multiple data centers. The goal of a CDN is to serve content to end-users with high availability and high performance. For example, in table 4.1 we have a CDN node in China and Canada, and we shows the different loading time between them when I access the webpages from Canada. The structure on the cloud will look like this in figure 4.31.

Page Name	Number of Requires	Loading Time(ca)	Loading Time(cn)
login.html	17	0.34s	3.17s
dataanalysis.html	74	1.23s	11.91s

Table 4.1: Loading speed test between different CDN nodes

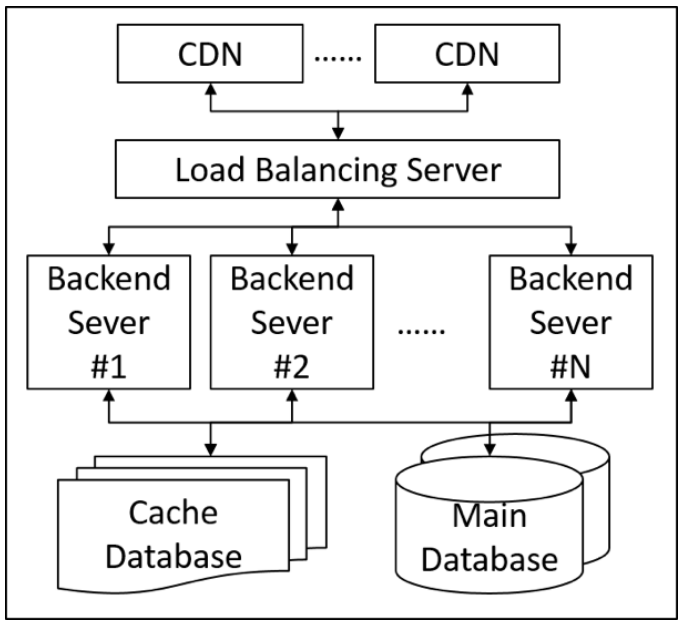


Figure 4.31: The structure in cloud

Chapter 5

Conclusion and future work

In this chapter, we summarize the major contributions and achievements of this thesis. We will also list some interesting future directions.

5.1 Summary of Contributions

In this thesis, we have proposed our City Digital Pulse (CDP) platform. As an end-to-end system for various applications in the context of smart city, the designs follow several principles which are essential for addressing the challenges raised by the large-scale heterogeneous data generated by different sensors. Both the detailed implementation and adopted solutions, ranging from data collection to data analysis, are elaborated. We have also instantiated the platform using social data as an example. The optimizations on the system designs are introduced. In order to demonstrate the usability of our proposed system, we have presented the Web application by considering the citizens' sentiments embedded in their posted social messages. The various facets explored from the large pool of tweets are vividly visualized in several dimensions (e.g., time and geo-location). In addition, we also integrate the knowledge from Web (e.g., Google) into our system. Thus

the users can easily understand the identified information and the underlying reasons. Our system has been deployed on both local server and Amazon cloud, that can be accessed by public.

5.2 Future work

Currently, we only consider the soft sensors: Twitter and Instagram. Future extension includes the integration of hard sensors such as the smart object in the smart city. More innovative applications can be developed on our platform. The possible issues raised are the fusion methods on the various contextually related data, and the system load balance as the query and data transition will be conducted among different types of databases. Another important issue to study the approaches on how to is utilize cloud-computing techniques to the data analysis, which is the most computationally expensive part in the system.

Besides the sentiment analysis discussed in this thesis, there are still several interesting and important issues that can be further investigated with the help of our contributed dataset. In addition, our dataset can be enriched to cover more fine-grained labels. We list several future works here.

- We have showed that there are many inconsistent labels between the text view and image view in the collected tweets. This suggests that future research should pay particular attention on the differentiation of emotional context with other contexts between two views, so that we can appropriately leverage the information from two views.
- The joint feature exploring the correlations between two views has been demonstrated to be helpful. However, the advantage seems to be not so obvious comparing to the traditional fusion strategies. This raises the issue that how to keep the specific

property of each view while modeling their contexts. It is important to purse an intermediate status, so that the discriminative information of single view will be kept as much as possible, meanwhile, the extracted contexts are not harmful.

- Our dataset is constructed by setting a global sentiment for each tweet without considering the entity-level sentiment. In addition, mixed sentiments which may appear in certain tweets are ignored in current work. Since the annotation on entity-level is so expensive, an appropriate interactive annotation tool is needed. For example, we can pre-filter the non-informative messages to reduce the number of messages.

Finally, in the future, the structure of the system can be changed into more light-weight framework. More companies are adopting Node.js for their products. Sharing code between the client and server is becoming a more common and natural choice, and in my opinion is the future of web development. This trend is enhanced by sharing templates through libraries like React. Since we already use the SPA framework we can push it futher to the Micro-Service Architecture and that will improve the system.

References

[1] Nikos Bikakis and Timos Sellis. Exploration and visualization in the web of big linked data: A survey of the state of the art. *arXiv preprint arXiv:1601.08059*, 2016.

[2] Damian Borth, Rongrong Ji, Tao Chen, Thomas Breuel, and Shih-Fu Chang. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In *ACM MM*, 2013.

[3] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[4] Maria Claudia Buzzi, Marina Buzzi, Daniele Franchi, Davide Gazz, Giorgio Iervasi, Andrea Marchetti, Alessandro Pingitore, and Maurizio Tesconi. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[5] Maria Claudia Buzzi, Marina Buzzi, Daniele Franchi, Davide Gazz, Giorgio Iervasi, Andrea Marchetti, Alessandro Pingitore, and Maurizio Tesconi. Facebook: a new tool for collecting health data? *Multimedia Tools and Applications*, pages 1–24, 2016.

[6] Luigi Di Caro and Matteo Grella. Sentiment analysis via dependency parsing. *Computer Standards and Interfaces*, 35(5):442–453, 2013.

[7] Miguel Castro1, Antonio J. Jara1, and Antonio F. G. Skarmeta. Smart lighting solutions for smart cities. In *International Conference on Advanced Information Networking and Applications Workshops*, 2013.

[8] Tao Chen, Dongyuan Lu, Min-Yen Kan, and Peng Cui. Understanding and classifying image tweets. In *ACM MM*, 2013.

[9] Tao Chen, Hany M. SalahEldeen, Xiangnan He, Min-Yen Kan, and Dongyuan Lu. VELDA: Relating an image tweets text and images. In *AAAI*, 2015.

[10] Yan-Ying Chen, Tao Chen, Winston H. Hsu, Hong-Yuan Mark Liao, and Shih-Fu Chang. Predicting viewer affective comments based on image content in social media. In *ICMR*, 2014.

[11] Tyson Condie, Paul Mineiro, Neoklis Polyzotis, and Markus Weimer. Machine learning for big data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 939–942. ACM, 2013.

[12] Carlos Costa and Maribel Yasmina Santos. Improving cities sustainability through the use of data mining in a context of big city data. In *Proceedings of the World Congress on Engineering*, 2015.

[13] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *WWW*, 2003.

[14] Swarnava Dey, Ankur Chakraborty, Soumitra Naskar, and Prateep Misra. Smart city surveillance: Leveraging benefits of cloud data stores. In *IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops)*, 2012.

[15] D Duncan, X Chu, C Vecchiola, and R Buyya. The structure of the new it frontier: Cloud computing. *Strategic Facilities Magazine*, 9:67–72, 2009.

[16] Meiyu Fan, Jian Sun, Bin Zhou, and Min Chen. The smart health initiative in china: The case of wuhan, hubei province. *Journal of Medical Systems*, 40(3):62:1–62:17, 2016.

[17] Xing Fang and Justin Zhan. Sentiment analysis using product review data. *Journal of Big Data*, pages 1–14, 2015.

[18] David Ferraiolo, Janet Cugini, and D Richard Kuhn. Role-based access control (rbac): Features and motivations. In *Proceedings of 11th annual computer security application conference*, pages 241–48, 1995.

[19] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.

[20] Parke Godfrey, Jarek Gryz, and Piotr Lasek. Interactive visualization of large data sets. Technical report, Technical Report EECS-2015-03 March 31 2015. Department of Electrical Engineering and Computer Science. York University. Toronto, Ontario. Canada, 2015.

[21] Ian Gregory, Christopher Donaldson, Patricia Murrieta-Flores, and Paul Rayson. Geoparsing, gis, and textual analysis: current developments in spatial humanities research. *International Journal of Humanities and Arts Computing*, 9(1):1–14, 2015.

[22] Vasileios Hatzivassiloglou and Kathleen McKeown. Predicting the semantic orientation of adjectives. In *ACL*, 1997.

[23] Jeffrey Heer and Ben Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10(2):30, 2012.

[24] Hugo Hromic, Danh Le Phuoc, Martin Serrano, Aleksandar Antonic, Ivana Podnar Zarko, Conor Hayes, and Stefan Decker. Real time analysis of sensor data for the internet of things by means of clustering and event processing. In *ICC*, 2015.

[25] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *SIGKDD*, 2004.

[26] Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*, 2013.

[27] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 277–281. ACM, 2015.

[28] Yu-Gang Jiang, Baohan Xu, and Xiangyang Xue. Predicting emotions in user-generated videos. In *AAAI*, 2014.

[29] Xiao-Yuan Jing, Ruimin Hu, Yang-Ping Zhu, Shan-Shan Wu, Chao Liang, and Jing-Yu Yang. Intra-view and inter-view supervised correlation analysis for multi-view feature learning. In *AAAI*, 2014.

[30] Zaheer Khan, Ashiq Anjum, and Saad Liaquat Kiani. Cloud based big data analytics for smart future cities. In *International Conference on Utility and Cloud Computing*, 2013.

[31] Li-Jia Li, Hao Su, Eric P. Xing, and Fei-Fei Li. Object bank: A high-level image representation for scene classification and semantic feature sparsification. In *NIPS*, 2010.

[32] Kar Wai Lim and Wray Buntine. Twitter opinion topic model: Extracting product opinions from tweets by leveraging hashtags and sentiment lexicon. In *CIKM*, 2014.

[33] Bing Liu. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.

[34] Patrizia Lombardia, Silvia Giordanob, Hend Farouhc, and Wael Yousefd. Modelling the smart city performance. *Innovation: The European Journal of Social Science Research*, 25(2):137–149, 2012.

[35] Shuangmei Ma and Zhengli Liang. Design and implementation of smart city big data processing platform based on distributed architecture. In *International Conference on Intelligent Systems and Knowledge Engineering*, 2015.

[36] Peter M. Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, United States, 2011.

[37] Michael S Mikowski and Josh C Powell. Single page web applications. *B and W*, 2013.

[38] Hye-Jin Min and Jong C. Park. Identifying helpful reviews based on customer's mentions about experiences. *Expert Syst. Appl.*, 39(15):11830–11838, 2012.

[39] A. Moreo, M. Romero, J.L. Castro, and J.M. Zurita. Lexicon-based comments-oriented news sentiment analyzer system. *Expert Syst. Appl.*, 39(10):9166–9180, 2012.

[40] Kristi Morton, Magdalena Balazinska, Dan Grossman, and Jock Mackinlay. Support the data enthusiast: Challenges for next-generation data-analysis systems. *Proceedings of the VLDB Endowment*, 7(6):453–456, 2014.

[41] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, 2004.

[42] Naila Murray, Luca Marchesotti, and Florent Perronnin. AVA: A large-scale database for aesthetic visual analysis. In *CVPR*, 2012.

[43] Teng Niu, Shiai Zhu, Lei Pang, and Abdulmotaleb El-Saddik. Sentiment analysis on multi-view social data. In *MultiMedia Modeling*, page 1527, 2016.

[44] Eiman Al Nuaimi, Hind Al Neyadi, Nader Mohamed, and Jameela Al-Jaroodi. Applications of big data to smart cities. *Journal of Internet Services and Applications*, pages 6–25, 2015.

[45] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2007.

[46] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*, 2002.

[47] AJ Patel. Red blob gameshexagonal grids. http://www.redblobgames.com/grids/hexagons/. Accessed April, 2016.

[48] Nataliia Plotnikova, Micha Kohl, Kevin Volkert, Andreas Lerner, Natalie Dykes, Heiko Ermer, and Stefan Evert. KLUEless: Polarity classification and association. *SemEval 2015 workshop*, 2015.

[49] Ivan Porres and Irum Rauf. Modeling behavioral restful web service interfaces in uml. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 1598–1605. ACM, 2011.

[50] Jonathon Read and John Carroll. Weakly supervised techniques for domain-independent sentiment classification. In *CIKM Workshop on TSA'09*, 2009.

[51] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. SemEval-2015 Task 10: sentiment analysis in twitter. *SemEval 2015 workshop*, 2015.

[52] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. Evaluation datasets for twitter sentiment analysis: A survey and a new dataset, the sts-gold. *ESSEM workshop*, 2013.

[53] Mukesh Saini, Kazi Masudul Alam, Haolin Guo, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. Incloud: a cloud-based middleware for vehicular infotainment systems. *Multimedia Tools and Applications*, pages 1–29, 2016.

[54] Ben Shneiderman. Extreme visualization: squeezing a billion records into a million pixels. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 3–12. ACM, 2008.

[55] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15(1):2949–2980, 2014.

[56] Kehua Su, Jie Li, and Hongbo Fu. Smart city and the applications. In *ICECC*, 2011.

[57] Moritz Sudhof, Andrés Goméz Emilsson, Andrew L. Maas, and Christopher Potts. Sentiment expression conditioned by affective transitions and social forces. In *SIGKDD*, 2014.

[58] Jussi Tarvainen, Mats Sjöberg, Stina Westman, Jorma Laaksonen, and Pirkko Oittinen. Content-based prediction of movie style, aesthetics, and affect: Data set and baseline experiments. *IEEE Transactions on Multimedia*, 16(8):2085–2098, 2014.

[59] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[60] Peter D. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, 2002.

[61] Yonggang Wen, Xiaoqing Zhu, Joel J. P. C. Rodrigues, and Chang Wen Chen. Cloud mobile media: Reflections and outlook. *IEEE Transactions on Multimedia*, 16(4):885–902, 2014.

[62] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *CIKM*, 2005.

[63] Eugene Wu, Leilani Battle, and Samuel R Madden. The case for data visualization management systems: Vision paper. *Proceedings of the VLDB Endowment*, 7(10):903–906, 2014.

[64] Wenxuan Xie, Yuxin Peng, and Jianguo Xiao. Cross-view feature learning for scalable social image analysis. In *AAAI*, 2014.

[65] Can Xu, Suleyman Cetintas, Kuang-Chih Lee, and Li-Jia Li. Visual sentiment prediction with deep convolutional neural networks. *CoRR*, 2014.

[66] Shintaro Yamamoto, Masahide Nakamura, and Shinsuke Matsumoto. Using cloud technologies for large-scale house data in smart city. In *International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012.

[67] Jiachen Yang, Shudong He, Yancong Lin, and Zhihan Lv. Multimedia cloud transmission and storage system based on internet of things. *Multimedia Tools and Applications*, pages 1–16, 2015.

[68] Yun Yang, Peng Cui, Wenwu Zhu, H. Vicky Zhao, Yuanyuan Shi, and Shiqiang Yang. Emotionally representative image discovery for social events. In *ICMR*, 2014.

[69] Quanzeng You and Jiebo Luo. Towards social imagematics: Sentiment analysis in social multimedia. In *MDMKDD*, 2013.

[70] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. Robust image sentiment analysis using progressively trained and domain transferred deep networks. In *AAAI*, 2015.

[71] Jianbo Yuan, Sean Mcdonough, Quanzeng You, and Jiebo Luo. Sentribute: Image sentiment analysis from a mid-level perspective. In *WISDOM '13*, 2013.

[72] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014.

[73] Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. Social media analytics and intelligence. *Intelligent Systems, IEEE*, 25(6):13–16, 2010.

[74] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *ECIR*, 2011.

[75] Yanyan Zhao, Bing Qin, Ting Liu, and Duyu Tang. Incloud: a cloud-based middleware for vehicular infotainment systems. *Multimedia Tools and Applications*, pages 1–18, 2014.