**United States Traveling Cities Recommendation System**

**Course project for CS586: Big Data Analytics**

**Author:** Chen Chen, Zishan Qin, Hongmei Zong

**Abstract:** The web application of United States travelling cities recommendation system offers customers a friendly user interface to set different criteria for their trip planning regarding to budget limitation, temperature preference, personal interests (such as different weights for dining, lodging and points of interest) and distance selections. The app will return the top traveling cities with a rank by score after applied all the user requirements. User can also look into detailed information about the dining, lodging and points of interest of all these satisfied travel cities. The application system architecture and implementation details are illustrated in the report.

**Keywords:** Big Data; Yelp; Yelp API; Hadoop; HDFS; Big Data Analysis; Web Application; MySQL.

Overview

The project of United States Traveling Cities Recommendation System is aimed to provide a user friendly interface for customers to search for potential interesting traveling cities based on customer’s budget, temperature preference, traveling distance and personal interests (such as dining, lodging, and points of interest favors). The system will offer a list of cities matching customers’ searching conditions with a score related to each city. Besides, customer can also view dining, lodging, and points of interest businesses’ detailed information such as address, business name, phone number etc. in a city based on searching keyword.

In this project, we collected business data using Yelp API 2.0, and other city related information such as temperature, budget, and geographical location from different websites. After parsing raw data, data were stored in both Hadoop Distributed File System (HDFS) and traditional database management system (MySQL). In data processing, we combined searching result both from Hadoop and MySQL based on customers’ searching condition passed from user interface. For displaying searching result and providing a user friendly graphical user interface, we developed a web application to interact with customers.

Here are the contents of our final project report:

Table of Contents

Overview 2

Background Materials 3

System Architecture 4

Design and Implementation 5

Issues 11

Validation of Our Approach 12

Lessons Learned 12

Member Contribution and Communication 13

Future Work 14

Conclusion 14

Background Materials

There are some background materials are listed as following:

**1. Web Crawler:**

Web crawler is a web search engine that browses the Word Wide Web and it copies and saves the useful information as it goes from URLs. For some social websites, like Facebook, Twitter, Yelp or LinkedIn etc. They offer APIs for public. User need to create an application and obtain an Client ID and Client Secret and next step we need to apply authorization code and request token in order to use these APIs from some social websites. Then users can copy and save information from those websites with some limitations.

**2. Yelp API:**

Yelp API is the API that provided by Yelp for developers to do searching related jobs. By using Yelp API, developers will be able to find out best results for searching by location, keywords or distance from over 50 million businesses in 27 countries. Also, some other functionalities include: sort results by best match, limit results by applying some conditions, and identify if a business has been claimed on Yelp.com.

**3. Eclipse:**

Eclipse is an integrated development environment (IDE). It provides tools for coding, building, running and debugging. It originally designed for java language, but now supports many other languages, such as C, C++, python, Ruby, etc. It contains a base workspace and an extensible plugin system for customizing the environment.

**4. Apache Hadoop Framework:**

Apache Hadoop is an open-source software framework written in java. It mainly consists two parts: HDFS, Hadoop File System and MapReduce and the processing part. Hadoop is a distributed file storage system and distributed processing for very large data sets across commodity clusters of computers. Hadoop splits files into large blocks and distributes them among the clusters. Hadoop uses a simple programming model, MapReduce to process the data independently for clusters.

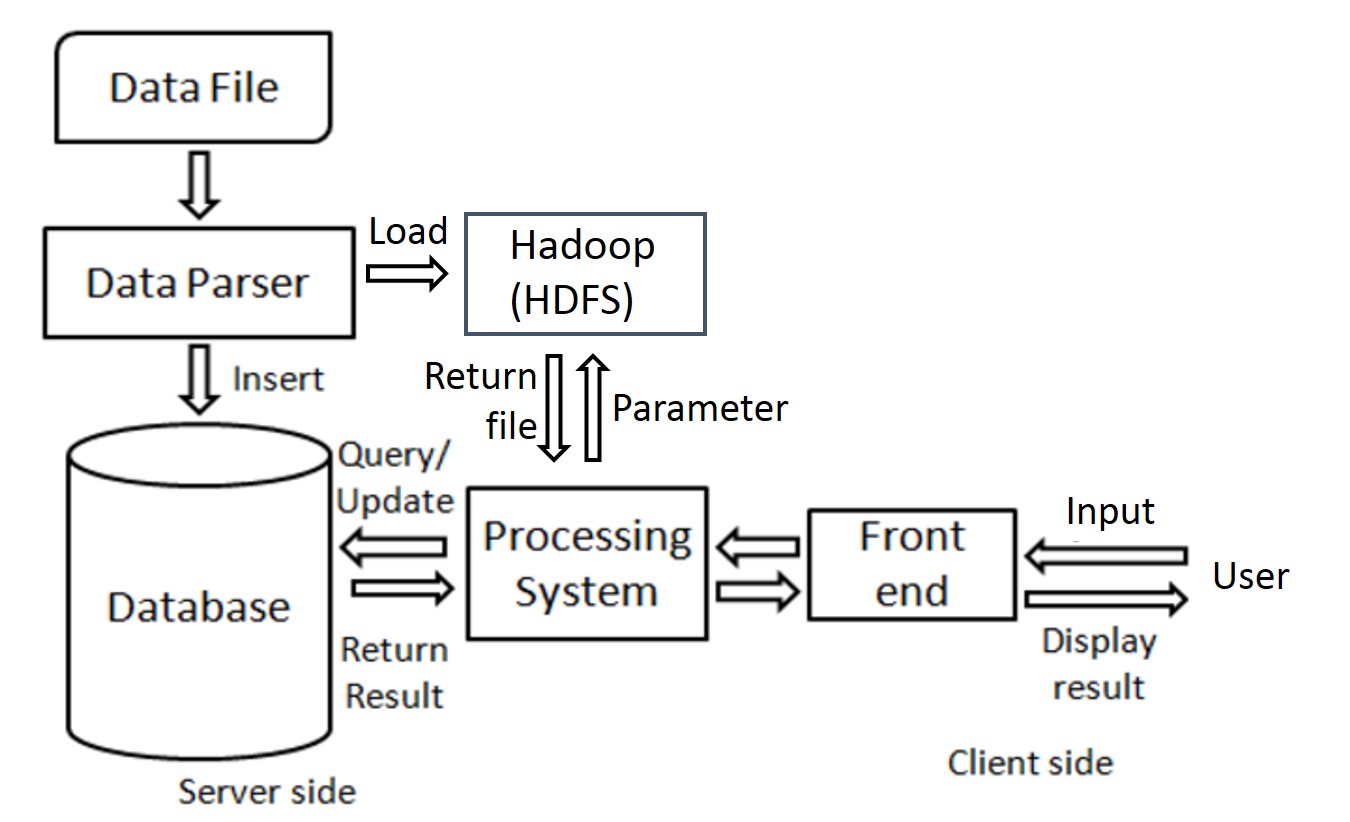
**5. Dynamic Web Application with JSP and Servlet**

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. JSP pages are responsible for interacting with html elements and messages passing to servlet. Servlet is responsible for back-end request processing. In this Recommendation System, we use several web development technologies, including: JSP, HTML, CSS, JavaScript, JQuery, Ajax, and Servlet.

System Architecture

The system is a client/server architecture. Data files, database, Hadoop files and processing modules are all in the server side. The client side is the webpage to accept users’ inputs for different searching criteria. Based on user’s input information, the server need to call different function modules to process the searching job. After the searching result returned from processing module, it passes to front end and display for users.

## Here is the architecture of our system as Figure-1 shows:



**Figure-1 System Architecture**

**1. Grab source data**

We use Yelp API 2.0 for crawling restaurants, hotels and points of interest data from Yelp website for different cities in United States. The historical temperature data is collected from [www.wunderground.com](http://www.wunderground.com). The average daily expense information for different cities are gathered from [www.budgetyourtrip.com](http://www.budgetyourtrip.com). The longitude and latitude data of cities are collected from [www.latlong.net](http://www.latlong.net) . All these information are the points that the travelers may interested in.

**2. Data parser**

The data of businesses of cities are returned in JSON format by Yelp API, We want a clean and well-organized format for Hadoop data analysis job. Therefore, for transforming JSON data file to comma separated records, we write a Java application for data parsing by incorporating Jackson Toolkit for such convenient transformation.

**3. Database**

We use MySQL as our database management system since it is free and easy to use. Besides, we installed MySQL Workbench development tool to help us create tables, view data and edit SQL queries easily. Workbench is also used to validate the user interface results by running queries at Workbench.

**4. Data processing**

In order to offer users cites that satisfied their personal interests, budget, weather preference and distance requirements, we need to implement different function modules in the background. For the personal interests regarding to dining, lodging and points of interests, we need to run Hadoop Map-Reduce job to pass in the different weights for these three categories and get the city ranking results. For the budget and temperature requirement, the SQL queries are generated by the query models and search from MySQL database. For the distance requirement, we need to calculate the distance from the departure city and return the cities within the limited distance. Finally, the cities satisfied all the criteria will be returned to user through the website.

**5. Data display**

For better user experience, we decided to use a interactive web application as our user interface so that we can use many great features like CSS, JavaScript, Ajax. User will input their preferences and personal customized information into the JSP web pages, and the pages will pass the messages to servlet for data processing, by HttpGet/HttpPost requests, and servlet will process the requests and forward to next pages and show the results in a well-formed format.

Design and Implementation

**1. Data File collection**

There are several datasets used for supporting our project, such as (a) dining-lodging-points of interest, (b) daily average expense (c) historical monthly-average temperature, (d) city’s longitude and latitude. All these data are collected from different websites.

We use Yelp API to search restaurants, hotels and points of interest of cities that travelers are more interested. In order to get more information regarding all these searches, we tried to use many related searching keywords, such as Chinese food, Japanese food, barbeque, sushi, etc. for restaurant search. For dining information search, it ends up at 28 keywords used for Yelp search API. For searching lodging and points of interest, we find out 21 keywords can be used through Yelp API. All these keywords and 50 United States city names combined being used for Yelp searching API and we get all the information regarding to dining, lodging and points of interest. The dataset is about 50MB. There are detailed information listed in every record, such as, id, location information, telephone number, total count of reviews, the average rating, categories, URL, etc. We do not need all the information to do the analysis, thus, in the next step, we will do the preparation for the analytical dataset.

The historical temperature data is collected from [www.wunderground.com](http://www.wunderground.com). We searched monthly average temperature from January to December for all 50 cites, and saved to a comma separated file (CSV). The average daily expense information for different cities are gathered from [www.budgetyourtrip.com](http://www.budgetyourtrip.com), also saved in a CSV file. The longitude and latitude data of 50 cities are collected from [www.latlong.net](http://www.latlong.net) and saved in a CSV file. All these data are stored in the MySQL database. For dining-lodging-points of interest dataset, it is used to do the analysis using Hadoop and generate the total score of the cities and get the cities’ rank.

**2. Data Preparation**

We have the dining-lodging-points of interest original dataset saved as a JSON file with many attributes in each record. We don’t care some part of the attributes, so we finally decide to remain the following interesting attributes, as Table-1 shows:

**Table-1 Attributes We interested**

|  |  |
| --- | --- |
| phone | the phone number of the store, used for recommendation |
| rating | the rating given by the customers, used for comparing the stores |
| review\_count | the number of reviews by the customers, used for comparing the stores |
| url | the site address of the store in Yelp, used for recommendation |
| snippet\_text | one comment of this store, used for recommendation |
| name | the name of the store, used for recommendation |
| id | the unique ID of the store, used for removing duplicates in data preprocessing |
| display\_addresses | The actual address of the store, used for preprocessing |
| categories | the categories of the store in Yelp site, used for partition the stores |

Additionally, we added our defined keywords and cities as two predefined attributes, since we need to use these information to divide the stores into three types, restaurants, hotels, and attractions to compute the recommendation ratings, as Table-2 shows:

**Table-2 Predefined attributes for data parsing**

|  |  |
| --- | --- |
| tagged\_city | the city of this record which we use when we call the API to get the result. The cities are defined in our Cities meta file. |
| tagged\_term | the term of this record which we use when we call the API to get the result. The cities are defined in our Keywords meta file. |

We write a program to parse the JSON records to our defined records, which will be used in Hadoop HDFS. The defined record is like the following, each attribute separated by comma, as Table-3 shows:

**Table-3 Attributes in Target format**

|  |
| --- |
| tagged\_city, tagged\_term, phone, rating, review\_count, url, snippet\_text, name, id, address, categories |

Then we can load the parsed data onto HDFS for Hadoop processing also to the processing module to save the dataset in MySQL .

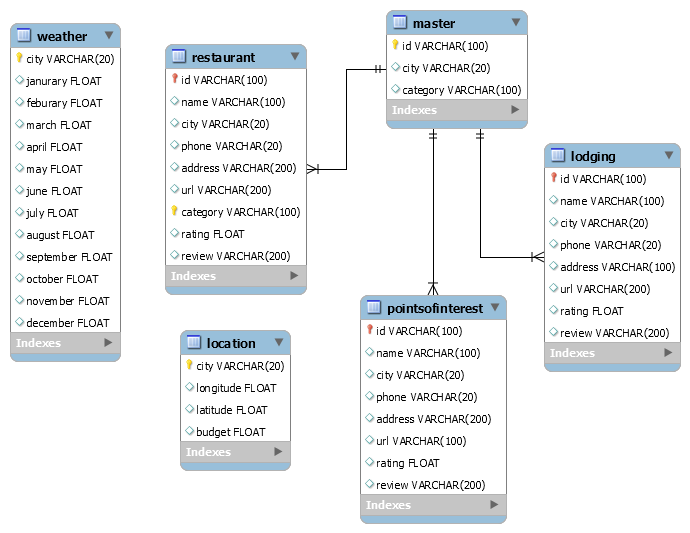
**3. Data analysis on Hadoop**

The parsed dataset is loaded to the HDFS. The first analysis for the dataset is to find out the total score of three categories: dining, lodging, points of interest for each city. There is one Map-Reduce job required to do this analysis. In the Mapper side, all the records are filtered and <city\_name category,rating> is output from mapper. Reducer sum the rating for each category and output the sum for each city as <city\_name dining\_score, lodging\_score, interest\_score>.

The second analysis is to rank the cities by the user’s interests. User can define the weights for three different categories, weight range from 1 to 3. The weight 3 denotes user caring that category very much and 1 is not care much, 2 is moderate. In Hadoop implementation, the mapper reads in the file of city with total score and user defined parameters, then do the calculation of parameter multiple category score. Last, sum three category score to total weighted score for each city. In the reducer part, we sort the city by it’s score and find the maximum score among all cities. Then each city’s score is divided by maximum score which is the rank score of that city. Reducer output<city\_name rank score>. There are total 27 Hadoop jobs executed for different combination of user defined parameters.

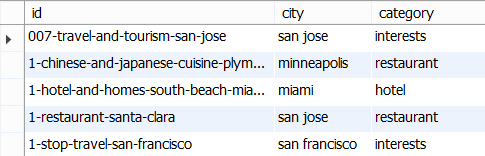
**4. Database Design**

We load data into MySQL database management system to store cities and businesses information. Here is database schema diagram as Figure-2 shows, showing how we designed the database:



**Figure -2 Database Schema**

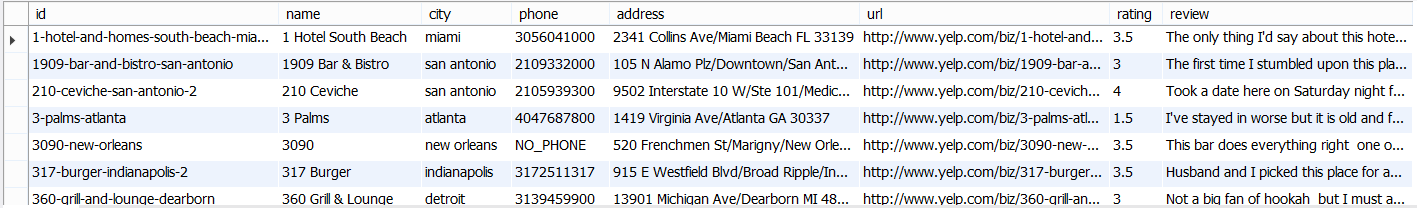
We have 6 tables in our database. master is primary table, storing business id, corresponding city and category. We divide all businesses into 3 tables: restaurant, lodging and pointsofinterest based on business’s type. In these 3 tables, id is the foreign key referring to id in master table. Several attributes are designed to represent business’s basic information such as name, phone, address, rating and so on. We also have 2 tables storing cities’ information. weather table is used to store each city’s temperature information, including average temperature in different month. location table is used to store each city’s longitude and latitude. Budget is also stored in location table since it is easier for us to write queries. Here are some snapshots for each table, as the figure 3~8 shows:



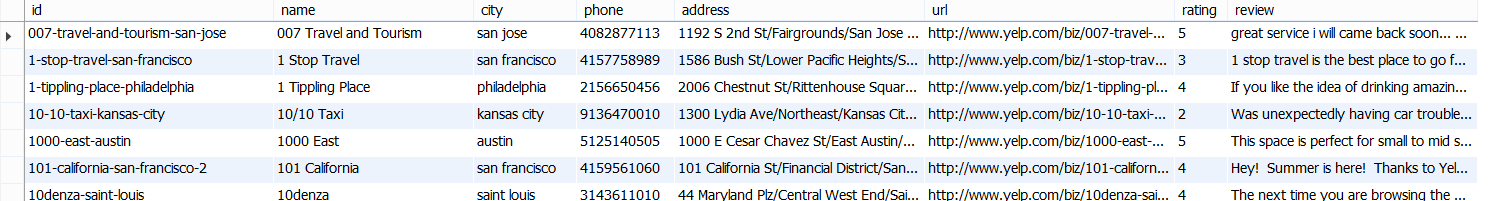
**Figure -3 Master Table**



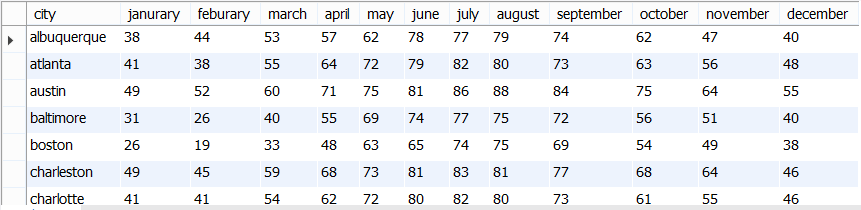
**Figure-4 Restaurant Table**



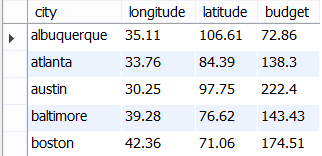
**Figure-5 Lodging Table**



**Figure-6 Pointsofinterests Table**



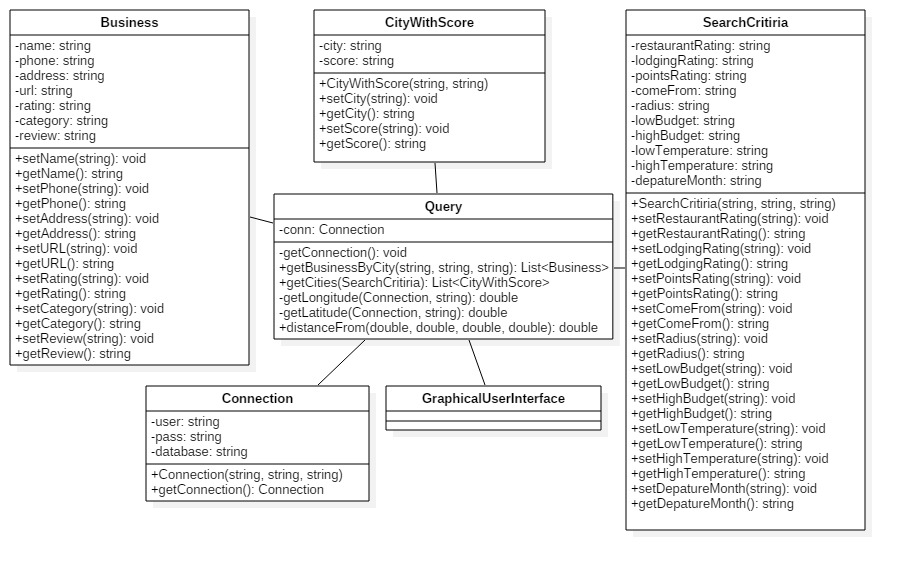
**Figure-7 Weather Table**

l

**Figure-8 Location Table**

**5. System Processing**

We implement Java code in back end to process queries and get results. Here is the UML diagram showing our classes, as Figure-9 shows:

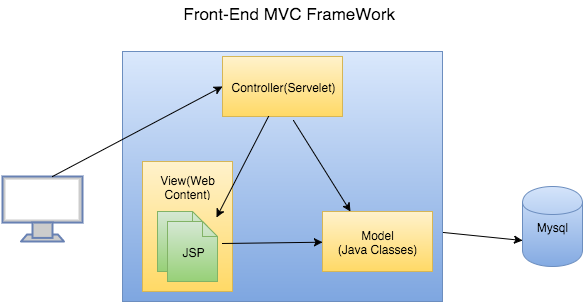


**Figure-9 Class Diagram of Business Logic**

Here is a brief work overflow of how system works. When user defines searching conditions such as rating to restaurant, traveling distance, budget range and so on, a SearchCritiria will be created. Then SearchCritiria object will be passed to Query class (getCities funtion). The function will load corresponding text file based on ratings to restaurant, lodging and points of interest into program. The text file was preprocessed by Hadoop system. After loading, a search query based on other searching conditions (except ratings to dining, lodging and points of interest) for finding matching cities will be created and a list of qualified cities will be returned. Then we do the intersection of the list and text file to return the final result list. When user wants to explore more details (restaurants, hotels or points of interests) of a city, city name, corresponding table name and search keyword (such as “sandwich”) will be passed to function getBusinessById in Query class. A business searching query will be created and executed. Finally, Business object will be returned containing all useful information regarding to business.

**6. User interface**

The front end module is used for interaction between user and our system. The front end is based on MVC architecture, as Figure-10 shows, which enables the separation from business logic and the display. Our front end module structure is shown as figure below: the JSP files are responsible for display, the servlets serve as the controller for handling the user requests, and our defined Java classes which are mentioned in the System Processing part are responsible for business logic processing. The general procedure is like, the user input their preferences as requests, and the servlets get these requests and process by Servlet doPost/doGet functions, the servlets processing ends, rendering the JSP files to user for showing the results.



**Figure-10 Front-End Architecture**

Issues

1. When inserting data from text file into database, we met some problems such as data type does not match. After examination, we found out that category attribute data are lost in some records. After discussion, we decided to regard searching keyword as category for records who lost category attribute.

2. When filtering cities based on distance, query is not direct since we only have latitude and longitude data for each city. Our solution is, instead of solving problem in one query, we divided distance calculation process into multiple steps. First, query longitude and latitude for each current qualified city, and called a helper function to calculate distance from departure city to current city.

3. There is a problem that how to pass in user defined parameters into map-reduce jobs, since we need to run many Hadoop jobs for three category’s weights defined by the user. We first try to use static variables in the map-reduce job class, but the mapper cannot get the updated value of these variables. We read many references regarding to this problem, and find out the method that use job configuration to set the variable values in the the phase of configuring a job, then get the value when run map function in the mapper. So, the problem is solved and we learned from it.

4. There is some difficulty when render the businesses details for restaurants, points of interests, hotels and filtered results, the records for each kind are all different. We need to present different data according to different requests. We tried several ways to figure out this difficulty, such as use multiple pages to show different results, refresh whole pages and load different data according to different requests and use only one page to dynamically load part of data according to different requests. Considering the user experience, we planned to use a user-friendly and efficient way to dynamically update the page and avoid to refresh the whole page or redirect to new pages . We applied Ajax call for overcoming this difficulty.

Validation of Our Approach

For city rank, we ran both Hadoop and Java code to compare the results when passing 3 rating parameters (ratings to dining, lodging and points of interests). After comparison, we find that the results are totally the same, which means our Hadoop map-reduce job was correct.

We went to yelp website for further validation. In our system, a city with higher score means current city has more businesses. We observed that Los Angeles’s rank is far higher than Nashville, which means Los Angeles should have more businesses than Nashville. After searching keyword “Chinese”, 100 pages of businesses return in Los Angeles comparing with 26 pages of businesses return in Nashville, which means our result is representative.

In our system, if user is interested in a city and click on “Details” button, a list of businesses with detailed information will return, including URL. For checking if the business really exits, we click on URL (which direct us to Yelp webpage) and compare business information such as name, phone, address and so on between our system and Yelp webpage. It turns out that our business result is accurate.

Lessons Learned

Chen Chen: As the team leader, the most important thing I learned is how to manage project processing and team communication. For project processing part, we encountered some technical troubles which we cannot solve them in time, so we switched our topic to make sure we can finish our final project by the end of semester. For team communication, everyone has his own schedule, so managing the weekly meetings is not so easy. Other things I have learned is how to use web crawling toolkit such as beautifulsoup (a python package) and how to use Yelp API to fetch data from yelp website. Also, how to parse json format file and extract useful information from raw data. Also, I have learned some base web developing knowledge from Zishan Qin.

Hongmei Zong: I have tried different approaches to crawl data from websites especially how to use APIs provided by the social network development groups to get data from their websites. I practiced MySQL Workbench and SQL queries through the project. My Map-Reduce design and implementation ability has been enhanced by solving the problems encountered in the Hadoop job design. I worked with my team member and learned java programming and webpage design knowledge.

Zishan Qin: For data processing, I enhanced the ability to process data to parse from some type to the format which is easier to analyze and learned many useful open source toolkits for data parsing. For the front-end module I learned how to design and implement a complete dynamic web application based on MVC architecture. Besides, I got deep understanding for how to incorporating the front end pages with the back end business logic. Furthermore, I also learned some handy and strong web development techniques like: JSP, HTML, CSS, JavaScript, Ajax, JQuery.

Member Contribution and Communication

We mostly work very closely with each other, and always report our progress and discuss problems with team members using E-mails, WeChat and regular face-to-face meetings. We have face-to-face meeting twice a week. We send messages through WeChat many times in a day. Everyone in the team are very actively take part in the discussion and response to others very promptly. We have no problem at all regarding communication in our team.

The individual contribution for the project are list below:

Chen Chen: As the team leader, I oversaw the development of the system, including assigning sub-tasks to each member, arranging meetings, and motivating my team members. During the project, I wrote raw code for getting data by using Yelp API 2.0. Also, I worked with Hongmei to figure out keywords for searching dining, lodging and points of interest business, and a city list with 50 city names. Besides, I designed database with my teammates, inserted data into database, and wrote backend code to generate queries and executed them to return appropriate result to front end. Finally, I participated in writing final project report and making presentation.

Hongmei Zong: I explored many ways to crawling data from website. I worked with Chen Chen tried a big variety of keywords for dining, lodging and points of interest, then finalize the keyword set and city name set used for our first big dataset searching. I collected average budgets dataset, average monthly temperature dataset and longitude and altitude dataset of 50 United States cities. I designed and implemented Map-Reduce jobs to do the dataset analysis and find out the city rank of 50 United States. I also involved to prepare UML, the final report and presentation PPT.

Zishan Qin: In this project, I participate the whole process and learned a lot of skills. The parts I took deep dive are data processing and the front-end modules development. For data processing, I implemented the methods for iteratively query records by Yelp API based on certain keywords and parsing data from JSON files to our targeted format. For the front-end developments, I implemented the web pages and the controllers for processing the requests and front display to realize the interaction between front end and back end. Finally, I participated in writing the final project report and making presentation.

Future Work

The next step for our work is to (1) extend traveling cities to all around the world; (2) get business information in real-time instead of loading from database; (3) use web crawler to fetch data from Yelp webpage and other similar websites (such as TripAdvisor) instead of using Yelp API; (4) update our data periodically (for example, 2 weeks).

Conclusion

We successfully developed a web application system to recommend top United States traveling cities. Users can define searching conditions such as budget range, temperature range, traveling distance limit and personal interests (such as ratings to dining, lodging and points of interest). Based on searching condition, we recommend potential traveling cities that user might be interested in. In our system, we combined both Hadoop and traditional database management system (MySQL) to do data analytics.

Our deliverable will include final project report, all the code, database and PowerPoint file.