Travel Itinerary Based on Sentiment Analysis of Yelp Reviews

Aniket Arun Chauhan
Computer Science and Engineering
State University of New York at Buffalo
Buffalo, New York, United States of America
ankietar@buffalo.edu

Anant Sarin
Computer Science and Engineering
State University of New York at Buffalo
Buffalo, New York, United States of America
anantsar@buffalo.edu

***Abstract* — We apply principles and techniques of Sentiment analysis recommendation systems to develop a model of Travel Itinerary for a location in USA. Using Yelp's reviews, we extract the sentiments of people to find a collaborative star rating for a business using the parallel processing through Hadoop file system.**

I. Introduction

What is a common problem that all of us inevitably face? When you plan a vacation, planning an efficient itinerary is important. You do not want to end up missing historical places, eating bad food, miss must see attractions. You spend a lot of time and effort in creating your perfect Itinerary for your perfect vacation. That is why we created this project. We thought why not let our system handle the itinerary part of your vacation and you handle just packing all the important stuff.

II. Architecture

As shown in the Figure our design is composed of six interleaving parts:

A. Data collection,
B. Data Pre-processing
C. Parallel Data Processing
D. Final Processing.
E. User Interface
F. Final Itinerary

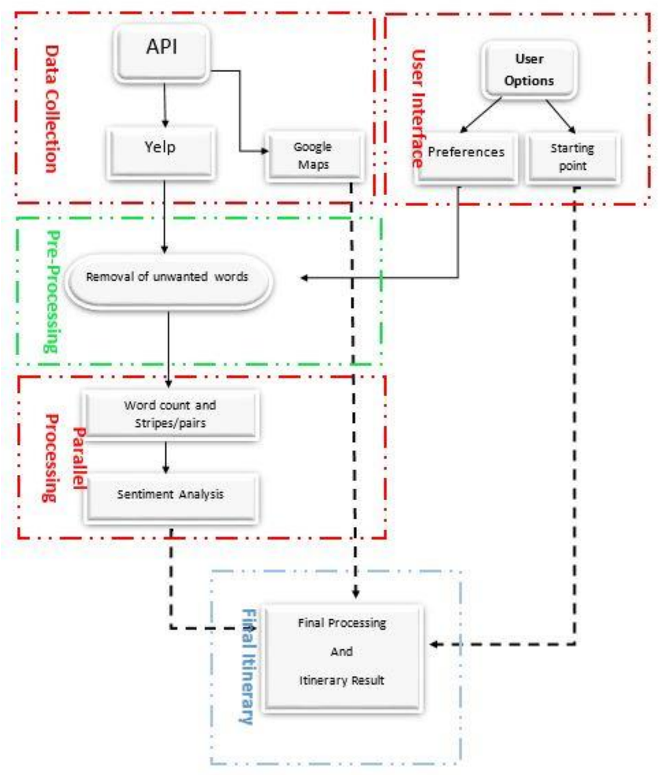


Figure 1.1 Architecture.

A. Data Collection

Data is an integral part of our project, after all our project is based on sentiment analysis. Data collection for this project is done using the help of the Yelp Developer API. While the API provides all the necessary information and more, it does not provide the most important part, the reviews. As far as rest of the data is concerned Yelp have easy API's through which the data we require can be accessed easily. As far as reviews are concerned, it is not an easy task.

We collected data for 14 U.S States in total. We took top 20 cities in these 14 states and top 20 businesses from each of the below listed categories.

- Religious Organization
- Arts
- Entertainment
- Food
- Restaurant
- Night Life
- Shopping
- Active Life
- Adventure
- Spas.

These categories help us in serving the User more efficiently based on his/her interests.

All this data that we collect get stored in the local MySQL Database. We have designed an efficient normalized database to help store us all the required information we get from the Yelp API's. Data collection is an ever ongoing process to make the result more reliable.

The hierarchy of the data base is as follows –

<u>ID</u>	int(11) AI PK
BusinessID	varchar(300)
Name	varchar(300)
URL	varchar(300)
Rating	double
ReviewCount	int(11)
Phone	varchar(50)
Address_Line1	varchar(300)
Address_Line2	varchar(300)
City	int(11)
State	int(11)
Country	int(11)
PostalCode	int(11)
ImageURL	varchar(300)
RatingImgURL	varchar(300)
RatingImgURLSmall	varchar(300)
RatingImgURLLarge	varchar(300)
Snippet	varchar(500)
SnippetImageURL	varchar(300)

Table 1.1 *business*

<u>ID</u>	int(11) AI PK
BusinessID	int(11)
CategoryID	int(11)

Table 1.2 *business_categories*

<u>ID</u>	int(11) AI PK
BusinessID	int(11)
YelpCategoryID	int(11)

Table 1.3 *business_yelpcategories*

<u>ID</u>	int(11) AI PK
BusinessID	int(11)
YelpCity	int(11)

Table 1.4 business_yelpcity

<u>ID</u>	int(11) AI PK
Category	varchar(100)

Table 1.5 *categories*

<u>ID</u>	int(11) AI PK
State	int(11)
City	varchar(100

Table 1.6 *cities*

<u>ID</u>	int(11) AI PK
Country	varchar(100)

Table 1.7 *countries*

ID	int(11) AI PK
City	int(11)
Postal_Code	varchar(15

Table 1.8 *postal_codes*

ID	int(11) AI PK
BusinessID	int(11)
Rating	double
Review	varchar(5000)

Table 1.9 *reviews*

ID	int(11) AI PK
Country	int(11)
State	varchar(100)

Table 1.10 *states*

ID	int(11) AI PK
Category	varchar(100)

Table 1.11 *yelp_categories*

ID	int(11) AI PK
BusinessID	int(11)
Rating	float
FamilyWordCount	int(11)
FamilyPositive	int(11)
FamilyNegative	int(11)
CoupleWordCount	int(11)
CouplePositive	int(11)
CoupleNegative	int(11)
FriendsWordCount	int(11)
FriendsPositive	int(11)
FriendsNegative	int(11)
ProcessedScore	float

Table 1.12 *preprocessed_data*

All the data we collect is stored in these tables. After we process our data through our data processing algorithms to perform sentiment analysis, the final processed data is stored in preprocessed_data table. This table contains the final rating based on our algorithms.

B. Data Pre-Processing

This module is concerned with making the data that we have available for final processing, hence the step is called as pre-processing. Here we remove the stop words from the comments, because we don’t want to end up using our resources on processing words that are not relevant. Removal of these stop words and other special characters gives great numbers and efficiency in our results. The stop words that we removed are listed as - {*a able about across after all almost also am among an and any are as at be because been but by can cannot could dear did do does either else ever every for from get got had has have he her hers him his how however i if in into is it it’s just least let like likely may me might most must my neither no nor not of off often on only or other our own rather said say says she should since so some than that the their them then there these they this tis to too was us wants was we were what when where which while who whom why will with would yet you your* },
And symbols are listed as -
{ *[-+^.:,%$#_+=!'?*@ () 0123456789\] ”*}.

After this preprocessing we create separate files programmatically for all businesses and store all the reviews as businessID.txt format to feed it into the parallel processing through Hadoop.

C. Parallel Data Processing

After preprocessing we have all the business files that we need. These file will now serve as input to the Parallel part of our project. In Hadoop we process the data to find out the emotional word counts and other counts to give input to the final algorithm for the sentiment analysis. In the Hadoop we used mapper and reducer class to find out our result , first of all folder containing the files is given to the mappers class , then the mapper divides the input files in many different number of chunks . Now these chunks are processed by the mappers one by one. In mapper function we used the current file name as the output key and a MapWritable map as output value to emit from the mapper. Map value will have different type of key-value inputs which consist of Text as key and count as value

e.g. map.put(word, rating)

where word can be some string *numberOfFamilyWords, numberOfCoupleWords, numberOfFriendsWords, friendsPositive, friendsNegative, CouplePositive, coupleNegative, FamilyPositive, FamilyNegative* and value which is shown as "one" is a FloatWritable constant 1.0 .

Now by setting these values in a map we then emit the key value pair from the mapper by writing it to the context

e.g. context.write(outputkey, map)

Here the output key represents the file name which we are reading from and the map represent the value to be emitted from the emitter and to be sent to the Reducer. After emitting the key-value from the mapper it reaches to the reducer. In Reducer , one key and the associated values will be received by a reducer at one time then all the values will be processed , in our case the key that reducer got will be the file name and the values associated with the key are the maps which itself contains the key-value pair . All the maps associated with the file name are taken in a loop and then all the values are added to the counters hence giving the total counts for a document/business so we can process the data in our algorithm. Hence, in Hadoop file system we find the count of positive and negative words for a business , number of words related to couple , friends and family and also we find the number of positive and negative words associated with couple , friends and family . These gives the numbers which will help us to characterize the data between the family, couple and friends.

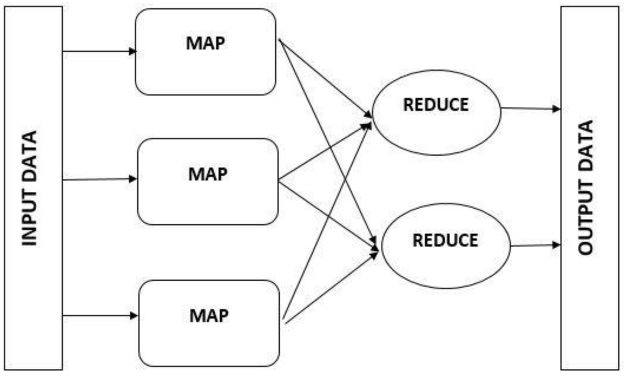


Figure 1.2 Map – Reduce Architecture

The next big thing in the processing is Score which is the most important base of our algorithm. Score is calculated when we are processing the data in the Hadoop clusters, every comment in a business is given a Score based on the formula of calculating S as shown below :

$$S = \frac{number\ of\ positive\ words\ in\ a\ comment}{number\ of\ negative\ words\ in\ a\ comment - number\ of\ negative\ words\ in\ a\ comment}$$

Then the value rating = S is inserted in a map

map.put(word, rating)

And this map is then emitted from the mapper through writing it into the context as

context.write(outputkey,map)

Now, just like all the values this key-value pairs this one also summed up in the reducer to find out a total Pre score by the formula –

$$Pre - Score = \sum_{i=1}^{t} rating$$

Using the above formula we get a score. This score is now divided by total number of reviews, which finally gives us a score of hundred in percentage.

$$Percent\ Score = \frac{Pre - Score}{Total\ Number\ of\ comments} \times 100$$

D. FINAL PROCESSING

Final processing deals with creating the final score out of 10 for each business. The percentage we get in parallel processing is now created to a score of 5 using the table stated below:

Score Percentage	Stars out of 5
0% to 10 %	0.5
11% to 20%	1
21% to 30%	1.5
31% to 40%	2
41% to 50%	2.5
51% to 60%	3
61% to 70%	3.5
71% to 80%	4
81% to 90%	4.5
91% to 100%	5

Table 1.13 Rating Conversion table(1)

This is the total score out of 5 that we get from the reviews. The number of reviews also plays a big role. A business with a score of 4 with 2 reviews and another business with a score of 4 with 1000 reviews definitely means the latter is better. Hence we give each business another score of 5 based on the number of reviews.

Number of Reviews	Stars out of 5
>1000	5.0
>800	4.5
>700	4.0
>600	3.5
>500	3.0
>350	2.5
>200	2
>100	1.5
>50	1.0
>0	0.5
=0	0

Table 1.14 Rating Conversion table(2)

These two scores out of 5 are now added to give a total score of 10, which is our final score for each business. This score is used when we decide which business to list when user makes a selection.

E. USER INTERFACE

Like all good projects, we too have a User Interface to showcase out final results. We have made the user interface using Java Servlets and deployed thee same using Tomcat Server.

The user interface we have, makes use of user inputs and selection to generate perfectly customized results for each user. The output is the Itinerary and a map to help user visualize the itinerary. The map is displayed with the help of Google Maps Direction API.

User inputs consists of State, City, State & End Point, Categories and Company. After giving this inputs and submitting the form the user gets the final output. The data stored in the MySQL Database with collected data is used in conjunction with the preprocessed data to populate the client side with the proposed itinerary.

F. FINAL ITINERARY

Final Itinerary is divided into two parts:

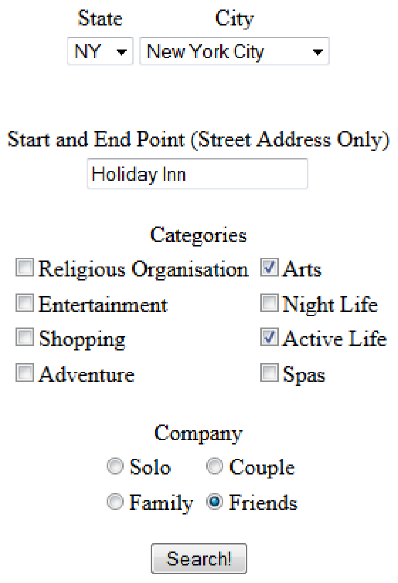
a) User Input

b) Final Result

a) User Input:

User input is taken from the user with the help of a web page with certain fields. The fields that need to be inputted from the user are:

- State
- City
- Start & End Street Address Only
- Interested Categories
- Company



State: NY

City: New York City

Start and End Point (Street Address Only): Holiday Inn

Categories

☐ Religious Organisation	☑ Arts
☐ Entertainment	☐ Night Life
☐ Shopping	☑ Active Life
☐ Adventure	☐ Spas

Company

○ Solo ○ Couple

○ Family [✓ Friends]

Search!

Figure 1.3 Input GUI form

State can be selected with the help of drop down menu on the web page. Only those states are listed for which the data is available. After the state is selected, all the city corresponding to that state are automatically populated, Then the user needs to enter the Street Address of his home/hotel for the above selected city. User then has the option to select all the categories that he/she is interested in visiting and also the company that the user plans to visit the place with. After selecting and inputting al the required values user can submit the page to see the desired results.

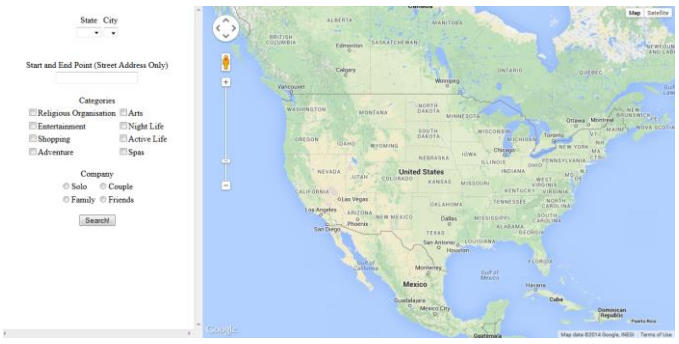


Figure 1.4 GUI welcome Screen

b) *Final Result:*

The final result is shown with the help of google maps API as below:

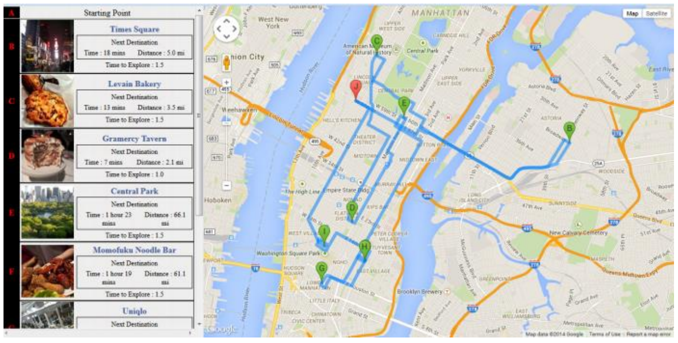
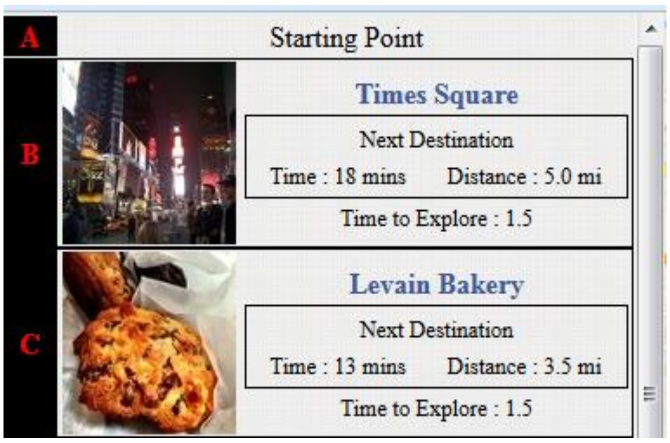


Figure 1.5 Output Screen

The page consists of two parts. One consists of details of each locations, with time to spend at that location and distance to next location.



A Starting Point

B Times Square

Next Destination

Time : 18 mins Distance : 5.0 mi

Time to Explore : 1.5

C Levain Bakery

Next Destination

Time : 13 mins Distance : 3.5 mi

Time to Explore : 1.5

Figure 1.6 Detailed Locations

The next part contains of the actual map, which gives a general idea of the commute as shown below:

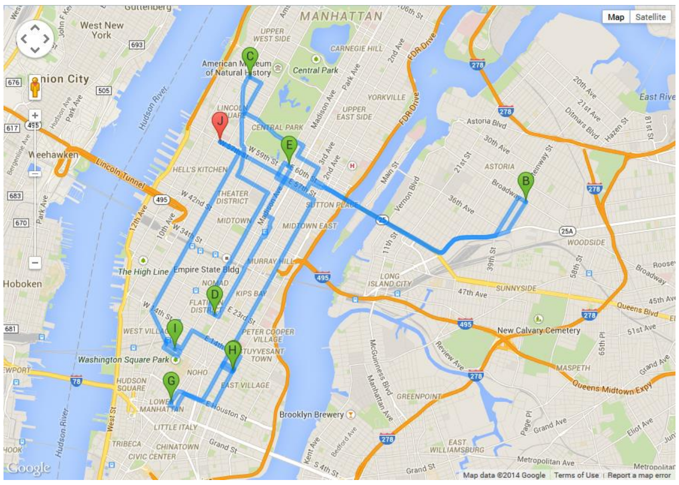


Figure 1.7 Output Map

III. Conclusion

Travel Itinerary Recommendation system applies the concept of sentiment analysis and developed a complete travel itinerary with respect to the preferences given by the user and the results of sentiment analysis, It gives a detailed plan with the collaborative star rating and time to spent at a place combined with the google route map of the Itinerary through GUI.

References

[1] Anant Sarin, and Mangesh .Data Intensive Computing final project , Big data content Retrival , Storage and Analysis foundation, April 2014.

[2] Google Developer API, www.developer.google.com

[3] Yelp Developer API ,www.yelp.com/developers .

[4] Rudy Prabowo, and Mike thelwell ,Sentiment Analysis : A Combined apoproch.

[5] Pragmatic Programming Techniques, horicky.blogspot.com/2008/11/hadoop-mapreduce-implementation.html

[6] Jeffrey Breen, Twitter Sentiment analysis words, github.com/jeffreybreen/twitter-sentiment-analysis-tutorial-201107

[7] MySQL Database , www.mysql.com