Restaurant recommendation system using the Hadoop Framework

Nikita Amartya
Dept. of Computer Science, NYU
New York
nn899@nyu.edu

Rahul Desai
Dept. of Computer Science, NYU
New York
rud206@nyu.edu

Suruchi Sharma
Dept. of Computer Science, NYU
New York
sss665@nyu.edu

Abstract—

Producing high quality recommendations has become a challenge in the recent years. The growth in the quantity of data involved in the recommendation process poses some scalability and effectiveness problems. Review rating prediction is one of the most challenging text mining problems that some commercial companies like Yelp want to explore. The rating prediction involves both feature extraction and classification process. Our analysis provides a list of all possible restaurants that the user might like, based on the inputs provided by the user and the weighted analysis of other users with similar taste who have reviewed similar businesses.

Keywords - Yelp, Recommendation Systems

I. Introduction

There are a lot of online restaurant review portals. Users generally browse through one of these trusted data sources (such as Yelp) for reviews while choosing a restaurant. In these reviewing systems, individual users pen down their own personal experience of a particular restaurant and the other users draw inferences by reading the reviews that they find relevant. Indeed this task is tedious as it involves a lot of work of going through all the reviews and reading them. To solve this problem most portals provide an overall rating along with the textual review. So, a user just reads the reviews which give a higher rating to a restaurant and make decisions based on a few of these high rated reviews.

Through our analytic, we want to make more specific recommendations based on the user's personal preferences. Using Hadoop framework we can process large amounts of data faster and in a more reliable way. We used related datasets available for free from Yelp and cleaned them to get the useful information that we needed for our project using Map Reduce. The resulting data was loaded in Hive. Natural Language Processing was used to identify and assign higher weightage to reviews from users with similar eating habits and taste. Finally, the available data was queried to analyze the possibility of a user liking a restaurant. The result shows as a percentage that can further be classified into ranges assigning names to each such as "Not recommended", "Highly recommended", etcetera.

II. Motivation

Consumers usually do not make full use of information available on websites and eventually overestimate or underestimate scores compared with their attitudes in review text. This situation often occurs when fresh users use review systems to make their first judgment.

Yelp dataset reviews can be used to classify and project the relevance of a user review and make recommendation. Our project aims at analyzing the text of all such reviews and providing recommendations based on reviews by users whose overall taste matches with a user's requirement.

Yelp provides data about businesses, reviews, users, and check-ins. Many recommendation system applications use only the items that customers review and explicitly rate to represent their interests, but they can also use other attributes, including items viewed, demographic data, location, etcetera. Our idea is to predict the overall rating of a business using the textual reviews and provide a more targeted recommendation service to Yelp users, which can assure them more relevant reviews.

III. Related Work

[6] Most recommendation algorithms start by finding a set of customers whose purchased and rated items overlapping the user's purchased and rated/reviewed items.
There are three common approaches to solving the recommendation problem: traditional collaborative filtering, cluster models, and search-based methods. Search-based methods and item-to-item collaborative filtering focus on finding similar items, not similar customers. For each of the user's purchased and rated items, the algorithm attempts to find similar items. It then aggregates the similar items and recommends them. Using collaborative filtering to generate recommendations is computationally expensive. It is O(mn) in the worst case, where m is the number of customers and n is the number of items. Also, the algorithm encounters severe performance and scaling issues. It is possible to partially address these scaling issues by reducing the data size. Unfortunately, all these methods also reduce recommendation quality in several ways. First, if the algorithm examines only a small customer sample, the selected customers will be less similar to the user. Second, item-space partitioning restricts

recommendations to a specific product or subject area. Third, if the algorithm discards the most popular or unpopular items, they will never appear as recommendations, and customers who have purchased only those items will not get recommendations. Cluster models can perform much of the computation offline, but recommendation quality is relatively poor. To improve it, it's possible to increase the number of segments, but this makes the online user–segment classification expensive.

Search (or content-based) methods treat the recommendations problem as a search for related items. Given the user's purchased and rated items, the algorithm constructs a search query to find other popular items by the same author, artist, or director, or with similar keywords or subjects.

We will be using hive and MapReduce for our analytics.

[1] MapReduce is a functional style programming model for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. It allows us to parallelize large computations and to use re-execution as primary mechanism for fault-tolerance. The map function emits each word plus an associated count of occurrences and the reduce function sums together all counts emitted for a particular word.

[2]Traditional data warehousing solutions were increasingly becoming expensive because of rapidly growing size of datasets and map-reduce programming model requires developers to write custom programs, which are hard to maintain and reuse. Facebook Data Infrastructure team built an open-source data warehousing solution for this - Hive. Hive is built on top of Hadoop and supports queries expressed in a SQL-like declarative language - HiveQL, which are compiled into map-reduce jobs executed on Hadoop. HiveQL supports custom map-reduce scripts to be plugged into queries and includes a type system with support for tables containing primitive types, collections like arrays and maps, and nested compositions of the same. The underlying IO libraries can be extended to query data in custom formats. Hive features Hive-Metastore for query optimization, which is a system catalog containing schemas and statistics.

IV. Design

1. *DATA PROCUREMENT AND REFINEMENT:*

a) Data Collection

Big data (i.e.) a large set of review data is collected from the official page of the Yelp Data-Set challenge.

The dataset contains information about the businesses, users, user reviews and the user check-in details. Details such as the name of the business, unique ID of the business, address of the business, working hours, various attributes such as Wi-Fi availability, parking availability, etc. are a part of the businesses information. The user data comprises the user ID, name and other yelp profile details of a user. Review data consists of business ID, user ID (the ID of the user who has reviewed the business), numerical rating on a scale of 5, textual review (comments), date when the user reviewed the restaurant and the votes that this review has been given by other users. The check-in data gives information about the number of check-ins made at a business on a certain time and day.

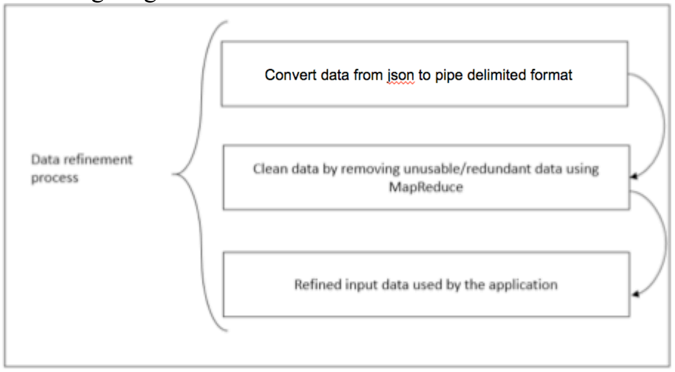
b) Data conversion to required format

The data was originally acquired in JSON format [3] and we applied a python script to convert it to a pipe-separated format. We chose pipe as the delimiter to make it easier for identifying each column in a row as some columns have data where comma is used (e.g., full address and categories). JSON file was parsed to extract names of attributes which became the column names in the pipe-separated file. The JSON file was read and the JSON values were written against each attribute for every JSON entry under the respective column names.

c) Data filtering

The information related to businesses was filtered to extract only the restaurant information to work with. Likewise, some modifications were made to remove any non-relevant information. A shell script was used to aggregate all the data that belongs to the restaurant category.

The data collection process can be summarized in the following diagram.


Data refinement process
Convert data from json to pipe delimited format
Clean data by removing unusable/redundant data using MapReduce
Refined input data used by the application

2. *ANALYTICS:*

a) Input collection:

User analytics:

The aim of these analytics would be to return to a user, with a certain degree of confidence, whether he would like or dislike the restaurant of his choosing. The end-user will provide information about a potential customer and the restaurant that he wants to visit.

Business analytics:

The aim of these analytics would be to return to a business a list of users who would appreciate their experience at the said restaurant. This would enable the restaurant owners to target potentially loyal customers.

b) Analysis:

User analytics:

Let U and R be the user and restaurant given as input, respectively.

The following pseudo-code explains the analysis being applied to the filtered input data to predict the likelihood of U liking R.

```
for every Restaurant r ∈ R
  let Y be the set of users who have reviewed r
  for every User u_1 ∈ Y
     compute s_index of User u_1 with User u_2
     compute the review_weight of User u_1 for Restaurant r
     store the s_index and the review_weight
  end for
  normalize the s_index of User u_2 with every other user that has been stored
  compute l_index for Restaurant r based on the normalized s_index and review_weights
end for

sort R based on the l_index
```

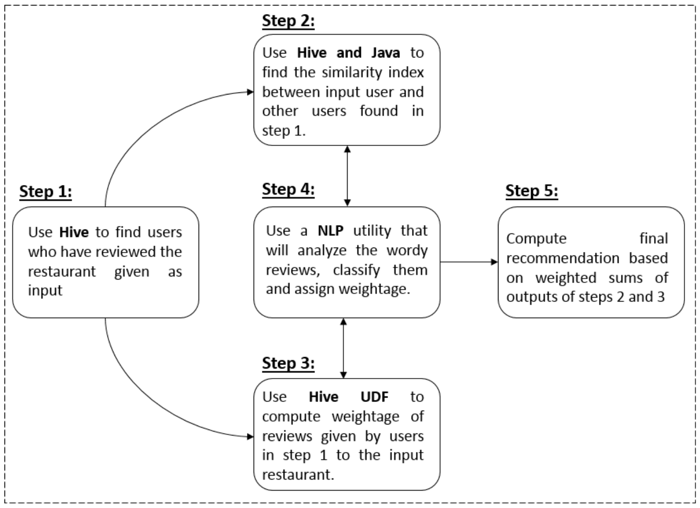
Business analytics:

Let R be the restaurant provided as input and U be the set of all users.

```
for every user u_1 ∈ U
  let Y be the set of users who have reviewed R
  for every User u_1 ∈ Y
     compute s_index of User u_1 with User u_2
     compute the review_weight of User u_1 for Restaurant r
     store the s_index and the review_weight
  end for
     normalize the s_index of User U with every other user that has been stored
     compute l_index of User U for Restaurant R based on the normalized l_index and review-weights
end for

sort U based on the l_index
```

The following diagram gives a pictorial representation of the analysis process being implemented by us. We will be using MapReduce and Hive extensively to perform the analysis.


Step 1:
Use Hive to find users who have reviewed the restaurant given as input
Step 2:
Use Hive and Java to find the similarity index between input user and other users found in step 1.
Step 3:
Use Hive UDF to compute weightage of reviews given by users in step 1 to the input restaurant.
Step 4:
Use a NLP utility that will analyze the wordy reviews, classify them and assign weightage.
Step 5:
Compute final recommendation based on weighted sums of outputs of steps 2 and 3

V. Results

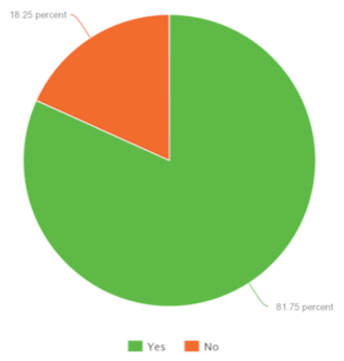
We tested our system for several users and following are some of the results that we got:

- **Result 1:**

  User: Brad

  Business: Mon Ami Gabi

  Recommendation: 81.75 %

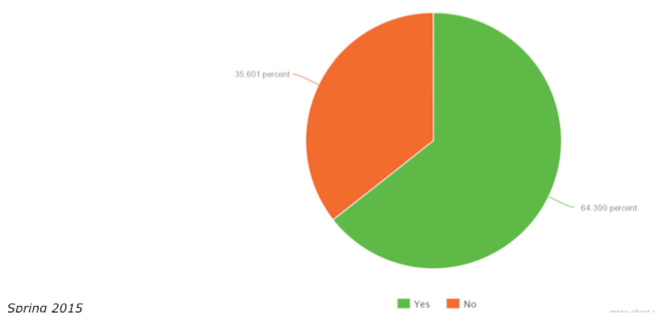


- **Result 2:**

  User: Westie

  Business: Serendipity

  Recommendation: 64.399 %

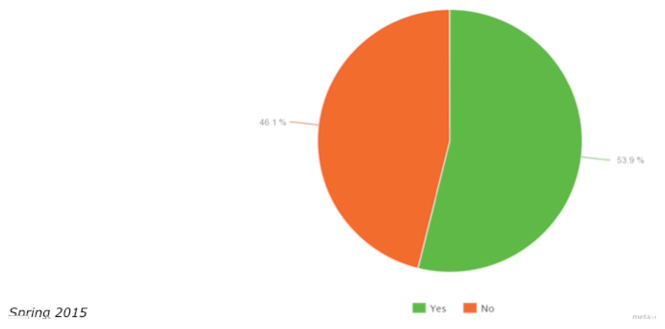


- **Result 3:**

  User: Jennifer

  Business: Arriba Mexican Grill

  Recommendation: 53.90%



VI. FUTURE WORK

Our project work can further be improved by the addition of a well-designed user interface. Also, our system is currently designed to be used on top of the logic that is already used by Yelp, giving more relevant recommendations in cases where other users with tastes similar to that of our user are found. We can further improve our project to show the result from Yelp in cases where similar users are not found in the system. Once hosted as a website, we can ask users to rate our recommendations in order to adjust the weight assignments used in our project for further fine tuning our output.

VII. CONCLUSION

We can run this analytic to predict whether a user will like a restaurant or not. Rather than just relying on star ratings or reading through several textual reviews, this recommendation system can be used to get an overall idea about the likelihood of a user liking a business.

A business can also take advantage of our approach to advertise itself to a set of users who are more likely to like the restaurant. We can run this analytic for all the users from the area where a business is located and if the prediction is in favor of the business for a user, that user is our target customer. More positive reviews mean more revenue.

ACKNOWLEDGMENTS

We are thankful to Prof. Suzanne Macintosh for her guidance and support during the course of this project, Amazon for the AWS voucher which gave us access to high performance computing environment and Yelp.com for putting up their extensive datasets for free online.

REFERENCES

[1] T. White. Hadoop: The Definitive Guide. O'Reilly Media Inc., Sebastopol, CA, May 2012.

[2] Hive - A Warehousing Solution Over a Map-Reduce Framework Authors: Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff and Raghotham Murthy from Facebook Data Infrastructure Team

[3] http://www.yelp.com/dataset challenge

[4] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. In proceedings of 6th Symposium on Operating Systems Design and Implemenation, 2004.

[5] Data Mining Yelp Data - Predicting rating stars from review text - http://www3.cs.stonybrook.edu/~cnaik/files/data_mining_report.pdf

[6] Amazon.com recommendations : Item-to-Item Collaborative Filtering by Greg Linden, Brent Smith, and Jeremy York

[7] S. Vinodhini, V. Rajalakshmi, B. Govindarajulu: Building Personalised Recommendation System With Big Data and Hadoop Mapreduce