



RESTAURANTS RECOMMENDATION SYSTEM

Team #5 - Wisdom

COMP-SCI 5560 (SS16): Knowledge Discovery and Management

First project report

Team Members

Samaa Gazzaz (9), Pooja Shekhar (38), Chen Wang (44), Dayu Wang (45)

Motivation

The difficulty of a recommendation system in artificial intelligence is the observation that even though the behavior of the user has been regarded as the most crucial clue and requirement to find out the best-fitting results, the actual results recommended may still not be satisfactory to the user, and sometimes the user itself does have a vague picture of what he/she really wants the results to be. Therefore, recommendation system is actually a complex human activity imitating system, including natural language processing, human interpretation (e.g. what females would probably like, where an artist would like to go to, etc.), and big data treatment.

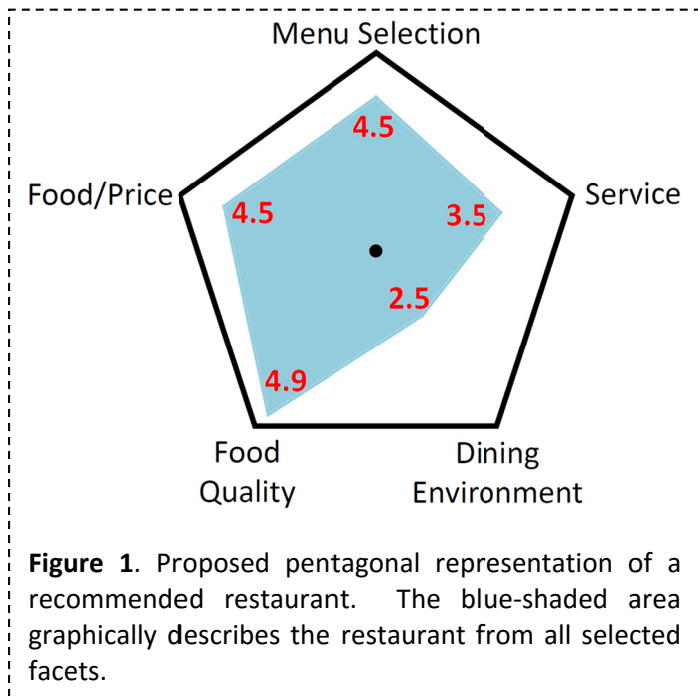
In this class, we decided to build up a small recommendation system from scratch, with the application of those existing services that smartly parse natural language input, and the existing database of targeted items that can be used for us to understand and analyze.

Finding a good restaurant to try new cuisines or even for a fun night out can be a really hard decision to make, especially with all the new restaurants competing to acquire new customers. Using restaurant recommendation apps or systems is very useful, especially when you are starving and would like to taste some local food in a top local restaurant in a city that you had never been there before. Some important features to current restaurant recommendation systems, however, are still missing. For example, when you are using most of the existing restaurant recommendation systems, recommended restaurants almost never change when searching from the same location—usually your home. This can be a problem inasmuch as the same user would want to find new restaurants when searching for a restaurant in two consecutive days. In addition, users usually have to read a lot of reviews in order to decide whether they found a restaurant that meets their needs, because in most cases, just looking at the star rating of a restaurant is an insufficient evidence to believe that the restaurant is on the top level.

Based on such observations mentioned above, we would like to build up a novel restaurant recommendation system that includes the following new features which can distinguish our system from the current existing restaurant recommendation systems.

First, natural language processing is smartly applied in our system. Let's take the Yelp system as an example. In Yelp, all users are encouraged to input a star-rate (1 to be the worst and 5 to be the best with 0.5 increment), and a text review with photographs. The problem is that it is not very scientific if the star-rating is used as the description of the level of the restaurant in all facets. Probably a user gives a 5 to a restaurant because the food was outstanding, but if another user really cares about the service quality of the restaurant, then the 5-star is misleading and meaningless to the second user. Therefore, all people have reached a consensus that customers' reviews are very significant.

In our system, we use natural language processing approaches to correctly understand customers' review and put the results into the overall rating of the restaurant. If most customers said that the service is not as splendid as the food in a restaurant, then such information will be summarized and negative evaluation will be applied to the "service" part of the overall restaurant recommendation score.



Second, we are inspired from the way that how social media describes a soccer player, they draw a pentagon and use each corner to represent the ability of the player in a specific facet, e.g. speed, attack, defense, shooting, and stability. Similarly, we can use this very awesome method to present our recommendation result (See Figure 1). This not only summarizes and visualizes the tremendous amount of big data, but also provides a friendly way to let the user know about the restaurant without looking at tons of tedious previous customers' reviews.

Third, behavior history of the user is important to our system, primarily because restaurant is special that no one would like to eat in the same restaurant every day. Therefore, not only do we keep the behavior history of a user with one week, but also we take comparison between the systematic recommendation list and the user's history, in order to decide whether recommendation index (which restaurant comes out first) should be altered.

Finally, based on the time limit of the entire project, we have decided to generate unidirectional server-client architecture for our system, in which two recommendation engines are built in the server and client sides. The engine in the server is the data training engine, which keeps the data being real-time once it is turned on. Also, it trains the formatted data into several relating indicators that represents the recommendation list of the restaurants. The server in the client part is the similarity analyzing engine. It compares the information sent

from the server with the user's requirements of the recommendation, and with the user's history of behavior. The engine thus rearranges the weighing score of those indicators and generates the final recommendation list of restaurants.

Based on the distinguishable designation mentioned above, we are quite confident about our restaurant recommendation system to be successful and practical, and useful. More details will be given in the following sections.

Objective

There are multiple necessary features that are still very much needed in currently existing restaurant recommendation systems. The objective of creating our own restaurant recommendation system is to try and fill the gap by providing those missing features. This system should be able to recommend new restaurants to users that match their needs even if users ask for recommendation from the same location and preferences as a previous search; recommendations should be different. In addition, this system will categorize and rank different features of restaurant using sentiment analysis of user review. This way, new users don't have to read all the reviews. Instead, they can just check a visual summary of the features and their ranking for each restaurant.

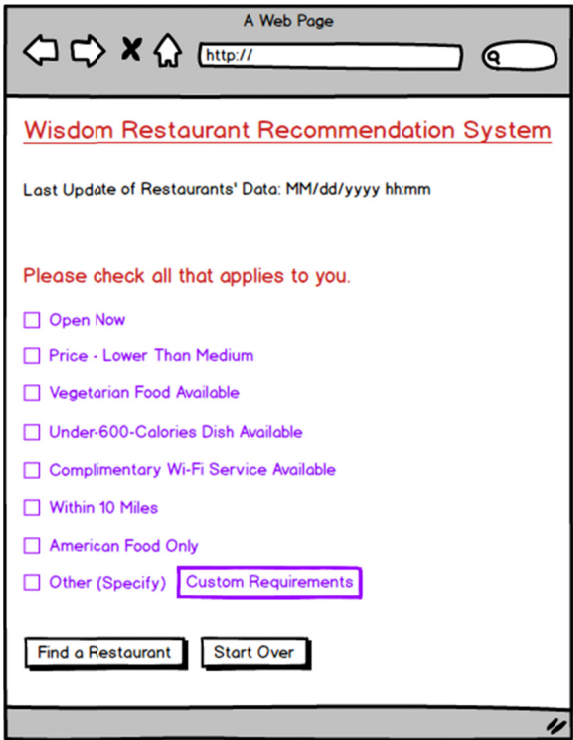
Our main objective in this class project is to learn, study, and understand the basic concepts of natural language processing, as well as all the implementation details of the usage of Apache Spark NLP APIs. We are not setting our goal to beat the current existing restaurant recommendation system (e.g. Yelp), since those mature system is build up by a large group of people and by a long-term research focused on this area. Instead, we would like people to distinguish our recommendation system by the novel features we are developing in our system. First, there is not any restaurant recommendation system can be easily found through the internet which applies Spark NLP. This could be a very proceeding research topic in the area of application of natural language processing. Second, it is very novel to use the "polygonal representation" to describe restaurant and present it to the user (see Figure 1). It is definitely clearer, faster, more readable, and more understandable to a user by reading pictures, instead of looking at tedious, bazaar text. Finally, server-client programming is another important thing that all of us would like to study in this project. Since restaurant recommendation is a service, so it is inevitable to apply server-client method of programming.

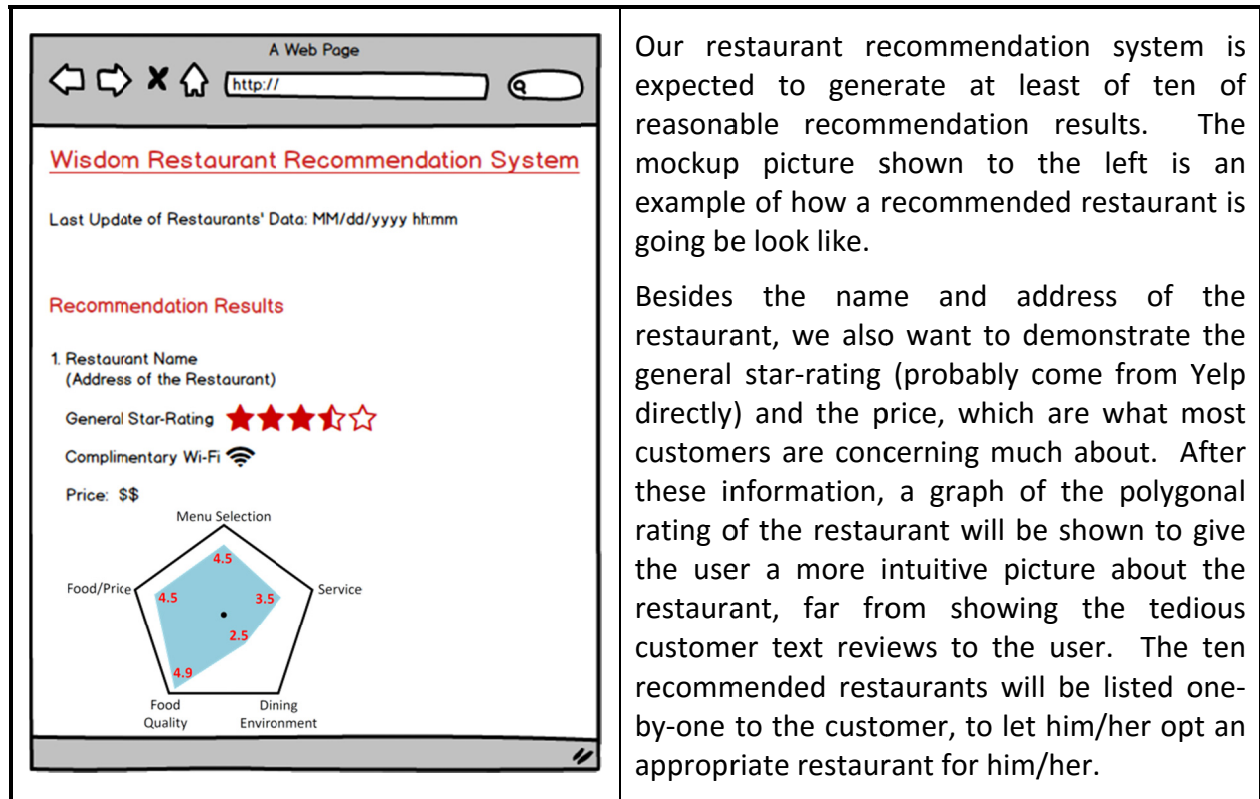
Expected Outcome

When we finish developing this system, it should be able to provide restaurant recommendation to users. Users should be able to specify different kind of preferences when asking for a recommendation such as location, type, closing time...etc. Recommended systems should be updated if the user asks for recommendation two consecutive days; not showing the same recommendations. In addition, ranking of different ranking of each recommended restaurant should be provided depending on analyzing the customer reviews. For example, in addition to five-star ratings, a restaurant should also have a ranking of cleanliness, noise level, friendliness and other features users might have mentioned in the reviews.

Visually, the outcomes of our restaurant recommendation system should look like the following (see Table 2) by the end of the semester. We have applied the Balsamiq Mockup Software to mimic the final user interfaces.

Table 2. Mockup and Explanation of the System Input/Output from the User's Interface.

Balsamiq Mockup	Explanation
	<p>The server keeps the restaurant's data to be always real-time, by updating the data in background every 5 minutes. In the user interface, the user can check some common requirements we set up for him/her, as well as customize his/her own requirements for the restaurants.</p> <p>It is also necessary to mention here that we assume that the user has already logged in. So we do not need to care about the login system, since in such small amount of time, we would like to focus on the main engine of the recommendation system to order to deeply understand the procedure of natural language processing and the recommendation system flow. Hence, the user interface will be made simply but clearly. Our final interface might look similar to the Balsamiq Mockup shown to the left.</p>



Our restaurant recommendation system is expected to generate at least of ten of reasonable recommendation results. The mockup picture shown to the left is an example of how a recommended restaurant is going be look like.

Besides the name and address of the restaurant, we also want to demonstrate the general star-rating (probably come from Yelp directly) and the price, which are what most customers are concerning much about. After these information, a graph of the polygonal rating of the restaurant will be shown to give the user a more intuitive picture about the restaurant, far from showing the tedious customer text reviews to the user. The ten recommended restaurants will be listed one-by-one to the customer, to let him/her opt an appropriate restaurant for him/her.

Project Domain

This project is going to be a recommendation system. Namely, it will be specialized in recommending restaurants to users. In addition, the specific domain is involved in restaurants and food chains. Not only will it provide the recommendation, our system will make it easy for users to decide where to eat by summarizing existing reviews by categorizing and raking different features mentioned in the reviews.

Data Collection

Dataset Used:

For this project, we will be using Yelp restaurant information. Yelp is a hybrid app that provides users with the ability to search for restaurant by specifying their preferences. Then, Yelp provides a list of restaurants which could be reorganized and sorted by rating, nearest location or other preferences. For efficiency purposes, we extracted a sample dataset to enable fast testing of the code. Sample datasets are included in the GitHub repository.

Collection Process:

Yelp provides developers with valuable resources and accessibility to static and dynamic datasets. For the static dataset, we will be using the challenge dataset provided by Yelp for developers that want to use their information in research and join a challenge at the same time. That data set is 2.2GB in size and includes 2.2M reviews and 591K tips by 552K users for 77K businesses. It can be accessed and downloaded through the following URL: https://www.yelp.com/dataset_challenge

Moreover, Yelp also provides a well-documented API for developers to access the real-time data featured on Yelp. This API provides access to search over 50 million local businesses from 32 countries. The API URL is: <https://www.yelp.com/developers/>

Tasks and Features

During this iteration, we implemented multiple features that will yield a better data/model training for our recommendation system. Those features enable better understanding of the dataset collected. Below, we discuss those features, process, input used and corresponding output:

Cognitive Services/Intelligent Services:

- User input - city name
- Service used – AlchemyApi
- Feature - Lists all the restaurants in the city with sentiment of the reviews given by users at Yelp.
- Mechanism – We have two json datasets available from Yelp- business.json and reviews.json. Former contains business_id, location, category of business and business name. We select business_id for restaurant category for the city inputted by the user. In the reviews.json, we search for reviews of each restaurant with that particular business_id, perform sentiment analysis.
- Output – Result has two columns restaurant names in the user selected location and the overall sentiment corresponding to each. Result dataset is uploaded github.

NLP:

For Natural Language Processing, our input would be the text part of user reviews. Users provide very insightful reviews on Yelp and being able to use those reviews in recommending a restaurant is a huge part of our project. In order to process the reviews, we first start by treating the JSON file and extracting only the needed information, that being the text attribute. Next, having the reviews file ready, we execute:

- Tokenization: in order to separate different words
- Lemmatization: this is an important step for later features (see TF-IDF)
- POS tagging: identified different parts of speech in users' reviews
- Entity recognition: this helps in identifying specific location users might be mentioning

All the information acquired from NLP is then provided as a text file for future references. This could be accessed on our GitHub repository.

Information Extraction/Retrieval Technology:

In order to be able to provide the insightful recommendation we aim for, we need to be able to understand reviews and extract information that will help the user make the decision. Using TF-IDF is very useful in this case as it will help us identify the most frequent words used by users in the reviews about a specific restaurant, enabling the identification of the most important features of that destination.

For the best results, we first lemmatize the reviews using NLP (see above). We also remove stop words. Next, we use N-gram method to identify N-gram phrases. These steps will provide the best input possible for our TF-IDF function which in part will return the most frequent words. The output of TF-IDF is the list of the most frequent words in the reviews. All described functionality is implemented and included in our GitHub repository.

Implementation Specification

Software Architecture:

The architecture of our recommendation system is generated based on the following assumptions, which comes from people's common sense when they are performing a web search.

First, the behavior of the user during the current searching is regarded as the most important reference to let the machine know what he/she wants. For example, if the user chooses to eat in a restaurant that has complimentary Wi-Fi service, then even though a restaurant has a 5-star rating, it will not appear on the top page of the recommendation list if it does not provide free internet access.

Second, the recommendation system looks at if any co-occurrence appears while it is comparing the user's search with the restaurants' stored data. Significant indicators will be computed out that attempt to imitate the user's taste.

Third, in order to satisfy text-based search, mathematical similarity comparing algorithm is underlined in the searching engine which enables the similarity connection amongst those indicator results.

Finally, the user only searches for a restaurant to eat, not for any other purpose (e.g. business).

The high-level architecture of the recommendation system is shown in Figure 3.

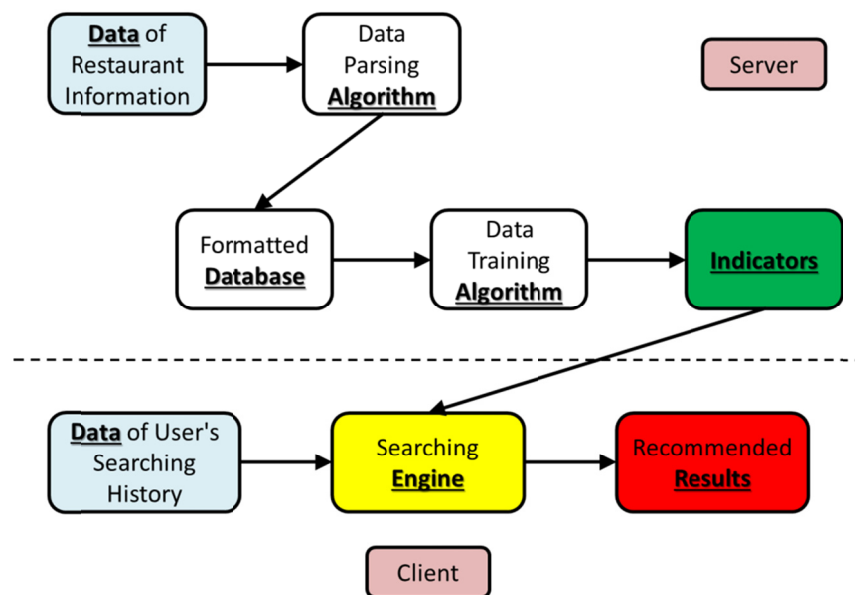


Figure 3. High-level architecture of restaurant recommendation system. Arrows indicate the flow direction of data. The black border of each component represents the interface that allows data exchange.

From the architecture in Figure 3, we can see that when the user performs a searching for a restaurant to eat, the recommendation engine generates the recommendation list from mainly two sources of data—restaurant information and user's searching history. The raw data of restaurants' information undergoes a process of parsing, and is then transformed to the format (json format) that will be used in the downstream component. This procedure does not contain data training.

Then, the data is trained using a certain algorithm to create weighted indicators that may be used for recommendation. The indicators will be compared with the user's searching history to generate the final recommendation list by mathematical similarity comparison.

UML Models:

Our architectural design is the unidirectional server-client architecture, which means only the server sends the information of restaurants to the client and a client never sends anything to the server. We chose this model since it is very clear to everyone and easy to implement. Our current UML diagram is shown in Figure 4.

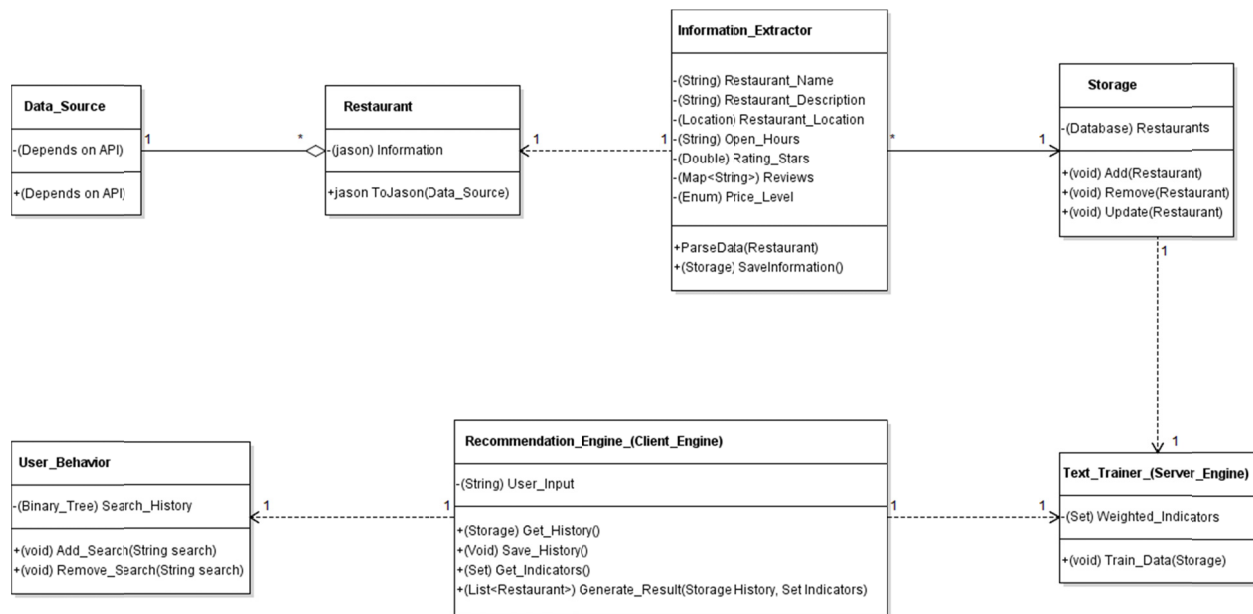


Figure 4. UML diagram for the restaurant recommendation system.

Workflow Diagrams:

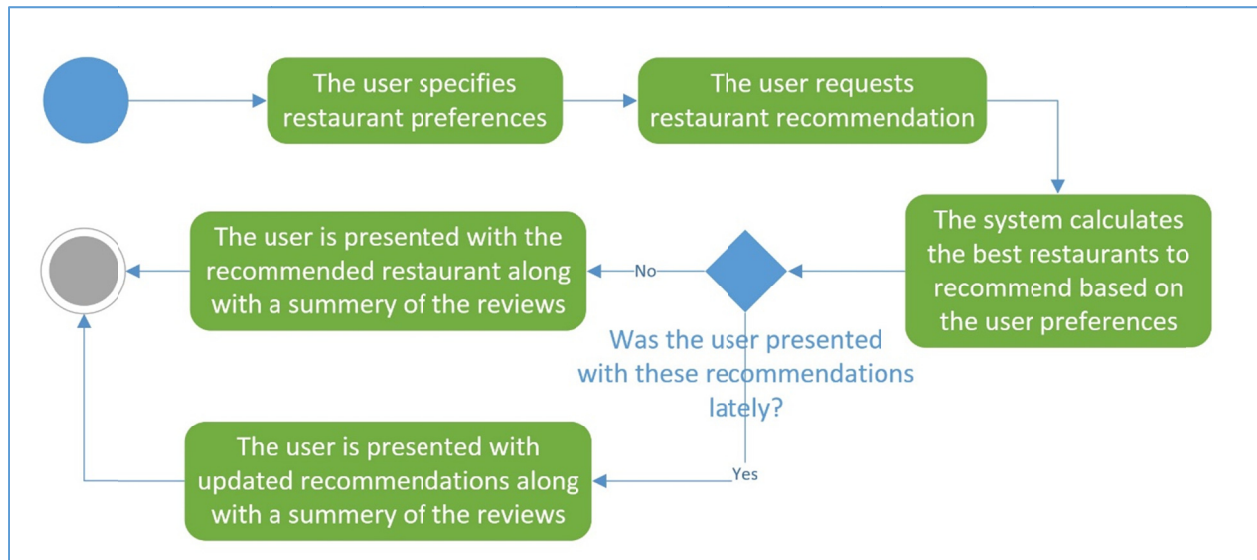


Figure 5. High-level workflow of the client side restaurant recommendation system.

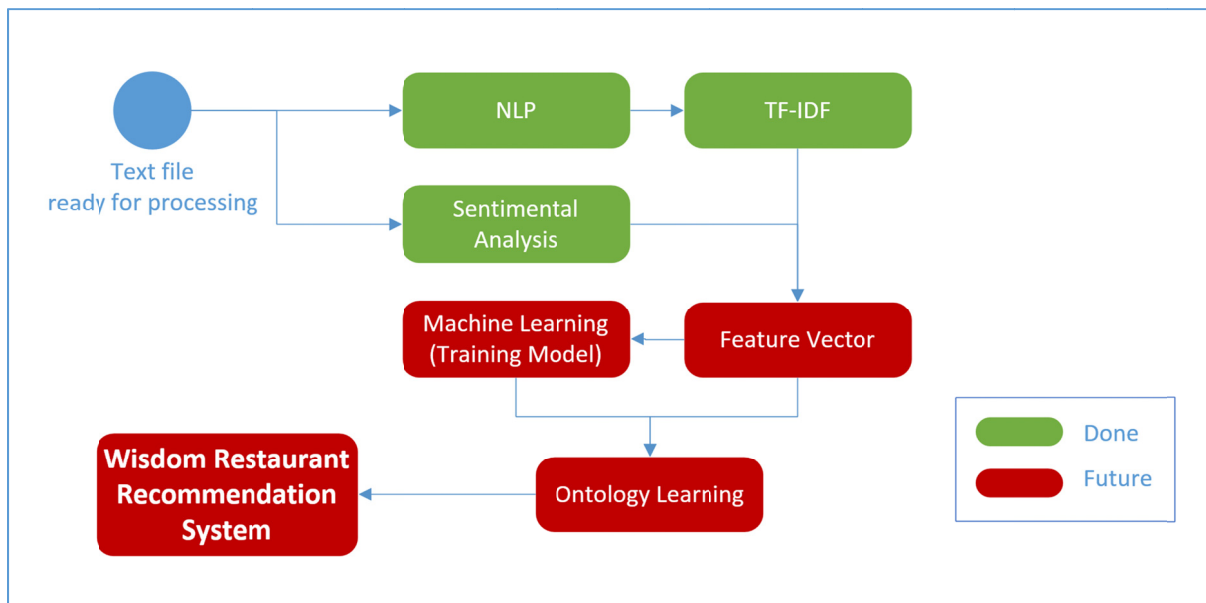


Figure 6. Workflow of the project progress

Existing Services:

Apache Mahout

Apache Mahout can build programming environment for a machine learning application which has scalable performance, offer a platform for the machine learning system, work on the filtering, clustering and classification. For the intelligent machine, the Apache Mahout can make it study and improve quickly and easily. For our recommendation system, Apache Mahout can help the system get learning and improve itself, for example, at the first time, the choices have been given by the system maybe not the best one for the customers, but through training the system constantly, it will feedback a better and better recommendation. So it's necessary to the restaurant recommendation system to realize the machine learning.

Elastic Search

Elasticsearch is a searching service developed by Java and based on Lucene. It can process scalable search and real time search. Elasticsearch is a common open source analyzing and searching engine that aim at analyzing logs, real time monitoring the applications and analyzing click streams. Through the supervisor console, we can easily set and deploy the Elasticsearch cluster. Amazon Elasticsearch Services is a kind of managed services which could help you deploy, operate and expand the cloud of AWS (Amazon Web Services). Amazon Elasticsearch Services deploy all the sources for the cluster and boot up it. This service detects and replaces the Elasticsearch node automatically. For our restaurant recommendation system, the Elasticsearch can provide a scalable search solution.

Google HighCharts

Highcharts is an application that we can use it to create interactive charts. And it can be used for the SaaS projects, web applications, intranets, and websites. And it's an open source application. For our restaurant recommendation system, the output will have a charts or graph that show the related restaurant's comments, ranking, and preference. All of them need a charts or graph tool to be implemented. And the google Highcharts is a good choice to help us to implement the system.

New Services/Features to be Implemented:

Our restaurant recommendation system will provide a new service that can show the preference of the restaurant, we collect the comments on the Yelp and processing those comments through NLP, and get a new style preference of the restaurant, so that we can accord the history of the customer searched, give a recommendation for the specific person. For example, we can have a pentagon to show some details about the restaurant, each of the corner can signify a specific character about the restaurant, such as price, food quality, service, dining environment, and the quantity of the menu choices. For our system, we can recommend restaurants to customers based on the preference graph. And of course, an average restaurant (no preference) with high quality is also a good choice.

Our system will recommend the restaurant which have no preference with a high quality or high quality with little preference. Because if a restaurant has a strong preference, it must be having one or more low characters in the graph. We can see it in the Figure 7 clearly. The first graph is the best condition, have the highest quality. But between the second graph and the third graph, we always recommend the third restaurant. Even though both of them have the similar average quality, but we will consider the Service and Dining Environment of the third restaurant is too low to have a good dinner. But if a restaurant just has a little preference, we will recommend it as the history of the customer's choices.

So this service can help the customer make a better choose for their preference, and also can help the customer remove some restaurants with a normal average score but have some obvious deficiencies in one or more aspects.

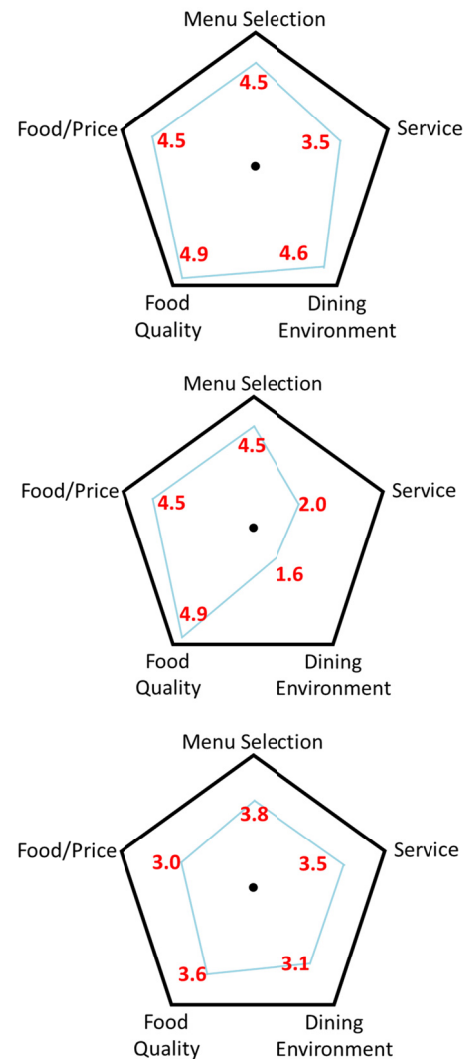


Figure 7. The different conditions of the preference of the restaurant recommendation system.

Project Management

Contribution of Each Member:

Member	Contribution		
Samaa Gazzaz	Documentation	Other	
	<ul style="list-style-type: none">• Motivation• Objective• Expected Outcome• Project Domain• Project Management• Workflow Diagrams	<ul style="list-style-type: none">• <u>Dataset collection</u>• Collecting Static Datasets• Sample Dataset Extraction (Secondary)<ul style="list-style-type: none">• <u>Input/output</u>• Information Extraction/Retrieval Technology• NLP (Implementation and documentation)<ul style="list-style-type: none">○ Code is included in the repository	
Pooja Shekhar	<ul style="list-style-type: none">• Intelligent Services- Sentiment Analysis of Reviews collected from Yelp<ul style="list-style-type: none">○ Code available in the repository• Sample Dataset extraction (Primary)		
Chen Wang	Documentation	Other	
	<u>Existing Services used</u> (Primary contribution) <u>New Services/Features Implemented</u> (Primary contribution)	<u>Put forward the preference theory</u> (Primary contribution) <u>Mock-up Pictures</u> (Secondary contribution)	
Dayu Wang	Project Report	Diagrams	Other
	<u>Motivation</u> (Secondary contribution) <u>Objective</u> (Secondary contribution) <u>Expected Outcome</u> (Secondary contribution)	<u>System Architecture</u> (Primary contribution) <u>System UML Diagram</u> (Primary contribution) <u>Mock-up Pictures</u> (Primary contribution)	<u>Proposal of “Polygonal Data Training Method”</u> (Primary contribution)

Version Control/Screenshots:

For this project, we used GitHub as the main version control tool. The whole project, in addition to documentation, is up on: <https://github.com/SamaaG/WisdomRecSys>

Concerns/Issues:

Although there are plenty of currently available recommendation systems that could serve as reference for this project, the time restraint imposes a huge concern on whether our system could be implemented on time. Since this is the first exposure for all our team members with recommendation systems, the time needed to get up to speed is going to take away from actual implementation time.

Future Work:

For the next iteration, we plan to implement at least one feature for the system. In addition, preparing the dataset should be done by the next report.