The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

A CUSTOMIZED REAL TIME RESTAURANT RECOMMENDATION SYSTEM

A Thesis in

Computer Science and Engineering

by

Rui Jiang

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

December 2015

The thesis of Rui Jiang was reviewed and approved* by the following:

Sencun Zhu

Associate Professor of Computer Science and Engineering

Thesis Advisor

Wang-chien Lee

Associate Professor of Computer Science and Engineering

Mahmut Kandemir

Professor of Computer Science and Engineering

Graduate Program Chair of Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

ABSTRACT

According to a Search Engine Land survey conducted in 2014, 88% of consumers have read online reviews to determine the quality of a local business and 72% consumers say that positive reviews make them trust a local business more. What is a local business? It can be a medical practice, a restaurant, or a gym. The result of this survey shows the important role of reviews in deciding to choose a local business. Among various types of business, restaurant is the most searched category by users in Yelp. However, oftentimes people just search a restaurant by using word “restaurant”, while the word “restaurant” means differently to different individuals. For an Asian, it can mean a “Chinese restaurant” or “Thai restaurant”. How to correctly interpret search requests based on people’s preference is a challenge. Building a machine-learning model based on activity history of a registered user can solve this problem. The activity histories used by this thesis are reviews and ratings from users.

This thesis introduces a data processing pipeline, which uses reviews from registered users to generate a machine-learning model for each business and each registered user. This thesis also defines an architecture, which uses the generated machine-learning models to support real-time personalized recommendations for restaurant searching.

In order to find a good machine-learning model, we have tried several collaborating filtering methods to predict ratings between restaurants and users. The methods we have implemented are Slope One, k-Nearest Neighbors algorithm, and multiclass SVM classification. Our evaluation shows that the multiclass SVM classification method outperforms the other methods. For rating prediction, we compare user-based and item-based collaborative filtering algorithms.

Finally, architecture is given to support the building of a real-time recommendation service.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to all the people who have helped me to go through the graduate program during the last three years.

I would first thank the Department of Computer Science and Engineering at the Pennsylvania State University to give me this precious opportunity to enroll as a master student.

I would thank all my friends who have studied and discussed class and computer technology with me.

I also feel very grateful to Dr. Sencun Zhu for guidance writing this thesis. He not only gave me valuable suggestions to solve problems, but also provided constructive opinions for the process of analysis, the way of thinking and writing. I would not have made it without Dr. Zhu's persistent encouragement and tremendous help.

I also want to thank Dr. Wang-Chien Lee for joining the committee and giving me suggestions on improving the thesis.

At last, I want to acknowledge the encouragement of my family all these years, and thank them for their support and help.

Chapter 1 Introduction

1.1 Introduction

Local businesses have a big influence on people's daily lives. What is a local business? It can be a medical practice, a restaurant, a tailor, and a gym. Finding a good local business was traditionally done by word-of-mouth. After the new trend of social networking emerged, people began to realize a different way to find trustworthy local businesses.

A Search Engine Land survey in 2012 showed a positive shift in consumer trust and appreciation of online reviews.

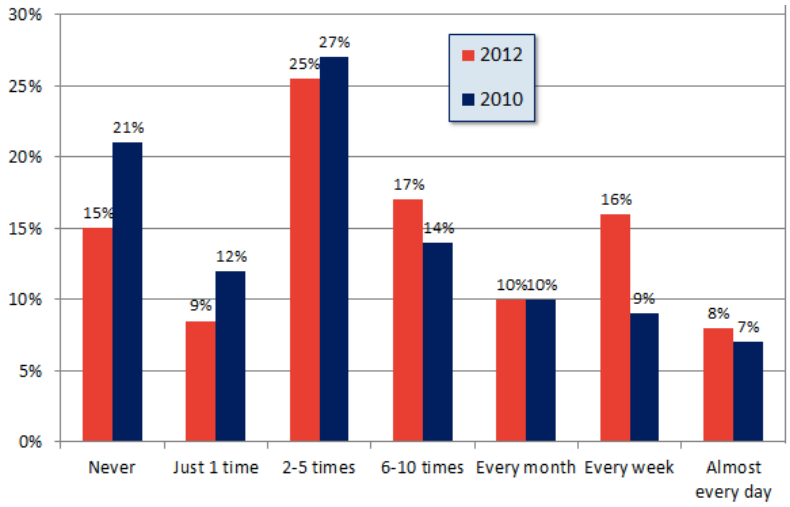

Figure 1. How many times consumers used the Internet to find a local business in 2012. (From Search Engine Land)

Figure 1 is the data from Search Engine Land's survey about how many consumers used the Internet to find a local business. Clearly, more people began using the Internet to

find local businesses. In 2012, 85% of consumers used the Internet to find local business. According to Figure 1, from 2010 to 2012, people who used the Internet to search local businesses weekly increased from 9% to 16%. The ratio of consumers who read 2~10 reviews to choose a business increased from 58% to 65%.

Survey Question	2012	2010
Consumers regularly use online review to determine which local business to use.	27%	22%
Never used online review	24%	29%
Consumer read 2~10 review to choose a business	65%	58%
Consumer read more than 20 reviews to choose a business	7%	12%
Consumer trusts a business which has positive only reviews	58%	55%
Consumer is more likely to use a local business if they have positive reviews	52%	50%
Consumer trusts online reviews as much as personal recommendations	72%	67%

Figure 2. Usage of review comparison between 2010 and 2012.

From Figure 2, the data showed an increased usage of reviews from 2010 to 2012. More consumers are reading online reviews, and more people began to trust reviews to find a local service.

In 2014, this trend continued.

Figure 3. 88% of consumers have read reviews to determine the quality of a local business

Figure 4. The confidence level of reviews from consumers.

Another great finding is 88% of consumers have read reviews to determine the quality of a local business. Also 88% of consumers say they trust online reviews as much as personal recommendations.

Among the many websites which host reviews for local businesses, Yelp is the most popular choice and the most influential local review site. This was based upon a survey conducted by Yelp and Nielson in 2014.

Before the creation of Yelp, people went to a restaurant usually by hearing from friends, or occasionally seeing an interesting place. Now, since websites like Yelp have been built, people can check information before directly visiting a restaurant. The available information can be reviews, ratings, price ranges and hours of operation. After evaluating prices, times and reviews, people make decisions about whether to go or not.

Although reviews provide valuable recommendations about restaurants, reading reviews for businesses can be time consuming. People go through several pages of recommendations published on Yelp just wanting to find a right restaurant, which has most potential to satisfy their personal appetite or their own food preference. Above that, when people visit a new place, move or travel, users have to go through the same search procedure again. This can be tedious and sometime annoying.

A problem we found is that Yelp cannot provide search results when people conduct their own searches.

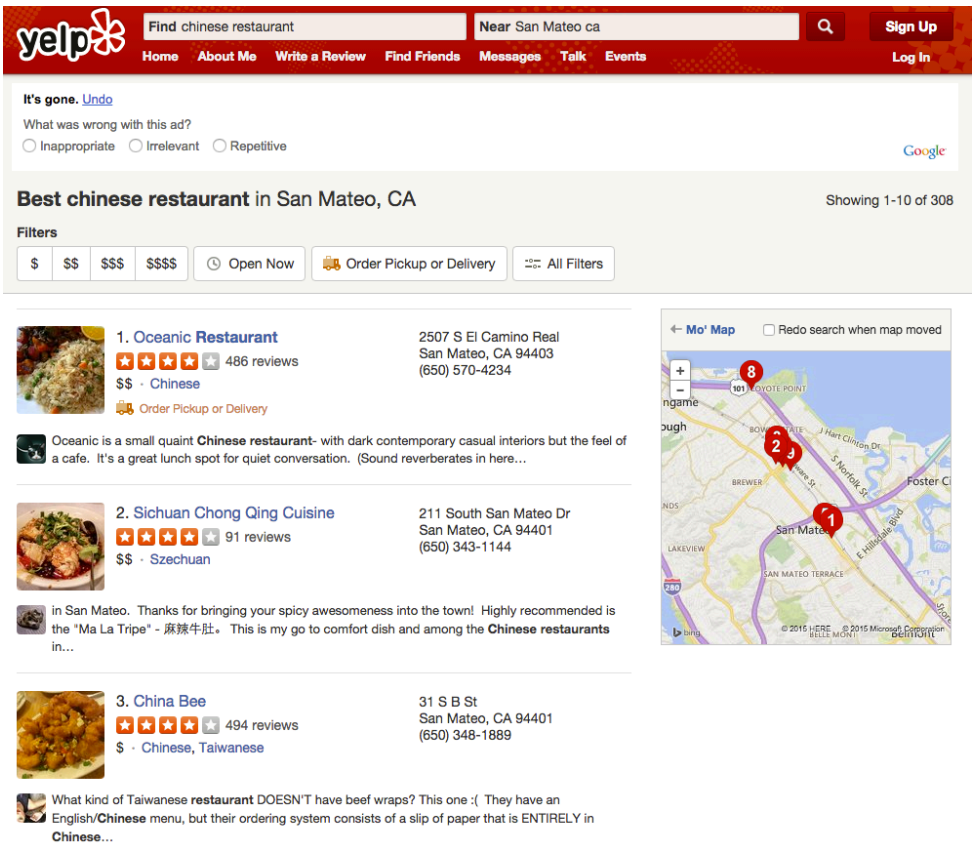

Figure 5. Search result of "Chinese Restaurant" in San Mateo CA neighborhood

Figure 3 shows the search results from Yelp for "Chinese Restaurant." But from this list, one famous Chinese Sichuan Style "Spicy Empire" is missing. We have to browse several pages to find it. For people who like Chinese Sichuan Food, "Restaurant" means "Chinese/ Asian style restaurant." If Yelp can understand this, that would be very helpful.

But how can this goal be achieved? Yelp needs to know people's preference. This not only indicates a login account, it also means Yelp needs to know a user's own taste. This understanding could be built from personal reviews, basically teaching Yelp to know people by training a machine-learning model on top of people's review data.

Another important requirement is the recommendations made by Yelp need to be in real-time. People change locations every day, and thus searching for restaurants or local businesses will happen in different locations.

In this thesis, we will explain how to use review data to build a real-time customized restaurant recommendation system.

Through a real-time customized recommendation system, recommendation for restaurants will be made to suit people's own preference and current locations. People will not need to browse several pages of search results to find a suitable business.

In this thesis, we will present a data process pipeline, which takes original reviews, extracts important words, builds a topic model, and trains a machine-learning model for each person or for each local business. The model will be stored in a database and be retrieved to provide recommendations in real-time.

Word extraction was done through a Natural Language Processing algorithm. Topic modeling was done through Latent Dirichlet Allocation (LDA).

In order to figure out the relationship between restaurants and users, several collaborating filtering methods were tested to predict ratings between restaurants and users, such as Slope One, k-Nearest Neighbors algorithm, and multiclass SVM classification. Our evaluation shows that the multiclass SVM classification method outperforms the other methods. We also compared user-based and item-based collaborative filtering algorithms. When a registered user doesn't have enough reviews to train a good personal machine-learning model, the item-based model can be used.

Based on a trained model, we designed and developed a database model to store user and restaurant modeling data, which can support a real-time recommendation service.

1.2 High Level View of System

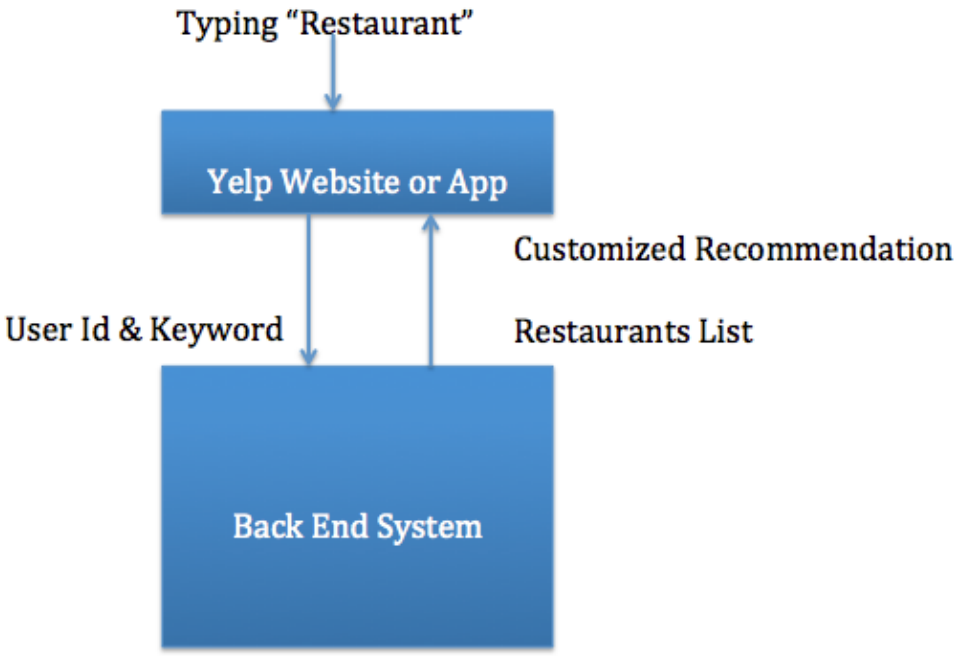

Figure 6. High Level View of Recommendation System

Figure 4 shows the workflow in high level.

1. User types "Restaurant" as key word
2. Yelp website or mobile app passed User ID & Keyword to back end system
3. Back end system returns a list of restaurants which are based on user's preference

All of the above steps will happen in few hundredths of a millisecond.

1.3 Data Process Pipeline & Workflow

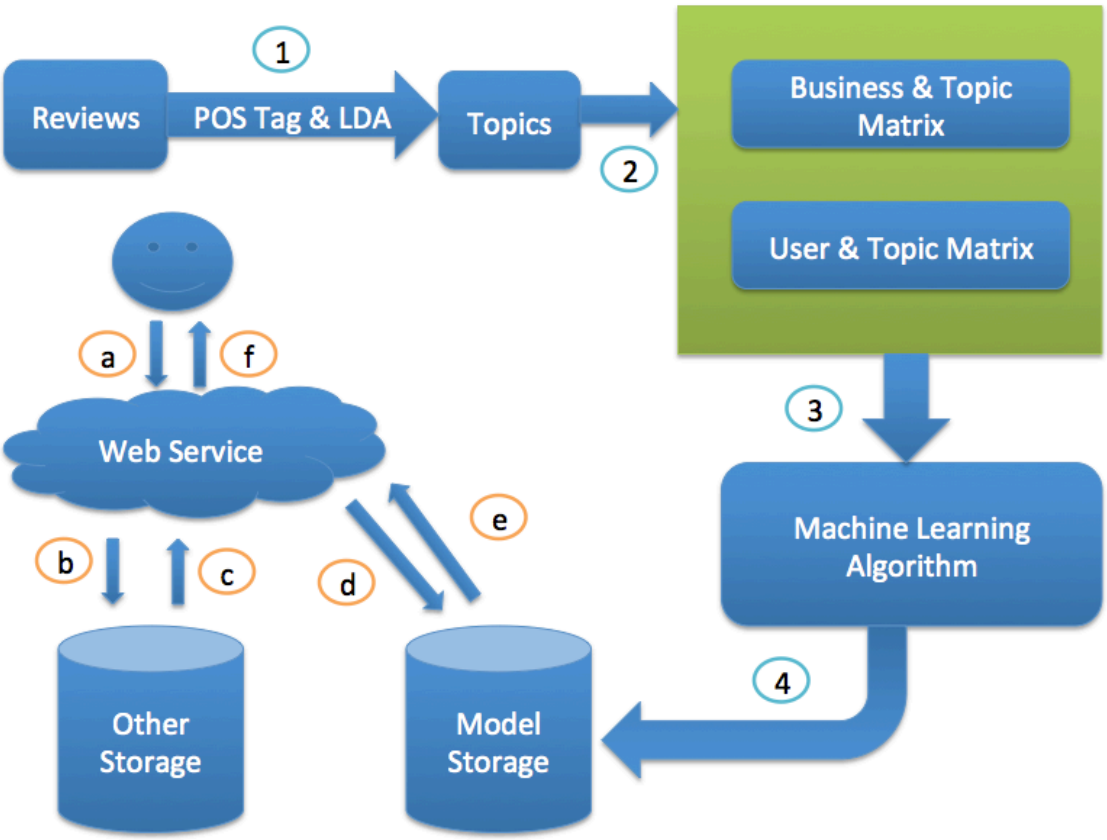

Figure 7. Data Process Pipeline and Workflow

In Figure 5, steps 1 through 4 belong to the data process pipeline. Steps a through f belong to workflow, which handles a user's request to generate a customized recommendation.

These are the steps in the data process pipeline.

Step 1: Extract POS tagging algorithm on review. Tagged words will be extracted and passed to Latent Dirichlet Allocation algorithm to do topic modeling.

Step 2: Based on topics extracted from LDA in step 1, a relationship among business, user, and topic can be built. The relationship will be stored as a matrix.

Step 3: The relationship matrix will be used to train the machine-learning model.

Step 4: After the model is trained, it is stored in model storage. Model storage can be a database.

Workflow (from a to f) is showing how user will get recommended restaurants.

Step a: Registered user types keyword search for restaurants.

Step b: Web service sends a request to other storage asking for a list of restaurants.

Step c: Storage sends back a list of restaurants.

Step d: Web Service retrieves user ID to model storage service asking for user model.

Step e: Model storage sends response back with trained machine learning for this registered user.

Step f: Web service returns a list of restaurants which are ranked by using a machine-learning model that has been trained based on the user's own data / reviews.

1.4 Recommendation System

In the Communications of the ACM 1997, Paul Resnick and Hal R. Varian introduced the concept of recommendation in [19]. In 2011, Recommender Systems Handbook [13] was published with a wide introduction about research paper and applications used by recommendation system. It not only has methods used by

recommendation system, but also contains how to design and evaluate recommendation system.

Recommendation systems used in memory item-based collaborative filtering algorithm were introduced in [14][15]. One system used content-based method is introduced in [16]. A scalable online collaborative filtering method is mentioned in [17] by Google to support news personalization. Collaborative filtering is the most popular method used in recommendation systems. Xiaoyuan and Taghi provided a great survey of it in [18].

1.5 Contribution

This thesis experiments with a complete data-processing pipeline, which uses reviews to predict whether a user will like a certain restaurant. It adopts a topic-modeling method to extract hidden factors from millions of reviews, and uses a multiclass classification SVM with an item-based and user-based collaborative filter method to predict the rating between user and restaurant.

It also introduces the database model to support real-time recommendation.

Chapter 2 Dataset

2.1 Introduction

Building a customized real-time recommendation system, data is the first prerequisite on how to build a machine-learning model. Yelp provided a test dataset for academic purposes, which allows people to run analysis and test new ideas on their data. This thesis is based on Yelp's dataset.

The following section introduces the structure of Yelp's data.

2.1.1 Dataset Overview

Here is the number of different elements in the dataset:

42,153 businesses (more than 30% of the businesses are restaurants)

320,002 business attributes

31,617 check-in sets

252,898 users (more than 73% of users have visited restaurants)

955,999 edge social graph (user's relationship)

403,210 tips

1,125,458 reviews (more than 70% of reviews are restaurants related)

From the summary, it is clear that the dataset is big enough to do analysis and restaurants, among different local businesses, are the major business for reviews.

This thesis will only focus on data related to restaurants. This thesis will only use the following data: business, business attributes, users, and reviews. Other data, such as social graph, are not in this thesis' research area.

2.2 Dataset Preprocessing

Before building a machine-learning model, the original data from Yelp needed to be stored and also needed to be easily retrieved for calculation. Let's look at what data format it is, and how it can be stored.

2.2.1 Dataset Format

Only three major data formats are listed here: business, users, and review. Yelp's dataset is all stored in text files in Json format. Data in this format cannot be analyzed in a scalable way. Thus, Yelp's data needed to be loaded into a database before any analysis could begin. Choices for databases included No-SQL Mango DB or relational database.

We chose a traditional relational database as storage for Yelp's data, which can easily support join operations.

2.2.1.1 Business

Yelp's Json Format for Business:

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': { (day_of_week): { 'open': (HH:MM), 'close': (HH:MM) }, ... },
    'attributes': { (attribute_name): (attribute_value), ... },
}
```

Table 1 is the database table model for Business. Here only the important fields in business are listed.

Column	Type	Comments
type	String	Value as “business”.
business_id	String	22 characters unique business id.
name	String	
stars	Real	Stars based from user’s review.

Table 1: Business Table

One thing that needs to be mentioned is attributes for restaurant. Here is an example of attributes: *{"Accepts Credit Cards": true, "Wi-Fi": "free", "Price Range": 3}*

It usually contains information like Wi-Fi status, credit card acceptance, open hours, price range, or parking conditions. Through attributes, people can get a more comprehensive understanding of a business.

2.2.1.2 User

Yelp’s Json Format for User:

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)], 'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': { (compliment_type): (num_compliments_of_this_type), ... },
    'fans': (num_fans),
}
```

Table 2 & 3 are the database table models for User to support the above data format:

Column	Type	Comments
type	string	Value as “user”.
user_id	string	22-character unique user id
name	string	
Stars	real	Stars based on user’s review

Table 2: User Table

Friendship is used to represent many-to-many relationship between two users.

Column	Type	Comments
User1	String	First user of friendship. Foreign key from User table.
User2	String	Second user of friendship. Foreign key from User table.

Table 3: Friendship Table

2.2.1.3 Review

Yelp's Json Format for Review:

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

Since this table has more than 1 million records, and the analysis usually requires querying this table frequently by user_id, business_id, adding indexes to these two columns would speed up the query dramatically.

Column	Type	Comments
type	string	Value as "review".
review_id	string	22-character unique id for review.
user_id	string	Foreign key to User table, 22 characters.
business_id	string	Foreign key to Business table, 22 characters.
stars	real	User's rating of business in this review.
text	string	User's text review for the business.

Table 4: Review Table

2.2.2 Data Preprocessing

In this part, data preprocessing would read lines from the Json file and translate data into corresponding records in the database.

The following database model is for storing business, user and review data.

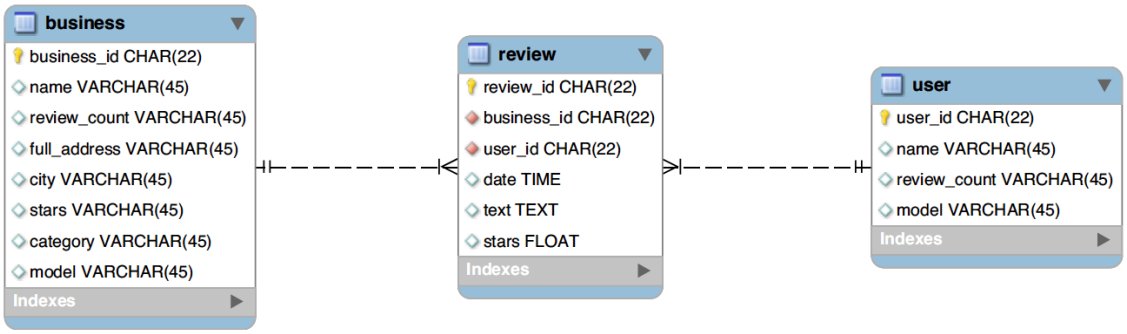

Figure 8. Database Schema

Chapter 3 Topic Modeling and Feature Extraction

3.1 Introduction

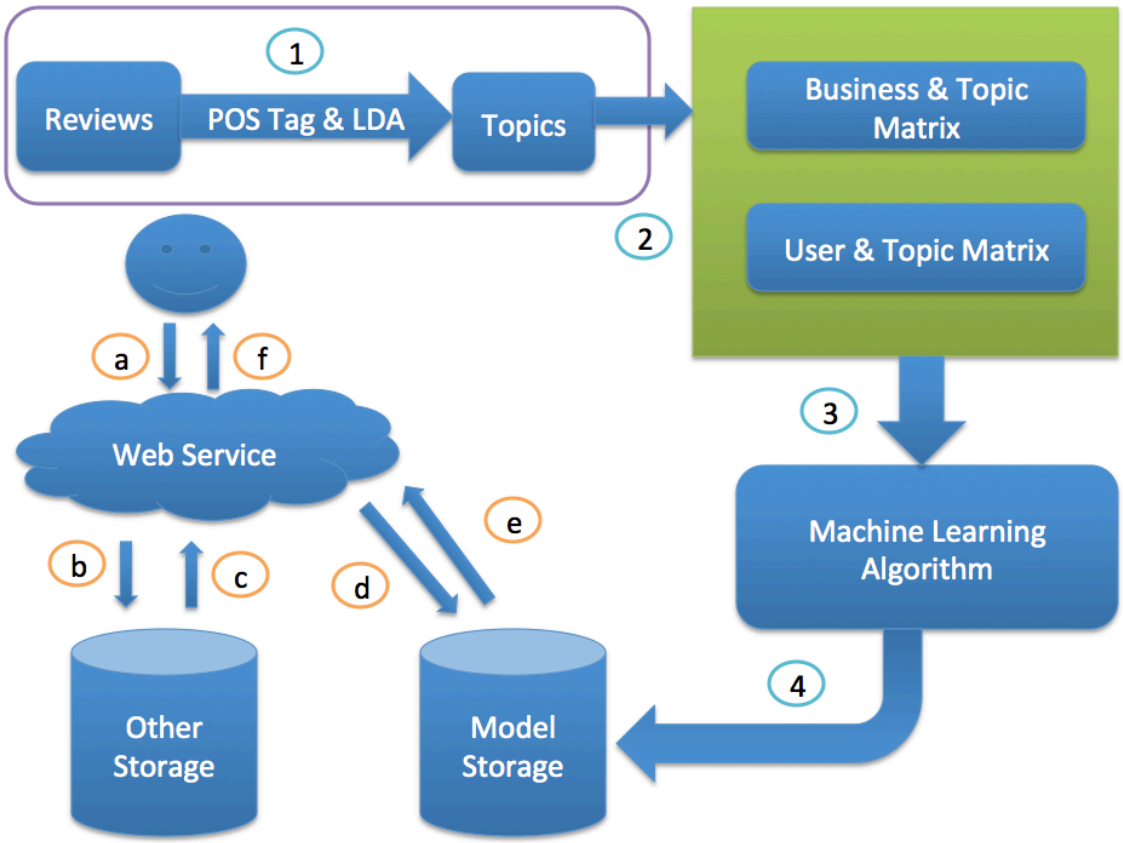

Figure 9. Step 1: Topic Modeling in data process pipeline

Topic modeling is the first step in the data process pipeline.

In order to predict whether a user will like a restaurant, taste preferences of users and attributes of the restaurant must be uncovered. For example, in order to tell whether a user would like a steakhouse, knowing the user's past experience and feedback to a similar place would be extremely helpful. To discover information in these dimensions,

users' feedback is required. In the context of Yelp, this information comes in the form of reviews and ratings.

Review text is disordered data. People can get the general idea of a review through reading it, but digesting a review by an automatic algorithm and retrieving what kind of food is provided by the restaurant, how quality of service at the restaurant is, and whether the restaurant is a good place to go for a group of people, is a challenge. A methodology called "topic modeling" is needed to solve this problem.

Topic modeling is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. For example, a restaurant serving classic American breakfast food would have a lot of reviews with words such as "egg," "coffee," and "bacon." For a bar providing alcohol and entertainment, the words "beer," "bar tender," and "drink" would appear more frequently among its reviews than non-bar restaurants.

Before we are able to do topic modeling, review sentences need to be transformed into words, which can be used as input. Extracting words is a task belonging to natural language processing.

The following figure shows the details inside step 1:

Figure 10. Topic Modeling Procedure

3.2 Natural Language Processing

Let's take a review as an example. The following is one review:

The food and service is exceptional, and the location and setting are superb. If you enjoy people watching and absorbing the ambience of the Las Vegas pastiche, then you must have Lunch or Dinner on the Terrace of Mon Ami Gabi.

Based on this review, we know that having lunch or dinner is a pretty good choice at this restaurant; the food, service and location are also superb here. How can we make a computer understand this? This is a challenge that natural language processing is facing. It's hard for an algorithm to know whether the food, service or location is good or not, but it is possible for it to distinguish that "food," "service," and "location" are nouns. This mechanism is called "grammatical tagging" or "word-category disambiguation."

Grammatical tagging is the process of marking up a word in a text as corresponding to a particular part of speech (POS), based on both definitions, as well as its context. I used a very popular tag called Penn Tree Bank POS Tag [3].

Here is the table defined POS tag type for words:

Number	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential *there*
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular

15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb

Table 5: Penn TreeBank POS Tag

After going through POS tagging, the previous review sample will be transformed into following format:

```
[('the', 'DT'), ('food', 'NN'), ('and', 'CC'), ('service', 'NN'), ('is', 'VBZ'), ('exceptional', 'JJ'), (',', ','), ('and', 'CC'), ('the', 'DT'), ('location', 'NN'), ('and', 'CC'), ('setting', 'VBG'), ('are', 'VBP'), ('superb', 'RB'), ('.', '.')]
[('if', 'IN'), ('you', 'PRP'), ('enjoy', 'VBP'), ('people', 'NNS'), ('watching', 'VBG'), ('and', 'CC'), ('absorbing', 'VBG'), ('the', 'DT'), ('ambience', 'NN'), ('of', 'IN'), ('the', 'DT'), ('las', 'NNS'), ('vegas', 'VBZ'), ('pastiche', 'PRP'), (',', ','), ('then', 'RB'), ('you', 'PRP'), ('must', 'MD'), ('have', 'VB'), ('lunch', 'NN'), ('or', 'CC'), ('dinner', 'NN'), ('on', 'IN'), ('the', 'DT'), ('terrace', 'NN'), ('of', 'IN'), ('mon', 'NN'), ('ami', 'NN'), ('gabi', 'NN'), ('.', '.')]
```

The food and service is exceptional, and the location and setting are superb. If you enjoy people watching and absorbing the ambience of the Las Vegas pastiche, then you must have Lunch or Dinner on the Terrace of Mon Ami Gabi.

Penn TreeBank POS Tagging

[('the', 'DT'), ('food', 'NN'), ('and', 'CC'), ('service', 'NN'), ('is', 'VBZ'), ('exceptional', 'JJ'), (',', ','), ('and', 'CC'), ('the', 'DT'), ('location', 'NN'), ('and', 'CC'), ('setting', 'VBG'), ('are', 'VBP'), ('superb', 'RB'), ('.', '.')]
[('if', 'IN'), ('you', 'PRP'), ('enjoy', 'VBP'), ('people', 'NNS'), ('watching', 'VBG'), ('and', 'CC'), ('absorbing', 'VBG'), ('the', 'DT'), ('ambience', 'NN'), ('of', 'IN'), ('the', 'DT'), ('las', 'NNS'), ('vegas', 'VBZ'), ('pastiche', 'PRP'), (',', ','), ('then', 'RB'), ('you', 'PRP'), ('must', 'MD'), ('have', 'VB'), ('lunch', 'NN'), ('or', 'CC'), ('dinner', 'NN'), ('on', 'IN'), ('the', 'DT'), ('terrace', 'NN'), ('of', 'IN'), ('mon', 'NN'), ('ami', 'NN'), ('gabi', 'NN'), ('.', '.')]

Figure 11. POS Tagging Result

POS tags have more than 30 tags. Here, we only care about nouns, since nouns have the most relevance to what people are talking about.

3.3 Topic Modeling

Topic modeling is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. For example, a restaurant serving classic American breakfast foods would have a lot of reviews on word of "egg," "coffee," and "bacon." For a bar providing alcohol and entertainment, words in reviews will have more "beer," "bar tender," and "drink."

Let's look at one of the simplest topic modeling method: bag-of-words.

3.3.1 Bag-of-Words

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order, but keeping multiplicity. The bag-of-words model is commonly used in methods of document classification, where the (frequency of) occurrence of each word is used as a feature for training a classifier.

Take one restaurant as an example: A restaurant named "Mon Ami Gabi," a French Steakhouse serving both breakfast and brunch. It has more than 4,000 reviews. Extract words from its review, and display them using word cloud (a display method for bag-of-words), and we get Figure 12:

Figure 12. Bag-of-words for a French steakhouse.

From the above picture, looking at the words bigger in shape, we can see "French," "good," "food," "steak," "service," and "great." Based on these words, it looks like this

restaurant is pretty good, and serves steak. After checking the actual rating of this restaurant from the Yelp website, its four-star rating confirmed our understanding of this restaurant; it serves steak and breakfast, and people are generally very satisfied by their services and food.

However, bag-of-words only uses the frequency or occurrence of each word as features to train a classifier. This can make the topic contains a long list of words. Recently, much attention has been given to generative probabilistic representations of the data that reduce description length and reveal inter- or intra-document statistical structure.

In this thesis, We adopt the Latent Dirichlet Allocation (LDA) [1] as our topic modeling method for review texts.

3.3.2 Latent Dirichlet Allocation

In natural language processing, latent Dirichlet allocation (LDA) [1] is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. In LDA, each document may be viewed as a mixture of various topics. The assumption is that documents are created by choosing some topics in a particular mixture. Words are drawn randomly from those topics. You throw these in a bag and you have your document. Order does not matter, and LDA seeks to reverse this process.

For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.

3.3.2.1 Intuition Behind Latent Dirichlet Allocation LDA

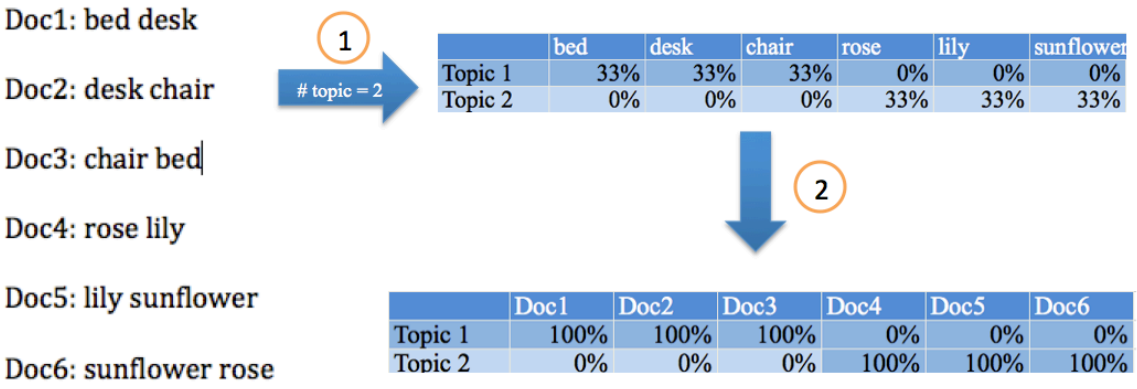

Figure 13. Intuition behind Latent Dirichlet Allocation

Suppose we have 6 documents. Each document only contains 2 words, as shown in the left side of the figure above. Clearly, there are 2 topics. Based on the word distribution, we can see what the possibility of each word is related to a topic. Also, for each document, we can get the chance about how much a document can be related to a topic.

After running the LDA model, we will get output from both step 1 and step 2.

3.3.2.2 Latent Dirichlet Allocation (LDA) Model

LDA is an unsupervised machine learning method to extract topics in a corpus and assign distributions of these topics over each document (as well as distributions of words over topics). LDA associates each document $d \in D$ with a K-dimensional topic distribution θ_d, which contains the part of words in d that is mentioned each topic. The probability of words in the document d related with topic k is $\theta_{d,k}$.

Each topic k has a corresponding word distribution ϕ_k, which explains the probability that a particular word is linked to topic k.

The final model includes word distribution for each topic ϕ_k, topic distribution for each document θ_d, a topic assignments for each word $z_{d,j}$.

Here is the notation and symbols for LDA model:

Symbol	Description
w_i	A word defined as an item from vocabulary indexed by $\{1, \dots V\}$.
N_d	Number of words in document d.
d	Document, a sequence of N words denoted by $\boldsymbol{w} = (w_1, w_2, \dots, w_N)$, where w_n is the nth word in the sequence.
D	A corpus is a collection of M document denoted by $D = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_M\}$.
θ_d	K-dimensional Dirichlet distribution on document d.
$\theta_{d,k}$	Words in the document d discuss topic k with probability $\theta_{d,k}$.
ϕ_k	Word distribution for topic k.
k	The number of topics is assumed to be fixed and known.
z_n	Topic $z_n \sim Multinomial(\theta)$.
α	α is a positive real number, and a parameter for Dirichlet process.
β	Word probabilities are parameterized by a $k \times V$ matrix β.
$\beta_{i,j}$	$\beta_{i,j} = p(w_j = 1 \mid z_i = 1)$ is treated as a fixed quantity that is to be estimated.

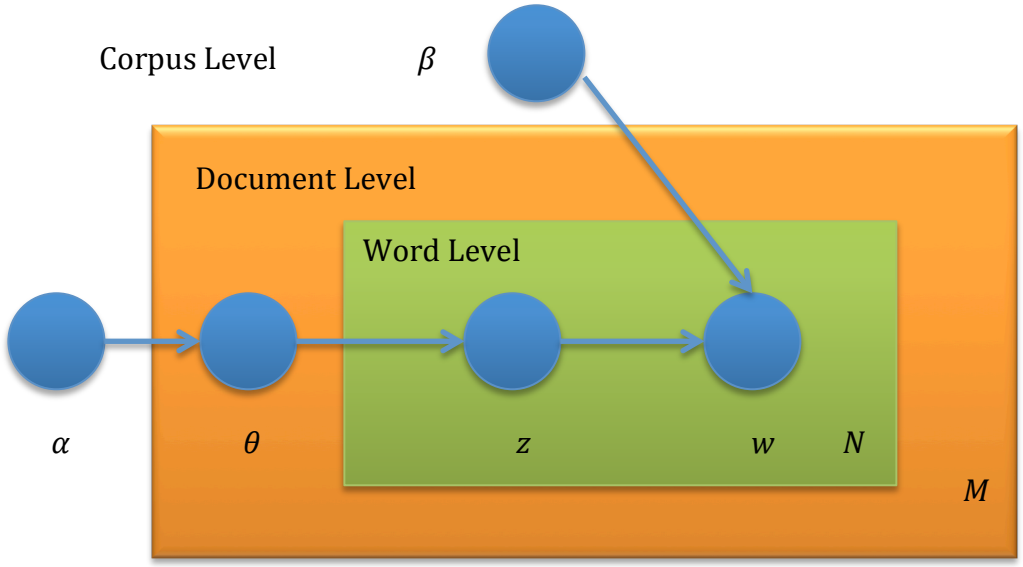

Figure 14. Graphical model representation of LDA.

The LDA model is represented as a probabilistic graphical model in Figure 14. Based on the figure, there are 3 levels to LDA representation.

Corpus level: α and β are assumed to be sampled once in the process of generating a corpus.

Document Level: θ_d is sampled per document.

Word Level: z_{dn} and w_{dn} are sampled once for each word in each document.

In LDA, the topic node is sampled repeatedly within the document, and documents can be associated with multiple topics.

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of N topics z, and a set of N words $\boldsymbol{w}$ are given by:

$$p(\theta, z, w\,|\alpha, \beta) = p(\theta|\alpha)\prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta)$$

$p(z_n|\theta)$ is simply θ_i for unique i such that $z_n^i = 1$.

Finally, the probability of a corpus is:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d\,|\alpha)\left(\prod_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn}, \beta)\right) d\theta_d$$

3.3.3 Number of Topics

When the number of topics and the number of keywords in each topic is specified, the LDA outputs a list of words with probabilities associated with each topic.

But we are still left with another problem: how to find a reasonable topic number among reviews. Based on [4], a natural number of topics with Latent Dirichlet Allocation can be found by using the Kullback-Leibler divergence function.

In probability theory and information theory, the Kullback- Leibler divergence (also called information divergence, relative entropy) is a non-symmetric measure of the difference between two probability distributions *P* and *Q*. Kullback–Leibler divergence of Q from P, denoted as $D_{KL}(P||Q)$ is a measure of the information lost when Q is used to approximate P.

Under this method, the LDA is viewed as a matrix factorization mechanism. Given a corpus *C* is split into two matrixes factor M_1 and M_2, such as

$$C_{d\times w} = M1_{d\times t}\times Q_{t\times w}$$

d ~ The number of documents present in the corpus.

w ~ The size of vocabulary.

t ~ The right number of topics chosen.

The quality of split is determined by t, the number of topics.

The following figure explains the relationship between the number of topics and its Kullback- Leibler divergence is non-symmetric. Here, WE calculated as

$$D_{KL}(P,Q) = \frac{1}{2}\left(D_{KL}(P||Q) + D_{KL}(Q||P) \right)$$

Then it is transformed into a symmetric measurement.

The following figure represents the relationship between number of topics and symmetric KL divergence on restaurant reviews.

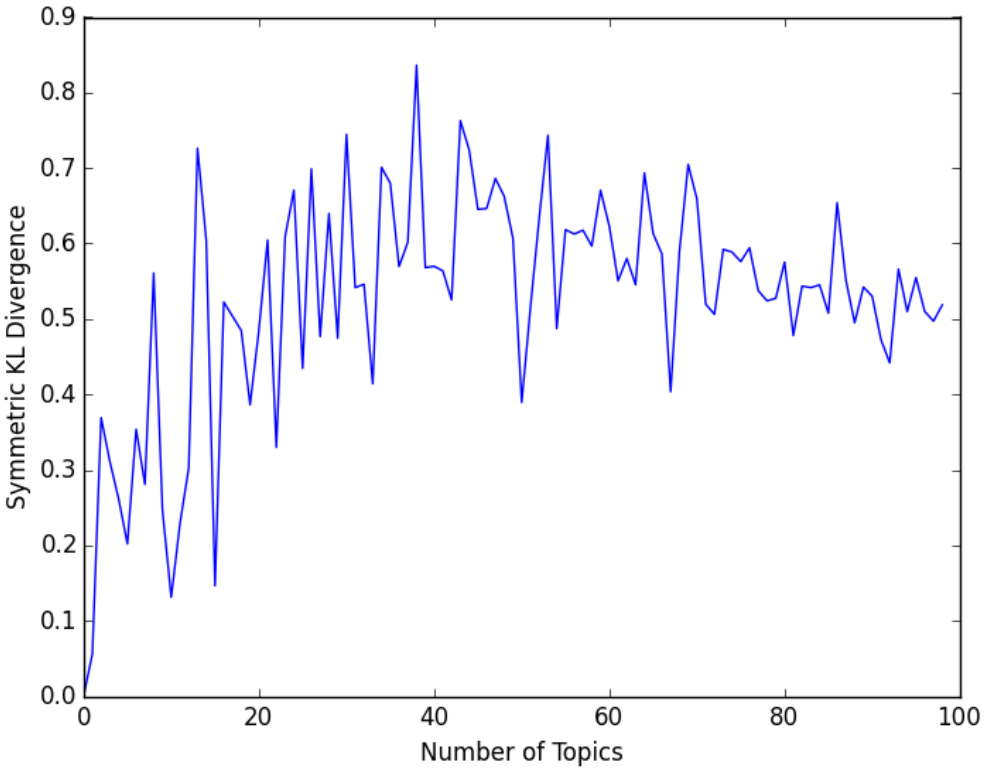

Figure 15. The relationship between number of topics and symmetric KL divergence on restaurant reviews.

From the graph, the optimal number of topics is relatively low. So WE chose the number as 22 and 50 to extract topics.

3.4 Topic Result

The following are the top 10 topics extracted from more than one million reviews linked with restaurants. To make it easier to understand by humans, a topic name has been added manually.

Take one topic as example:

0.102*egg + 0.079*breakfast + 0.047*bacon + 0.030*place + 0.030*brunch + 0.029*pancake + 0.027*morning + 0.025*toast + 0.022*sunday + 0.021*waffle

WE called this topic "breakfast," because "egg," "breakfast," and "pancake" are all food eaten in breakfast.

Number	Topic Name	Key Word and Its Possibility Associated with Corresponding Topic
1	Lunch dinner buffet	0.092*food + 0.066*lunch + 0.049*price + 0.034*place + 0.031*service + 0.029*buffet + 0.021*quality + 0.019*selection + 0.018*dinner + 0.017*option
2	Mexican food	0.095*taco + 0.062*food + 0.048*place + 0.027*bean + 0.023*mexican + 0.017*restaurant + 0.014*tortilla + 0.014*flavor + 0.014*rice + 0.010*street
3	Wine environment	0.034*wine + 0.026*restaurant + 0.023*nice + 0.022*patio + 0.020*menu + 0.015*area + 0.014*place + 0.014*service + 0.014*date + 0.013*staff
4	Asian food - Thai	0.033*soup + 0.029*rice + 0.027*bowl + 0.026*spicy + 0.025*dish + 0.025*sauce + 0.024*noodle + 0.022*thai + 0.021*flavor + 0.019*curry
5	American fast food	0.275*burger + 0.143*fry + 0.032*onion + 0.023*bun + 0.023*ring + 0.014*bacon + 0.014*park + 0.011*jam + 0.010*place + 0.010*design
6	Service	0.068*place + 0.060*happy + 0.056*food + 0.052*hour + 0.034*service + 0.033*drink + 0.031*awesome + 0.023*staff + 0.021*super + 0.020*price

7	Bar	0.078*bar + 0.056*beer + 0.040*place + 0.030*drink + 0.018*bartender + 0.017*night + 0.014*food + 0.014*time + 0.013*game + 0.013*selection
8	Location	0.089*vega + 0.052*strip + 0.023*station + 0.015*trip + 0.015*casino + 0.015*hotel + 0.013*mall + 0.013*airport + 0.010*location + 0.010*floor
9	Location & food	0.085*location + 0.046*tuna + 0.044*pho + 0.029*wrap + 0.026*chipotle + 0.019*pastor + 0.016*sprout + 0.016*honey + 0.015*store + 0.014*scottsdale
10	Sea food	0.051*shrimp + 0.046*ice + 0.037*crab + 0.035*cream + 0.030*chef + 0.029*dessert + 0.027*dish + 0.025*seafood + 0.019*appetizer + 0.018*menu

Chapter 4 Rating Prediction And Training

4.1 Introduction

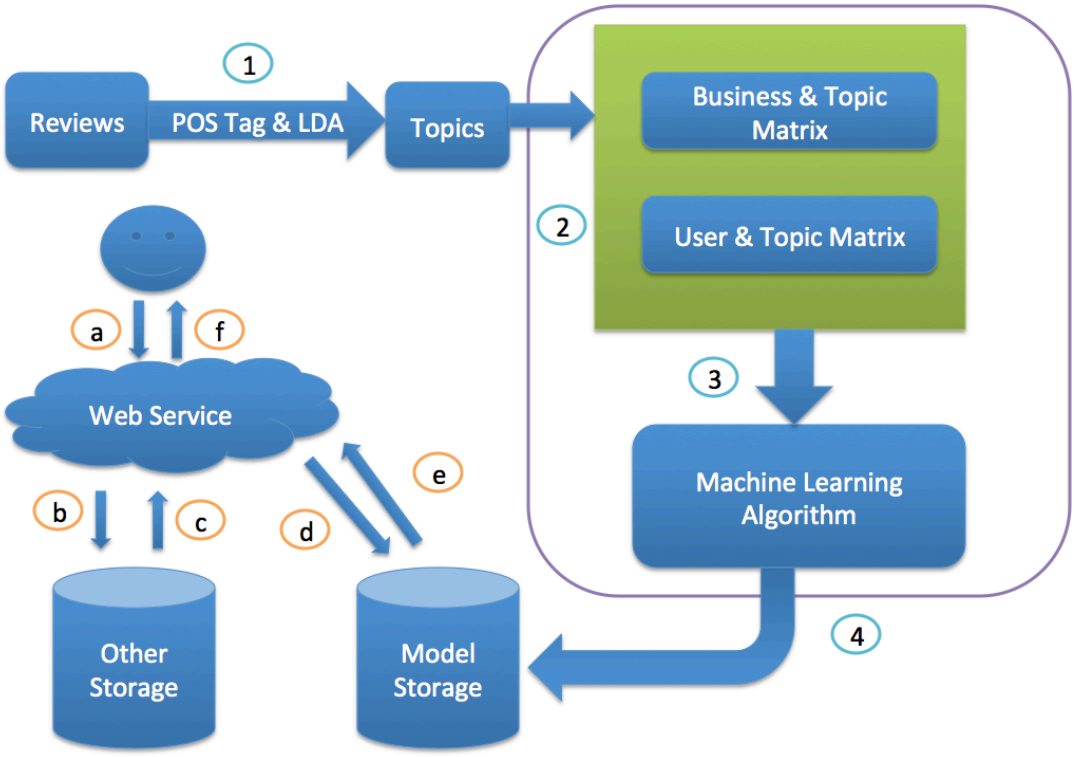

Figure 16. Chapter 4 will explain steps 2 and 3 of the data process pipeline.

Through topic modeling, topics were extracted. However, this is still not enough to run a machine-learning algorithm on it. Input data needed to be built in a certain format. This section will explain how to build the input data so that it's ready to be picked by machine learning algorithm.

Step 3 is for finding a feasible machine-learning algorithm. The major task for this chapter will explain what types of algorithm are available, which algorithm was used, how the performance is and which algorithm was chosen.

4.2 Rating

The rating has two aspects. One is rating for restaurants to topic, and the other is for users to topic.

The rating for restaurant to topic is the performance of a restaurant in the topic perspective. The rating for a user to a topic can be understood as how much the user cares about this topic.

From Yelp's dataset, we get topic ratings for each restaurant. Based on the relationship between topic and restaurant, and restaurant's actual rating, WE can calculate a rating or score for each topic for each restaurant.

But there is no such rating for a user to a topic. The relationship between user and topic can only be obtained from LDA modeling. In this section, WE explain how to get a rating for a user to a topic.

4.2.1 Topic of reviews

From Chapter 3, topics are extracted across all the restaurants' reviews. Let's assume the number of topics is *t*. The Topic model, gotten from Chapter 3, is TM.

Picking one $\mathrm{rev_k}$, passing selected POS tags as input to TM (Topic model), the output is the related topic $\mathrm{t_j}$ and the possibilities for this topic. It can be represented as following:

$$p_{rev_{k,j}} = TM(w_{rev_k})$$

w_{r_i} ~ The words from review rev_k.

t_j ~ The *j*th topic in *t* number of topics.

$p_{rev_{k,j}}$ ~ The probabilities between review rev_k and *j*th topic.

For each relationship between topic t_j and review rev_k, adopt review's rating $rating_{rev_k}$ as this relationship's rating ($rating_{rev_k,t_j}$). It represents as following:

$$review\_topic\_rating_{rev_k,t_j} = review\_rating_{rev_k}$$

4.2.2 Topic Rating of Items (Restaurants)

For each restaurant r_i, it has a set of reviews $Rev = \{rev_1, rev_2, \dots, rev_n\}$. Based on a set of review topic rating ($review\_topic\_rating_{rev_k,t_j}$), the rating between restaurant r_i and topic t_j is calculated as following:

$$restaurant\_topic\_rating_{r_i,t_j} = avg\left(\sum_{k=1}^{N} review\_topic\_rating_{rev_k,t_j}\right)$$

$$= avg\left(\sum_{k=1}^{N} review\_rating_{rev_k}\right)$$

For each topic t_j, it can calculate $restaurant\_topic\_rating_{r_i,t_j}$ by taking the average of $review\_rating_{rev_k}$ if the underlying review has link with topic t_j.

4.2.3 Topic with Users

Between topics and users, WE adopted the same logic from above, and applied it to users' data.

For each user u_i, it has a set of reviews $Rev = \{rev_1, rev_2, \dots, rev_n\}$. Based on a set of review topic ratings ($review\_topic\_rating_{rev_k,t_j}$), the rating between user u_i, and topic t_j is calculated as following:

$$user\_topic\_rating_{u_i,t_j} = avg\left(\sum_{k=1}^{N} review\_rating_{rev_k,t_j}\right)$$

$$= avg\left(\sum_{k=1}^{N} review\_rating_{rev_k}\right)$$

4.3 Collaborative Filtering

4.3.1 Introduction

After getting all the relationships among restaurants, users, and topics, we are ready to find a good machine-learning algorithm, train the model, and do prediction. A question comes. Which method or which category method should be used? Looking back at what problem this thesis tries to solve, the goal is to automatically recommend restaurants by using a personally customized feature. Collaborative filtering (CF) is a popular technique used by recommender system.

Collaborative filtering [5] has two types: a general and narrow.

In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.

Narrow collaborative filtering, is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from

many users (collaborating). In this thesis, WE have adopted narrow collaborative filtering.

The underlying assumption is that if a person *A* has the same opinion as a person *B* on an issue, *A* is more likely to have *B*'s opinion on a different issue *x* than to have the opinion on *x* of a person chosen randomly.

Another way to categorize collaborating filtering is based on the comparing object (also known as item). In this thesis, there are restaurants and users. Restaurants are treated as items.

When the filtering is performed on computing restaurant-to-restaurant similarities, this is called item-based collaborative filtering. Another way filtering is performed is on calculating user-to-user similarities, such as what kind of restaurant the user has been to and how it was rated; this is categorized as user-based collaborative filtering.

The following sections show how a good method was found, and will also show a comparison of the methods with which WE have experimented. It also compares the performance of item-based and user-based collaborative filtering.

4.3.2 K-Nearest Neighbors (k-NN)

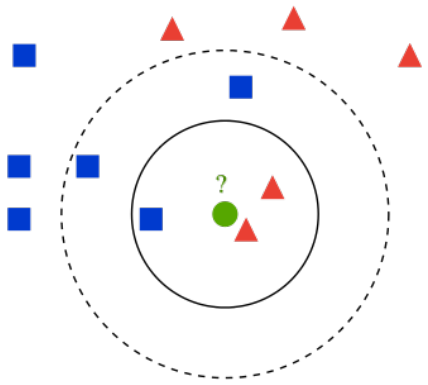

Figure 17. K-Nearest Neighbors Example

Among the simplest collaborative filtering machine-learning algorithms, the k-Nearest Neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space.

Take the above figure as an example. There are two classes; one is blue and the other is red. The green dot is an unknown sample. The way to categorize the green dot is by looking at its k-Nearest Neighbors to tell which category it is. If k=2, the green dot will be treated as red, since there are 2 red triangle very close. If k=3, the green dot will be judged as blue, because there are 3 blue squares close to it.

Both for classification and regression, the k-NN would weight the contributions of neighbors, so that the nearer neighbors contribute more to the average than the more distant ones.

For the k-NN method, user, restaurant, user to restaurant rating and relationship between user and topic are used. The k-NN method is running at a per restaurant level, another way of saying it is an item-based k-NN method.

The possible value for rating is limited as 1, 2, 3, 4 and 5, and k-NN treats these five values as five classes. Given a restaurant, an unvisited user, visited user relationship with topic, and rating from visited users, the output or prediction is the rating for the unvisited user.

4.3.3 Pearson Reference Scheme

The Pearson reference scheme [6] uses the Pearson correlation coefficient to predict the rating from user to item.

In statistics, the Pearson Product-Moment correlation coefficient is a measure of the linear correlation between two variables X and Y, giving a value between +1 and -1 inclusive, when 1 is total positive correlation, 0 is no correlation, and -1 is total negative correlation. The Pearson correlation coefficient is widely used in the sciences as a measure of the degree of linear dependence between two variables.

The prediction in the Pearson reference scheme for user u on item i is calculated in the following equation. The predictions depend implicitly on the training set χ.

$$P(u)_i = \bar{u} + \frac{\sum_{v \in S_i(\chi)} \gamma(u,v)(v_i - \bar{v})}{\sum_{v \in S_i(\chi)} |\gamma(u,v)|}$$

Notation:

u ~ The ratings from a given user, is called evaluation represented in an array.

χ ~ The set of all evaluations in the training set.

$S(u)$ ~ The subset of the set of items consisting of all those items which are rated in u.

$card(S)$ ~ The number of elements in a set S.

$\bar{u}$ ~ The average rating in an evaluation u.

$S_i(\chi)$ ~ The set of all evaluations $u \in \chi$ such that they contains item i ($i \in S(u)$).

$\sum_{i \in S(u) \cap S(v)} u_i v_i$ ~ Given two evaluation u and v, the scalar product $\langle u, v \rangle$

γ is a similarity measure computed from Pearson's correlation

$$Corr(u,w) = \frac{\langle u - \bar{u}, w - \bar{w} \rangle}{\sqrt{\sum_{i \in S(u) \cap S(w)} (u_i - \bar{u})^2 \sum_{i \in S(u) \cap S(w)} (w_i - \bar{w})^2}}$$

u_i ~ The rating of user u gives to item i.

From $Corr(u,w)$, calculate $\gamma(u,w)$.

$$\gamma(u,w) = Corr(u,w)|Corr(u,w)|^{\rho-1}$$

From [7], ρ is the Case Amplification power, set as 2.5. Case application reduces noise in the data. If the correlation is high, say $Corr = 0.9$, then it remains high ($0.9^{2.5} \cong 0.8$ after Case Amplification). If it is low, say $Corr = 0.1$, then it becomes negligible ($0.1^{2.5} \cong 0.003$).

4.3.4 Slope One

The Pearson reference scheme only takes into account information from other users who rated the same item, while the Slope One schemes take both account from users who rated the same item and the other items rated by the same user (like the Per User Average).

Given a training set χ, and two items j and i with rating u_j and u_i respectively in some user evaluation u (annotated as $u \in S_{j,i}(\chi)$), the average deviation of item i with respect to item j as:

$$dev_{j,i} = \sum_{u \in S_{j,i}(\chi)} \frac{u_j - u_i}{card(S_{j,i}(\chi))}$$

$card(S_{j,i}(\chi))$ ~ The number of elements in a set $S_{j,i}(\chi)$.

$S_{j,i}(\chi)$ ~ The set of users which have rated both items j and i.

Note that any user evaluation u, not containing both u_j and u_i, is not included in the summation. The symmetric matrix defined by $dev_{j,i}$can be computed once and updated quickly when new data is entered.

Given that $dev_{j,i} + u_i$ is a prediction for u_j given u_i, the prediction is the following formula:

$$P(u)_i = \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i} + u_i$$

$R_j = \{i | i \in S(u), i \neq j, card\left(S_{j,i}(\chi)\right) > 0\}$ ~ The set of all relevant items.

For a dense enough data set, where almost all pairs of items have ratings, $\bar{u} = \sum_{i \in S(u)} \frac{u_i}{card(S(u))} \cong \sum_{i \in R_j} \frac{u_i}{card(R_j)}$ for most j, the prediction for the Slope One scheme can be simplified as the following formula:

$$P^{S1}(u)_j = \bar{u} + \frac{1}{card(R_j)} \sum_{i \in R_j} dev_{j,i}$$

This equation means that Slope One doesn't depend on how the user rated individual items, but only on the user's average rating and crucially on which items the user has rated.

4.4 Support Vector Machine (SVM)

In machine learning, support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. They are one of the most popular machine learning algorithms.

4.4.1 Introduction

Given a set of training examples, each marked as belonging to one of two categories; an SVM training algorithm builds a model that assigns new examples into one category or the other.

A support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, since in general the larger the margin, the lower the generalization error of the classifier.

Linear SVM is the basic SVM algorithm. Given some training data D, a set of n points of the form $D = \{(x_i, y_i) | x_i \in R^p, y_i \in \{-1, 1\}\}_{i=1}^n$, where the y_i is either 1 or -1, indicating the class to which the point x_i belongs. Each x_i is a p-dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points X satisfying:

$$\boldsymbol{w} \cdot x - b = 0$$

$\cdot \sim$ The dot product.

$w \sim$ The normal vector to the hyperplane

$\frac{b}{\|w\|} \sim$ It determines the offset of the hyperplane from the origin along the normal vector w.

If the training data are linearly separable, two hyperplanes can be selected in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called “the margin”. These two hyperplanes can be described by the equations:

$$\boldsymbol{w} \cdot x - b = 1$$

and

$$\boldsymbol{w} \cdot x - b = -1$$

The following figure would explain the relationship among hyperplane, margins and support vectors.

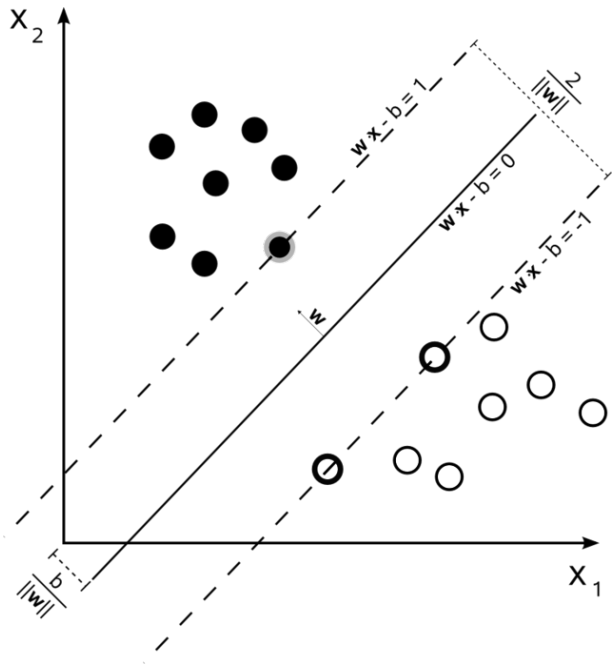

Figure 18. Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

By using geometry, the distance between these two hyperplanes is $\frac{2}{\|w\|}$, so the goal is to minimize $\|w\|$. Also, it needs to prevent data points from falling into the margin, and then the following constraints are added:

$w \cdot x_i - b \geq 1$ for x_i of the first class

or

$w \cdot x_i - b \leq -1$ for x_i of the second class

Then we have an equation: $y_i(w \cdot x_i - b) \geq 1$, for all $1 \leq i \leq n$.

It can be transformed to an optimization problem:

Minimize $\|w\|$ subjects to (any $i = 1, \ldots, n$) $y_i(w \cdot x_i - b) \geq 1$.

SVM is an effective method to do supervised learning for distinguishing between two categories. However, this result is not enough for the use case here. Here, the rating has five options, from 1 to 5.

Luckily, SVM can be extended to support multiclass classification. The following section will shed some light on how.

4.4.2 Multiclass classification SVM

For multiclass classification SVM [8], the labels are drawn from a finite set of several elements. In this case, there are 5 labels.

In order to achieve multiclass classification, usually a combination of several binary SVM classifiers is used to solve the problem. Popular methods for doing this are: 1) one-versus-all method, which uses winner-takes-all (WTA) strategy and 2) one-versus-one method by max-wins voting (MWV).

For a given multiclass problem, M will denote the number of classes and ω_i, $i = 1, 2, \ldots M$ will denote the M classes. For binary classification, the two classes can be referred as positive and negative.

4.4.2.1 One-Versus-All SVM

One-versus-all SVM constructs M binary classifiers. The ith classifier output function ρ_i is trained taking the examples from ω_i as positive and the examples from all other classes as negative. For a new example x, One-versus-all SVM assigns it to the class with the largest value of ρ_i.

4.4.2.2 One-Versus-One SVM

This method constructs one binary classifier for every pair of distinct classes. Together, $M\ (M-1)/2$ binary classifier are constructed. The binary classifier C_{ij} is trained taking the examples from ω_i as positive and the examples from ω_j as negative. For a new example x, if classifier C_{ij} says x is in class ω_i, then the vote for class ω_i is added by one. Otherwise, the vote for class ω_j is increased by one. After each of the $M\ (M-1)/2$ binary classifiers makes its vote. This strategy will assign x to the class with the largest number of votes.

Based on the comparison result from [8], the performances from one-versus-all and one-versus-one don't have a significant difference. So the following section will use one of them as multiclass SVM method.

4.4.3 Item-based Multiclass classification

As mentioned before, collaborative filtering can be performed as item-based or user-based. In item-based method, the multiclass SVM model is trained for each restaurant. Why do we want to use the item-based method? Based on the data from Yelp, there are 14,303 restaurants. Among them 5,157 restaurants have more than 30 reviews, around 36%.

More data means more accuracy. Item-based method customizes recommendation at the restaurant level.

4.4.4 User-based Multiclass Classification

According to Yelp's data, there are 252,898 users. Among this number, only 2,803 people have posted more than 30 reviews, about 1%. Although only a small number of people's reviews have enough data to build a model on, it plays an important role for performance.

In the following evaluation section, we will be able to compare the performance of item-based and user-based methods.

4.5 Evaluation

4.5.1 Mean Squared Error (MSE)

To measure the performance for the item-based collaborative filtering method, the top 300 restaurants were chosen, so each restaurant can have reviews from more than 300 users. For the user-based collaborative filtering method, the top 1000 reviewers were chosen to ensure each reviewer had more than 50 reviews.

In each dataset, for each item or user, WE selected $p\%$ data as training set χ, $1 - p\%$ as testing data T_χ to validate the performance.

There are two ways to calculate MSE; one is by getting the average of all the predicted values, the other is the average of the MSE for each item or user.

1) Average of all the predicted difference

$$MSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(predict_i - actual_i)^2}$$

$n \sim$ The total number of predictions has been made.

$predict_i$ ~ The predicted rating for the test data set.

$actual_i$ ~ The actual rating from the test data set.

2) Average of the MSE for each item or user

$$MSE = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

n ~ The total number of elements. For item-based method, n is 300 restaurants. For user-based method, n represents 1000 users.

MSE_i ~ The MSE for one restaurant or user.

The first method cannot show the performance at an elementary level, while the second can display the variance of performance with more detail. Although the same method is used, some restaurants or some users' predictions have a much better prediction ability.

4.5.2 K-Nearest Neighbors Evaluation

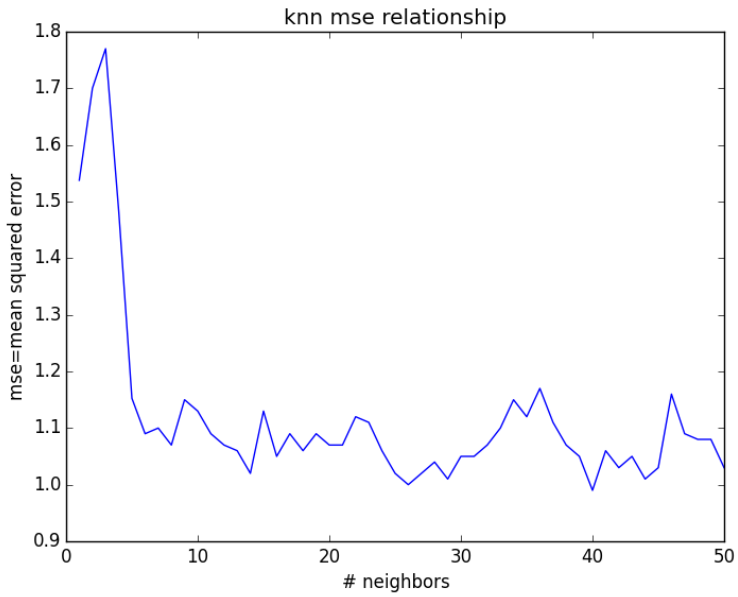

Figure 19. The relationship between MSE and the number of neighbors chosen for k-NN prediction

From the above figure, we can see that the number of neighbors plays an important role for the quality of performance. In the above figure, one restaurant that has the most reviews was chosen. From the result, when k is less than 10, MSE has a large variance. But after k increased to certain level, MSE doesn't have a very big improvement.

4.5.3 Pearson & Slope One Evaluation

Pearson and Slope One methods are both memory–based models, which require loading the information between user and restaurant. These two methods both use user similarity to predict ratings. Because there is no range restriction on the rating, the performance is not as good as expected.

The following is a part of the output for Slope One.

Actual	Prediction
4	5.047619048
3	4.795730373
5	5.035294118
3	4.967619048
4	5.013333333
4	5.307142857
4	4.934432234
4	4.685714286
2	5.361584596
5	4.945629371
5	5.105322129
3	5.783116883
2	5.157777778
4	4.576190476
5	5.310891623
3	4.843650794
4	5.234444444
5	5.352222222
4	5.6375
2	6.033333333

Table 6: Piece of output from Slope One

It is clear that something is not right. The value in prediction has exceeded the possible rating. Rating can only be between 1 and 5. Why? Because the calculation for Pearson and Slope One are not designed to set prediction boundaries, the result can be outside of the 1 to 5 rating.

That's why we switched to Multiclass SVM, which can set boundaries by specifying how many classes it can handle.

4.5.4 Multiclass SVM Evaluation

For the item-based method, 300 restaurants were chosen. For the user-based method, 1000 users were chosen. Each restaurant had more than 100 reviews, and each user had more than 30 reviews.

Based on two MSE calculation methods, the following is the comparison:

Suppose 90% data is used in training set, and 10% data is used as test data.

	Number of Topics	MSE1	MSE2
Item-Based	22	1.2879	1.2615
Item-Based	50	1.2884	1.2624
User-Based	22	1.1372	1.1095
User-Based	50	1.1376	1.1101

Table 7: Item-Based SVM Performance

From the above table, we can see that user-based SVM has better performance than item-based SVM, which make sense. The goal of this system is to make customized recommendations based on user's preference, taste and experience. To make the recommendation or prediction more effective, the challenge lies in obtaining enough data for every user.

Also, in general, MSE2 is smaller than MSE1. The way MSE2 is calculated implicitly indicates calculating the average twice, while MSE1 only requires once. But the results are still very close between MSE1 and MSE2.

Another finding from the above table is that more topics chosen to be in the model do not ensure the enhancement in the performance.

The following figures show how the quantity of data impacts the accuracy of recommendation. The number of topics chosen in the model is 22.

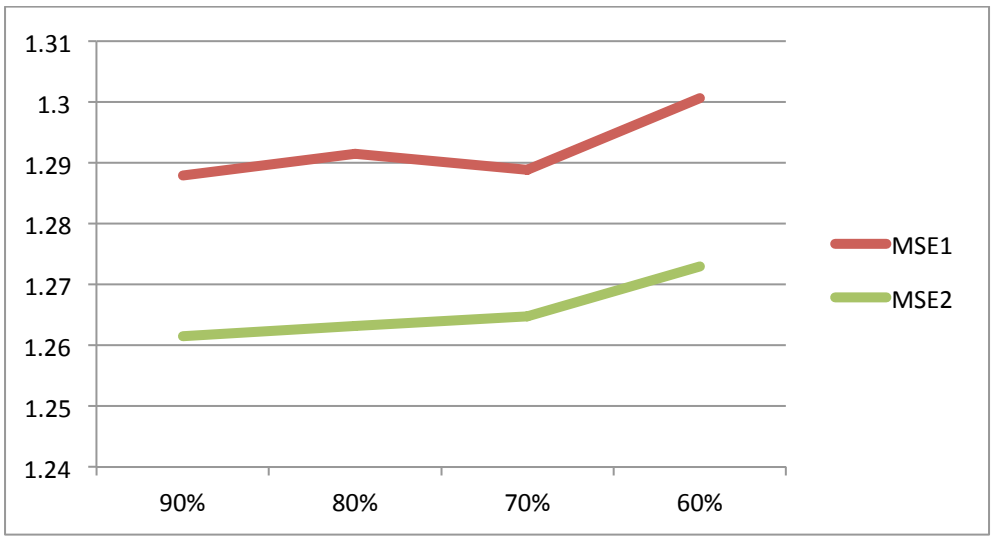

Figure 20. Item-Based Performance with different rate of training data.

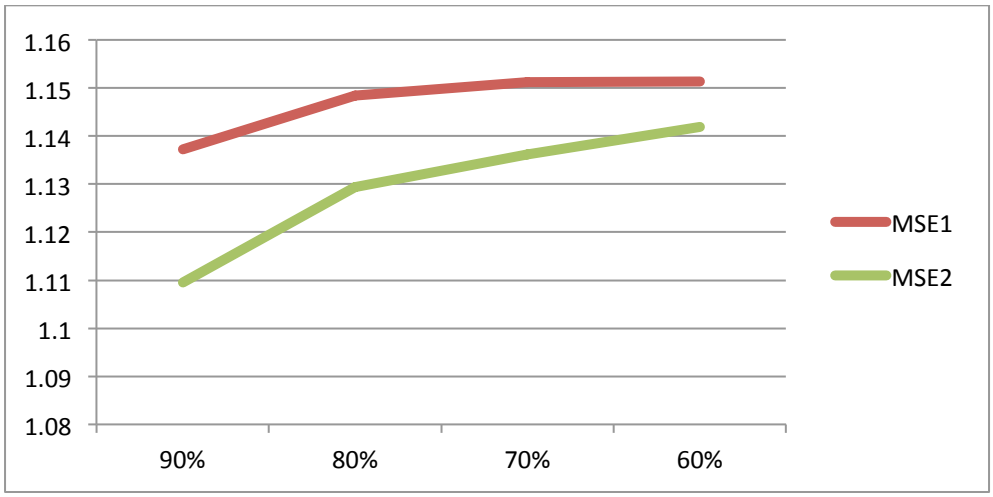

Figure 21. User-Based Performance with different rate of training data.

Comparing figure 20 and figure 21, we can find two MSE values in user-based model increased a lot when the percentage of training data to total data changed from 90% to 80%.

Here is the comparison of the performance variance between item-based SVM and user-based SVM.

Method	Training Data ratio Change	Difference Ratio $\Delta MSE1\%$	Difference Ratio $\Delta MSE2\%$
Item-Based SVM	From 90% to 80%	0.27%	0.14%
	From 80% to 70%	-0.21%	0.12%
	From 70% to %60	0.92%	0.65%
User-Based SVM	From 90% to 80%	0.98%	1.78%
	From 80% to 70%	0.24%	0.61%
	From 70% to %60	0.01%	0.5%

Table 8: MSE change ration based on training data ratio

$$\Delta MSE\% = \frac{MSE_{cur} - MSE_{pre}}{MSE_{pre}} \times 100\%$$

From table 7, the user-based SVM's MSE1 and MSE change ratio has dropped more than the item-based SVM. Why did this happen? Let's recall the data chosen for the evaluation. The top 1000 users were selected, and each user had more than 100 reviews. For restaurants, the top 300 were chosen and each restaurant had more than 400 reviews. The 10% data change has more impact on the user-based model, because the absolute number of reviews is larger for restaurants.

This issue is derived from the real challenge for Yelp: how to acquire more user-based data. People easily use Yelp to find restaurants, but only a relatively small fraction of them leave a rating, and even less of them bother to write a comment to explain the rating. So in real life, to achieve a good performance of

customized recommendation, both models need to be used. In the next chapter, WE describe how the database schema will be designed to support real-time customized recommendation.

Chapter 5 Real-Time Recommendation Model

5.1 Introduction

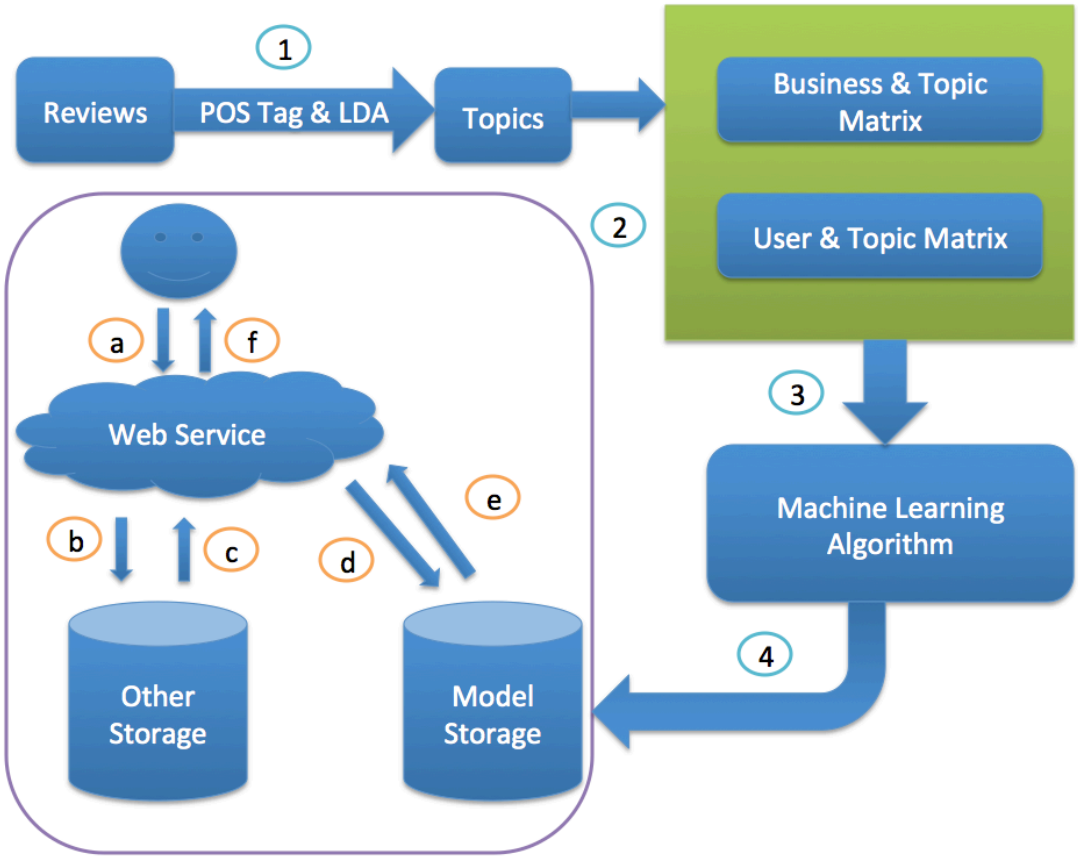

Figure 22. Real-time recommendation system implementation

This chapter will explain the details concerning how to implement the real-time recommendation implementation as steps from *a* to *f*.

Today, recommendations are everywhere online. Amazon, the biggest online retail store in the USA, provides recommendations in many different ways. Google enhances search results based on its knowledge of the user's past searches to find

the most relevant result. In this chapter, a real-time customized recommendation model will be introduced.

In the previous chapters, multiclass classification SVM was found to be an efficient algorithm to predict what a user would like based on 5 different rating ranks. The next step is to build an application framework, which will use the result of multiclass classification SVM to make recommendations.

5.1.1 Why Real-Time?

Visiting a restaurant is not like buying a commodity from Amazon.com. Buying from Amazon.com does not have a distance restriction. It would be unrealistic to have a lunch or dinner at a restaurant that is more than a 4-hour drive away from the user's current location. If it's just a lunch during the weekdays (not including holidays), people may just want to find some place to go within walking distance. Due the busy working schedules, people tend to limit their restaurant options to within 10 minutes' walking distance. This distance restriction leaves a limited number of restaurants a user can choose from a neighborhood.

Sometimes, finding the right place to go is not an easy job, and even time-consuming for a food-lover. There are can be various sizes of places, big or small. Different foods like ramen, Chinese food, Japanese sushi, Italian food, and American fast food can be served. All this information is hard to build before a user types "restaurant" in Yelp. The location determinant to the search result is also hard to predict.

Basically, geo-location-based recommendation has to be implemented to support real-time response.

5.2 Database Model

To build a real-time recommendation system, the system needs to be able to store a comprehensive view of the users. Moreover, we need to be able to retrieve data about a particular customer quickly in order to produce recommendations as they interact with Yelp's website or mobile app.

From the previous section, we found multiclass classification model is useful to predict the score of unvisited restaurants. Passing a list of unvisited restaurants to the existing model of the user, show the list of restaurant based on the predicted ranking score.

From the database or storage point of view, the model needs to be stored per user. Based on the findings in Chapter 4, only a small amount of users has enough reviews to have an effective trained model. So, using the item-based model is necessary when the review count on a user is below a certain threshold.

The following is a simplified database model for real-time recommendation.

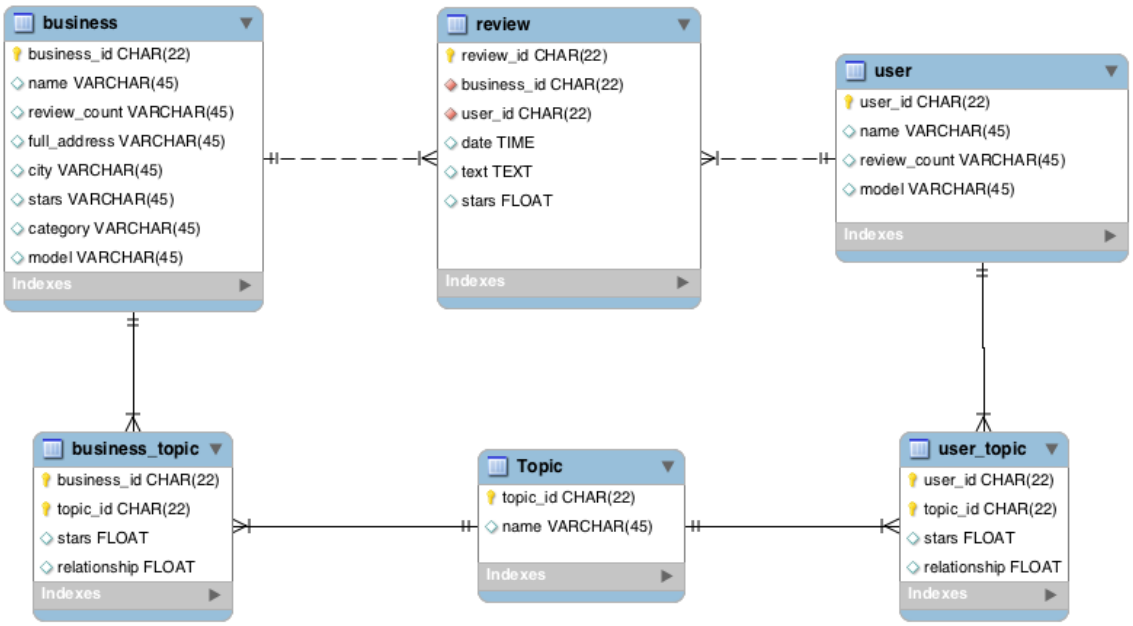

Figure 23. Simplified Real Time Recommendation Model

Business (Restaurant):

The major piece in the following field is a model, which will save the trained model for future recommendation.

Column Name	Type	Comments
business_id	string	Unique ID for business (restaurant).
name	string	Restaurant Name
review_count	integer	Number of reviews of this restaurant.
stars	float	The rating.
category	string	A list of category name.
model	blob	The ready trained model

Table 9: Business Table

User:

Same as the business table, the model was trained and saved in "model" column.

Column Name	Type	Comments
user_id	string	Unique ID for user.
name	string	Restaurant Name
review_count	integer	Number of reviews of this restaurant.
model	blob	The ready trained model

Table 10: User Table

5.3 Real Time Recommendation Procedure

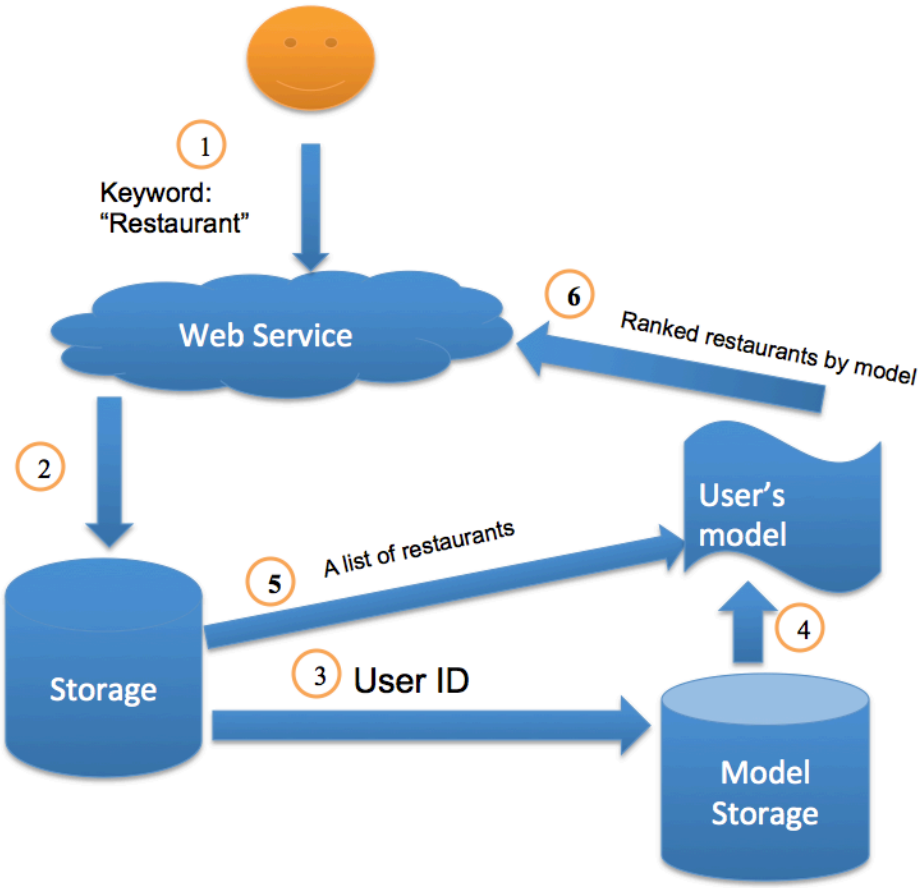

Figure 24. Real-time Recommendation Implementation

Here is the list of steps breaking down how the real-time recommendation was made:

1) The website or app may get a general request as “restaurant.”
2) Based on the current location information from user, a list of restaurants is searched from back end system. This list can be optimized by the distance or other known characteristics.
3) User ID is passed to model storage.
4) Based on User ID, user’s model is retrieved from model storage.
5) Run user’s machine-learning model against a list of restaurants.
6) Get a list of restaurants ranked by user’s machine learning model.

5.3.1 Performance Factor

Several places have influence on the performance in terms of time.

1) Handle http request: Determined by network traffic and sever responding speed.
2) Loading Model: User-based method is faster, because only one model is needed. Item-based method needs to load more models.
3) Loading Existing data: This is determined by database speed.
4) Model Prediction: Prediction speed depends on selected algorithm.

Chapter 6 Further Discussion

6.1 Introduction

Real time recommendation system is becoming more useful these days. However, there are several challenges to make it more efficient and feasible.

In the following section, we will discuss them one by one.

6.2 Challenges

6.2.1 Topic Modeling

A computer understanding a customer requires both natural language processing and topic modeling to advance further. In this thesis, topic modeling was built upon extracted words. This method cannot detect people's opinions. In [10], the study presents a method using Stanford typed dependencies to extract option phrases.

One example is:

This place is amazing. I come here at least once a month & am never disappointed. Food & service is always great. The buffalo burger it TDF as well as the bruschetta. Outside seating is so cute with a lights (great for date nights) live music inside is a wonderful touch. This place is great to meet with friends, family or date night.

The extracted opinion phrases are:

[“place amazing”, “service great”, “burger buffalo”, “seating cute”, “touch wonderful”, “seating Outside”, “place great”]

In this system, WE used a calculated rating to represent people's opinions. This method might be helpful to enhance the prediction performance.

6.2.2 Rating Prediction

The major challenges for rating prediction are feature extraction and machine-learning algorithm. To improve the performance for machine learning, feature analysis or feature selection can be a direction to get better performance.

6.2.3 Real-Time Framework

The concern for real-time framework is the efficiency of loading data and computing. Nowadays there are several popularly used real-time data analysis tools, such as Spark[10] or Storm[11]. HBase as a distributed, scalable, big data store can be an ideal way to save the pre-trained model.

6.3 Limitation

In this thesis, there are several preconditions before recommendation can be made.

1) User must have a login for Yelp
2) User must have activity recorded on Yelp before searching, such as reviewed a restaurant.

If user related data could be obtained external of Yelp, then it might be possible to provide recommendations without the above pre-conditions. One option can be a user's cookie data in browser.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation.” 2003.

[2] Pern Hui Chia, Yusuke Yamamoto, and N. Asokan. “Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals.” 2012.

[3] Penn Tree Bank Pos Tag. https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[4] Arun, R., Suresh, V., Madhavan, C. V., & Murthy, M. N. “On finding the natural number of topics with latent dirichlet allocation: Some observations.” 2010.

[5] Loren Terveen, Will Hill. “Beyond Recommender Systems : Helping People Each Other.” 2001

[6] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. “Grouplens: An open architecture for collaborative filtering of netnews.” 1994.

[7] Daniel Lemire, Anna Maclachlan. “Slope One Predictors for Online Rating-Based Collaborative Filtering.” 2005.

[8] Kai-Bo Duan, S. Sathiya Keerthi. “Which Is the Best Multiclass SVM Method? An Empirical Study.” 1998.

[9] Samaeh Abbasi Moghaddam. “Aspect-Based Opinion Mining In Online review.” 2013.

[10] Spark: https://spark.apache.org

[11] Stom: https://storm.apache.org

[12] HBase http://hbase.apache.org/

[13] Francesco Ricci, Lior Rokach, Bracha Shapira. “Recommender Systems Handbook” 2011

[14] Greg Linden, Brent Smith, Jeremy York. “Amazon.com Recommendations Item-to-Item Collaborative Filtering” 2003

[15] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Item-Based Collaborative Filtering Recommendation Algorithms” 2001

[16] Michael J. Pazzani, Daniel Billsus. “Content-Based Recommendation Systems” 2007

[17] Abhinandan Das, Mayur Datar, Ashutosh Garg. “Google News Personalization: Scalable Online Collaborative Filtering”. 2007

[18] Xiaoyuan Su, Taghi M. Khoshgoftaar. “A Survey of Collaborative Filtering Techniques” 2009

[19] Paul Resnick, Hal R. Varian “Recommender Systems” 1997