Yelp Personalized Recommendation System

Xun Xue, Zhangyu Liu, Lingyu Zhang
Electrical Engineering
Columbia University
xx2241@columbia.edu, zl2513@columbia.edu, lz2494@columbia.edu

Abstract—**In this paper, combining the results of collaborative filtering and topic modeling with Yelp dataset, we introduce a new interactive recommendation system, which can meet users dynamic requirements and personalize the recommendation. We also analyse the results build a tiny demo to demonstrate our system.**

Keywords: Personalization, interactive recommendation, Collaborative Filtering, Topic Modeling

I. Introduction

As a popular online consumer review application, Yelp relies on the recommendation system to give customer reasonable advice. One of the most popular recommendation algorithm is Collaborative Filtering, which can provide a reasonable prediction of customers' preferences based on each user's rating in this system. However, specific feelings towards each particular business can't be fully express only by ratings, we can explore more detailed information in the reviews. Moreover, collaborative filtering algorithm treats user's characteristic as a consistent feature, but in reality user's mood is different every day, consistent recommendation results can not meet the dynamic requirement.

Topic Modeling is a useful tool allowing us to learn the latent subtopics in review texts, with which we can extract the key-words for each restuarants to represent the detailed features[1][2]. In our personalized recommendation system, firstly we use collaborative filtering to generate potential recommendations. Then we perform topic modeling for each of the potential recommendations to explore specific features. We allow user to input key-words and match the key-words with those potential recommendations in order to further filter the results. Therefore, our recommendation system can interact with users and meets theirs dynamic requirements.

In this paper, we will explain our methodology in detail and demonstrate our implementation with Apache Spark as the main tool. We will show the novelty and explore the potential business value of our personalized recommendation system.

II. Related Works

In this part, we will demonstrate the relevant research including topic modeling, collaborative filtering and text feature matching, which are also the three most import part in our project.

1. Collaborative filtering

Collaborative filtering is widely used in recommendation systems. This can be applied in combining different source of data and filtering the pattern through extracting the preferences information of the multiple sources. [3] proposes an item-based recommendation algorithm using collaborative filtering. This paper analysis the performance of different approaches for calculating the similarity values between two items. Experiment in it compares the performance of item-based collaborative filtering recommendation algorithm with the basic k-nearest neighbor search. [4] discusses the evaluation approach of collaborative filtering recommendation system. In this article, several key decisions for the recommendation system have been reviewed including the size of the dataset, the way to measure the prediction accuracy. Based on one content, the empirical result from the analysis of different accuracy matrices have also been presented.

2. Topic Modeling

Topic modeling is an approach which helps us discover the topics that occur in one document. It is based on the counting of the word frequency and give us the output of a bag of words with the largest frequency of the occurrence. This is of great important for us to realize the hidden semantic structure of a text document. LDA is a general tool for us to do topic modelling. [5] proposes the basic structure to extract the topics with the words from the collection of documents. We firstly define a topic to be a distribution over a set of words. for the collection of documents, we randomly choose a distribution over topics. Since the topics for each document are of different distribution. Each word in one document is extracted from one specific topic which is randomly selected from the document distribution over topics. With this procedure, we can assign several topics with a set of words as well as the distribution probabilities for each document. [6] proposes another method for topic modeling called LSI. It is based on latent semantic analysis which assumes that similar words in meaning would occur in the similar position of one text document. [7] clearly discusses the difference between LDA and LSI.

3. Text Matching
Text Matching is a procedure which can help us find the similarity score of two text words. It can be applied to multiple areas. [8] proposes the examination which aims at testing the hypothesis that the ability for user to learn from a text is based on the matching between of the user's background knowledge and the difficulty of the text information.

III. System Overview

In this part, we describe our system in detail, including data extraction, collaborative filtering, topic modeling and text matching, the relationship between these processing be can demonstrated in Fig. 3.1.

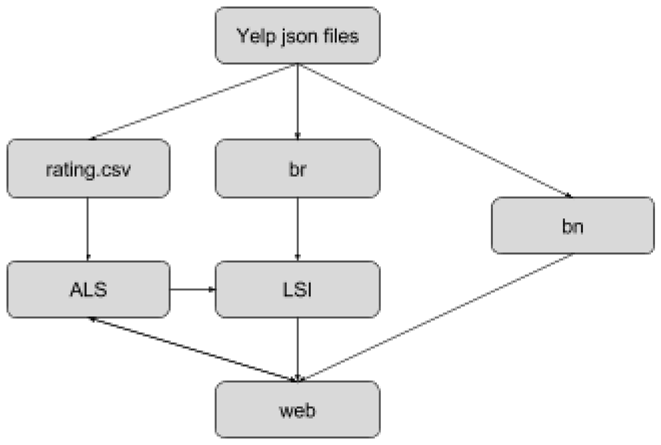


Figure 3.1

1. Data extraction
The Yelp challenge dataset we used is huge and contains about 86K businesses, 566K business attributes, 687K users and 2.7M reviews. We decided to focus our attention on restaurants in Las Vegas. The data was split across multiple JSON files. Each of these files contained attributes that were not of use to us since they would not be involved in the recommendation algorithm. Hence, we concentrated on major attributes like user and business IDs, review texts, ratings, etc.

We initially converted the JSON data from Yelp into CSV format. This is because we didn't need all the attributes that were present in the data such as hours open, whether the place is still functional, whether the user is elite, etc. We focused our attention on necessary information such as ids, names, stars, and review texts since these were required for our algorithms.

We wrote a python script to parse the JSON data and extract the attributes that we needed. We first created a dictionary of business ids that were in Las Vegas after which we extracted the ratings and review texts for the businesses, dumped business ids and ratings to csv file, joined reviews into a dictionary named br which indexed with business ids. While extracting the reviews, we made another dictionary named bn of business names indexed by business ids. Then dump dictionaries to pickle files.

2. Collaborative Filtering
Spark MLlib provides a collaborative filtering technique by using Alternating Least Squares algorithm.
The cleaned and integer indexed rating.csv file formed from the previous step were fed to the Spark python script, which carried out the user based collaborative filtering algorithm described later by . The result from Spark could be combined with LDI part, which makes the the recommendation more reasonable and suitable. The details about collaborative filtering will be discussed in part IV.

3. Topic Modeling
The preprocessing for LSI modeling can be divided into 4 steps including tokenization, stop words, stemming and constructing matrix. The dataset we have contains different reviews of different business. For one text review, we regard it as a document. The first thing to do is to segment the whole document into separate items which are all single words. After tokenization, we get a list of tokens. Since we are aiming at extracting the features of the text reviews, we need to delete the meaningless words such as "to", "my", "is". For our approach, we use stop words package from Pypi. After that, we are supposed to find out the words which are similar to avoid pleonastic procedure in the training part. We use the Porter Stemmer module from NLTK to help us discover the similar words and delete repeated one. The last thing is to convert the stemming result to matrix and prepare the result to pickle list for LSI.

We use LSI model in gensim to perform LSI algorithm. We calculate the LSI results for the review of each restaurant and append them as a list to prepare for text matching. The detail of which will be explained in part IV.

4. Text matching

Firstly we need to convert the words to vectors which can later be used to measure the similarity in meaning for different words. This procedure is based on Word2vec algorithm in gensim. Each specific word in the big corpus is represented by a vector in a multi-dimensional vector space. After training the Word2vec model, we can analysis the semantic relationship of different words. For our project, we do matching on the user's keywords typed and the keywords of the business. Through calculate the semantic similarity scores, it helps us find the best-matching business based on user's personalized information.

5. Objective

The objective of our system is consist of two parts. One is basic recommendation, the other is personalized recommendation. In the user interface, we allow user to input his/her userID and keyword. If the user only input userID, we will perform the basic recommendation, which is the top 10 results for collaborative filtering. If the user input his/her userID and keyword, we will perform the personalized recommendation. In this process, firstly we generate 100 results by collaborative filtering, then we conduct topic modeling for the reviews of these 100 results. After that we extract the key-words generated by topic modeling and match them with the user's keyword input, calculating word similarity. Finally, we output the top 10 results sorted by word similarity. We will demonstrate this process in detail in part V.

IV. Algorithm

1. Collaborative Filtering

In recommendation part of our project, we used the collaborative filtering technique provided from the Spark machine learning library. The idea behind the user-based recommendation is taking the similar users' preferences on certain products to predict certain user's choice. In Spark MLlib, an implementation of Alternating Least Square (ALS), a popular collaborative filtering algorithm, was provided.

In a matrix factorisation (MF) framework, MF optimises:

$$argmin\|R - U^T V\|^2 + \lambda(\|U\|_2^2 + \|V\|_2^2)$$

where R partially represents the user-item preference matrix, U and V represent user and item feature factors separately.

ALS solves least squares by initializing U with average values and then keep U fix and optimize V and vice versa. There are two main benefits of this approach:

a. very easy to parallel.
b. much more efficient in comparison with other techniques like SGD, etc.

In this project, we first measured errors for different parameters with a small test dataset, choose the best combination automatically, then trained model for the complete dataset.

2. Topic Modeling

In machine learning and natural language processing, topic modeling is a frequently used text-mining tool for discovery of hidden semantic structure in a text body. The key idea of topic modeling is to find low-dimensional descriptions of high-dimensional text [9].

In our implementation, we use Latent Semantic Indexing (LSI) (aka Latent Semantic Analysis) with model in python package gensim. Before performing LSI algorithm, some basic data preprocessing is necessary, such as tokenization and setting stop words. We conduct data preprocessing with natural language processing toolkit NLTK and stop_words. After we preprocess the data, we construct a document-term matrix, then we can analyse using Singular Value Decomposition (SVD).

SVD is the generalization of eigendecomposition of a positive semidefinite normal matrix to any $m \times n$ matrix via an extension of polar decomposition. The approximate SVD formulation of LSI is $X \approx TSD'$. Since D and T are orthonormal, we can represent term similarity matrix as $XX' = TS^2T'$ and document similarity matrix as $X'X = DS^2D'$. For results retrieval, LSI compute cosines between query vector X and apply threshold to determine operating point.

In gensim, the LSI model can SVD can be updated with new observations at any time, for an online, incremental, memory-efficient training. It can perform analysis of large corpora effectively even though corpora size is much larger than RAM. Since gensim use one-pass algorithm which don't need to even temporarily store corpora. Each document can only be seen once and process immediately. This model is very suitable for key-words extraction from large dataset, therefore we choose it in our topic modeling implementation.

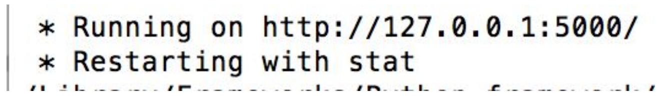
V. Software Package Description

Here is the repository of source code.
https://github.com/Sapphirine/Yelp_Personalized_Recommendation_System
The source code contains three parts, data_preprocessing folder, train_model folder and main folder.

1. Preparation for the demo
Firstly, we need to preprocess the dataset. The dataset we use is Yelp Dataset Challenge from https://www.yelp.com/dataset_challenge.
We use “final_data.py” in data_preprocessing folder to generate a csv file "yelp_reviews.csv" preparing for collaborative filtering and two pickle file “business_num_review” and “business_num_name” for topic modeling. Then we train the collaborative filtering model with “train_model.py” in train_model repository. After we finish our training, we should save the collaborative filtering model.

2. Launch the demo
In the main folder, we use preprocess.py to prepare dataset for topic modeling, which include tokenization, setting stop words, stemming(we choose not to use it in the final demo) and generating document-term matrix. this procedure will use “business_num_review” and “business_num_name” pickle file generated previously. Then we can perform topic modeling, the function of which is defined in “topic_modeling.py”. We can use “buildmodel.py” for research purpose or use Apache Spark to run “main.py” and launch the app.

3. Use the demo
After we launch the app, we will get a url as shown in Figure 5.1.



```
 * Running on http://127.0.0.1:5000/
 * Restarting with stat
```

Figure 5.1

Coming to this url, we can click the submit button launch the model as shown in Figure 5.2. After we click it, the backend will load the collaborative filtering model which we have already trained in advance.

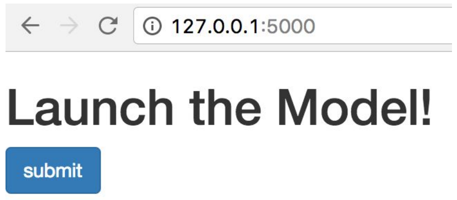


Figure 5.2

After we launch the model, the demo will route to another url with a simple interface allowing use to input their userID and keyword as shown in Figure 5.3.

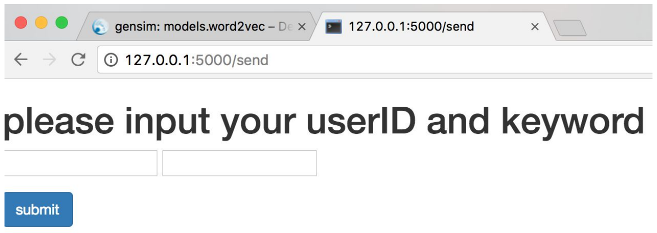
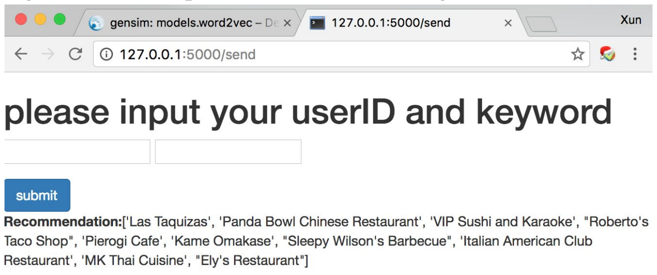

gensim: models.word2vec – De
127.0.0.1:5000/send
127.0.0.1:5000/send
please input your userID and keyword
submit

Figure 5.3

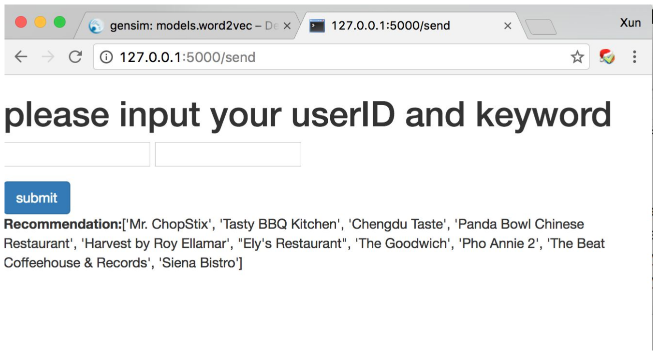
If the user only input their userID, the system will give him/her 10 restaurant's’ name which is the top 10 results generated by collaborative filtering, as the basic recommendation in this system. The result is shown in Figure 5.4 (the input userID is 110 for Figure 5.4 result).


gensim: models.word2vec – De
127.0.0.1:5000/send
Xun
127.0.0.1:5000/send
please input your userID and keyword
submit

Recommendation:['Las Taquizas', 'Panda Bowl Chinese Restaurant', 'VIP Sushi and Karaoke', "Roberto's Taco Shop", 'Pierogi Cafe', 'Kame Omakase', "Sleepy Wilson's Barbecue", 'Italian American Club Restaurant', 'MK Thai Cuisine', "Ely's Restaurant"]

Figure 5.4

If the user input the userID and a keyword simultaneously, the system will firstly generate 100 results by collaborative filtering, then conduct topic modeling for the review of these 100 restaurants. After that, it will extract the key-words generated by topic modeling and calculate the similarity between user’s input and those key-words. Finally, the system will output the top 10 results sorted by word similarity. The results for is shown in Figure 5.5.(for the same userID 110 and the keyword Chinese). Notice that the keyword must be meaningful, otherwise the system will cast an exception.



please input your userID and keyword

Recommendation:['Mr. ChopStix', 'Tasty BBQ Kitchen', 'Chengdu Taste', 'Panda Bowl Chinese Restaurant', 'Harvest by Roy Ellamar', "Ely's Restaurant", 'The Goodwich', 'Pho Annie 2', 'The Beat Coffeehouse & Records', 'Siena Bistro']

Figure 5.5

The result of Figure 5.5 will be discussed in part VI.

In this demo, when we perform topic modeling algorithm, the results of the key-words of each restuarants will display in terminal for demonstration purpose. However, in practical we should perform topic modeling for every restuarants in advance and extract the data directly to save time.

VI. Experiment Results

1. Collaborative filtering result

To test the collaborative filtering recommendation system, we perform cross-validation over the dataset and test the predicted ratings with the actual ratings of the businesses. We used the root mean squared error (RMSE) to test these ratings:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(r_{1,t} - r_{2,t})^2}{n}}$$

Firstly, we split the 70% of the ratings as the training set the remaining 30% as the testing set. Then we train the model and evaluate it by the testing set.

In Figure 6.1, with fixed iteration and λ, each rank improves the RMSE score of the dataset, and it converges after about 35. Then with fixed rank and λ, each iteration improves the RMSE score of the dataset, and it converges after about 25, which shown as Figure 6.2.

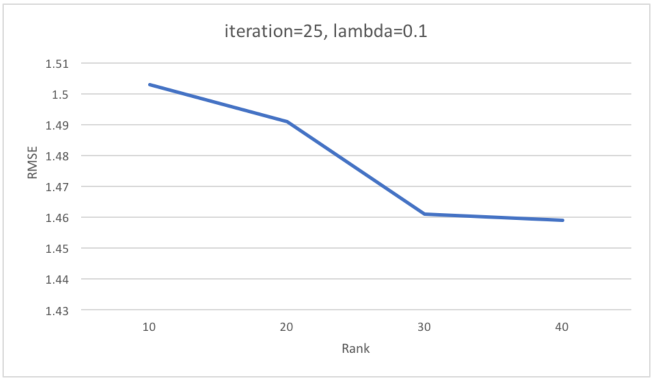


Figure 6.1

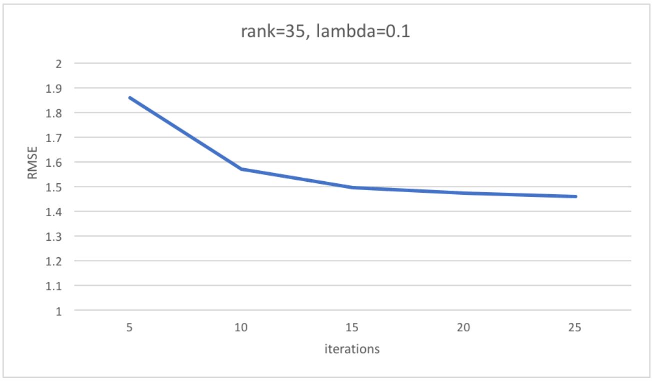


Figure 6.2

Empirically we found that for fixed rank , the convergence RMSE score is a convex function of λ, and the optimal value of λ is monotone decreasing with respect to rank . Based on these observations, in Figure 6.3, we were able to find the best value of λ=0.4 for optimal rank.

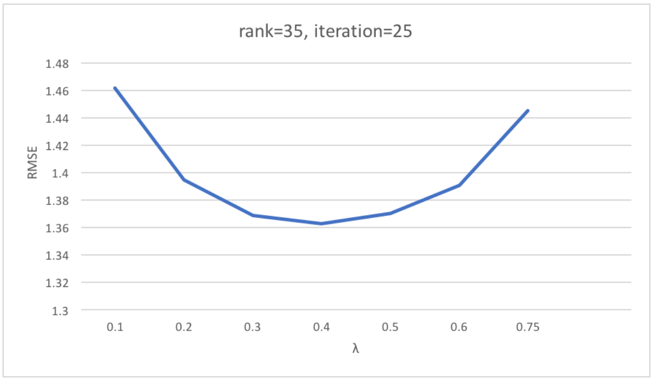
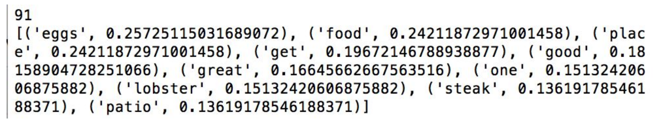


Figure 6.3

Finally, we get the optimal ALS model RMSE=1.36 by setting the parameters discussed above. It represents one of the top pure-ALS-method performance of this raw dataset according to our knowledge. By combining with LSI, we can make better prediction.

2. LSI result

Since what we need is the key words for each resturants, we output the 10 key-words and don't limit the number of topics. Figure 6.4 show the output for restaurant 91. As we can see from the results, the key words such as "eggs", "steak", "lobster" can be very helpful to represent the feature of the restaurant. However, some words such as "food" is useless. We can further optimize the results by modifying stop words.



```
91
[('eggs', 0.25725115031689072), ('food', 0.24211872971001458), ('plac
e', 0.24211872971001458), ('get', 0.19672146788938877), ('good', 0.18
158904728251066), ('great', 0.16645662667563516), ('one', 0.151324206
06875882), ('lobster', 0.15132420606875882), ('steak', 0.136191785461
88371), ('patio', 0.13619178546188371)]
```

Figure 6.4

3. Text matching result

We need firstly extract all the keywords in the LSI result without any float numbers. Then we convert the word list to the vectors to calculate the semantic similarity.

Since each topic contains multiple keywords. For the matching result of a specific user's keywords to a set of topics. Here we simply add the similarity scores of each keywords in the topic to do comparison of different topics.

4. Personalized recommendation result

The result of a personalized recommendation is shown in Figure 5.5, which is the recommendation for user 110 who wants to have Chinese food. As we can see from Figure 5.5, the results make sense, since some restaurants such as "Mr.ChopStix", "Chengdu Taste", "Panda Bowl Chinese Restaurant" must be chinese restaurants. Other restaurants also have many key-words relevant to "chinese" in their reviews.

VII. Conclusion

To summarize, our project firstly perform collaborative filtering algorithm to generate basic recommendation, and then realized the personalized recommendation by adding another filter with topic modeling, which can meet user's dynamic requirement.

For future works, we can modify the algorithm and tune parameters in order to achieve better results, and our primitive web interface also need to be upgraded.

All members in our team work hard and conducted great jobs. Therefore we assign each person 33.3% contribution.

Acknowledgment

We are so grateful for our professor and TAs' help. Thank professor Lin for teaching us this semester, giving us so many advices about how to improve our project. Thanks TAs, who offering support in assignments, carefully checking our homework and giving us many constructive comments patiently. At least, thanks all people who directly or indirectly helped us complete our project.

References

[1] Huang J, Rogers S, Joo E. Improving restaurants by extracting subtopics from yelp reviews[J]. iConference 2014 (Social Media Expo), 2014.

[2] Linshi J. "Personalizing Yelp Star Ratings: a Semantic Topic Modeling Approach"[J]. Yale University, 2014.

[3] Herlocker, Jonathan L., et al. "Evaluating collaborative filtering recommender systems." *ACM Transactions on Information Systems (TOIS)* 22.1 (2004): 5-53.

[4] Wolfe, Michael BW, et al. "Learning from text: Matching readers and texts by latent semantic analysis." *Discourse Processes* 25.2-3 (1998): 309-336.

[5] Blei, David M. "Probabilistic topic models." *Communications of the ACM* 55.4 (2012): 77-84.

[6] Blei, David M., and John D. Lafferty. "Topic models." *Text mining: classification, clustering, and applications* 10.71 (2009): 34.

[7] Vinokourov, Alexei, Nello Cristianini, and John S. Shawe-Taylor. "Inferring a semantic representation of text via cross-language correlation analysis." *Advances in neural information processing systems*. 2002.

[8] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001.

[9] Eugene Weinstein. "Introduction to Topic Models." October 21st, 2008.