Recommendation System with Aspect-Based Sentiment Analysis

Qiming Du
University of Southern California
Los Angeles, California
qimingdu@usc.edu

Dinghan Zhu
University of Southern California
Los Angeles, California
dinghanz@usc.edu

Wenjing Duan
University of Southern California
Los Angeles, California
duanw@usc.edu

ABSTRACT

Nowadays, a good recommendation system can not only help merchants make more profit, but also help customers find what they need. We are trying to build a recommendation system based on Yelp Dataset. We use content-based, memory-based, model-based and hybrid models to recommend businesses to users. We also apply some natural language processing algorithms and sentiment analysis on reviews to find out what they think of the business in different aspects. Then we can make recommendations from the users' own perspective.

KEYWORDS

Recommendation System, NLP, GloVe, Sentiment Analysis

1 INTRODUCTION

With the development of information technology and the growth of the Internet, information overload is becoming a big problem for customers. Although search engines can help customers to find things they want, there are still some situations where customers don't even know what they want or they don't want to spend time on looking for interesting things.

That's why recommendation system is very important nowadays. A good recommendation system can save users' time on looking for things they want and satisfy their needs properly. A good recommendation system can also help merchants to make more profit by recommending suitable businesses to proper users according to their profile and rating history.

Our recommendation system is built on Yelp Dataset. Given a user, it can recommend a bunch of businesses based on his or her rating history, similar user's preferences and businesses' information. The number of recommended businesses can be designated by users. Users can even sort those recommended businesses according to any of five aspects, including food, service, costs, environment, and location.

2 DATASET

We are using the dataset from Yelp Dataset Challenge round 12[1]. There are nearly 6 millions reviews and we randomly divide them into training set and testing set with 8:2 ratio. Review text is used for sentiment analysis and review stars are used for recommendations retrieval.

Businesses' category and location are used in our recommendation system as known features. When a business is registered into a system, their basic information will be immediately available. Therefore, this kind of information should be available during the cold start.

[1]http://www.yelp.com/dataset

3 METHODS

3.1 Hybrid Memory-Based Model

User-Based Collaborative Filtering and Item-Based Collaborative Filtering are Memory-Based approaches of Collaborative Filtering. Memory-Based CF uses user rating data to compute the similarity between users or items. It's effective and easy to implement.

3.1.1 User-Based Collaborative Filtering. In our task, we want to recommend several businesses that haven't been visited by user before but might be liked to user. We could assume that similar people will have similar taste. Suppose user1 and user2 have visited the same business, and they rated them all almost identically. But user1 has not been to "Sun Nong Dan-a famous Korean restaurant" and user2 has. If user2 love that restaurant, it sounds logical to think that user1 will too. With that, we have created an artificial rating based on their similarity.

User-Based CF recommends items by finding similar users to the active user (to whom we are trying to recommend a business). A specific application of this is the user-based Nearest Neighbor algorithm. This algorithm needs two tasks:

(1) Calculate User Similarity using Pearson Correlation, as shown in equation 1
(2) Select top-k users that are similar to target user called neighbors
(3) Compute a prediction from a weighted combination of the selected neighborsâĂŹ ratings, weights are similarity according to equation 2

$$w_{u,v} = \frac{\sum_{i \in I} \left(r_{u,i} - \overline{r_u}\right)\left(r_{v,i} - \overline{r_v}\right)}{\sqrt{\sum_{i \in I} \left(r_{u,i} - \overline{r_u}\right)^2}\sqrt{\sum_{i \in I} \left(r_{v,i} - \overline{r_v}\right)^2}} \tag{1}$$

$$P_{a,i} = \overline{r_a} + \frac{\sum_{u \in U} \left(r_{u,i} - \overline{r_u}\right) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|} \tag{2}$$

3.1.2 Item-Based Collaborative Filtering. Instead of focusing on similar users, we could focus on what items from all the options are more similar to what we know the user enjoys. This new focus is known as Item-Based Collaborative Filtering.

The same, we first calculate similarity among businesses the user has been rated before using Pearson Correlation as presented in equation 1.

Then we choose top-k businesses that are most similar to target business. Last we do Item-Based Predictions Using a Simple Weighted Average as shown in equation 3

$$P_{a,i} = \frac{\sum_{u \in U} r_{u,i} \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|} \tag{3}$$

3.1.3 The Hybrid Approach. We combined Item-Based and User-Based collaborative filtering, set neighbor size of User-Based to 1, set neighbor size of Item-Based to 3, combined the final prediction of the two with confidence.

Although this approach is easy to implement, context independent and generally, the prediction result is more accurate compare to content-based recommendation. It faces several problems:

(1) Sparsity: The percentage of people who rate business is low, so new user or new business will have no information. We assigned average rate of the business or average rate rated by the user to solve it which could result in inaccurate prediction.
(2) Scalability: The more K neighbors we consider (under a certain threshold), the better our prediction should be. However, the more users or business are in the system, the greater the cost of finding the nearest K neighbors will be. We tried another way: Model-Based Recommendation System to solve new user and new business problem.

3.2 Alternating Least Square

Model-Based recommendation system involve building a model based on the dataset of ratings. In other words, we extract some information from the dataset, and use that as a "model" to make recommendations without having to use the complete dataset every time. This approach potentially offers the benefits of both speed and scalability. There are three generic modeling approaches: 1) Clustering based algorithm; 2) Matrix factorization based algorithm; 3) Deep learning.

We adopted Spark MLLib library as our model-based recommendation system library, which uses alternating least squares (ALS)[4] algorithm, a matrix factorization algorithm to build the model.

3.3 LightFM

3.3.1 Model Introduction. LightFM was introduced by Toine Bogers and Marijn Koolen in 2015, which is a hybrid model of content-based model and model-based model. This model was motivated by two considerations[5]:

(1) The model must be able to learn user and item representations from interaction data: if items described as "ball gown" and "pencil skirt" are consistently all liked by users, the model must learn that ball gowns are similar to pencil skirts.
(2) The model must be able to compute recommendations for new items and users.

The model fulfills the first requirement by optimizing a model which produces similar embeddings for users or businesses with similar interactions, and produces unrelated embeddings for users with totally different interactions. This method is, as stated by the author [2], over and above what pure content-based model using dimensionality reduction techniques can achieve.

The second requirement is fulfilled by representing user/items as linear combinations of their features extracted from contents. For example, the representation for denim jacket is simply a combination of the representation of denim and the representation of jacket.

3.3.2 Design of Content Features. For Yelp reviews, we need to design the categorial features for both user and businesses in order to make use of the cold start features of LightDM model.

Unfortunately, the Yelp dataset did not provide any categorial data on users. When a new user registered, none of the ratings or compliment information will be available. Therefore, all users will be given a single same category for the model to work properly.

And for business, we are using two fields of the business data as business features. The first field used is category. If a user is favor of one category of restaurants, it is more likely that he will like another one. Another business data used in the model is location, specifically, the state field. There are so many different cities, neighborhoods or postal code present in data that the representation can be too sparse to make sense. This category information could make sense because each state is developed in different level and have different style. Some states may prefer spicy food or tacos, while some may prefer sweet food more.

3.4 Spotlight

Another library experimented is Spotlight[6]. Which is first introduced on GitHub in 2017 as a replacement and upgrade to LightFM. In this library, both explicit matrix factorization and implicit matrix factorization methods are supported, and more loss functions are supported than LightFM model.

However, spotlight does not support user and business features, which can potentially be a huge disadvantage. Another problem is that this library utilized PyTorch for deep learning models, and it takes more than tolerated time to train and test both the explicit model and the implicit model. It takes more than 50 times of training time as to the LightFM model on a similar scale model, which is about 20 hours to train a model with 10 components for 5 iterations on a Macbook Pro.

3.5 Aspect-Based Sentiment Analysis on Reviews

In this part, we are trying to extract useful information about every business from reviews given to that business. Specifically, we want to know how good every business is in five aspects, including food, service, costs, environment, and location. We will not perform any evaluation method on our natural language processing results because Yelp dataset does not have any independent rating in these aspects.

After careful consideration and comparison, we choose spaCy[2] as our main natural language processing tool. Because spaCy makes the hard choices about algorithms for us, providing state-of-the-art solutions, and it is a production-ready library with great performance, especially on part-of-speech tagging, which exactly meets our main needs.

The first thing we need to do is to extract all adjective-noun pairs from reviews. There are mainly two kinds of grammatical structures that give us adjective-noun pairs. The first grammatical structure, also the most common one, is "ADJECTIVE + NOUN" (E.g. "friendly staff"), and the second one is "NOUN + Verb + ADJECTIVE" (E.g. "the service is great", "the food tastes good"). These two kinds of grammatical structures seem to be very simple and straightforward but it is not easy to extract from reviews in practice. Because our

target words can be far away from each other and separated by irrelevant nouns, adverbs, prepositions, and so on in real life reviews. There are many situations where we have conjunctions in reviews as well. For example, we should output four adjective-noun pairs (i.e. "good, service", "great, service", "good, environment", "great, environment") for the sentence "Both service and environment are good and great." In addition, there is a negation grammatical structure problem to be solved. Considering the example of "the food is not bad", if we only extract adjective-noun pair, it will give us "bad, food" which is negative, but our target pair should be "not bad, food" which is positive. To solve this problem, we should extract corresponding adverbs for all adjectives if there are some.

There are three main issues we mentioned above: continuity, conjunction, and negation. Our solution is based on two main linguistic features in spaCy: part-of-speech tagging and dependency parsing. spaCy will help us to generate a syntactic tree that has part-of-speech tagging and syntactic dependency in it for every sentence in every review. Then we will traverse syntactic trees for every review to extract all adjective-noun pairs.

After we get all adjective-noun pairs from reviews, we should only consider pairs that are relevant to five aspects we mentioned above. Firstly, we will convert nouns in adjective-nouns pairs to their base format. For example, "fish", "Fish", "fishes" will be converted to "fish". This conversion is done with the help of spaCy, which is based on WordNet[7] and Lexiconista. We have 171,318 different nouns in our training set One thing to notice is that, pronouns are converted to the same special base format "-PRON-" in spaCy, which has the largest appearing times, 4,901,856 times in total, in our training set. Before we start to aggregate relevant nouns in every aspect, we will do a simple filtering, that is removing all pronouns, and removing all nouns that appear less than 1,000 times in our training set. Because we think those nouns have little influence in describing how good a business is. After filtering, we have 2,516 different nouns to be considered in our next step.

To find relevant nouns in those five aspects, we need to convert all nouns to word vectors. Then we can perform some distance measurement (E.g. Euclidean distance, cosine distance) among word vectors to decide whether they are similar or not. The word vectors model we use is the "en_core_web_lg" model in spaCy. The model contains English multi-task CNN trained on OntoNotes 5[3], with GloVe[8] vectors trained on Common Crawl. It provides 300-dimensional GloVe vectors for over 1 million terms of English. After we convert all nouns to word vectors, we will use cosine similarity over those vectors as our semantic similarity. Because most of nouns are outliers which are irrelevant to any aspect of our five aspects, clustering algorithms will not work in our case. We develop an algorithm that will take a few initial nouns in every aspect as input, find words that have more than a certain similarity (0.7 in our case) with any words in our initial words set, expand our words set by adding those words, and keep doing the process until there is no new relevant word being found. Finally, we get 354 relevant nouns in five aspects, 333 nouns in "Food", 7 nouns in "Environment", 4 nouns in "Costs", 8 nouns in "Service", 2 nouns in "Location". There are so many words in "Food" because besides general food nouns like "drinks", "buffet", "lunch", there are specific food nouns like "peach", "curry" as well.

In the end, we need to perform sentiment analysis on those adjective-noun pairs whose noun's base format is in our relevant nouns set. After careful consideration and comparison, we decide to use TextBlob as our sentiment analysis tool. Sentiment analysis in TextBlob is based on NLTK and Pattern[10]. TextBlob has a very large subjectivity lexicon for English adjectives. Polarity calculation in TextBlob will take negation and adverbs into account. Every adjective will have a polarity (negative/positive, -1.0 to +1.0) after sentiment analysis. For example, the pair "bad, movie" will get a polarity of -0.699 while the pair "not bad, place" will get a polarity of 0.349. After polarity calculation on every relevant adjective-noun pair, we will aggregate adjective-noun pairs in reviews with same business ID and calculate the average polarity on every aspect of that business.

4 EVALUATION

4.1 Evaluation of Retrieval

The evaluation is performed on the 20% testing data. For each review, all of our models will predict a rating and a ground truth rating. For these testing results, we consider the following metrics to evaluate.

4.1.1 Rooted Mean Square Error. Rooted Mean Square Error (RMSE) is a method used to evaluate the accuracy of predictions. The lower the score is the better. However, we cannot rely all the evaluation on this metric, since our purpose is not to predict the exact rating number, but to retrieve a set of businesses that may be recommended. Therefore, although RMSE can represent the model quality in some cases, there is some mismatch between the meaning of RMSE and our purpose.

One another problem is that RMSE will no longer be valid after we perform the standard derivation normalization, since every user has a different scale, we cannot directly calculate the mean error upon the prediction results.

4.1.2 Receiver Operating Characteristic Curve, Area Under Curve. The Receiver Operating Characteristic (ROC) curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied[5]. Area under Curve (AUC) is one popular way to analyze the quality of the model with a ROC curve. In a recommendation system, AUC can represent the performance of the model regardless of how the threshold of retrieval is determined and what value it is. The AUC score is semantically the probability that a positive sample has a higher score than a negative sample.

The ROC curve is created by plotting the true positive rate against the false positive rate at different possible threshold settings. True positive rate is represented by TP/P , which is also called recall. And false negative rate is represented by FP/F, which is also called fall-out. The area under the curve is calculated accumulatively.

The key point we need to determine if we need to use AUC to evaluate our model, is that we need to convert our ground truth, which is rating from 1 to 5 (-5 to 5 after normalization), into binary label, i.e. true or false, to apply the AUC metrics. In our case, we use the users' average rating as threshold, whichever reviews with rating higher than his threshold is considered positive, and any below is considered negative. If needed, this threshold can also be

Table 1: Experiment of Parameters on LightFM

NumComponents	Loss Function	Learning Rate	NumIterations	Lambda	RMSE	AUC
50	bpr[9]	0.1	50	1e-5	1.441	0.672
80	bpr	0.1	50	1e-5	1.441	0.672
50	**logistic**	**0.1**	**50**	**1e-5**	**1.443**	**0.677**
50	warp	0.1	50	1e-5	1.423	0.657
50	bpr	0.1	50	5e-5	1.448	0.668
50	bpr	0.1	50	5e-6	1.443	0.671

Table 2: Experiment of Parameters on Memory-Based Hybrid Model

K in User-Based Model	K in Item-Based Model	RMSE	AUC
1	3	1.492	0.461
2	4	1.486	0.457
3	**5**	**1.487**	**0.476**

Table 3: Experiment of Parameters on ALS Model

Rank	NumIterations	Lambda	RMSE	AUC
2	24	0.3	1.576	0.730
2	20	0.35	1.592	0.720
4	**24**	**0.3**	**1.459**	**0.733**

set as the median rating of the user, or the 80th percentile of the ratings, etc.

Also, AUC can be calculated separately for each user, for a higher accuracy and make more sense without normalization. However, this method does not work on this data set. Out of the 134k users in testing set, 109k of the test cases have only one kind of label when mapped from ratings to binary labels, in which case the ROC curve is not defined. Therefore, although this metric variation makes more sense and shows a higher performance on the limit testing data available, it cannot represent the whole model.

5 RESULTS

5.1 Memory Based Hybrid Model

In the hybrid model, we experimented several configurations including different hybrid combination methods, and different parameters. In this model, the most important parameters are the number of nearest neighbours retained in both model. Table 2 compares the performance of different settings of these parameters.

From table 2 we can see that although this model makes sense in theory, its performance on recommendation, especially the AUC score is not satisfactory, which is below 1. This model is treated as a baseline for our other models.

5.2 Alternating Least Square Model

This model is implemented with the Spark MLLib. Three important parameters are rank, lambda (L2 penalty) and number of iterations. Table 3 shows some results about the best :

5.3 LightFM Model

The LightFM model is an hybrid model of implicit matrix factorization and content-based model. In the Python implementation, AdaGrad[1] is used to optimize the results, and several parameters are available, including number of latent variables, loss function, learning rate, L2 penalty for users and items, number of iteration, etc. Table 1 shows the impact that different parameters have on the performance of the model.

6 CONCLUSION

During the experiments, we found that the LightFM model has a higher RMSE than the ALS model, and the ALS model has a higher AUC than the LightFM model. Therefore, a combination by making a union of the top recommendations from each model will probably give better results. We will use combined results as our recommendations. Our sentiment analyzer will provide independent ratings in five aspects for every business, so users can sort recommended businesses according to their preference by any of five aspects.

7 APPENDIX

Source codes and individual contributions can be found from here.

REFERENCES

[1] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[2] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear* (2017).

[3] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, 57–60.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[5] Maciej Kula. 2015. Metadata Embeddings for User and Item Cold-start Recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015. (CEUR Workshop Proceedings)*, Toine Bogers and Marijn Koolen (Eds.), Vol. 1448. CEUR-WS.org, 14–21. http://ceur-ws.org/Vol-1448/paper4.pdf

[6] Maciej Kula. 2017. Spotlight. https://github.com/maciejkula/spotlight.

[7] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[8] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[9] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[10] Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *Journal of Machine Learning Research* 13, Jun (2012), 2063–2067.