香港城市大學
City University of Hong Kong

City University of Hong Kong
SDSC4016 Fundamentals of Machine Learning II
2023-2024 Semester B

Building a Recommender System Based on the Yelp Dataset

Team members:

JIN Yiyao (57124923)

contact: yiyaojin2-c@my.cityu.edu.hk

LIU Jiaruo (57124542)

contact: jiaruoliu2-c@my.cityu.edu.hk

1. Motivation

A recommender system, or a recommendation system, is a subclass of information filtering system. It is an artificial intelligence algorithm that uses big data to suggest items most appropriate to a particular user. Recommender systems are trained to understand the preferences, previous decisions, and characteristics of users and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases.

Recommender systems drive personalized user experiences, deeper customer engagement, and robust decision support tools in retail, entertainment, healthcare, finance, and other industries. On some of the largest commercial platforms like Amazon, recommendations account for as much as 30% of the revenue. A 1% improvement in the quality of recommendations can translate into billions of dollars in revenue.

While there are a vast number of recommender algorithms and techniques, most fall into these broad categories: collaborative filtering and content-based filtering. Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. Content-based filtering uses item features to recommend other items similar to what the user likes based on their previous actions or explicit feedback. In this project, we want to leverage their respective strengths and combine them to build a hybrid recommender system.

2. Background of Related Works

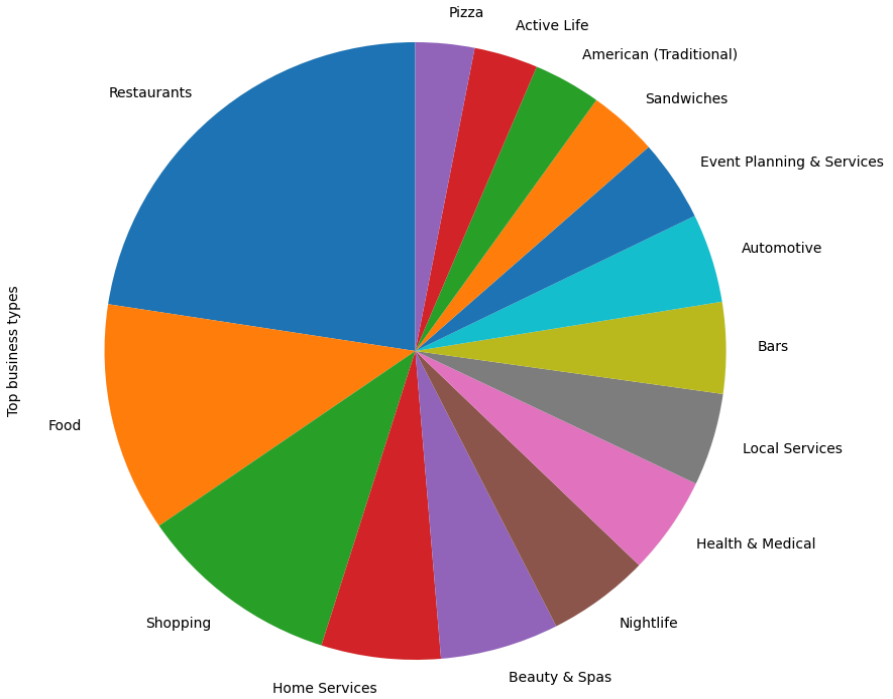
Applying machine learning to building recommender systems has a long history and is a broad field both in width and in-depth. Among all the methods used, one line of research is specifically focused on using graph neural networks (GNNs) to tackle tasks of recommendation. One reason for choosing this method is that most user-item interaction data have graph structure inherently, and GNN have good performance on learning this kind of graph data. Another reason is that GNN explicitly encode collaborative signal and show an improvement in comparison to traditional methods that only learn the implicit signals (Wu et al., 2022).

According to Wu et al. (2022), GNN based recommender systems can be divided into four categories, user-item CF, sequential recommendation, social recommendation, and KG-based recommendation. Here we focus on the user-item CF field. In June, 2019, Haoyu Wang and his team proposed the framework of DGCN-BinCF which uses knowledge distilling to yield better result than traditional graph convolutional networks. In July, 2019, Xiang Wang and his team proposed the Neural Graph Collaborative Filtering (NGCF) method, which leverages high-order connectivity between users and items to propagate embeddings. In July, 2020, Xiangnan He's team proposed the LightGCN model based on NGCF, which removed the redundancy of feature transformation and nonlinear activation in propagation layers and further improved performance. LightGCN is also the model we are going to adopt in our approach.

Another line of research in the field is applying machine learning to content-based filtering and leverage the reviews and other text information. There are word-based methods like ConvMF (Kim et al., 2016), DeepCoNN (Zheng et al., 2017), topic-based methods like HFT (Julian and Leskovec, 2013), RBLT (Tan et al., 2016), and also sentiment-based methods including EFM(Zhang et al., 2014), JMARS (Diao et al., 2014) and others. We will also be implementing a content-based filtering method based on text information to aggregate with the aforementioned LightGCN model.

3. Dataset

The data we use is from the 2022 Yelp Open Dataset, which is a subset of Yelp's businesses, reviews, and user data. This dataset contains a total of 150,346 distinct businesses, 1,987,897 distinct users, and 6,990,280 reviews. A column named ‘categories’ in the business dataset indicates the business type. Each business can fall into different categories. For example, a business named ‘Middle East Delicious Food’ belongs to both ‘Food’ category and ‘Restaurants’ category. The pie chart that shows the top 15 business types is shown below.



We found that among these businesses, restaurants account for the highest proportion. To reduce memory usage and running time, we narrowed down our recommendation candidate set to include only restaurants.

4. Methods for Implementation

4.1 Description

In this project, we will build a collaborative filtering recommender system and a content-based filtering recommender system, respectively, and further aggregate the result to build a hybrid recommender system. The specific model selection and reasons will be discussed in the respective introduction section.

4.2 Collaborative Filtering: LightGCN

4.2.1 Introduction to the Model

For collaborative filtering, we leveraged the existing model of LightGCN, which is a state-of-the-art model proposed by Prof. Xiangnan HE and his team in 2020, and has well-recognized performance that transcended the baseline methods and many existing methods in the field.

We directly used the original authors’ code on GitHub for the model-building, both the PyTorch version and the TensorFlow version were experimented with in order to compare the

potential differences in performance, but we eventually used only the PyTorch implementation because the TensorFlow version used was outdated and some functions were deprecated. It is worth pointing out that the data used for the original paper was Yelp 2018 Dataset, and our implementation makes use of the newest Yelp 2022 Dataset.
The LightGCN model can be roughly divided into three parts: light graph convolution, layer combination, and prediction.

During the light graph convolution stage, each user and each item is initially assigned an embedding. Then, for each convolutional layer, the user(item) will get a new embedding by integrating the information from its neighboring item(user) from the previous layer. Note that this process does not include integrating its own embedding from the previous layer. In the end, we get k+1 embeddings for each user and each item, where k is the number of convolutional layers.

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}$$
$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}$$

In the layer combination stage, for each user and each item, the k+1 embeddings were summed together using a uniformly weighted sum to yield a final embedding.

$$\mathbf{e}_u = \sum_{k=0}^{K} \alpha_k \mathbf{e}_u^{(k)}; \quad \mathbf{e}_i = \sum_{k=0}^{K} \alpha_k \mathbf{e}_i^{(k)}$$

For prediction, to get user A's rating for item B, the final embedding of A and B were multiplied and the dot product of them is the final predicted rating.

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i$$

4.2.2 Data Processing

The model of LightGCN takes the positive interactions between users and items as input, and outputs a top k recommendation list for each user, where k is a self-defined parameter. Each entry without a positive interaction (no matter whether it is a negative or an empty entry) is treated as a negative interaction.

Thus, the data we need is users' ratings on different restaurants. We made use of the Yelp Academic Review Dataset, which is a .json file provided in the Yelp Open Dataset. Specifically, the 'user_id', 'business_id', and 'stars' columns in the data were employed.
In the original paper for LightGCN, the authors treated all ratings (from 1 star to 5 stars) as positive interactions. However, after trying both methods, we decided to only treat the ratings from 3 stars to 5 stars as positive interactions as they yielded better performance.

Sparsity is an inherent problem of review data, and after observation, we found that the Yelp data did come with this. To deal with it, we kept only the users with more than 20 ratings so that there can be enough instances in both the training and testing set. This method is supported by the original paper since after looking into their code, it was obvious that they

have also filtered their data based on the number of ratings. The total number of users left in the pool was 31788 after filtering.

Another inherent problem with data used for collaborative filtering is the cold-start items. In this specific case, the 'cold-start items' refer to restaurants with either no interactions or very little interaction. The solution to this issue was mentioned in the paper of LightGCN where they filtered out the cold-start items in the testing set. To realize this, we first created a list of the business IDs of the restaurants with more than 20 ratings, then compared the list with the restaurants within our testing set. After filtering, 46398 restaurants were left in the pool. Even after all the methods mentioned above, the data we obtained still have a sparsity of around 0.0004. It is low but the usability is ensured since we have checked the sparsity for other datasets used in the paper of LightGCN, and some like Gowalla have a sparsity of similar order of magnitude.

The training and the testing set are formatted in a way that each user and their ratings compose a row. We further divides each row so that roughly 50% of the ratings are in the training set, and the other 50% of the ratings are in the testing set. This division ratio is chosen based on severe sparsity. As mentioned before, all entries without a positive interaction is considered as negative interaction, but this may seem unreasonable if we come to think about it. A large proportion of the data is composed of empty entries, that means the user have not visited the restaurant. We need to ensure that there is enough data in the testing set so that the evaluation result is not largely degraded due to the lack of test samples.

4.3 Content-based Filtering

4.3.1 Introduction to the model

A content-based (CB) recommender suggests items to users based on the attributes or features of the items that the user prefers. The content-based model can make personalized recommendations, so it does not need data about other users. In this project, we want our content-based model to predict each user's rating of a certain restaurant. We follow the following three steps to build the CB model:

Firstly, we dived into the item(restaurant)-matrix building. The rows of the matrix represent each restaurant, and the columns show restaurants' features. We leveraged two kinds of data to form the restaurant features: the 'attributes' data in the business dataset, and the 'text' data in the review dataset. The 'attributes' data gives explicit restaurant attributes, and the 'text' data shows users' comments on restaurants. We processed these text data and converted them into vectors. The detailed processing steps will be elaborated in the next section.

Secondly, we built the restaurant similarity matrix. Based on the vector representation of restaurant features, we calculated the cosine similarity of each restaurants pair. The similarity of restaurant A and B is calculated as follows:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The value of cosine similarity ranges from -1 to 1. The larger the value, the more similar the two restaurants are.

Lastly, we built the user profiles. For each target restaurant, we found each user's top k most similar restaurants to it, and used the average of the user's previous ratings for these k restaurants as the predicted rating of the target restaurant. We used mean squared error (MSE) as the evaluation metrics and tuned the value of k to find the optimal value.

4.3.2 Data Processing

Since the content-based model also faces the cold start problem, we do the same filtering process as the LightGCN, to only keep 'active' users and restaurants, which have review records more than 20 pieces.

For the 'attributes' data, the original presentation in the dataset is an extensive dictionary, with over 1.2 million business attributes like hours, parking, availability, and ambience (casual, romantic, etc.). We split the aggregate attributes column into many separate columns, with each representing a dummy variable that has a value of 0 or 1. Other numeric variables in the dataset that also show restaurants' features, like latitude and longitude, were standardized to have mean=0 and variance=1. Then, a truncated Singular Value Decomposition (SVD) was followed to reduce the features' dimension. We kept the first 10 singular values, which could explain more than 90% of the total variations.

After dealing with the attributes data, we split the whole dataset into a training set, a validation set, and a test set. Our splitting was based on the chronological order, that is, for each user, we left out the last review record in the test set, and the second last review in the validation set. We trained the model on the training set and then used the validation set to tune parameters. Finally, we combine training and validation sets to train the model with the optimal parameters, and evaluate the model performance on the test set.

Then, we started to do the 'text' data processing. To avoid data leakage, we only used the text data in the training set. The word cloud in the training set is shown below:

The first step in natural language processing is to remove punctuations, special words (like # & *), and stop words (like the, and, on, etc.). Then, we converted all numeric information in the text into textual representation. Stemming and lemmatization were followed to unify words that look different but have the same meaning so as to facilitate subsequent processing

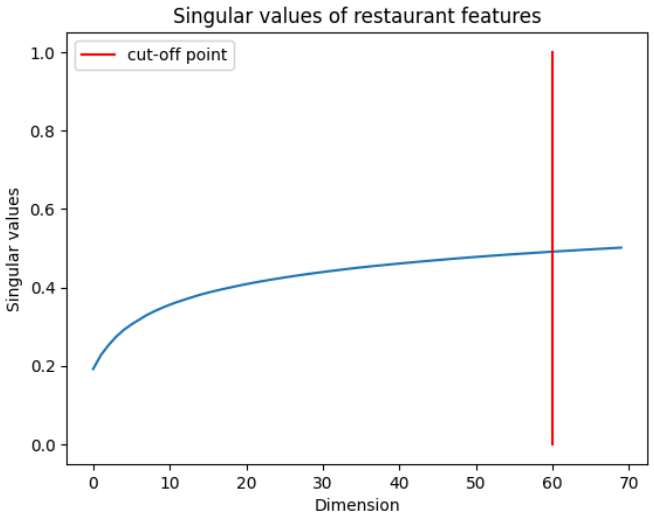
and analysis. After this, we converted the text data into vectors. We applied two different vectorization methods: TF-IDF (term frequency-inverse document frequency) and LSI (Latent Semantic Indexing). TF-IDF vectorizer uses similar logic to one-hot encoding. First, it calculates the TD-IDF value for each word in the training text data. TF-IDF value is a product of TF value and IDF value, where TF indicates the frequency of a word i to a document d, and IDF indicates how common or rare a word i is in the entire corpus.

$$TF(i,d) = \frac{number\ of\ times\ i\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(i,d) = log(\frac{total\ number\ of\ documents\ N}{number\ of\ documents\ containing\ i})$$

$$TF - IDF(i,d) = TF(i,d) \times IDF(i,d)$$

Then, for each review record, instead of using value 0 and 1, TD-IDF vectorizer used the tf-idf value to encode. This resulted in an enormous dimension (4311) vector space. So we also performed truncated SVD to reduce the dimension to 60 which explains more than half of the variations.



LSI can identify hidden topics in different texts. It uses SVD to decompose the TF-IDF matrix. The three matrices after SVD are document and topic, topic and word meaning, and word meaning and word. Through different interpretations of the three matrices, meaningful interpretation can be achieved on the basis of reduced dimensions. The number of topics is a parameter in the model that needs to be tuned.

4.3.3 Ensemble CB model

We have built three content-based models: the text and tf-idf-based model, the attributes-based model, and the text and LSI-based model. We further ensembled the three models to include more information in a complete model. The detailed ensemble method will be discussed in the 'results' section.

4.4 Hybrid Model

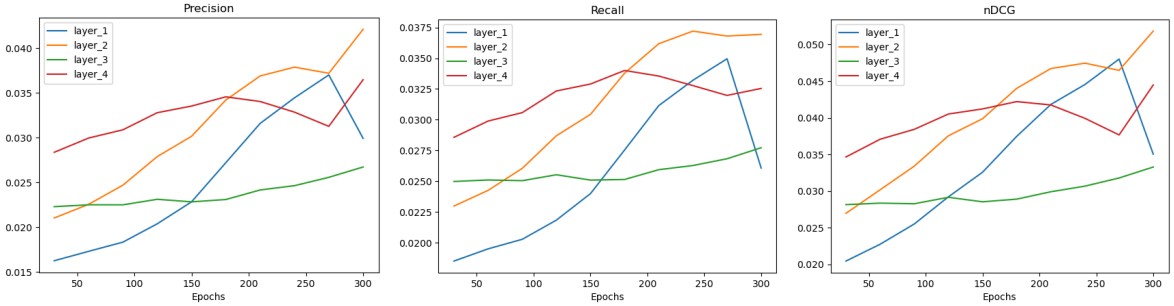
Since both the collaborative filtering and the content-based filtering model yielded the results of the top 20 recommended restaurants for each user, we will combine the two recommended list to produce the hybrid result.

Due to the slight difference in the procedures of data processing, the users covered by the two models have intersection but are not the same. So in order to successfully combine the two results, we do an update to the collaborative filtering results based on the results of content-based filtering. For each user in the LightGCN result, we would check whether he/she is also present in the content-based filtering result, if yes, then the original recommendation list for this user will be updated.

Since the format of the output in the hybrid model is the same as in collaborative filtering model (LightGCN), we chose the same metrics as LightGCN to evaluate the hybrid model.

5. Results and Observations

5.1 Collaborative Filtering: LightGCN



The result of applying LightGCN with 4 different number of convolutional layers is shown above. The k value we chose is 20, and the evaluation criteria are precision@20, recall@20, and nDCG@20. We can see that the model with 3 convolutional layers performs the worst, while the model with 2 convolutional layers performs the best, with the highest result in all of the criteria by the end of 300 epochs.

	Precision@20	Recall@20	nDCG@20
1 Layer	0.0370	0.0350	0.0480
2 Layers	0.0421	0.0372	0.0518
3 Layers	0.0267	0.0277	0.0333
4 Layers	0.0365	0.0340	0.0445

The graph above shows the best performance for each criterion for each model structure. It is shown that the layer-2 model outperformed all other models. The numerical results may seem low, but are actually in line with the experimental results of the original paper using the Yelp2018 dataset.

In order to further illustrate the strength of LightGCN on the Yelp2022 dataset, we used its predecessor, the NGCF model, on the same data to yield a baseline performance.

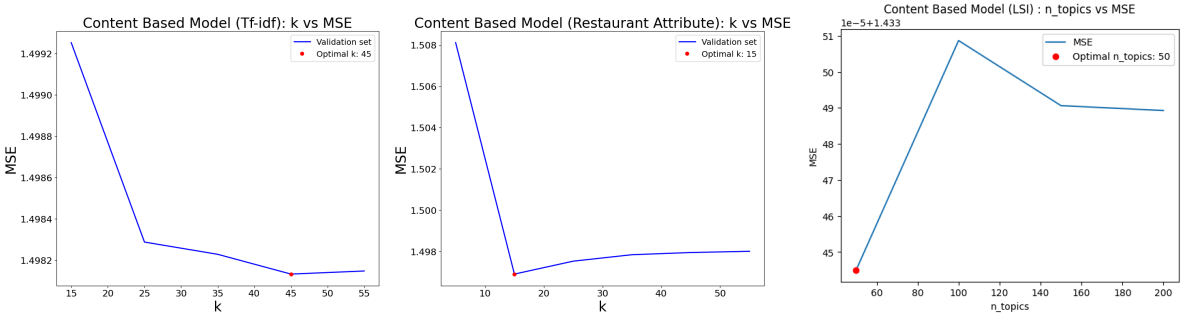
	Precision@20	Recall@20	nDCG@20
NGCF-3	0.0370	0.0296	0.0449
LightGCN-2	0.0421	0.0372	0.0518

According to the paper of the NGCF model, the structure with 3 convolutional layers yielded the best performance on the Yelp2018 dataset, so we applied the 3-layer model on Yelp2022. The results are shown in the table. Precision@20, recall@20, and nDCG@20 had risen 13.8%, 25.7%, and 15.4% respectively.

5.2 Content-based Filtering

5.2.1 Parameter Tuning

There are 2 parameters need to be tuned. The first one is k, which means finding a user's top k most similar restaurants to the target one. The other one is n, which is the number of latent topics in the LSI-based model. After tuning, we find the optimal k for the text and tf-idf-based model is 45, the optimal k for the attributes-based model is 15, and the optimal n for the text & LSI-based model is 50.



5.2.2 Final CB Model Result

The results of these three single models are shown below.

	Attributes-based	text & tf-idf-based	text & LSI-based
MSE	1.4392	1.4354	1.4335

We applied a weighted average ensemble of the three prediction matrices, where the weights of each model are the inverse of its MSE score. By doing so, we obtained a smaller MSE score of 1.4296, indicating better recommendations.

5.3 Hybrid Model

	Precision@20	Recall@20	nDCG@20
LightGCN	0.0421	0.0372	0.0518

Hybrid Model	0.0457	0.0489	0.0413

After comparing the results from different combination methods, it was found that taking the first 10 recommendations from the LightGCN model and the first 10 recommendations from the collaborative filtering model would give the best performance among all. For both precision@20 and recall@20, we saw an improvement from the original LightGCN model, while for nDCG@20, the performance degraded. This may be due to the messing up of order of items during concatenating.

In general, the aggregation is perceived as successful since two of the three metrics were improved.

6. Discussions

In this project, we implemented a hybrid model for building a recommender system, and this model combines the results from a collaborative filtering model and a content-based filtering model. While both the two models show decent performance standalone, the aggregated result shows an improvement. Due to the time limit, we did not go out of our way to change some hyperparameters like the weights in weighted sum of the layer combination stage of the LightGCN model, since the authors stated that the existing parameters have the best performance. However, since we implemented the model on a different dataset, the parameters may yield different performances and tuning is worth trying. In addition, the hybrid method used in the project is quite naive and straightforward, other more complex means may yield even better results.

As novices who touch upon this topic for recommender systems for the first time, we have found this topic very attractive throughout the process. Even though the task of recommending to users falls into the category of predicting, its results are hard to assess. Users preferences can vary rapidly and some people may give positive response to trying out new things while others do not. It remains unclear whether a list of recommendation is of use or not until it is actually promoted to the targeted user. Nevertheless, researchers are still devoted to finding the best algorithm for recommender systems and making efforts to translate machine learned patterns into revenue.

In the project we tried to develop a model that recommends places to dine for customers. Similar implementations in reality can make people’s life more convenient and hopefully shorten the time for making decisions. In general, this technology can promote living standards and create a more efficient style of living.

Other than doing business and generating revenue, recommender systems can also be applied to other aspects of society. For example, the technology can help content creators online to reach a wider range of audience and help them gain visibility.

If implemented properly, recommender systems can even help to promote attitudes, like advocating environmentally friendly products. In conclusion, it is certainly a promising field of study.

7. References

Nguyen, L. V., Nguyen, T. H., & Jung, J. J. (2020, October). Content-based collaborative filtering using word embedding: a case study on movie recommendation. In *Proceedings of the international conference on research in adaptive and convergent systems* (pp. 96-100).

Si Gao, Xinli Gu, Xirui Fu, and Wenxin Zhang. 2020. Yelp Restaurant Recommender System. https://github.com/Yelp-Recommender-System/FancyYelpers

Srifi, M., Oussous, A., Ait Lahcen, A., & Mouline, S. (2020). Recommender systems based on collaborative filtering using review texts—a survey. *Information*, *11*(6), 317.

What is a Recommendation System? (n.d.). NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/recommendation-system/

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: a survey. ACM Computing Surveys, 55(5), 1-37.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In SIGIR. 639–648.

Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In SIGIR. 165–174.