Movie Recommendation on Douban Datasets

Baichuan LIU, 16307130214

Binghui XIE, 16307130163

Jiayang CHENG, 17307110242

1 ABSTRACT

Modern audiences are inundated with choices. Stream media providers offer a huge selection of movies, with unprecedented chance to meet users' needs. Matching users with most appropriate movies is not trivial, yet it is a key in enhancing user satisfaction.

Precise item recommendation to target users has long been a sought-after task in the academic circle and among enterprises. Modern recommender systems evolved differently in these years. Such systems were usually constructed based on matrix factorization method, one of the most popular models, and newly-born knowledge graph model also proved itself important.

We implement Probabilistic Matrix Factorization [1], Factorization Machines [2] and Ripple Network [4] to deal with the recommendation problem. The three major models we choose are developed in time sequence. PMF is one of the classic models of early matrix factorization attempts. FM, and various methods derived from it, conduct matrix factorization with feature engineering. And Ripple Network, though proposed recently, has shown its significant capability on solving recommendation tasks by utilizing enormous entity-relation-entity triples in real social networks with deep learning methods. When working on FM models, Skipgram-based DeepWalk [5] and Item2Vec [6] algorithms are applied to pre-train the model and thereby help to achieve higher convergence rates. We applied the algorithms and models mentioned above, and came out with considerably good experimental results.

2 INTRODUCTION

2.1 An Outline for Social Network Mining

Nowadays, tens of millions of people share their experience and viewpoints, posting messages and doing all kinds of daily activities online, making the Internet a young and dynamic place. Enormous information is collected, organized and released by citizen journalists, and in the meantime read, responded and propagated by others. Social media platforms provide us with unprecedented opportunities to analyze human behavior patterns, letting us have a better understanding to humanity. Such missions had been impossible before the necessary technologies happened.

Social media mining provides a computable way to represent and measure the virtual world of social media communities, and establish models to help us understand interactions within them.

2.2 Douban Social Network

Similar to *IMDb* and *Rotten Tomatoes*, Douban is renowned for its movie rating service in Mandarin communities. A large amount of other user generated content, book reviews or music comments, creating vast data flow every second. People chat, make friends, write reviews and record their interesting moments of life on Douban. Almost all the information and topics are created and developed by its users.

By 2012, the number of monthly unique visitors (MUV) on Douban has been reportedly exceeding 100 million, with 160 million daily visits (PV). With hundreds of millions of users, vast amount of information is created and being created, making training recommender systems possible. However, following **the power-law distri-**

LIU, B., XIE, B., and CHENG, J., 'Movie Recommendation on Douban Datasets,' *Game & Puzzle Design*, vol. 1, no. 1, 2015, pp. 1–9.

bution, the data sparsity still exists, since most users have little or no interacting records, while a small group of them creates most information. Therefore, perfecting recommender systems on Douban social networks is not only challenging, but of great practical significance.

3 DATA COLLECTING

3.1 Web Crawler Design

We devised an **automatic parallelizable** web crawler to gather information from Douban website. In order to provide a complete picture of the social networks in reality, Breadth-First Search was adopted. 10 users were sampled randomly from the website, and their following users were added to a queue, and then the following users of these users, so on and so forth. While collecting data, a proportion of users were abandoned stochastically to achieve better comprehensiveness of sampling and guarantee efficiency.

Our crawler is automatic, since the implementation of Verification Code Recognition enables automatic recognition, and help us circumvent the anti-webcrawler mechanism that occurs once in a while when collecting data. Moreover, since we easily extend it into a multiprocessing version, it is parallelizable. As a result, the crawler can run efficiently.

3.2 Verification Code Recognition

One of the common obstacles in collecting data is the anti-crawler mechanism on the target website, which forces the program to enter a verification code on a box whenever it finds anomaly. Fortunately, the verification code pictures on Douban are relatively simple that they can be easily recognized through two processes.

First, *pytesseract* [7] is applied so as to recognize alphabets from the picture. Alphabet combinations obtained are usually meaningless, which necessitates further process.

Second, *SpellChecker* is used to project the alphabet combination into a meaningful word. The projection might not always be correct. However, as long as the accuracy is larger than 0.1, it is acceptable in practice. All we need to do is repeat the above two steps for several times if it fails to generate correct answer.

Combining the two steps, we found the algorithm achieves an accuracy of 0.28, which means it only tries 3.57 times on average before giving the correct code.

3.3 Parallelizability and Multi–Processing

Our web crawler is parallelizable. It is possible to apply multiprocessing scheme through some small changes in its functions. Package *multiprocessing* was used to provide system level support.

3.4 Data Storage and Retrieval

The data are gathered from the Internet and then stored into MYSQL database. Fast retrieval is possible by *pymysql*.

4 DATASET

4.1 Statistical Information

An overview of the datasets we use is given in Table 1.

On user graphs, both the SMALL set and the BIG set are sparse, for their densities are far smaller than 10%. The SMALL set is even sparser, which may partly account for our models' relatively bad performance on it.

Moreover, the introduction of *entity–relation–entity* triples may help to overcome the difficulty of cold start brought by data sparsity, as we can see in the table.

4.2 Visualization

We used *Gephi* [**?**] to visualize the user graph.

Fast Unholding algorithm [8] was applied to discover communities in the user graph. As shown in Figure 1, Different colors represent different communities, and the sizes of vertices are proportionate to their degrees.

Note that there are enormous isolate nodes on the edge of the figure (see Figure 1), which corresponds with the **power–law** distribution. Eliminate these nodes, and we can see the core communities in the centre (see Figure 2). Several core communities are closely surrounded by many. In addition, the closer we get to the centre, the denser it is.

The *Fast Unholding algorithm* fails to reflect the relationships between different communities, as it doesn't focus on the distance between nodes. To obtain better effect, we eliminated isolate nodes, and apply *Force–Directed Layout algorithm* [9] to

Table 1. Stastical Information

	USER	MOVIE	GRAPH edge	GRAPH density	ENTITY	RELATION	TRIPLE
SMALL	1000	1000	574	0.115%	4528	9	10836
BIG	3897	48848	84438	1.112%	115013	9	717008

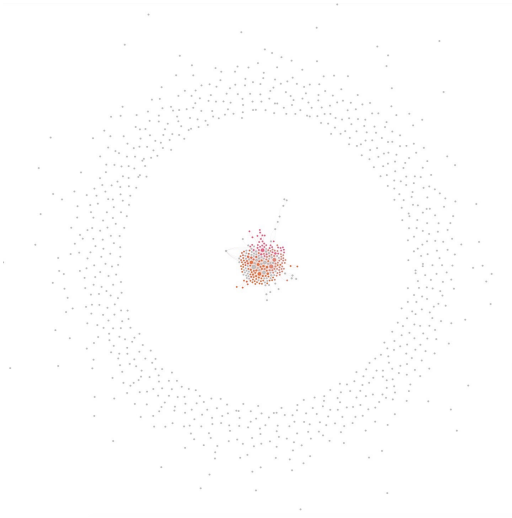

Figure 1. User graph generated by Fast Unholding algorithm.

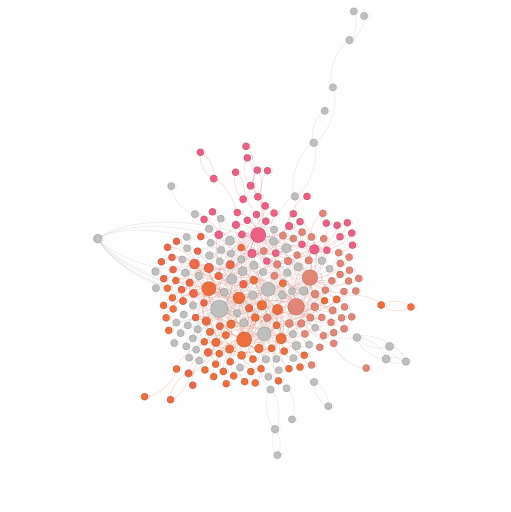

Figure 2. User graph generated by Fast Unholding algorithm.

user communities (see Figure 3). The visualization result from this algorithm outperforms the previous one, since nodes with closer relationship (more connected edges) tends to stay closer, and thus the relationships between nodes are displayed in the plot.

Figure 3. User graph generated by Force–Directed Layout algorithm.

4.3 Network Analysis

4.3.1 Degree Distributioin

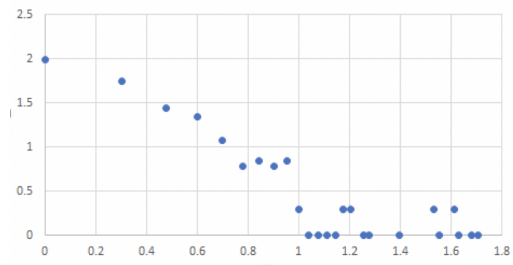

Figure 4. Degree distribution.

The degree distribution (see Figure 4) with double logarithmic coordinates obeys the power–law distribution.

4.3.2 Average Clustering Coefficient and Path Length

The average clustering coefficient is 0.117, and the average path length is 3.994.

4.3.3 Similarity with Preferential Attachment Model

Our analysis shows that the Douban network has low clustering coefficient, relatively short average path length, and its degree distribution obeys the power–law distribution.

Although we sampled the data from the real world community, the three metrics above indicate that the Douban network resembles the network generated by *the preferential attachment model*.

If our sampling is unbiased, there exists strong Matthew Effect in Douban network. In reality, and more often than not, people use Douban to get movie information rather than to socialize with others. Therefore, they may tend to follow more popular network writers than normal users, who create far less movies information. Also, low social pressure between users may account for the low clustering coefficient.

5 RECOMMENDER SYSTEMS

In this section we discuss models and algorithms we use when fulfilling the task. We also present experimental results of our approaches.

5.1 Pre–training Methods

The two pre–training methods here are both derived from SkipGram, a language model that maximizes the co–occurrence probability among the words.

5.1.1 DeepWalk

[5]

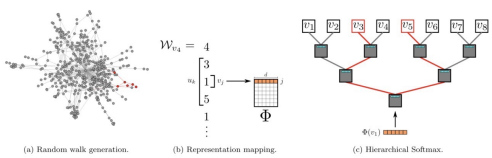

Figure 5. DeepWalk outline.

DeepWalk is an online learning method of social representations by Bryan Perozzi et al. It operates on homogeneous graph networks. Running DeepWalk on Douban network yields latent representations, and these representations **encode social relations** in a continuous vector space, which can be easily exploited by other models. This is crucial, as applying DeepWalk enables us to make the most of social information between users, which is hard to utilize by other methods.

DeepWalk consists of two parts. In general, it generates a set of short truncated random walks as the input for SkipGram (see Figure 5). The set of random walks are considered the corpus, and the graph vertices its own vocabulary.

5.1.2 Item2Vec

[6] *Item2Vec* is an item-based Collaborative Filtering method that produces embeddings for items in a latent space. Similar to DeepWalk, it also uses SkipGram as its core algorithm.

5.2 Probability Matrix Factorization

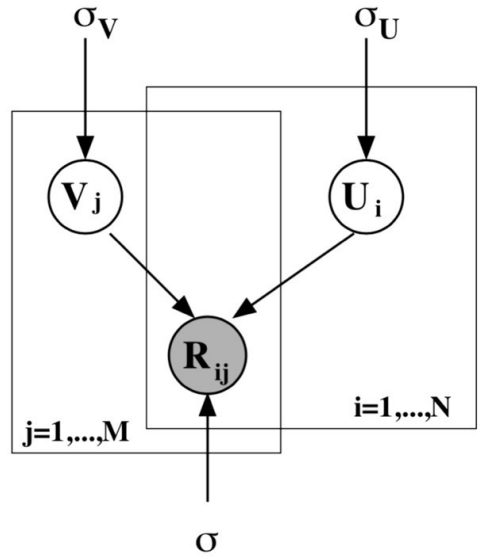

Figure 6. Matrix R is used to learn U and V.

Probabilistic Matrix Factorization is a classic approach of matrix factorization in collaborative filtering. It yields user and movie embedding matrices U and V by maximizing the log-posterior over movie and user features. Essentially, it finds the best U and V that minimizes $||R - U^T V||$, where R is the rating matrix. Once U and V is learned, we can conduct matrix product $\hat{R} = U^T V$ to predict users' ratings towards different movies.

For simplicity, and given that the datasets we have are relatively small, we adopted GD and SGD as the optimizing methods. After trying several values for the learning rate and experimenting with various values of D, we chose to use a

Table 2. Experimental Results for PMF

	NDCG(test)	RMSE(test)
SMALL	0.4017	0.3144
BIG	0.4511	0.6213

learning rate of 0.001, and $\lambda_U = \lambda_V = 0.01, D = 32$. After 450 iterations, the model converged.

As shown in Table 2, PMF achieved a RMSE of 0.3144 and a NDCG of 0.4017 on the SMALL set. Note that the density of the SMALL set user graph is only 0.115%, probably responsible for the bad performance of NDCG. As the size of dataset increases, both NDCG and RMSE increases. The NDCG on the BIG set reaches 0.4511, slightly higher than that on the SMALL set.

It should not be forgotten that PMF only makes use of rating information, yet it even outperforms FM in some cases.

5.3 Factorization Machines

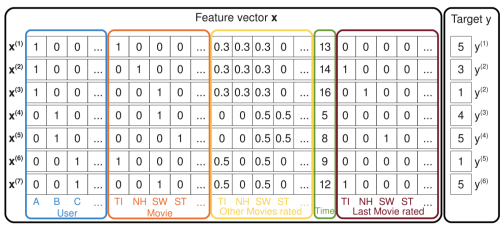

Figure 7. Different feature vectors are concatenated together.

PMF only utilizes rating information. However, there are much more dimensions of information in our dataset. For example, directors, actors, type, countries of movies, and social relations between users. It is reasonable for us to make full use of these characteristics to improve the recommendation performance.

Factorization Machines (FM) simulates most matrix factorization with feature engineering methods. Its core formula is

$$\hat{y}(X) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i+1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

where X is the input feature vector, x_i is the i-th element of X, $\hat{y}(X)$ is the predicted value when given X, and w_i, v_i are parameters.

FM has several advantages. First, as a general algorithm, there is no special requirement for input vector X, which means we can concatenate as many different feature vectors together as we like (see Figure 7). As for our task, all the features can be included in FM to enhance performance. And second, FM shows great ability to process sparse data. Even though X is sparse, its output—-low dimensional vectors v_i—- are dense, and are only of linear scale.

5.3.1 Some Improvements

Pre–training Embeddings

We applied DeepWalk and Item2Vec algorithms respectively to pre-train our FM model. To be specific, we ran DeepWalk on user graph, yielding the latent embeddings of users. By doing so, we made use of the social relations among users. Similarly, we used Item2Vec to acquire the embeddings of movies.

The introduction of pre–training embeddings sped up later training, and improved the performance.

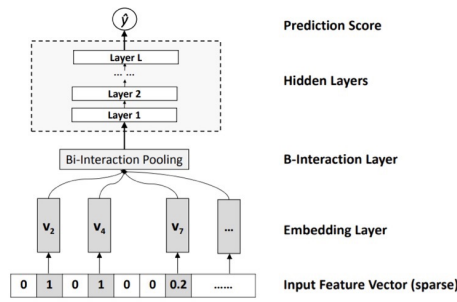

Figure 8. Neural FM structure.

Neural FM [3]

Neural FM is a deep learning extension to FM, proposed by Xiangnan He et al. It adopts neural networks to do non–linear modeling on x_i, x_j (see Figure 8), and thus achieve better performance.

5.3.2 Experimental Results

On the SMALL set, the best performance occurs when no feature is eliminated, probably because of the very limited amount of data. Under such circumstances, leaving out any feature will lead to the loss of essential information, and hence lower the performance. The results are summarized in Table 3.

With limited computation resources, dealing with all features on the BIG set is a formidable task. It is observed that the performance with no feature eliminated on the BIG set is worse

Table 3. Experimental Results of FM on Different Sets

	SMALL			BIG		
	All features	–Rate	–Year –Type	All features	–Rate	–Year –Type
RMSE(test)	1.73	2.14	2.19	1.91	0.84	0.95
NCDG(test)	0.38	0.25	0.12	0.29	0.91	0.87

Table 4. Experimental Results from Different Matrix Factorization Methods

	FM(test)	Neural FM(test)	FM(train)	Neural FM(train)	FM with DeepWalk Embedding	PMF(test)
RMSE	0.84	1.41	0.77	0.72	1.34	0.31
NCDG	0.91	0.88	0.85	0.89	0.56	0.40

than that on the SMALL set. Moreover, it is noteworthy that the performances on the BIG set when some of the features are eliminated (see Table 3 BIG -Rate, BIG -Year-Type) are considerably higher.

When pre–trained embeddings were added to our model, the performance was enhanced (see Table 4). We tried DeepWalk and Item2Vec, and found that embeddings from Item2Vec made no obvious improvement on the performance. Since DeepWalk utilizes the social relations between users, we suspect the reason is that the relations between users outweigh that between movies in terms of determining one's preference for movies.

The Neural FM outperforms FM on the training set, while it performs badly on the testing set, indicating the existence of overfitting. We tried Early Stopping and reducing the number of hidden layers to alleviate the extent of overfitting, but failed to make improvement. It might be an evidence that Neural FM is not suitable for the use of recommender system under our experimental settings, while its basic version—-FM—-would be a better choice.

5.4 Ripple Network

To address the sparsity and cold start problem of collaborative filtering, we have tried to make use of side information such as social relations or item attributes (DeepWalk and Item2Vec pre–training for FM). Another source of the side information is the knowledge graph, a heterogeneous network that takes entities as its vertices and relations between entities as its edges. There are tremendous triples in the knowledge graph (see Table **??**).

5.4.1 Details of the Model

Ripple Network is an end–to–end framework that naturally incorporates the knowledge graph into recommender systems. Similar to actual ripples propagating on the surface of water (see Figure 9) , Ripple Network stimulates the propagation of user preferences over the set of knowledge entities by automatically and iteratively extending a user's potential interests along links in the knowledge graph. The multiple "ripples" activated by a user's historically liked items are thus superposed to form the preference distribution of the user with respect to a candidate item, which could be used for predicting the final preference probability [4].

Table 5. Vast Amount of Tripples in Knowledge Graph

	ENTITY	RELATION	TRIPLE
SMALL	4528	9	10836
BIG	115013	9	717008

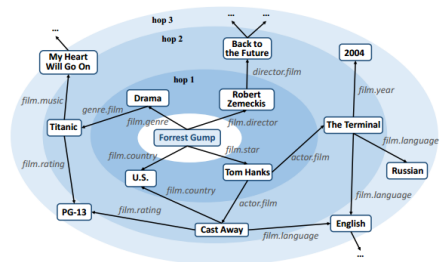

Figure 9. Ripples on the Knowledge Graph.

As shown in Figure 11, each item v is assigned an embedding $v \in \mathbb{R}^d$, where d is the dimension of embeddings. Then, each triple (h_i, r_i, t_i) is assigned a relevance probability by

$$p_i = softmax(v^T R_i h_i) = \frac{exp(v^T R_i h_i)}{\sum\limits_{(h,r,t)\in S_u^1} exp(v^T R h)}$$

After obtaining the relevance probabilities, we

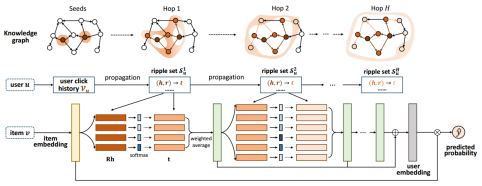

Figure 10. An Outline for Ripple Network.

take the sum of tails weighted by the corresponding relevance probabilities:

$$o_u^1 = \sum_{(h_i,r_i,t_i)\in S_u^1} p_i t_i$$

where $t_i \in \mathbb{R}^d$ is the embedding of tail t_i.

The embedding of user u with respect to item v is calculated by combining the responses of all orders:

$$u = \alpha_1 o_u^1 + \alpha_2 o_u^2 + ... + \alpha_H o_u^H$$

where α_i are positive trainable mixing parameters for measuring the importance of i-order response.

Finally, the inner product of the user embedding and item embedding is computed used to generate the predicted probability. In our setting, the probability represents how likely the user enjoys the movie.

$$\hat{y}_{uv} = \sigma(u^T v)$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the sigmoid function. [4]

5.4.2 Experimental Results

As shown in Table 6, the Ripple Network shows overall higher performance on the BIG set than on the SMALL set. This is not surprising, as the size of triples in a knowledge graph grows geometrically with respect to the size of users and items. Richer information on the BIG set helps to achieve better results.

On the SMALL set, we note that the Ripple Network tends to predict negative samples, which leads to the low Recall. Quite a lot of the prediction reaches a probability of less than 0.0001, yet none of them is over 0.97. However, the reason remains unclear.

Like Neural FM, the Ripple Network is deep learning–based; but unlike Neural FM, the Ripple Network takes full advantage of all features. As shown in Table 6, every time we eliminate a feature, the performance declines, which means every feature makes positive contribution to the model. In contrast, the performance of Neural FM is enhanced each time we eliminate some features (see Table 3).

Moreover, *Type*, of all features, shows the biggest influence on the performance. We give our explanation below. At including type information, our model is somewhat capable of recommending movies according to users' preferences for certain types, which helps to reduce the amount of candidates and hence improve the performance.

6 CASE STUDY

In this section, we chose a user, and generated several movies from the above models as our recommendation.

Figure 11. Word Cloud for the Types of Movies in the User's Viewing Record.

See Figure 11, types of movies in the user's viewing history are extracted and used to create a word cloud. The user's favorite types are *action*, *story*, *comedy* and *romance*. We then used PMF and Ripple Network to generate some recommendations.

The recommendations generated from the two models varied from each other. Note that the results from PMF focuses on action and comedy films, with little suspense film included, which coincides with the user's historical preferences. In contrast, many story and suspense films occurred in the recommendations from the Ripple Network. We then found the reason behind this phenomenon on the user's homepage.

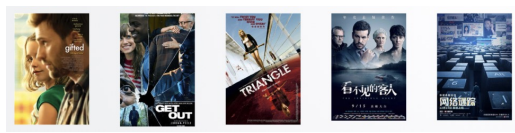

Figure 12. Wish List on the Homepage.

Table 6. Experimental Results for Ripple Network on Douban Datasets

	BIG						SMALL					
	All features	–Rate	–Director	–Actor	–Year	–Type	All features	–Rate	–Director	–Actor	–year	–Type
Accuracy	**80.69**	79.45	77.08	76.23	79.33	73.28	78.30	76.84	77.64	77.34	77.52	76.53
Precision	84.53	84.12	82.13	81.56	82.74	80.56	**89.13**	86.92	85.93	88.64	89.13	87.26
Recall	**75.05**	72.56	71.24	71.36	73.42	70.36	63.88	62.60	65.72	62.52	62.80	61.85
F1	**79.50**	77.91	76.30	76.12	77.80	75.12	74.42	72.78	74.48	73.32	73.68	72.39

Table 7. Recommendations from PMF

	MOVIE	TYPE
1	Residence Evil: The Final Chapter	Action, Science-fiction, Horror
2	Sing	Comedy, Anime, Musical Drama
3	The Wasted Times	Story, Action, Suspense
4	Hacksaw Ridge	Story, Bio, Action
5	Operation Mekong	Story, Action, Crime
6	Heidi	Story, Family, Adventure
7	Train to Busan	Action, Thriller, Disaster
8	Kapoor and Sons	Story, Comedy, Romance

Table 8. Recommendations from the Ripple Network

	MOVIE	TYPE
1	The Wasted Times	Story, Action, Suspense
2	Manchester by the Sea	Story, Family
3	La La Land	Story, Romance, Musical Drama
4	Nocturnal Animals	Story, Suspense, Thriller
5	Arrival	Story, Suspense, Science-fiction
6	Mr.Donkey	Story, Comedy
7	Doctor Strange	Action, Fantasy, Adventure
8	Billy Lynn's Long Halftime Walk	Story, War

As shown in Figure 12, the user had chosen 5 movies in his/her wish list recently. They were *Gifted, Get Out, Triangle, The Invisible Guest and Searching*, all with the tag of *story*, and 4/5 were tagged *suspense*. This somehow proved the Ripple Net's validity, as well as its ability to reflect the change in users' taste in time sequence. And PMF, since it only takes into account the rating information, failed to predict what the user wanted to watch recently.

7 CONCLUSION

By far, we have devised an efficient web crawler to gather information from Douban, visualized the data, analyzed the social network and implemented several models to build the recommender system.

The data we obtained were sparse, with the user graph density of 0.115% and 1.112% on the SMALL set and BIG set respectively. Further data visualization and network analysis revealed some important characteristics. The degree distribution obeyed the **power–law** distribution, corresponding with what we could see in the user graph—-many isolate nodes lie on the edge of the figure, while a small group of nodes with dense edges locate in the centre. The Douban social network also had small clustering coefficient (0.117) and relatively short average path length (3.994). The degree distribution, clustering coefficient and path length indicated a network generated by preferential attachment model, which might account for the formation of Douban social network, and also gave a sketch for the behavior pattern of Douban users.

We then implemented several models and algorithms. SkipGram-based methods DeepWalk and Item2Vec were used for pre–training purpose, where DeepWalk made use of the social relations between users and Item2Vec utilized the information of movies. Both generated embeddings for further training. We first implemented Probabilistic Matrix Factorization, a classic approach of matrix factorization in collaborative filtering. PMF performed well in terms of RMSE, while failed to achieve good NDCG. Given that PMF only took into account the ratings, we turned to Factorization Machines, which was capable of dealing with all kinds of features. Pre–trained embeddings were incorporated into FM to speed up its training. Also, Neural FM, as one of the extensions of FM, was implemented. For FM, when some of the features were eliminated, we observed significant improvement in performance on the BIG set, comparing with that on the SMALL set. Pre–trained embeddings were proved to be effective in enhancing the performance of FM, while Neural FM outperformed FM **only** on the training set, and failed on the testing set. Finally, to address the sparsity and cold start problem of collaborative filtering, we tried Ripple Network, a deep learning framework that incorporated the knowledge graph into recommender systems. The Ripple Network shew an overall higher performance on the BIG set than on the SMALL set. In addition, it made the best of all the features.

References

[1] Ruslan Salakhutdinov, Andriy Mnih, *Probabilistic Matrix Factorization*, Toronto, 2007.

[2] S. Rendle, *Factorization machines*, In ICDM, 2010.

[3] Xiangnan He, Tat–Seng Chua, *Neural Factorization Machines for Sparse Predictive Analytics*, Tokyo, 2017.

[4] Wang H , Zhang F , Wang J , et al. *RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems*[J]. 2018.

[5] Bryan Perozzi, Rami Al-Rfou, Steven Skiena, *DeepWalk: Online Learning of Social Representations*, 2017.

[6] Oren Barkan and Noam Koenigstein, *Item2vec: neural item embedding for collaborative ltering*, In MLSP Workshop, 2016.

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, *Fast unfolding of communities in large networks*, 2008.

[8] Noack, Andreas, *Modularity clustering is force-directed layout*, 2009.

[9] Gephi–The Open Graph Viz Platform, https://gephi.org

[10] Tesseract-OCR, https://github.com/tesseract-ocr/tesseract/wiki/Documentation