Report of Douban Movie Recommendation

Jiancong Gao
School of Data Science
Fudan University
Shanghai, China
15300180050@fudan.edu.cn

Linyang He
School of Data Sciences
Fudan University
Shanghai, China
lyhe15@fudan.edu.cn

1 Data Visualization

In this project, we use the *Gephi* application to implement the Douban user data visualization. Gephi is an open-source network analysis and visualization software package written in Java on the NetBeans platform. It was initially developed by students of the University of Technology of Compigne in France.

Considering that the whole dataset is quite slow to run the visualization program, and the whole dataset will not promise a good visualization result, we randomly extract a subset of the user dataset with 246 users, and we will count the following users and followers of this user, which makes the total number of users is 12678. Besides, we can build an *adjacent matrix* to represent the directed graph. The following and followed information can be denoted in a directed graph as the figure showing. In summary, there're 12678 nodes representing the users and 16144 edges representing the following information. We will illustrate some important information the graph can tell.

1.1 Pointed Edge

The single pointed arrow between U_2 and U_1 suggests that U_2 follows U_1 but U_1 does not follow U_1. Also, the double pointed arrow between U_1 and U_3 means that these two users follow each other.

1.2 Node Size

As you can see from the figure, the sizes of these three nodes are different. We use the size information to denote the *In-Degree* rather than just degree. In this situation, users with more fans or followers can have a big size.

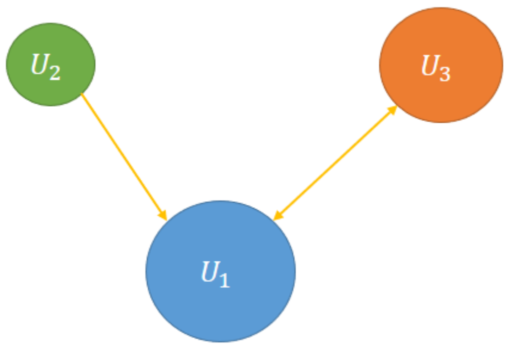


Figure 1

1.3 Node Color

In this project, a quite meaningful feature of the visualization is the color. We are not coloring the nodes randomly, in contrast, we use different colors to represent different communities. That is, we implement a community division algorithm through the data visualization. We introduce the Fast Unfolding algorithm to make it. Before we demonstrate the algorithm, we will first introduce the Community Division task.

Community Division The main goal of the community division is to make the connection in the same community more dense while make the connection among different communities more sparse. This can "split" the graph into several parts in general based on their following and followed information.

Modularity Broadly speaking, modularity is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use. Modularity is quite an important concept in social network mining. It describes how meaningful the community

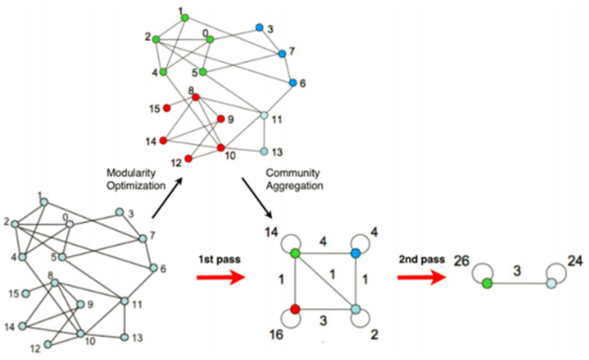


Figure 2

division algorithm is. We can get the modularity as

$$Q = \sum_{c}[\frac{\sum_{in}}{2m}(\frac{\sum_{tot}}{2m})^2]$$

If a community division is reasonable, it should have a relatively higher modularity.

Fast Unfolding Algorithm We will illustrate the algorithm as following steps.

1. Initialization

   Just make all the nodes belonging to different communities. That is, if we have N nodes, we have N communities initially.

2. Modularity Optimization

   For each node, try to divide each point into the community where its neighboring point is located, and calculate the degree of module at this point. Judging whether or not the modularity before and after the division is increased, if it is yes, then accept this division.

3. Community Aggregation

   This is the core of the algorithm. In this step, we will turn all the nodes in a same community into one new big node.

4. Iteration

   Iterate the algorithm until the directed graph does not change.

Using such algorithms and all the methods above, we can get the Data Visualization result in Figure 3 and Figure 4

Different colors denote different community. And we can know that the biggest pink node in the middle has the most followers, (shown in Figure 5)

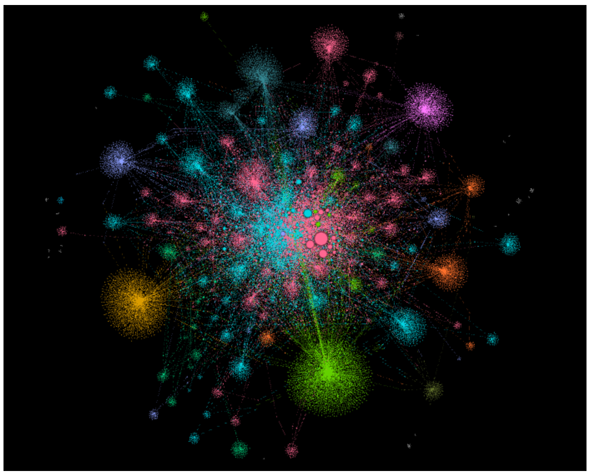


Figure 3

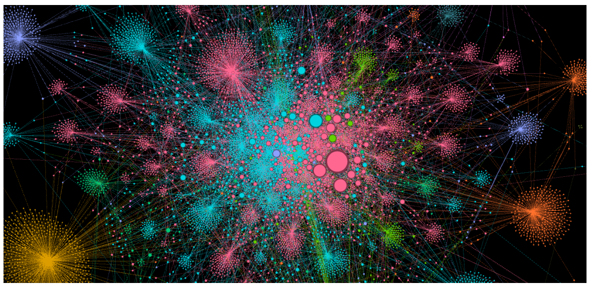


Figure 4

Zoom in, we can find the user's information: user's ID, number of followers and number of following. Besides, we highlight all the related nodes for ease of observation. (shown in Figure 6)

2 Collaborative Topic Regression

In this section, we implement the collaborative topic regression (CTR) model(**?**). CTR combines traditional traditional collaborative filtering with topic modeling. More specifically, it is a combination of Probabilistic Matrix Factorization (PMF) and Latent Dirichlet allocation (LDA). First, we get to know a bit about backgrounds of this model.

2.1 Bachgrounds

Recommendation by PMF Matrix Factorization is a simple but effective latent factor methods. We represent users and items in a shared latent low-dimensional space of dimension K. User i is represented by a latent factor u_i while itme j is represented by a latent factor v_j. The rating is then computed as

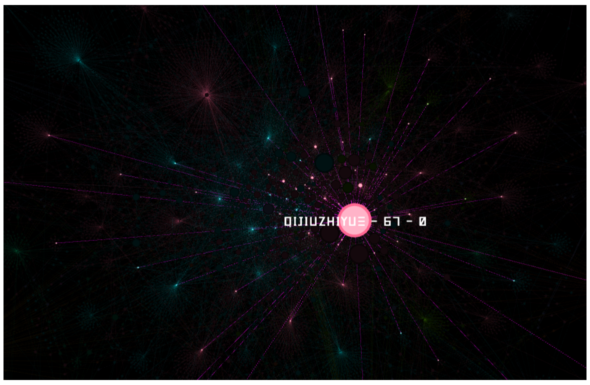


Figure 5

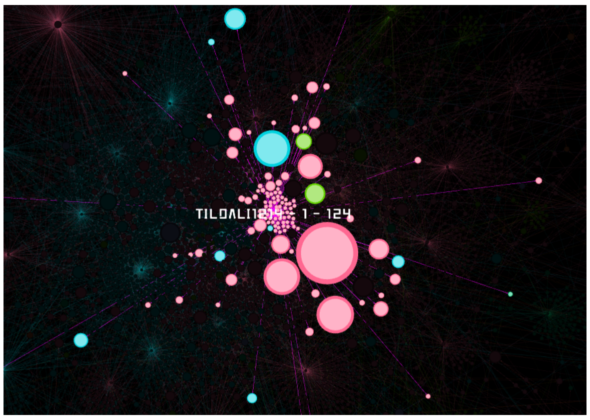


Figure 6

$$\hat{r}_{ij} = u_i^T v_j$$

The method is named matrix factorization, because Rating matrix R is estimated by product of user matrix $U = u_i$ and item matrix $V = v_j$.

$$R = U^T V$$

PMF, as its name infered, brings in the idea of probability. The model assume the following generative process,

1. For each user i, draw user latent vector $u_i \sim N(0, \lambda_u^{-1} I_K)$.

2. For each item j, draw user latent vector $u_i \sim N(0, \lambda_v^{-1} I_K)$.

3. For each user-item pair (i, j), draw the response
$$r_{ij} \sim N(u_i^T v_j, c_{ij}^{-1})$$
where c_{ij} is the precision parameter for r_{ij}.

Here c_{ij} is a confidence parameter. If c_{ij} is large, we trust r_{ij} more, so we give real ratings higher value of c_{ij}.

Topic Models: LDA Topic models are a useful model in NLP. It is used to discover a set of ”topics” from a large collection of documents. From one aspect, topic models provide an interpretable low-dimensional representation of the documents.

LDA is the simplest and most-frequently-used topic model. LDA assumes there are K topic $\beta_{1:k}$ is a distribution over a fixed vocabulary. The model process is as follows:

1. Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$

2. For each word ,

    - Draw topic assignment $z_{jn} \sim Mult(\theta_j)$
    - Draw word $w_{jn} \sim Mult(\beta_{z_{jn}})$

LDA models on document-theme distribution over word-theme distribution and learn the parameters α, β with EM algorithm.

2.2 Model Introduction

From the previous bachgrounds, we found that MF or PMF simply ignore the item information and thus, the learnt latent space cannot be interpreted. Also, it cannot genneralize to completely unrated items. A good way to solve these two drawback is to assign items with latent factors with meaning. A simple way is start with item content and extract valuable features. The first approach is replacing the latent item factor v_j with the its theme factor θ_j and generate ratings as,

$$r_{ij} \sim N(u_i^T \theta_j, c_{ij}^{-1})$$

but it cannot distinguish topics for explaining recommendations from topics important for explaining content.

Thus, we introduce collaborative topic regression (CTR). It represents users with topic interests and assumes that documents are generated by a topic model. CTR additionally includes a latent variable θ_j that offsets the topic proportions ϵ_j when modeling the user ratings. Details will be illustrated in next section.

2.3 Model Details

The graphic model of CTR is shown in Figure 7. The generative process is as follows,

1. For each user i, draw user latent vector $u_i \sim N(0, \lambda_u^{-1} I_K)$.

2. For each item j,

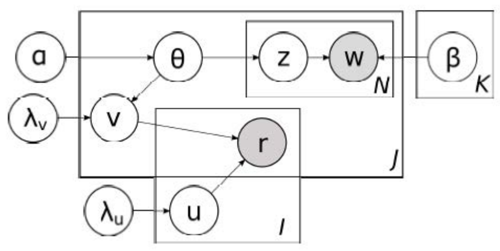


Figure 7

- Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$
- Draw item latent offset $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$ and set the item latent vector as $v_j = \epsilon_j + \theta_j$
- For each word w_{jn},
  1. Draw topic assignment $z_{jn} \sim Mult(\theta)$
  2. Draw word $w_{jn} \sim Mult(\beta_{z_{jn}})$

3. For each user-item pair (i, j), draw the rating

$$r_{ij} \sim N(u_i^T v_j, c_{ij}^{-1})$$

The key is how v_j is generated. $v_j \sim N(\theta, \lambda_v^{-1} I_K)$.

Learning parameters The parameters are learned by EM algorithm to learn the maximum a posteriori (MAP) estimates.

$$u_i \Leftarrow (VC_iV^T + \lambda_u I_K)^{-1} VC_iR_i$$

$$v_i \Leftarrow (VC_jV^T + \lambda_v I_K)^{-1} (UC_jR_j + \lambda_v \theta_j)$$

and a good estimate of θ is the estimate from vanilla LDA.

After we estimate U, V, θ, we can optimize β,

$$\beta_{kw} \propto \sum_j \sum_n \phi_{jnk} 1[w_{jn} = w]$$

Prediction After we have estimate of $U*, \theta*_{1:J}$and β. We will provide two ways to predict rating: in-matrix and out-matrix.

For in-matrix prediction which both user and item has record, we predict as

$$r_{ij}^* = (u_i^*)^T (\theta_j^* + \epsilon_j^*) = (u_i^*)^T v_j^*$$

For out-matrix prediction which item is new, we predict as

$$r_{ij}^* = (u_i^*)^T \theta_j^*$$

3 Experiments

In experiment, we use the average rating $\bar{r}_i j$ of a movie and set $r_i j$ in the model to be the difference of user's rating and average rating $r - \bar{r}_i j$. This shows the user's own perferance and can give a more accurate estimate.

Because CTR takes huge computational cost, we only perform it on small dataset provided in the first time. We conduct 5-Fold Cross Validation.

Here we adopt same parameter setting in (**?**), but the question is how to define a document for a item (movie).

In this Douban Movie Recommendation task, items are movies. Clearly, we cannot extract features through video. But, we can use item information provided by Douban and other DBPedia, like directors, actors, types, countries and so on. Moreover, we can go onestep further besides using tag-information listed above. Movie summaries mainly summarize the plot of the movie and contain lots of information about the movie. We produce several setting for three different document selection

- all: 'title', 'directors', 'year', 'actors', 'type', 'countries', 'summary'
- summary: 'summary'
- other: title', 'directors', 'year', 'actors', 'type', 'countries'

The results is shown in Table 1

Table 1: Recommendation Results

	all	summary	other
RMSE	0.9217	0.8790	0.8867
MAE	0.7458	0.7279	0.7303

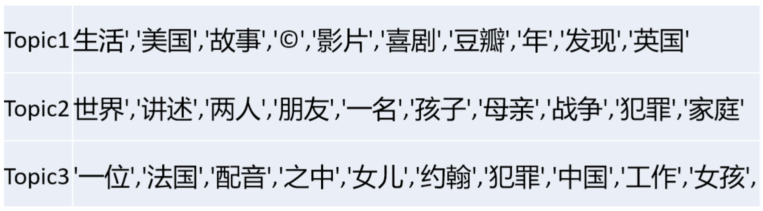
Surprisingly, CTR with all information performs worst among three.

Here we extract top-10 words from certain generated topics and it shows that our topic model give explicit split to different kinds of movies.

We provide an example for top-10 words in certain topic, and we can see it is effective.

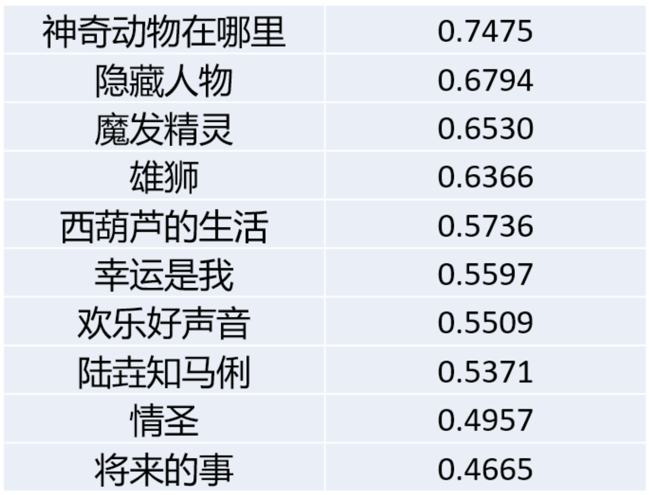
4 Another Idea of Feature Extraction

The basis of the content-based recommendation system is that the user's interests should match the description of the system's recommended items.



Topic1	生活','美国','故事','©','影片','喜剧','豆瓣','年','发现','英国'
Topic2	世界','讲述','两人','朋友','一名','孩子','母亲','战争','犯罪','家庭'
Topic3	'一位','法国','配音','之中','女儿','约翰','犯罪','中国','工作','女孩',

Figure 8



神奇动物在哪里	0.7475
隐藏人物	0.6794
魔发精灵	0.6530
雄狮	0.6366
西葫芦的生活	0.5736
幸运是我	0.5597
欢乐好声音	0.5509
陆垚知马俐	0.5371
情圣	0.4957
将来的事	0.4665

Figure 9: Example of recommendation via preference

In other words, the more similar the user's interest is to the description of the item, the more interested the user may be in the recommended item. The content-based recommendation system achieves this by measuring the similarity between the description of a project and the user's personal information. The higher the similarity, the greater the chance of the project being recommended.

We use some algorithms to embed the information of the movie and the user's information in the same semantic space.

4.1 Movie Information Embedding

Specifically, the embedded movie vector consists of provided movie information. We have adopted information including the movie's publication time, ratings, screenwriters, directors, and actors.

In particular, in this project, **we proposed an original method of embedding actors and directors in Low-dimensional vector space.** We label both the director and the actor as artists, build an adjacency matrix from the number of movies they have worked with, and then use the SVD matrix decomposition to get a low-end matrix. This method not only greatly reduces the dimension, facilitates subsequent calculations, but also has certain semantic information embedded, which is a more reliable method.

As the saying goes, things are gathered together and people are divided into groups. We know that often-co-directed directors and actors may have similar film styles. For example, the actors often appearing in Qiong Yao's TV series may have similar artistic styles and they all like to play soap operas. And the audience may be similar to their preferences.

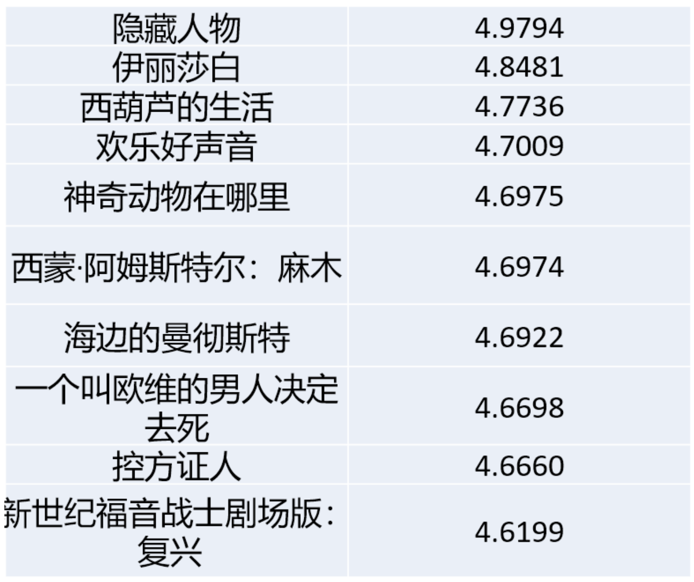
This algorithm's inspiration comes from the word2vec algorithm. In the word2vec algorithm, semantically similar words are closer in semantic space. Considering the following sentences:

- The tiny is so cute!

We will easily think of cat, puppy, dog, etc according to semantic environment. And it can be proved as the Bing search result as below.

So in the movie situation, considering the following artists:

- Tim Burton, Johnny Deep,



隐藏人物	4.9794
伊丽莎白	4.8481
西葫芦的生活	4.7736
欢乐好声音	4.7009
神奇动物在哪里	4.6975
西蒙·阿姆斯特尔：麻木	4.6974
海边的曼彻斯特	4.6922
一个叫欧维的男人决定去死	4.6698
控方证人	4.6660
新世纪福音战士剧场版：复兴	4.6199

Figure 10: Example of recommendation via overall rating

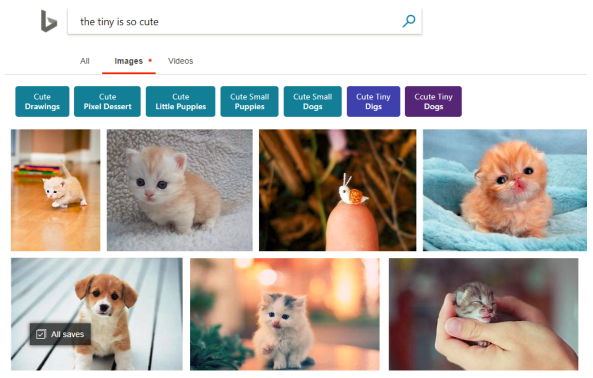


Figure 11

- Daniel Radcliffe, Alan Rickman, JK Rowling.
- Fan BingbingZhao WeiSu Youpeng

You might think of Helena Bonham Carter, Emma Watson, Lin Xinru respectively. And you will never think of a Chinese actor in the first movie environment just like you will never think of a word like "bottle" in the "The tiny XXX is so cute!" semantic environment. Then environment entails the information! Therefore, in word2vec algorithm, we will use the words around a particular word as the "environment" to train the word embedding model. And in the "movie environment", we believe the co-operation entails the environment. We will build a "Co-operation matrix" M like below.

[]@llll@

Co-operation Matrix	Tim Burton	Johnny Deep	Helena Carter
Tim Burton	16	9	7
Johnny Deep	9	17	8
Helena Carter	7	8	15

If we have 10000 artists in total, we can have a such 10000 *10000 dimension "Co-operation matrix". And then we can SVD the matrix:

$$M = U \times S \times V$$

We will choose the top n (here 100) important dimension of all U,S,V matrix to get off the noise as the figure shows. We denote them as U', S', V' respectively. We will rebuild the M matrix.

$$U' \times S' \times V' = M'$$

In this situation, each **artist can be represented by a 100-dimension vector.**

$$v_{artist_i} = M'_{row_i}$$

And finally, we have

$$v_{movie_i} = cat(v_{artist_i}, v_{rating_i}, v_{date_i}, \ldots)$$

In the experiment, we selected all actors (about 2,000) who performed more than 8 times in 10,000 movies and established such a "cooperation matrix" and then reduced it to 100 dimensions through PCA. This gives each artist's expression. Finally, we use the TSNE algorithm to reduce the

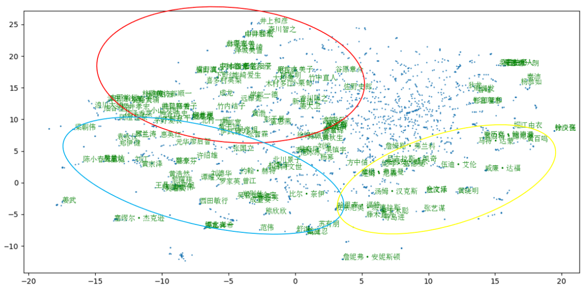


Figure 12

matrix to 2 dimensions and visualize it. As can be seen in the figure below, like the semantic space of word2vec, the low-dimensional representation of our cooperation matrix dimension reduction can indeed express a good cooperation relationship.

As shown in the figure, the red region is basically a Japanese artists community, the blue region is for Chinese artists, and the yellow one is mainly a European and American region.

4.2 User Information Embedding

For every user, we have their rating information. We can suppose that they love those movies which they rate them as 5 score. So we can use all the movies they love to represent them. In this situation, we will denote the user embedding vector as the average of their 5-score movies' embedding vector.

$$v_{user} = \frac{1}{n} \sum v_{movie_{5-score}}$$

We can use this method to build the users' denoting.