Movie Recommendation Using Graph Neural Networks

NTDS Final Project Team 7

Kuan Tung*, Chun-Hung Yeh†, Hiroki Hayakawa‡ and Jinhui Guo§
Institute of Electrical Engineering*†‡, Institute of Material Science & Engineering§
École Polytechnique Fédérale de Lausanne
Email: *kuan.tung@epfl.ch, †chun-hung.yeh@epfl.ch, ‡hiroki.hayakawa@epfl.ch, §jinhui.guo@epfl.ch

***Abstract*—A movie recommendation is important to our life not only due to its strength in providing enhanced entertainment, but also its economic profits. Such a system can suggest a set of movies to users based on their interest, or the popularity of the movies. In this project, we first analyze data from Movielens 100k [1] to find out the hidden network structures of movies and users, and then emphasize on building a recommendation system using graph based machine learning to predict unknown ratings. The best result is from the graph convolutional matrix completion (GC-MC) with features model, which possesses the lowest root-mean-square error (RMSE) of 0.901.**

I. Introduction

It is always great for people to spend a couple of hours watching a good entertaining movie. However, looking for a suitable movie is not an easy task. There are many other users' ratings on the internet, but they are not always reliable. On the other hand, movie companies are also interested in predict customers' preference. Thus, it is of great necessity to build such a recommendation system that can achieve this win-win situation.

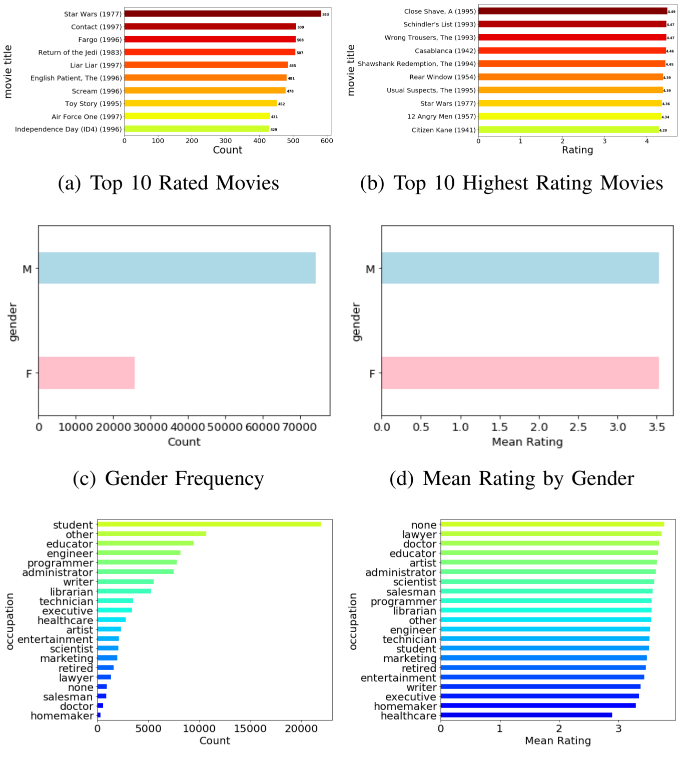
Collaborative and content filtering techniques are two major methods to the recommendation system. One of the main problems of such a system is matrix completion. The purpose of our project is to design a personalized movie recommendation system and evaluate its performance. In this work, we mainly apply the graph machine learning called Graph Convolutional Matrix Completion [2] to build the system, which will predict users' rating to a unseen movie based on their ratings to watched movies and pick up new movies with high ratings as recommendation to the user. Moreover, we will compare our system's performance with several existing methods to see if ours outperforms. Aside from the recommendation system, we will investigate the dataset to identify the network structures of movies and users as well.

II. Data Exploration

The dataset, MovieLens 100k, consists of a list of movies with their meta-data (e.g. rating, genres, and release date) and a list of users with their relevant information (e.g. age, gender, job, and occupation). More precisely, there are 100,000 ratings (1-5) from 943 users on 1682 movies. Luckily, the original data has already been cleaned up, removing users who had less than 20 ratings or incomplete demographic information. Hence, we could work on the initial data exploration without any worry.

A. Exploratory Data Analysis

In order to better understand our data, we must look into some properties of our features in detail. Just as Fig. 1, we found out that most rated movies are those in recent times when the project was ongoing. That is, users mostly rated the movies that were released after 1995.



(a) Top 10 Rated Movies

(b) Top 10 Highest Rating Movies

(c) Gender Frequency

(d) Mean Rating by Gender

(e) Occupation Frequency

(f) Mean Rating on Occupation

Fig. 1. Data Exploration

In addition, we noticed that among all users there is no difference in rating between males and females though more men rated than women did. In this way, the influence by the uneven number of audience should not be a big problem.

Another interesting finding is that exceedingly more students gave ratings to movies than all the other job roles. Nevertheless, we observed that students are more captious to give high ratings compared with other occupations corresponding to high social status.

Lastly, we did some exploration in the genres (Fig.2). The most appearing word is drama, followed by comedy, which are three and two times higher than the third most appearing word. From our notebook `data-exploration.ipynb`, we could see that Drama and Comedy are always people's favorite regardless of gender and occupation. This probably indicates

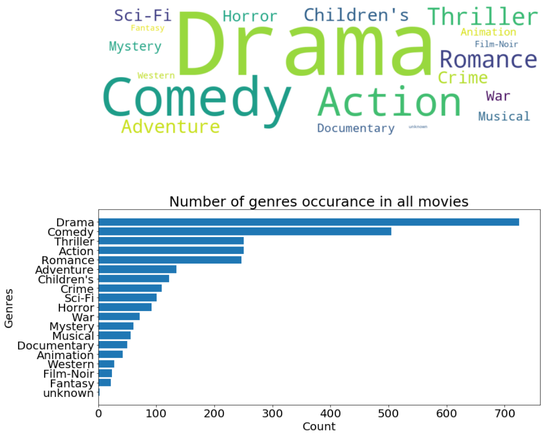


Fig. 2. Number of Genres Occurrence in All Movies

that the movies in these genres can be safely recommended to audience.

B. Graph Creation

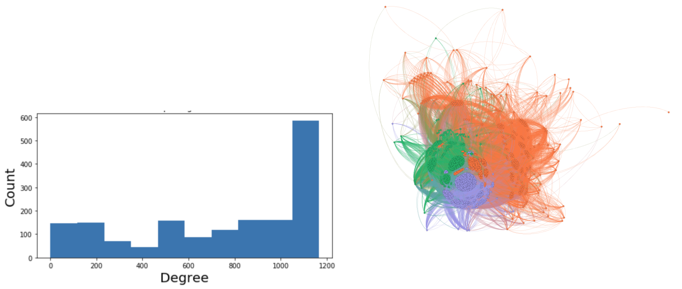
In this part, we would present several graphs for movies and users mainly based on their genres and occupations respectively.

1) Movie Graph: We built a movie graph where each node stands for an individual movie (shown via its ID) and every edge is determined by the distance between two movies as computed based on one-hot vectors of their genres. Hence, according to Fig. 3[1], this graph contains 1682 nodes and 642422 edges. Also, we found out that more nodes have degrees beyond 1000. We speculated that there might be numerous giant components in the movie graph. More detailed graph statistics are shown in Table I.

TABLE I
MOVIE GRAPH STATISTICS

Statistics	Value
Nodes	1682
Edges	642422
Average Degree	763.879
Diameter	6
Density	0.454
Modularity	0.168
Clustering Coefficient	0.797
Connected Component	1

In addition to building the movie graph, we created circular graphs from the co-occurrence matrix and the correlation matrix of movie genres. From Fig. 4, most genres do not appear jointly. The most apparent pair that two genres co-occur is documentary and drama as their edge weight amounts to 0.6. As for the graph formed on the correlation matrix displayed in Fig. 5, numerous genres are strongly positively correlated. For instance, Children's is more correlated with Animation and Adventure. Most genres are slightly negatively correlated to each other. However, Drama is the one that strongly negatively correlated with many genres, such as Comedy, Action and Adventure.



(a) Degree Distribution

(b) Movie Graph Colored via Modularity Shown by Gephi

Fig. 3. Movie Graph

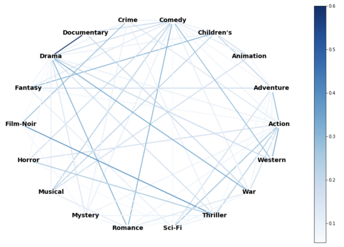
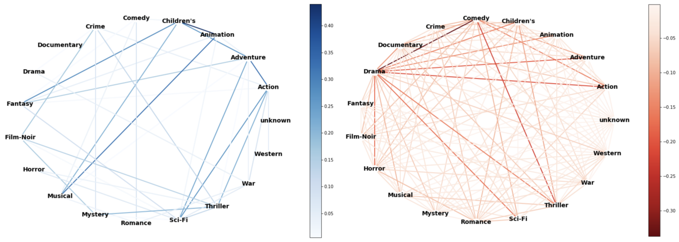


Fig. 4. Genres Graph Using Co-occurrence Matrix



(a) Circular Graph (Positive Weights Only)

(b) Circular Graph (Negative Weights Only)

Fig. 5. Genres Graph Using Correlation Matrix

2) User Graph: The user graph we made incorporates 943 nodes and 41326 edges. Each node represents a user (shown by the user's ID) while every edge is determined by the distance between two users as calculated based on one-hot vectors of their occupations. The degree distribution and its corresponding graph are presented in Fig. 6[2]. The degree distribution mostly ranges from 0 to 125; however, there exist some hubs with roughly 175 degrees. Thus, our user graph accommodates some giant components, formed as numerous isolated clusters. More detailed graph statistics are shown in Table II.

III. DATA EXPLOITATION

A. Similar Movies Related to Genres

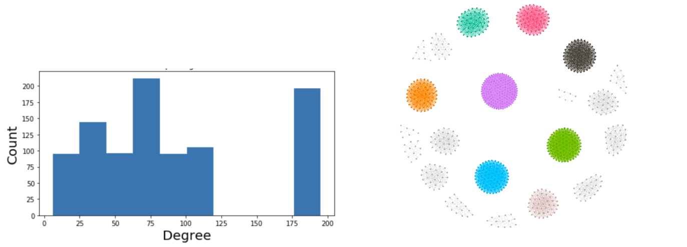
The first attempt in this section is to explore similar movies by using clustering method. It is important and useful to recommend the audience other movies according to relevant

[1] Interactive visualization link of the movie graph

[2] Interactive visualization link of the user graph

TABLE II
USER GRAPH STATISTICS

Statistics	Value
Nodes	943
Edges	41326
Average Degree	87.648
Diameter	1
Density	0.093
Modularity	0.744
Clustering Coefficient	1.0
Connected Component	21



(a) Degree Distribution

(b) Graph Colored via Modularity Shown by Gephi

Fig. 6. User Graph

genres. In our dataset, each movie is attached by at least one genre, up to 6 genres.

1) Dimension Reduction and Clustering: In this part, we construct movie clusters with related genres. Since Euclidean distance tends to become inflated in high dimension, we need to consider dimension reduction before clustering by method such as K-means. We compared the embedding methods of principle component analysis (PCA), Isomap and t-Distributed Stochastic Neighbor Embedding (t-SNE), and found t-SNE works the best due to its properties of non-linearlty and circumvention of crowding problems. The we use K-means method to cluster similar movies. The results after dimension redcution on our dataset in shown in Fig. 7.

The clustering seems to work well, however, whether the clusters make sense requires us to refer to the statistical result of genres numbers in each. From Fig. 8, we see cluster number 3, 4 and 5 have one prominent genre, which are drama, comedy and documentary, respectively. For cluster number 0 and 2, they have two obvious genres in each, which are *Romance and Drama* and *Romance and Comedy* respectively. Others clusters also have one significant genre in each, while they have more association with other genres, which is also reasonable, since one movie can have at most 6 genres in one movie in our dataset.

B. Prediction of Movie Ratings

The task of building a recommendation system can be expressed as designing a method for matrix completion or prediction of unknown entries in a rating matrix M whose entries are ratings from users to movies and shape is $N_u \times N_v$, where N_v is the number of users and N_v is the movies respectively. An undirected bipartite User-movie interaction graph represents the rating matrix with the form $G = (\mathcal{W}, \mathcal{E}, \mathcal{R})$. A collection of nodes $\mathcal{W}$ is union of user nodes $u_i \in \mathcal{U}$ and movie nodes $v_i \in \mathcal{V}$. The edges $(u_i, r, v_j) \in \mathcal{E}$ connect a user and a movie and carry rating information $r \in \{1, \ldots, R\} = \mathcal{R}$. The matrix completion task is equivalent to a link prediction of the bipartite graph. The project adopts an approach with Graph convolutional encoder proposed in [2] for the link prediction. The schematic of the approach is shown in Figure 9 and it consists of a graph convolutional encoder that transfer rating information from movies to users or users to movies and a bilinear decoder that makes rating prediction with a form of a matrix. In this section, we introduce the encoder and the decoder.

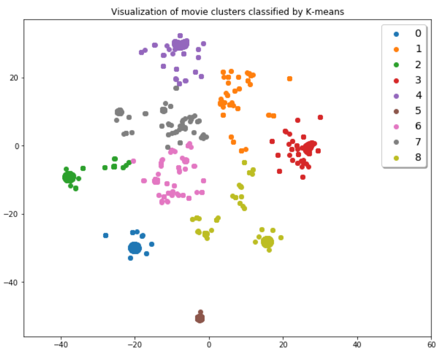


Fig. 7. Visualization of movie clusters classified by K-means (cluster number from 0 to 8)

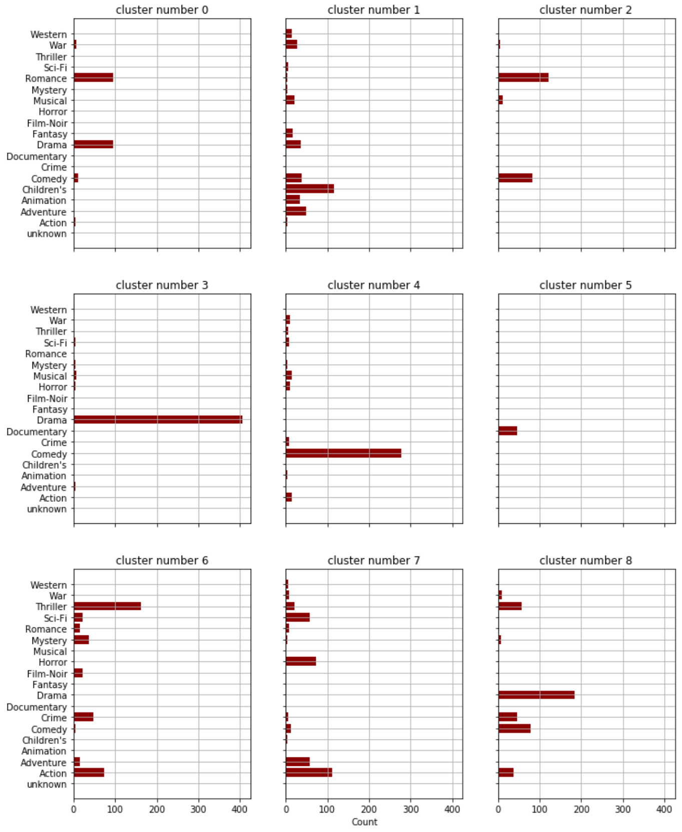


Fig. 8. Statistical Results of Genres in Each Cluster

1) Graph Convolutional Encoder: The graph convolutional encoder is formed as $[U, V] = f(X, M_1, \ldots, M_R)$ taking $(N_u + N_v) \times D$ feature matrix X and rating matrices $M_r \in$

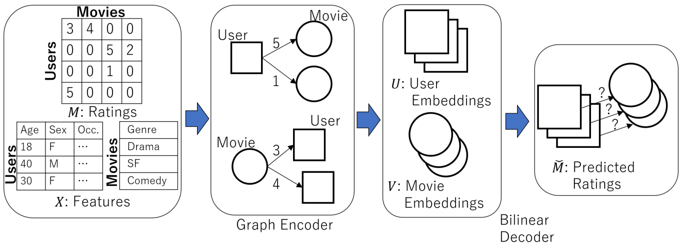


Fig. 9. Schematic of Graph Convolutional Matrix Completion Method Proposed in [2]

$\{0,1\}^{N_u \times N_v}$ whose entries are 1's if the original ratings M has r value at the same location of entries or 0's otherwise. The output U and V are user and movie embedding matrices with shape of $N_u \times E$ and $N_v \times E$. Here, we look at the process of graph convolution. A transferring rating information from a movie i to a user j is formed as follows.

$$\mu_{j \rightarrow i,r} = \frac{1}{c_{ij}} W_r x_j,$$

where c_{ij} is a normalization factor and can be chosen from $|\mathcal{N}_i|$ (left normalization) and $\sqrt{|\mathcal{N}_i||\mathcal{N}_j|}$(symmetric normalization) and $\mathcal{N}_i$ is neighbor set of node i. In addition, W_r is a parameter matrix and x_j is the feature vector of node j. The transferred information is accumulated with a following way.

$$h_i = \sigma \left[\text{accum} \left(\sum_{j \in \mathcal{N}_{i,1}} \mu_{j \rightarrow i,1}, \cdots, \sum_{j \in \mathcal{N}_{i,R}} \mu_{j \rightarrow i,1} \right) \right]$$

Incoming information from neighbors are summed up and then accumulating operation such as stack($\cdot$) is performed to form them into a vector. The activation function σ like ReLU acts elementwisely. Finally, a single user embedding u_i is formed as follows.

$$u_i = \sigma(W h_i)$$

Here, W denotes parameter matrix. The movie embedding v_i can be calculated in analogous way.

2) Feature as Side Information: Feature information can be used as input of graph convolutional encoder. We need to be cautious about the contents of the features. If the features contained limited amount of information and it is hard to distinguish users or movies, which leads to a unnatural information flow. To avoid it, we can use features as side information as follows.

$$u_i = \sigma(W h_i + W_2^f f_i) \text{ with } f_i = \sigma(W_1^f x_i^f + b),$$

where W_1^f and W_2^f are weight parameter matrices, x_i^f is feature vectors for node i, and b is a bias. In this case, an identity matrix is injected into a graph convolution layer instead. In this project, given feature information is not enough; therefore we use them as side information. For the efficient computation with matrix multiplication, the graph encoder is vectorized as follows.

$$[U,V]^T = f(X, M_!, \ldots, M_R) = \sigma([H_u, H_v]^T W^T),$$

$$\text{with } [H_u, H_v]^T = \sigma(\sum_{r=q}^{R} D^{-1} \mathcal{M}_r X W_r^T),$$

$$\text{and } \mathcal{M}_r = \begin{pmatrix} 0 & M_r \\ M_r^T & 0 \end{pmatrix},$$

where D is $\text{diag}(|\mathcal{N}_i|)$ or $\text{diag}(\sqrt{|\mathcal{N}_i||\mathcal{N}_j|})$.

3) Bilinear Decoder: As a bilinear decoder a following softmax function is applied to produce a probability that the estimated rating $\check{M}_{ij}$ between user i to movie j is r.

$$p(\check{M}_{ij} = r) = \frac{e^{u_i^T Q_r v_j}}{\sum_{r \in R} e^{u_i^T Q_r v_j}},$$

where Q_r is parameter matrix with shape of $E \times E$. The rating prediction is generated with estimation operation as follows.

$$\check{M}_{ij} = g(u_i, v_j) = \mathbb{E}_{p(\check{M}_{ij}=r)}[R] = \sum_{r \in R} r p(\check{M}_{ij} = r)$$

Vectorization of bilinear decoder is implemented in an analogous way.

4) Loss Evaluation: The negative log likelihood of the predicted ratings $\check{M}_{ij}$

$$\mathcal{L} = - \sum_{i,j;\boldsymbol{\Omega}_{i,j}=1} \sum_{r=1}^{R} I[r = M_{ij}] \log p(\check{M}_i j = r)$$

is minimized for a model training. We denote $I[k = l] = 1$ if $k = l$ and zero otherwise and the masking matrix $\Omega^i n\{0,1\}^{N_u \times N_v}$ seals unobserved ratings. Thus, the minimization is performed over given ratings.

C. Experiments

We conducted experiments on five recommendation systems. Two of them are using the previously introduced Graph Convolutional Matrix Completion (GC-MC) methods. The other three are based on matrix factorization [4] and are used for comparison. We will first give an introduction to the experiment settings and then discuss the results.

1) Matrix Factorization Settings: Instead of basic matrix factorization (MF), we also implemented MF + DNN and MF + DNN with additional features. For the MF part, we set the latent dimensions of users and movies as 16 since it yielded the best performance. The final prediction is the sum of the dot value of two latent factors, the user bias term, and the movie bias term. The user bias term corresponds to the tendency of a user to give better or worse ratings than the average. Similarly, the movie bias term describes how well a movie is rated compared to the average.

As for the DNN part, the input of the model is the concatenation of user and movie latent factors. If additional features are used, user and movie features will be concatenated with them as well. The model consisted of four fully connected layers, and their hidden sizes were 256, 256, 256, and 1, respectively. Batch normalization and dropout (dropout rate = 0.5) were applied to the first three layers. The activation function was ReLU for all layers. We used mean squared error as the loss function. We trained 100 epochs and set the batch size to 256 for these three models. The train/validation/test split was 72/8/20, and Adam [5] was chosen as the optimizer.

2) Graph Convolutional Matrix Completion Settings: We used two GC-MC based models. One is without features, and the other is with features. The code we used was from the GitHub repository [6] uploaded by the authors of this work [2]. We made some modifications based on it. The hidden sizes of the stack graph convolution layer and dense layer were set as 500 and 75. The hidden size of the dense layer for the with features case was set as 64. Dropout (dropout rate = 0.7) was applied to the stack graph convolution layer and dense layer. Softmax cross-entropy was the loss function since the problem was viewed as a classification problem with five classes (1-5 ratings) in this case. We trained 1000 epochs and did not apply batch training for both models. The train/validation/test split was again 72/8/20. Adam was selected as the optimizer.

D. Results

The results are shown in Table III. We can see that models with additional components like DNN or adding features achieve lower RMSE. These additional components would result in more training time, but the extra computation efforts are not wasted. Graph-based models (GC-MC) outperform MF-based models with less model parameters. It shows that the utilization of graph convolution is useful. GC-MC with features is the best model with the lowest RMSE (0.901).

TABLE III
RECOMMENDATION SYSTEM RESULTS

Model	RMSE
MF	0.954
MF + DNN	0.942
MF + DNN with features	0.923
GC-MC	0.908
GC-MC with features	0.901

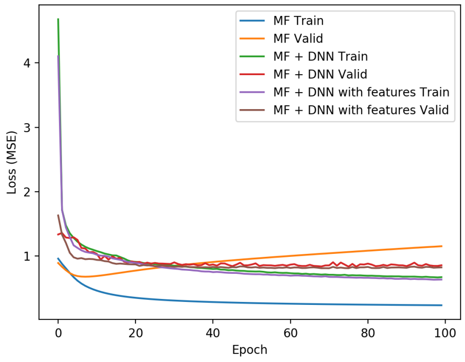


Fig. 10. Loss History of MF Based Models

The training history of loss (MSE) for MF based and RMSE for Graph-based models are shown in Fig. 10 and Fig. 11. For the pure MF model, the validation loss increases very early at around the 8th epoch. It shows that the relatively simple MF model overfits the training data quite easily. The validation losses of the other two MF based models converge at the end. Likely, we would not be able to get better results if we kept training. The validation losses of both graph-based models are surprisingly lower than the training losses, and they keep decreasing. It is possible that we could get better results if we kept training.

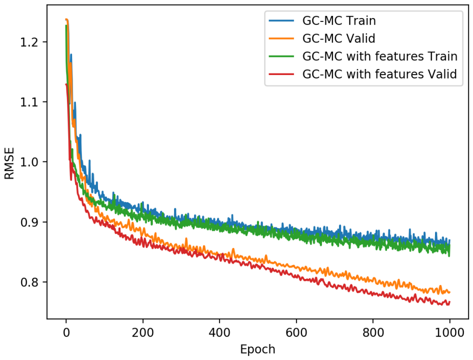


Fig. 11. Loss History of Graph Based Models

IV. CONCLUSIONS

The project conducted graph-based analyses on Movie Lens 100k, and applied both non-graph and graph-based approaches to generate recommendations of movies.

Basic data analyses found out the fact that both men and women have the same mean rating and offered us the user occupation ranking in terms of mean rating value and list of top-rated movies and movies with the highest rating score. Utilizing the features such as movie genres and occupations of uses, distance between movies or uses were calculated and a graph for movie and a graph for users were created. Many of nodes in movie graph, (i.e. each movie) have degree of more than 1000. As for the user graph, some hubs with degrees more than 175 exist so giant components are confirmed. Co-occurrence and correlation between movie genres were also investigated. We found out that co-occurrence is rare for most pair of genres but numerous genres are positively correlated.

K-means accompanied by t-SNE gives us a good result on movie clustering. However, whether this method can give us a better result is still need exploration and probably we need more data. For the movie recommendation system, rating prediction were implemented with five methods, MF, MF-DNN, MF-DNN with features, GC-MC, and GC-MC with features. GC-MC with features scored the lowest RMSE value of 0.901. Overall, graph convolutional methods outperformed matrix factorization methods in terms of RMSE vales while the latter methods are of computational complexity and additional components, DNN and features, improves the results. Graph convolutional methods showed the tendency of decreasing even after 1,000 epochs of iterations whereas matrix factorization methods converge at 100 epochs. Tuning model configurations such as size of hidden layers, accumulation method, the number of training epochs, and optimization methods (e.g. SGD) are considered as future works.

References

[1] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets: History and Context. 2015.

[2] Berg, Rianne van den, Thomas N. Kipf, and Max Welling. Graph Convolutional Matrix Completion. 2017.

[3] Monti, Federico, Michael Bronstein, and Xavier Bresson. Geometric Matrix Completion with Recurrent Multi-graph Neural Networks. Advances in Neural Information Processing Systems. 2017.

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

[5] Kingma, Diederik P., and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[6] Riannevdberg. (2018, November 15). riannevdberg/gc-mc. Retrieved from https://github.com/riannevdberg/gc-mc.