Jurnal Ilmu Komputer dan Informasi (Journal of Computer Science and Information)
15/1 (2022), 55-67. DOI: http://dx.doi.org/10.21609/jiki.v15i1.1066

SGCF: Inductive Movie Recommendation System with Strongly Connected Neighborhood Sampling

Jatmiko Budi Baskoro[1], Evi Yulianti[2]

Faculty of Computer Science, University of Indonesia, Indonesia

Email: jatmiko.budi.personal@gmail.com[1], evi.yulianti@cs.ui.ac.id[2]

Abstract

User and item embeddings are key resources for the development of recommender systems. Recent works has exploited connectivity between users and items in graphs to incorporate the preferences of local neighborhoods into embeddings. Information inferred from graph connections is very useful, especially when interaction between user and item is sparse. In this paper, we propose graphSAGE Collaborative Filtering (SGCF), an inductive graph-based recommendation system with local sampling weight. We conducted an experiment to investigate recommendation performance for SGCF by comparing its performance with baseline and several SGCF variants in Movielens dataset, which are commonly used as recommendation system benchmark data. Our experiment shows that weighted SGCF perform 0.5% higher than benchmark in NDCG@5 and NDCG@10, and 0.8% in NDCG@100. Weighted SGCF perform 0.79% higher than benchmark in recall@5, 0.4% increase for recall@10 and 1.85% increase for recall@100. All the improvements are statistically significant with p-value < 0.05.

Keywords: *recommendation system, collaborative filtering, graph neural network*

1. Introduction

Recommendation system is a mechanism that helps users discover relevant items for them [1]. It has been used in business as a part of marketing strategy. One key strategy to increase online sales is to recommend customers with a short list of items that is relevant to their preferences [2]. This recommendation process can be automated using several approaches, such as content-based methods and collaborative filtering [3], and usually involving computation of user and item vectors that serve as a basis for predicting the extent to which a user prefers a particular item [4]. Recent development in recommender system suggests that relationships between users are useful for determining user and item vectors [5].

Graph-based recommendation system has emerged to utilize information connection represented in the graph [6] [7].

It utilizes connectivity between nodes to infer similarities between nodes and utilize it to generate vector representation for each node. There are differences in generating vector representations when graph data is used. For example, in collaborative filtering approach without graph data, similar user preferences are represented as latent vectors that close to each other. Compared to graph-based recommendation system, user that have similar preferences are directly represented as 2 connected nodes. The advantage of this approach is that the representation has less restriction defining similarities.

Graph embedding is an approach to encode interaction between nodes into vector space. [8] develop graph neural networks with nodes being represented as vectors computed from convolution operations. [8] further argue that the proposed convolution mechanism is able to capture the underlying graph structure and the connectivity between nodes

Graph embedding is one of many approach to encode interaction between data point into vector space in graph data. In graph neural network approach for example, nodes are represented as vector derived from convolution operation. The convolution operation are able to capture the graph structures and connectivity information between the nodes. This convolution operation is used in [8] work. However, convolution operation in [8] is transductive, meaning that the learned latent vector may not able to generalize unseen data. This issue may cause the recommendation system need to be retrained whenever batch of new data come. To solve this problem, some research has proposed inductive graph embedding such as GraphSAGE [9] and PEAGNN [10]. However, neighborhood sampling is conducted uniformly, which may introduce noise or irrelevant neighbors into the aggregation process. In this paper, we aim to solve the problem by adding local neighborhood weights. These weights will alter the sampling distribution so that relevant neighbors has higher probability to get sampled.

In this paper, we build the model specifically for movie recommendation. The task in this work is defined as item ranking task where historical user interaction with items represented as pair (User ID, Item ID) with $ratings$ ranged from 1-5 as label that represent user preference towards items used as input. The output will be list of ranked items that fit user taste for all user in the input. This work addresses the problem of using inductive-based embedding generation method for computing user and item vectors. Graph based embedding generation model is used to generate embedding for each entity in inductive manner. Our contribution in this work is proposing inductive graph embedding and adding connection weight to alter neighborhood sampling distribution.. In this paper, 1 research question is proposed:

1) How does proposed user and items graph embedding with weights affect recommendation system performance?

2. Related Works

This work is related to the graph embedding approach and previous neural network-based recommendation systems. Previous work that utilizes graph structures for recommendation systems is discussed in this section.

2.1. Matrix Factorization Approach

Matrix factorization technique proposed by [4] predicts user and item into latent vector space U and V. Suppose we have rating matrix R where each cell $r_{ui} \in R$ represent user $u_i \in U$ rating towards item $v_j \in V$. Matrix factorization technique predicts user rating towards item by applying dot product of user latent vector U and item latent vector V. Other work such as [11] use matrix factorization technique to capture social network context for recommendation system.

With the emergence of deep learning techniques in recent years, deep learning-based recommendation systems have emerged to improve latent vector generation.

2.2. Neural Embedding approach

Collaborative filtering approach represent users and items as latent vector U and V, with similar objective function as matrix factorization-based recommendation system [12]. Neural collaborative filtering [13] predicts rating by concatenating user latent vector U and user latent vector V and passed into fully connected feed forward neural networks to generate recommendation. Similar method for item-based collaborative filtering proposed by [14] use deep learning to generate top-n recommendations. Other methods such as [15] [16] utilize deep learning approach to generate user and item latent vector.

Deep learning models proposed by [15] are able to attain maximum a posteriori estimates (MAP), which the authors claims as an advantages of this model. Other deep learning models in [16] utilize convolutional neural network (CNN) to generate latent vector representations and implement attention mechanism to predict user rating towards an item. Recent works [5] in deep learning-based recommendation system has utilized graph to generate latent vectors and shows promising results.

2.3. Graph Embedding Approach

A graph is a data structure that contains relational information between data points (nodes). However, utilizing information within large graph in adjacency matrix form is challenging, so graph embedding is developed to represent relational data within the graph as vector representations [8]. In recent years, other work has explored graph-based deep learning models to generate latent vector representations. For example, graph convolution network [8] and [17] has been proposed to generate embedding matrix of graph data by applying convolution definition. Other works such as [9] stated that convolution operation in graph proposed by [8] is transductive process that may not be able to generalize towards unseen data.

Before we discuss graph convolution networks, first we discuss convolution operations in graphs. Consider a graph $G = (V, E)$ where V is a set of nodes and E is a set of edges. Also consider attribute matrix of $X^{|V| \times F}$ where X is matrix contains user with F attributes. A graph convolution will summarize value of $x \in X$ based on x neighbors.

Graph convolution neural network (GCN) is proposed in [8] for node classification purpose. The authors stated that GCN can be used for other tasks besides classification. Graph convolution operation is further improved by [17] [18]. The authors implemented low-pass collaborative filtering with improved GCN. The success of GCN to generate latent vector still need to be improved, as stated by [9] . The authors stated that graph convolution operations are transductive processes which may not be able to generalize unseen data.

In [9], node embedding is generated by sampling and aggregating node attributes. The aggregation operation can be represented by a differentiable model or aggregation operation such as maximum and pooling operation. In graphSAGE, the node's neighbors are uniformly sampled; the node's vector representation is predicted by the sampled neighbors. The mechanism of generating inductive latent vector using graph data has also been proposed by [5].

Other graph-based approach using neural collaborative filtering framework has also been proposed by [19]. This work used a transductive Graph Convolutional Network (GCN) to create user-item embeddings in one graph and then passed the embedding to a multi-layer perceptron (MLP) and predicted the relevance scores. This work approach is nearly identical to our work; the difference is we use inductive graph embedding and we alter the sampling distribution by adding connection weights both in the user graph and item graph.

Meta-path graph such as PEAGAT [10] has also been proposed to fuse information on each user and item nodes. PEAGAT uses a single graph that contains user-item relationship where user node and item node have linked information node which will be aggregated as meta-path. PEAGAT uses inductive graph embedding. However, the graph used in PEAGAT differs from this work where we separate user and item embeddings.

In this work, latent vectors generated by neural diffusion networks are used to generate recommendations. Compared to this work, our work utilize both user and item relation information, whereas in [5] only user relation information is utilized. We also modified the neighborhood sampling method in the graph embedding.

3. Proposed Networks

In this section, we discussed problem formulations and built recommendation systems using graphsage on both user network and item network. We start by defining the problems and constructing graphs for both users and items and our reasoning behind construction rules. The graph embedding for both user and item is discussed next. Finally, the overall architecture with neural network classifiers will be discussed to generate the recommendation system.

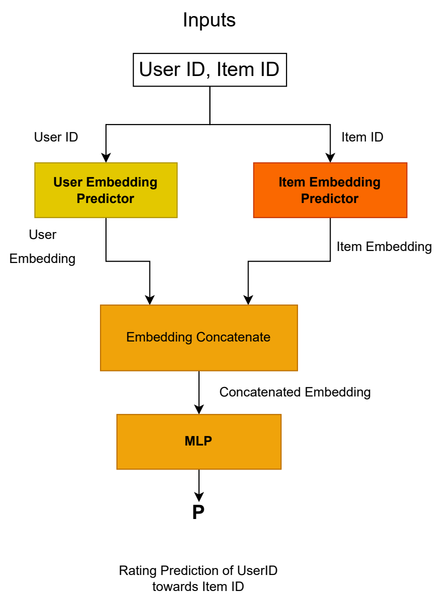


Figure 1. SGCF Computation flow. The input is a pair of user id and item id. Each id will be fed to corresponding embedding and then used to predict user's rating towards the item

3.1. Problem Formulations

The task is defined as ranking task where historical user interaction with items represented as pair (User id, Item id) with $ratings$ ranged from 1-5 as label that represent user preference towards items used as input. The output will be list of ranked items that fit user taste for all user in the input.

To formalize the task, let recommendation input as set of pair of user U and item V pair $S(U, V)$ with user feedback toward item in the pair R. Let recommendation system be $F(U)$. Given user id, the recommendation system $F(U)$ will give a set of k ranked item $[v1, v2, ..., v_k]$ where the top n item, $n \leq k$ is the most relevant item to user u. This problem is formulated as ranked problem. The task is illustrated in figure 1

Figure 1 illustrate the model flow from input to its final output. The final output is a rating prediction which will be used to construct the recommendation list.

In the implementation, we didn't directly optimize to rank list of item for every user in dataset. Instead, we use regression approach to let the model predict relevancy score for each user item pairs. This information produced by the model will be used to sort the item-based on specific user's preferences.

3.2. Graph Construction

Our works utilize graph structured data to generate recommendation. Previous works models user and item similarity using matrix factorization technique and embedding technique where bot approach let the machine determine the similarity of users-based on user-item interaction history. In our case, we directly provide similarity information between users and items by connecting them. For example, consider a pair of item i_1 and i_2 with n attributes $\{x^1_1....x^1_n\}$, $\{x^2_1....x^2_n\}$.

To build user graphs, historical user-item interaction similarity is used to determine whether or not two users are connected. To illustrate this, consider user $u1$, $u2$ with historical interacted items $\{i_1...i_k\}$, $\{i_1...i_l\}$ where $k \leq m$ & $l \leq m$, m is the number of items. User $u1$ and $u2$ are connected when both users has interacted with at least 1 similar item. The similarity of an item are able to be determined by a function that yield similarity metrics which able to freely defined to suit certain use case.

Similar mechanism also applied in building item network. A pair of items i_1 and i_2 are connected if item i_1 is similar to item i_2. This item similarity is measured the same way with function that we mentioned. This approach allows the model to generate representation where similar user and items are close to each other.

3.3. Feature Extraction

In this section, we discuss user and item features used to generate latent vector for both item and user and how the we calculate the edge weights. User features are calculated by averaging user rating over all possible genre. Min max normalization is then applied so that the values range between 0 and 1. Movies features binary vector of it's genre. The feature's value is 1 if the movie belong in a genre. If a movie has more than one genre, then the feature value that represent the genres are 1 and the others is 0. This feature will be used to predict the user and item vector representation from graph.

There are 19 different movie genre available in MovieLens data. Each category is represented as one hot encoding for item graph. In user graph, each movie that user interacted with will be used as user's movie preference attributes. Let X_m be one hot encoded movie m genres and R_{mu} be rating that user u give to movie m. The user attribute Z_u is calculated in equation 1

$$Z_u = \frac{\sum(x_m)(R_{mu})}{|R_u|}, \tag{1}$$

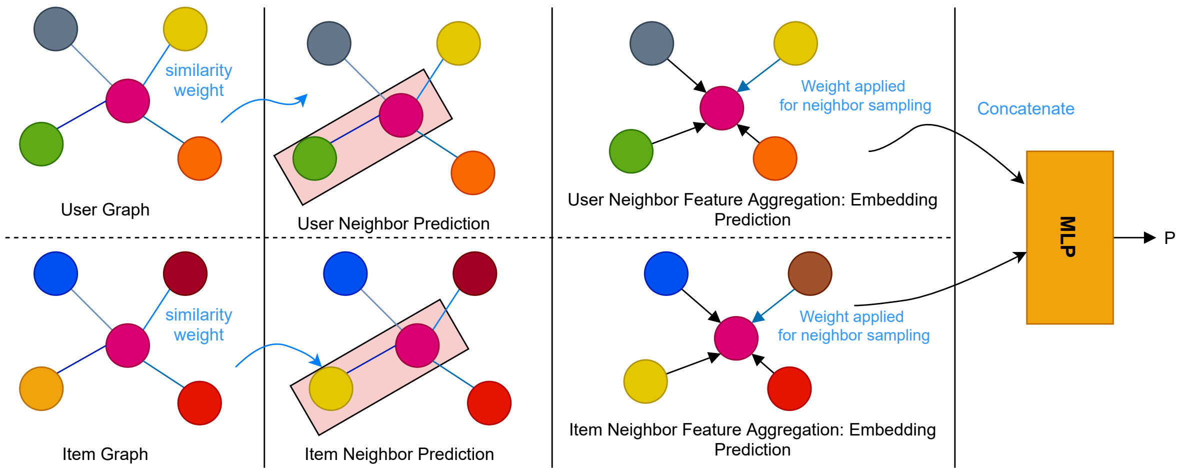
for each user u, we calculate the user attributes in training data to prevent data leakage. In graph construction, we add weight to represent connection strength between entities.The weight between users are calculated by measuring the cos distance of users attribute Z_u. The movies weight are calculated by counting the number of similar genres. Note that the user and item connections are build by sampling the potential neighbor of each users and item. The objective of this sampling mechanism is to reduce the number of edges and reduce the computational cost. The weight in each graph will be used for local neighborhood sampling.

3.4. Overall Architecture

In this section we discuss the model used to generate the recommendation. The item graph and user graph are treated as different entity and will have vector representation independent from user. We call our proposed model graphSAGE Collaborative Filtering (SGCF). SGCF has similar framework with neural collaborative filtering, the embedding result of both user and item will be concatenated and fed to dense neural network. The graph embedding of SGCF is trained with edge weights, changing its neighbor sampling distribution. If the edge weight is uniform, all node neighbor will have uniform sampling probability distribution. Changing the weight will shift the sampling probability distribution to neighbors that has larger weight. In our case, the weight represent user taste similarity and item similarity, therefore the most similar item and user will have higher probability to be sampled.

In this paper, SGCF is trained with link prediction and regression problem formulation. First the the embedding for item graph and user graph are trained. Then the recommendation system neural network is trained with trained graph embedding. Notice in figure 2, both user and item embedding are trained separately and the resulting embedding model used to predict the rating score of pair user and item. The embedding is trained with loss function similar with graphSAGE, which formulated as follows:

$$\begin{aligned} J(z_u) = -log(\sigma({z_u}^T z_v)) - Q.\mathbb{E}_{v_n \sim P_n(v)} \cdot \\ log(\sigma(-{z_u}^T z_{v_n})), \end{aligned} \tag{2}$$



similarity weight
User Graph
User Neighbor Prediction
Weight applied for neighbor sampling
User Neighbor Feature Aggregation: Embedding Prediction
Concatenate
MLP
P
similarity weight
Item Graph
Item Neighbor Prediction
Weight applied for neighbor sampling
Item Neighbor Feature Aggregation: Embedding Prediction

Figure 2. SGCF Architecture

where v is node that co-occurs near u on fixed random walk, σ is a sigmoid function, P_n is negative sampling distribution and Q is the number of negative samples. In our case the sampling probability distribution $P_n(v)$ is not uniform. The weighted edges will alter the sampling probability distribution so that neighbors that have higher weight are more likely to be sampled. The graphSAGE model is used as the embedding generator.

Let a graphSAGE model M, a graph $G(V,E)$, node attrributes $x_v, v \in V$, model depth K, weight matrices $W_k, k \in \{1, ..., K\}$ and differentiable aggregator $AGGREGATE_k, k \in \{1, ..., K\}$. for each depth 1 to K, the model will uniformly sample the node neighborhood and aggregate the attributes using differentiable aggregator $AGGREGATE_k$ that yield the embedding vector of that node.

Notice that there are 2 steps to train SGCF. First the embedding is trained with link prediction framework, where each graph try to predict similarity between neighbor based on its feature and neighbors feature. This training framework will make the embedding of connected nodes are close to each other while non connected node has farther distance.

This model predicts embedding based on its feature and its neighbor's feature. To achieve this, the model aggregates node's neighborhood features and concatenate it with it's own and then fed it to a neural network which will produce the embedding based on equation (2). Our model use average operation to aggregate neighbors feature. The aggregator is formulated in equation 3

$$h_v^k = \sigma(\mathbf{W}.MEAN(\{h_v^{k-1}\} \cup \{h_u^{k-1}, \forall u \in N(v)\})), \quad (3)$$

where h^k denotes embedding value at k^{th} layer or k-hop neighbor embedding, σ denotes sigmoid function, $\mathbf{W}$ denotes trainable aggregator weights, and $N(v)$ denotes node v neighbors. In our model, we change the aggregation sampling so that the model morelikely to extract local neighborhood feature using strongly connected neighbors. In our case, this is a user who has the most similar preference of movies and movies who has the most similar genre. With the embedding model defined, our model concatenate users and item embedding and feed it to MLP as illustrated in figure 2. The MLP will act as rating predictor which will predict user rating to a movie. This prediction rating is used to construct item recommendation for user u as illustrated in figure 1. The MLP model is trained using mean squared error as the cost function.

To illustrate how our model works, take a look at figure 1. In the figure, there are 3 columns that separate each steps to train the embeddings. SGCF architecture takes use 2 graph to produce embeddings depicted in the leftmost column in figure 1. First, we calculate edge weight using cosine similarity of node features defined in equation 1 for user graph, and one hot encoding of genres for item graph. Second, we train the embedding for user graph and item graph using features (2nd and 3rd columns). The task to train the embeddings are predicting whether or not two nodes is connected. To formalize this let z_i be item features, z_u user features, G_i be item graph nodes, G_u be user graph nodes, h_i^k is item graph embedding and h_u^k is user graph embedding.

$$h_u^0 = z_u \forall u \in G_u \quad (4)$$

$$h_u^1 = \sigma(\mathbf{W}.MEAN(\{h_{u'}^0\} \cup \{h_{u'}^0, \forall u' \in N(u), \forall u \in G_u\})) \quad (5)$$

$$h_i^0 = z_i \forall i \in G_i \quad (6)$$

$$h_i^1 = \sigma(\mathbf{W}.MEAN(\{h_{i'}^0\} \cup \{h_{i'}^0, \forall i' \in N(i), \forall i \in G_i\})) \quad (7)$$

With the embedding defined, to calculate the loss in training, we put h_i^1 for the item graph and h_u^1 to the cost function as input for each graph embedding. After training the graph embedding for each user graph and item graph, we train the recommendation using the graph embeddings as input and give the recommendation model task to predict movie ratings. Note that the illustration above only uses 1 hop neighborhood. The embeddings produced by SGCF can be seen in 7. The embeddings behave exactly as intended in the cost function, where similar entities, in this case similar items with similar genres, are projected close to each other.

In the next section we will explain how we evaluate our model performance and how the experiment is conducted.

4. Experiments

In this section, we explain our experiment objective and methods to achieve the objective. First we explain our main objectives followed by detailed method used for the experiment. After that we explain how the recommendation list is constructed and how to evaluate the recommendations.

4.1. Experiment Settings

There are 2 main objective of this experiment. First objective is we want to investigate SGCF performance compared to neural collaborative filtering. Second, we want to investigate SGCF performance if we alter the problem formulation and embedding optimization. We choose neural collaborative filtering as baseline because our model is a variant of neural collaborative filtering where we modify the embedding part to graph based embedding. To achieve this objective, we evaluate the recommendation results with precision metric and recall metric. We use NDCG [20] [21] as precision metric and recall as recall metric. Both precision and recall metrics evaluated at top 5, 10, and 100.

First, we separate training and testing data in user level by 80:20. This way every user will have ground truth interaction stored in testing data. We use training data to extract features that requires user item interaction information. The graph embedding model is trained by generating negative samples of connection in each graph. The embedding models are trained for 10 epochs with binary cross entropy as the loss function.

In the experiment scenario, we train the embedding model with weight calculated from feature extraction phase and without. Furthermore, in training recommender system phase, we freeze the embedding result and retrain the embedding vector, lastly we experimented with SGCF where user and item graph are not separated, and the connection is determined from training data. In this case, we train the embedding and score prediction simultaneously in recommendation system training phase.

The recommendation system training phase is trained with validation randomly sampled 20% from training data. Evaluating the recommendation system is done by sampling 10000 users and compare the recommendation list to ground truth. We evaluate the recommendation by NDCG, and which defined in equation 10

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \quad (8)$$

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{log_2(i+1)} \quad (9)$$

$$NDCG = \frac{DCG_p}{IDCG_p}, \quad (10)$$

where P in equation 8 represent the number of query, in our case is the number of user which recommendation is compared to. The rel_i in equation 8 9 represent relevancy score, which in our case is the predicted ratings. The REL_p is recommendation ground truth and p is movie rank in the predicted recommendation list. Our reasoning behind using NDCG and recall to evaluate movie recommendation system is that NDCG can evaluate ranking quality while recall can evaluate movies which the user would watch. An ideal recommendation system would recommend the most likely item a user would interact first, which is why the order of recommendation matters. This quality can be measured using NDCG [20]. Recall is used as metric to make sure the recommendation system able to retrieve list of item user would likely to watch.

To construct the recommendation list, for each user we sample 1000 item the user haven't interacted (both in training and testing) and add all item in test data for that user as ground truth. We rank the user item pair based on the predicted scores. This way we can fairly compare the model ranking ability to the ground truth. This recommendation list is evaluated at top 5, 10 and 100 cut off from the recommendation system towards ground truth in test data.

The experiment is conducted by comparing SGCF variants to the baseline. We choose Neural Collaborative Filtering (NCF) as our baseline because the computation flow is very similar to it. The NCF model has both user and item embedding, where each embedding is represented as a matrix and trained simultaneously in recommendation system training phase.

4.1.1. Neural Collaborative Filtering. The neural collaborative filtering [13] proposed trainable interaction between user and item embeddings. In previous methods, the interaction between user and item embedding is formulated as dot product. The authors of NCF believe that replacing dot product operation with generic trainable model such as MLP able to boost the recommendation performance.

4.1.2. NGCF. Similar with neural collaborative filtering, neural graph collaborative filtering [19] has idea to replace user and item embedding as graph that contains interactions between user and items. The embeddings is calculated utilizing GCN. However, the GCN used in this methods is transductive.

4.1.3. PEAGAT. The e MetaPath- and Entity-Aware Graph Attention Network (PEAGAT) [10] propose unified framework to process relational information in graph. This method aggregates information over multiple metapath-aware subgraphs and fuse the aggregated information to obtain node representation using attention mechanism.

The SGCF variants we mentioned above are SGCF with edge weight, SGCF without weight, SGCF with frozen graph embedding, means the embeddings are not trained in recommendation training phase. The last variant is SGCF with link prediction formulation, where user and item are constructed in one graph.

4.2. Dataset

This work use movielens 25M dataset [22] to perform experiment. The Movielens 25M dataset contains user historical interaction with movies, user information and movie information. The data has approximately 62K movies, 162K users and 25 million rating / historical user interaction data. There are 19 different movie genres where movie can have more than 1 genre. Those genres are Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, (no genres listed). The category data are stored in string format. There also review information and synopsis that linked to other sites. However we decided to not to use user review and synopsis because it's outside this work's focus.

4.3. Result

In this section we will answer 2 research question. We answer research question 1 by our model with 3 baselines along with significance test to measure how significant our result compared to other methods. Research question 2 will be answered by conducting multi-label classification both on item and users. The task is to predict movie genre and user preferred movie based on the feature extractions.

4.3.1. Recommendation System Performance. We conduct experimentation on 4 different models, SGCF with edge weight, SGCF without edge weight, SGCF with the embedding locked (not changed), and NCF [13] and NGCF [19] as the baseline.

Our experiment result is illustrated in table 1. Both SGCF which embedding is optimized in recommendation system training has better performance than NCF in precision metric (NDCG) and recall metric. However, SGCF with link prediction framework and frozen embedding perform significantly worse than baseline, except in top 100 recall in SGCF with link prediction framework. Note that all model are trained in uniform setting training and test data split. Our weighted SGCF perform better than the baseline, however the weighted SGCF and SGCF has difference in which they're perform best at.

If we compare the two, weighted SGCF are better at retaining items in the ground truth however it wasn't able to outperform SGCF in the ranking task. Our experiment shows that SGCF are better at predicting the relevancy scores.

In figure 3, we illustrate weighted SGCF loss in training and validation for 10 epoch training. We add early stopping to prevent the model to be over-trained and the model loss stop decreasing at epoch 10. Our model is trained in 2 phase, embedding training phase and downstream task fine-tuning. The embedding training phase is conducted in unsupervised manner, where for each graph, the

Method	NDCG@5	NDCG@10	NDCG@100	Recall@5	Recall@10	Recall@100
Weighted SGCF	$0.5923^{\alpha\beta\gamma}$	$0.6027^{\alpha\beta\gamma}$	$0.7060^{\alpha\beta\gamma}$	**0.6480**$^{\alpha\beta\gamma}$	**0.6482**$^{\alpha\beta\gamma}$	$0.8969^{\alpha\beta\gamma}$
SGCF	**0.5942** $^{\alpha\beta\gamma}$	**0.6029** $^{\alpha\beta\gamma}$	**0.7079**$^{\alpha\beta\gamma}$	$0.6476^{\alpha\beta\gamma}$	$0.6456^{\alpha\beta\gamma}$	$0.8886^{\alpha\beta\gamma}$
SGCF (frozen graph embedding)	0.1839	0.2025	0.4516	0.2108	0.2377	0.8705
SGCF (Link Prediction)	0.0865	0.1227	0.4711	0.0985	0.1610	**0.9294**
PEAGAT	0.5372	0.5475	0.5850	0.6388	0.6440	0.8784
NGCF	0.4472	0.4866	0.5230	0.6088	0.6140	0.7758
Neural Collaborative Filtering	0.5870	0.5990	0.6992	0.6401	0.6450	0.8780

Table 1. Experiment Results. The superscript symbols represent statistical significance $p < 0.05$ against baselines where α : PEAGAT, β : $NGCF$ and γ : NCF

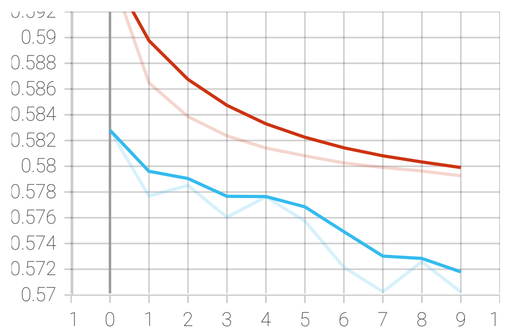


Figure 3. SGCF training epoch. Y axis denote mean squared error loss. X axis denote epoch. Red line denote validation loss, blue line denote training loss. Note that the thick line is smoothed, the thin line is the real value

Table 2. Recommendation Ground Truth

user	rating	ground truth title
26811	5.0	The Hateful Eight (2015)
26811	4.5	28 Days Later (2002)
26811	4.5	Inglourious Basterds (2009)
26811	4.5	Grand Budapest Hotel, The (2014)
26811	4.5	Deer Hunter, The (1978)
26811	4.0	Death Proof (2007)
26811	4.0	Hot Fuzz (2007)
26811	4.0	Dallas Buyers Club (2013)
26811	4.0	The Revenant (2015)
26811	4.0	Seven Samurai (Shichinin no samurai) (1954)

embedding is tasked to predict if two entities in the graph are connected or not. Then the fine tuning is conducted to predict the relevancy score of user and item pairs. The fine tuning is performed by taking trained embedding and combine the embedding to a downstream task which is predicting relevancy scores.

In order to achieve our experiment objective, we compared our baseline and SGCF model embedding result and recommendation results on 2 random picked user with top 10 recommended items against the ground truth in table 2. We focused on our first objective because our experiment result have given the evidence on other SGCF variant performance against the strongest baseline that use graph embedding (PEAGAT).

We compare top 10 recommendation result from SGCF and PEAGAT. To conduct the comparison, We sample 1 random user id and we pass the id to the recommendation system and generate top 100 recommended movies the user might like, which we cut off to top 10 for analysis purpose. Furthermore, we compare the 2 recommendations from SGCF and PEAGAT to ground truth. User with id 26811 has strong affinity (top 10 movie genres) with genre Film-Noir, Western, Thriller, Mystery, Drama, Comedy, Horror, Crime, War, Action in that order. SGCF model able to hit 5 out of 10 movies recommended to user 26811. On the other hand, PEAGAT model able to hit 3 out of 10 movies recommended to user 26811. SGCF model have better recall compared with PEAGAT.

4.3.2. Embedding Quality Analysis. In this section, we will investigate the quality of user and item embeddings by comparing them with NCF. We didn't compare it with PEAGAT and NGCF because both method use single graph that contains relationship between user and item, whereas our method use them separately. However, We are able to compare recommendation result for weighted SGCF and PEAGAT. We use PEAGAT as comparison because PEAGAT is the strongest baseline.

First, take a look in figure 6 and figure 7 on horror genre attached in the appendix, the embedding visualization are projected using TSNE algorithm [23] in the same parameter (100 iterations and 63 perplexity) without supervision. The NCF embedding tend to tightly group similar categories together such as comedy/horror movies and horror/sci-fi movies. On the other hand, SGCF embedding projects the items more sparsely and more aware on its subcategory. For example, the horror/sci-fi genre movies are projected close to horror/fantasy. This behavior shows that the embedding produced by SGCF model able to learn "nuance" of the movie which we didn't explicitly give the information. For example, The ability to learn the movie nuance and interact it with user projections is one reason why SGCF perform better in all of the performance metrics against the baseline.

Table 3. Recommendation Result Comparison

Methods	user	score	title	NDCG@10	Recall@10
		4.37	City of God (Cidade de Deus) (2002)		
		4.35	Deer Hunter, The (1978)		
		4.32	Lost in Translation (2003)		
		4.28	Seven Samurai (Shichinin no samurai) (1954)		
		4.19	Wrestler, The (2008)		
SGCF	26811	4.16	Grand Budapest Hotel, The (2014)	0.7930	0.8000
		4.13	Rome, Open City (a.k.a. Open City)		
		4.12	The Hateful Eight (2015)		
		4.09	Perfect Candidate, A (1996)		
		4.08	28 Days Later (2002)		
		4.73	Seven Samurai (Shichinin no samurai) (1954)		
		4.39	Lost in Translation (2003)		
		4.33	Deer Hunter, The (1978)		
		4.32	Rome, Open City (a.k.a. Open City) (Roma, città aperta) (1945)		
		4.28	City of God (Cidade de Deus) (2002)		
PEAGAT	26811	4.21	Frank (2014)	0.5964	0.5000
		4.21	Confessions of a Dangerous Mind (2002)		
		4.20	Personal Journey with Martin Scorsese Through American Movies, A (1995)		
		4.19	Grand Budapest Hotel, The (2014)		
		4.16	Shirkers (2018)		

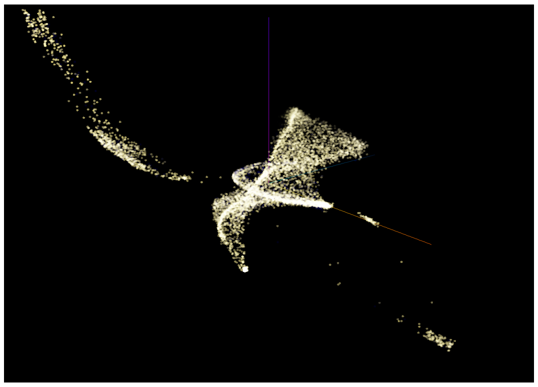


Figure 4. NCF user embedding visualization using T-SNE algorithm. The illustration highlight users who like horror movies. Brighter dots means higher affinity to Horror genre

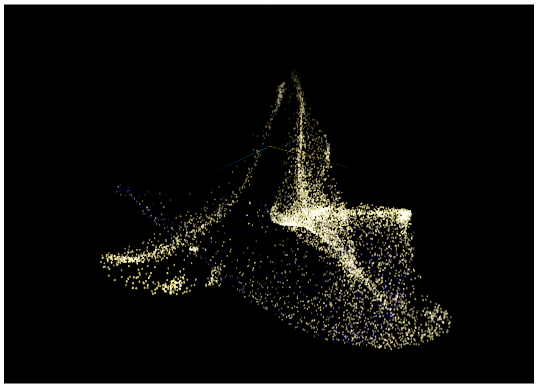


Figure 5. SGCF user embedding visualization using T-SNE algorithm. The illustration highlight users who like horror movies. Brighter dots means higher affinity to Horror genre

Table 4. Multi-label classification result

Embeddings	Average Precision@19	Average F1
User	0.9435	0.9215
Movie	0.9620	0.9450

This behavior also shown in user embeddings. Notice that in figure 4 and figure 5 user embedding on NCF model tend to be densely grouped. On the other hand, SGCF embedding are more sparse and more specific to the user's taste of movies, which is another evidence that can explain why SGCF perform better compared to NCF.

If we investigate further on the SGCF itself, the SGCF model are designed very similar to NCF model. The main difference between them is the starting point to optimize the embedding, the NCF model is initialized by random while SGCF model initialized from graph embedding. Our experiment result shows that the graph embedding alone does not able to perform better than NCF. We further tested it with other SGCF variant where we trained the SGCF using link prediction framework which perform worse than SGCF where the embedding is not trained in recommendation system training phase.

Furthermore, to provide evidence that the generated embeddings are able to capture useful signals, we will conduct experiment on both embeddings. The embeddings will be given a downstream task to predict user preference and movie genre. The task will be formulated as multi-label classifications with one vs the rest settings.

The experiment result illustrated in table 4 shows that both embeddings can capture useful signals, supported by the average precision for 19 class and the average f1 scores. The embedding used in this experiment is embeddings that produced in the recommendation system training phase. There are information loss in the embedding if we compare them to embbedings that produced before recommendation system training phase. Note that the embeddings before recommendation system training phase has the downstream task label as feature which will make the embedding has 100% accuracy. To conclude this experiment analysis, we shows that SGCF model are able to discern movies based on its nuance rather than solely relies on movies genre. This behavior is supported by evidence that we observed on the embedding projections. We also provide the evidence on the embedding quality, where the embedding can capture useful features which has been shown by downstream task experiment. We also shows that the weighted inductive graph embedding is useful for recommendation system task.

5. Conclusion

We have experimented with the inductive graph embedding based recommendation system SGCF. We also analyzed the SGCF recommendation performance and its variations compared to baseline. Both SGCF with edge weight and without weight perform better than baseline. The difference between both SGCF model variants lies in its ability in precise ranking and recall. The SGCF model with weight is better at retaining top-n recommendation from ground truth, whereas SGCF model without edge weight are better at precise ranking. This observation is supported by our experiment result on table 1. We also show that graph embedding is able to capture useful signals based on projection and downstream task testing, which makes the embedding effective. Based on the evidence that we have obtained so far, we believe that graph structures contribute positively to recommendation given that we feed informative relational data. We also confirmed the effectiveness of the SGCF model compared to baseline and analyzed SGCF performance.

References

[1] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The youtube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 293–296.

[2] J. Freyne, M. Jacovi, I. Guy, and W. Geyer, "Increasing engagement through early recommender intervention," in *Proceedings of the Third ACM Conference on Recommender Systems*, ser. RecSys '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 85–92.

[3] F. Isinkaye, Y. Folajimi, and B. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 235–244.

[6] K. Yang and L. Toni, "Graph-based recommendation system," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 798–802.

[7] C. Feng, Z. Liu, S. Lin, and T. Q. Quek, "Attention-based graph convolutional network for recommendation system," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7560–7564.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1025–1035.

[10] Z. Han, M. U. Anwaar, S. Arumugaswamy,

T. Weber, T. Qiu, H. Shen, Y. Liu, and M. Kleinsteuber, “Metapath- and entity-aware graph neural network for recommendation,” *CoRR*, vol. abs/2010.11793, 2020. [Online]. Available: https://arxiv.org/abs/2010.11793

[11] C. Xu, “A novel recommendation method based on social network using matrix factorization technique,” *Information Processing & Management*, vol. 54, no. 3, pp. 463–474, 2018.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW ’01. New York, NY, USA: Association for Computing Machinery, 2001, p. 285–295.

[13] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[14] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, “Deep item-based collaborative filtering for top-n recommendation,” *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 3, pp. 1–25, 2019.

[15] H. Wang, N. Wang, and D.-Y. Yeung, *Collaborative Deep Learning for Recommender Systems*. New York, NY, USA: Association for Computing Machinery, 2015, p. 1235–1244.

[16] S. Seo, J. Huang, H. Yang, and Y. Liu, “Interpretable convolutional neural networks with dual local and global attention for review rating prediction,” in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 297–305.

[17] W. Yu and Z. Qin, “Graph convolutional network for recommendation with low-pass collaborative filters,” in *ICML*, 2020.

[18] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, “Graph convolutional networks: a comprehensive review,” *Computational Social Networks*, vol. 6, no. 1, p. 11, 2019.

[19] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, “Neural graph collaborative filtering,” in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 165–174.

[20] Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu, “A theoretical analysis of ndcg type ranking measures,” in *Conference on learning theory*. PMLR, 2013, pp. 25–54.

[21] N. Craswell, *Mean Reciprocal Rank*. Boston, MA: Springer US, 2009, pp. 1703–1703.

[22] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, Dec. 2015.

[23] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.

Appendix A.
Embedding Visualization

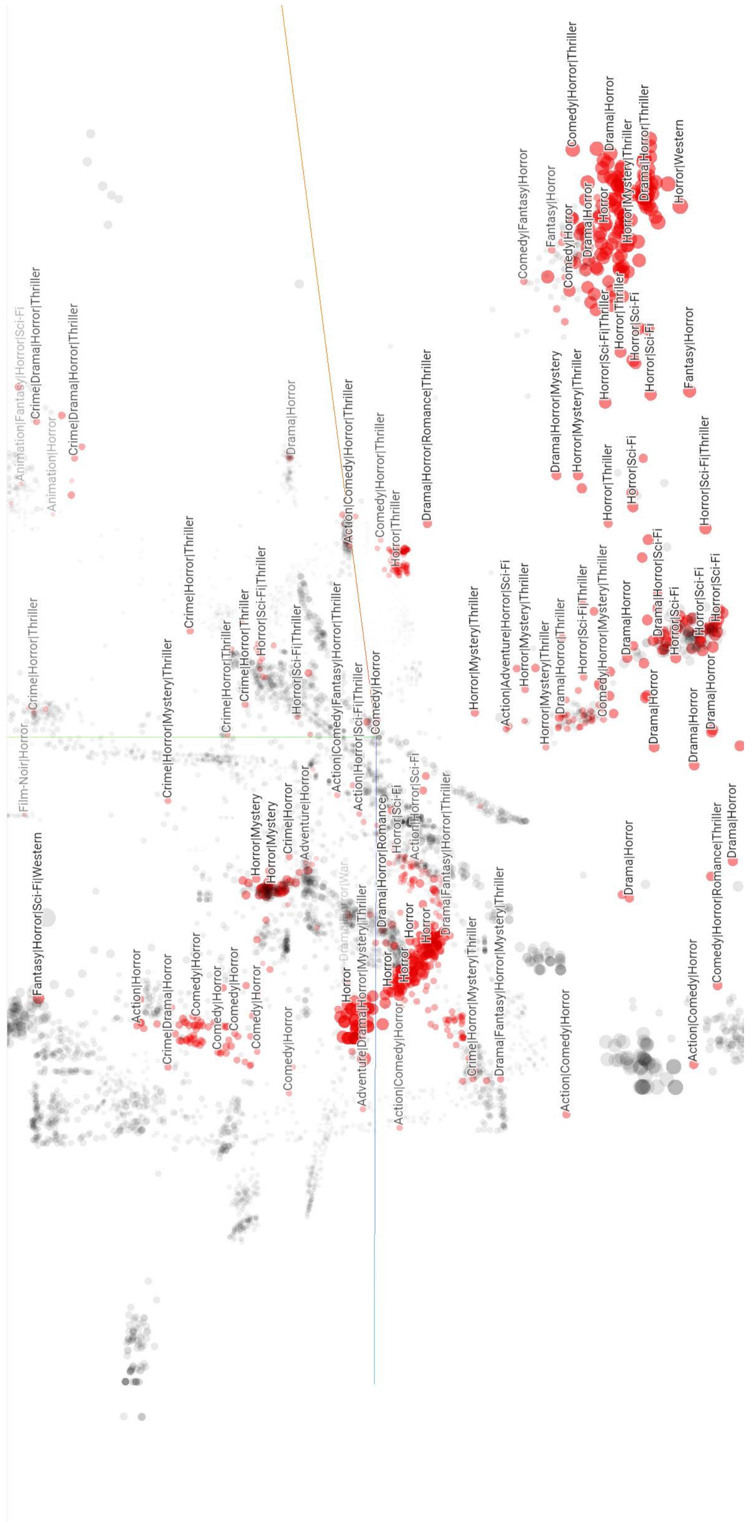


Figure 6. NCF movie embedding visualization using T-SNE algorithm. The illustration highlight different movies with horror genre as one of its genre

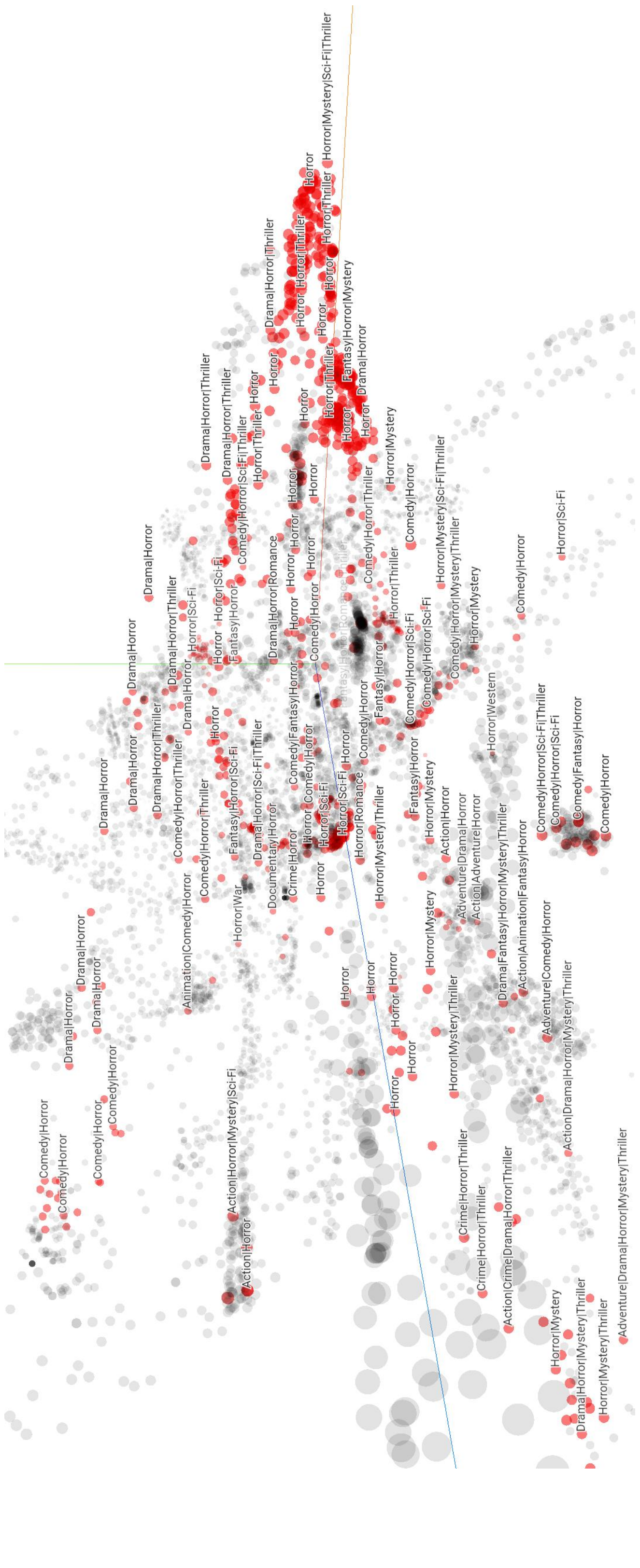


Figure 7. SGCF movie embedding visualization using T-SNE algorithm. The illustration highlight different movies with horror genre as one of its genre