# GRAPH-BASED RECOMMENDER SYSTEM FOR BIPARTITE NETWORK

**June 19, 2020**

XIN SUN - YUERUI YAN
University of Heidelberg
Department of Mathematik und Informatik & IWR

# Contents


0.1 Abstract . . . 2
0.2 Recommender System — Xin SUN . . . 3
0.2.1 Baseline Movie Recommender System using Graph-based approaches . . . 4
0.2.1.1 Content-based filtering approach — Xin SUN . . . 4
0.2.1.2 User-based filtering approach — Yuerui YAN . . . 5
0.3 Project Experiment . . . 6
0.3.1 Dataset and Graph Network Analysis — Xin SUN & Yuerui YAN . . . 6
0.3.1.1 Dataset Description . . . 6
0.3.1.2 Graph Construction . . . 6
0.3.1.3 Graph Analysis . . . 7
0.3.1.3.1 Degree distribution . . . 7
0.3.1.3.2 Degree centrality distribution . . . 7
0.3.1.3.3 Clustering coefficients . . . 7
0.3.1.3.4 Community detection . . . 7
0.3.1.3.5 Most Favourite Movie for User . . . 8
0.3.2 Graphical Neural Network based Movie Recommender System — Xin SUN . . . 9
0.3.2.1 Inspiration . . . 9
0.3.2.2 Graph Embedding . . . 9
0.3.2.2.1 Random walk . . . 9
0.3.2.2.2 Language model - Skip-gram . . . 11
0.3.2.2.3 DeepWalk Algorithm . . . 13
0.3.2.3 Additional Graphical Features . . . 14
0.3.2.4 Construction of Neural Network . . . 15
0.3.2.5 Result Analysis . . . 16
0.4 Conclusion . . . 18

## 0.1 Abstract

Recommendation system is an assistive model for users with the intent of suggesting a set of new items to view (e.g., movie, news, research articles etc.) or buy (e.g., book, product etc.). Nowadays it has altered the way of seeking out the things of our interest by using information filtering approach. Recommender systems usually make use of either or both collaborative filtering and content-based filtering (also known as the personality-based approach) as well as other systems such as knowledge-based systems. Collaborative filtering approaches build a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete, pre-tagged characteristics of an item in order to recommend additional items with similar properties. Current recommender systems typically combine one or more approaches into a hybrid system.

In the proposed system, we designed a hybrid model-based way based on graph attributes from used data, including DeekWalk algorithm for embedding user and movie nodes as vector, degree of nodes, community detection result, nodes demographic attributes, etc. After extracting all these graphic features by constructing an undirected bipartite graph from raw data and doing graph analysis, we then input all prepared features into a MLP neural network and finally output the result indicating whether we should recommend one specific movie to a specified user. And this can also being seen as a link prediction problem between two nodes in graph. And the MovieLens dataset was used to explore the proposed hybrid system.

Besides, we also re-implement a baseline recommender system: a content-based movie recommendation which is automatically made using a graph based approach according to past film preferences of users and alternatively using demographic information of users. And finally a combination on outputs of these two techniques reveals more precise recommendations concerning movies.

## 0.2 Recommender System — Xin SUN

A recommender system or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering approach that collects ratings or preference information from users to make new suggestions to them for example in movie, video, book, music album recommendation or to recommend social elements such as people or groups to follow using a model built from the characteristics of an item or the users' demographic information and their social environment.

Recommender systems are utilized in a variety of areas and are most commonly recognized as playlist generators for video and music services like Netflix, YouTube and Spotify, product recommenders for services such as Amazon, or content recommenders for social media platforms such as Facebook and Twitter. These systems can operate using a single input, like music, or multiple inputs within and across platforms like news, books, and search queries. There are also popular recommender systems for specific topics like restaurants and online dating. Recommender systems have also been developed to explore research articles and experts, collaborators and financial services. Also Recommender systems are a useful alternative to search algorithms since they help users discover items they might not have found otherwise.

Normally there are some frequently used approaches for recommender system:

- Collaborative filtering
- Content-based filtering
- Model-based filtering
- Hybrid recommender systems

Each type of system has its strengths and weaknesses. In the above approaches, Collaborative and Content-based filtering require a large amount of information about a user to make accurate recommendations. This is the example of the cold start problem, and is common in collaborative filtering systems. Whereas some other methods need very little information to start, it is far more limited in scope (for example, it can only make recommendations that are similar to the original seed). Nowadays, we can use Hybrid recommender systems which ensemble several different traditional recommender system approaches and this hybrid methods normally can give us better result than only single approach in recoomender system.

### 0.2.1 Baseline Movie Recommender System using Graph-based approaches

In baseline system, we applied two basic approaches, that are used for the construction of recommendation systems, are content-based and collaborative filtering techniques. A content-based recommendation system takes item's descriptions, which can be genre of a movie in case of movie recommendation or category of a book for book recommendation etc., into consideration so that similar items which are of particular interest to the user are likely to be suggested as a recommendation. The history of user interactions such as watched / rated movies or purchased items are collected to deduce user preferences. On the other side, a collaborative filtering handles information gathered from other people similar to the respective user who the system will recommend an item. The idea is that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. Two versions of collaborative filtering approaches are commonly used as user-based and item-based filtering.

#### 0.2.1.1 Content-based filtering approach — Xin SUN

The first neighbors of each movie node point out the users who rated / watched this movie and conversely, the movies that were rated by users are placed in the first neighbors of each user node. Using this information, movies which were watched and not watched by each user were extracted from the graph. This step is important because the most preferred film genres for each user are determined by looking through the frequency of the watched film genres. The film genres are ordered from the most preferred to the least one so that the proposed system suggests new movies to the user from the not watched film set of the corresponding user by selecting from the most preferred movie genre of him / her. This step can be stated as content-based recommendation technique as our first baseline system since "genre" feature of the movie item is predicated on to make suggestion. As a result of this technique five movie recommendations are made for each user from their most favored category and with the ones with highest watch counts.

| | userId | rec_movie1 | rec_movie2 | rec_movie3 | rec_movie4 | rec_movie5 | rec_movie6 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | [Toy Story (1995)] | [Bug's Life, A (1998)] | [E.T. the Extra-Terrestrial (1982)] | [Toy Story (1995)] | [American Beauty (1999)] | [Shakespeare in Love (1998)] |
| 1 | 20 | [Star Wars: Episode IV - A New Hope (1977)] | [Star Wars: Episode V - The Empire Strikes Bac... | [Star Wars: Episode IV - A New Hope (1977)] | [Star Wars: Episode V - The Empire Strikes Bac... | [L.A. Confidential (1997)] | [2001: A Space Odyssey (1968)] |
| 2 | 100 | [Star Wars: Episode IV - A New Hope (1977)] | [Star Wars: Episode V - The Empire Strikes Bac... | [Star Wars: Episode IV - A New Hope (1977)] | [Star Wars: Episode V - The Empire Strikes Bac... | [American Beauty (1999)] | [Schindler's List (1993)] |
| 3 | 500 | [American Beauty (1999)] | [Shakespeare in Love (1998)] | [E.T. the Extra-Terrestrial (1982)] | [Toy Story (1995)] | [Toy Story (1995)] | [Bug's Life, A (1998)] |
| 4 | 1000 | [American Beauty (1999)] | [Schindler's List (1993)] | [Schindler's List (1993)] | [Star Wars: Episode V - The Empire Strikes Bac... | [Star Wars: Episode IV - A New Hope (1977)] | [Star Wars: Episode V - The Empire Strikes Bac... |

**Figure 1:** Recommended movies by content-based filtering

#### 0.2.1.2 User-based filtering approach — Yuerui YAN

The second baseline recommendation technique takes the user's demographic data such as age and occupation into consideration. After extracting similar user groups, new film suggestions are made. Because the age attribute is continuous, it was discretized into five groups as the intervals of 7-17, 18-24, 25-35, 36- 50, and 51-73. Later, the most watched film categories for each age group were identified. Also, as a result of two applied techniques, a combination of movie suggestion list was formed. The first three films are selected from content-based approach; fourth and fifth suggestions come from user-based approach.

## 0.3 Project Experiment

### 0.3.1 Dataset and Graph Network Analysis — Xin SUN & Yuerui YAN

#### 0.3.1.1 Dataset Description

The MovieLens database, that is available for public use, was chosen in our experimental study. The selected version of the dataset consists of 9125 movies, 671 users who rated movies between January 9, 1995 and October 16, 2016, 100004 ratings from the respective users. After removing the movies that are not rated by users, the remaining part covers 9066 movies. All selected users rated at least 20 movies. 18 genres are present (Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western) in the given context. Each dataset (Movies, Users and Ratings) has different attributes which are displayed in Figure 1.

| Dataset | Attributes |
|---|---|
| Movies | movieID, title, genres |
| Users | userID, movieID, rating, timestamp |
| Ratings | UserID, age, gender, occupation, zipcode |

**Figure 2:** Used MovieLens Datasets and Their Attributes

#### 0.3.1.2 Graph Construction

The core part of the application is representing the system as a graph model. Graph G is composed of a pair of sets (V, E) where E is edges and V is vertices or nodes of the given graph. In

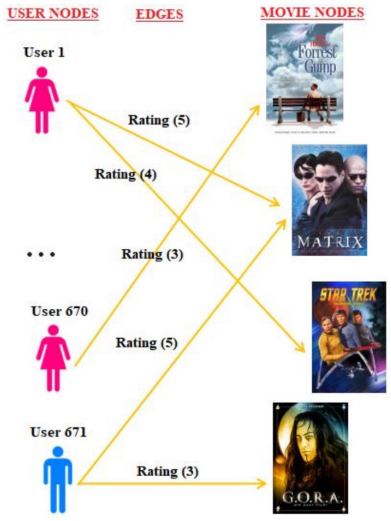


**Figure 3:** Movie recommendation system as a graph

order to construct movie recommendation system, movies and users should be placed into the vertices and edges should represent ratings (1 to 5) given by users to movies. Each node has its own features such that users have their respective IDs, age, occupation and gender information; movies have their respective IDs, title, year of the release, and genre information. Since the dataset contains Users and Movies with UserID, MovieID and users' ratings for each movies, we can build a bipartite graph between Users and Movies. And the link between user and movie is rating score which can been seen as the weight of edge. Figure. 2 displays the visual representation of the movie recommendation system as a bipartite graph.

#### 0.3.1.3 Graph Analysis

**0.3.1.3.1 Degree distribution** In graph theory, the degree (or valency) of a vertex of a graph is the number of edges incident to the vertex, with loops counted twice. From Figure 3, we could see the coarse plot for degree distribution.

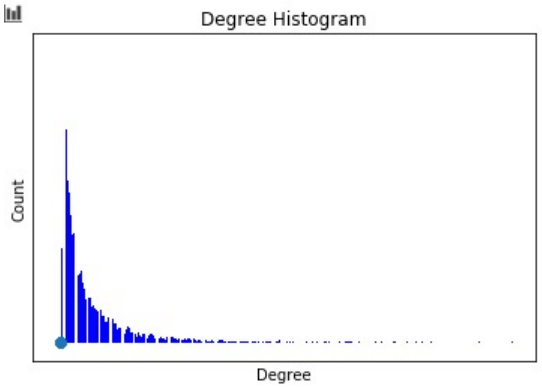


**Figure 4:** Degree distribution

**0.3.1.3.2 Degree centrality distribution** We define the degree centrality for a node v is the fraction of nodes it is connected to.

**0.3.1.3.3 Clustering coefficients** We computed the clustering coefficient on each nodes and this Clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together.

**0.3.1.3.4 Community detection** We find communities in graph using Clauset-Newman-Moore greedy modularity maximization. This method currently supports the Graph class and does not consider edge weights. Greedy modularity maximization begins with each node in its own community and joins the pair of communities that most increases modularity until no such pair exists. And after that, we get 34 communities in our data.

**0.3.1.3.5 Most Favourite Movie for User** There are 21 different kinds of occupations for users, and also 13 categories for movies, therefore we can explore which movie category is the most favourite type for each different occupations.

| Occupation | The Most Favourite Movie Genre | Occupation | The Most Favourite Movie Genre |
|---|---|---|---|
| administrator | Drama | marketing | Drama |
| artist | Drama | none | Comedy |
| doctor | Drama | other | Drama |
| educator | Drama | programmer | Drama |
| engineer | Drama | retired | Comedy |
| entertainment | Drama | salesman | Drama |
| executive | Drama | scientist | Drama |
| healthcare | Drama | student | Drama |
| homemaker | Drama | technician | Drama |
| lawyer | Action | writer | Drama |
| librarian | Drama | | |

**Figure 5:** The most Favourite Movie Categories for User Occupations

### 0.3.2 Graphical Neural Network based Movie Recommender System — Xin SUN

#### 0.3.2.1 Inspiration

We need approach for learning latent representations of vertices in a network. These latent representations encode social relations in a continuous vector space, which is easily exploited by statistical models. Deep- Walk generalizes recent advancements in language mod- eling and unsupervised feature learning (or deep learning) from sequences of words to graphs.

The sparsity of a network representation is both a strength and a weakness. Sparsity enables the design of efficient discrete algorithms, but can make it harder to generalize in statistical learning. we introduce deep learning (unsupervised feature learning) [2] techniques, which have proven successful in natural language processing, into network analysis for the first time. We develop an algorithm (DeepWalk) that learns social representations of a graph's vertices, by mod- eling a stream of short random walks. Social representa- tions are latent features of the vertices that capture neigh- borhood similarity and community membership. These la- tent representations encode social relations in a continuous vector space with a relatively small number of dimensions. DeepWalk generalizes neural language models to process a special language composed of a set of randomly-generated walks. These neural language models have been used to capture the semantic and syntactic structure of human lan- guage [6], and even logical analogies.

DeepWalk takes a graph as input and produces a la- tent representation as an output. The result of applying our method to the well-studied Karate network is shown in Figure 1. The graph, as typically presented by force-directed layouts, is shown in Figure 1a. Figure 1b shows the output of our method with 2 latent dimensions. Beyond the striking similarity, we note that linearly separable portions of (1b) correspond to clusters found through modularity maximization in the input graph (1a) (shown as vertex colors).

#### 0.3.2.2 Graph Embedding

**0.3.2.2.1 Random walk** "Random Walk is a technique to extract sequences from a graph. We can use these sequences to train a skip-gram model to learn node embeddings."

The goal of Random Walk is to discover neighborhoods in the network. To do so, we generate a fixed number (k) of random walks starting at each node. The length of each walk is pre-determined. Thus, when this stage is finished, we obtain k node sequences of length l. The gist of random walk generation is the following assumption:

"Adjacent nodes are similar and should have similar embeddings."

Based on this assumption, we will accept nodes that co-occur in a path as similar nodes. This means that the frequency of co-occurrence in random walks is an indicator of node similarity. Note that this is a reasonable assumption since edges, by design, mostly denote similar or interacting nodes in a network.

Let me illustrate how Random Walk works. Let's consider the undirected graph below:

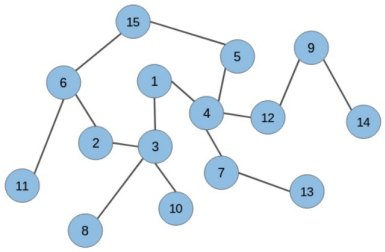


**Figure 6:** Undirected graph

We will apply Random Walk on this graph and extract sequences of nodes from it. We will start from Node 1 and cover two edges in any direction:

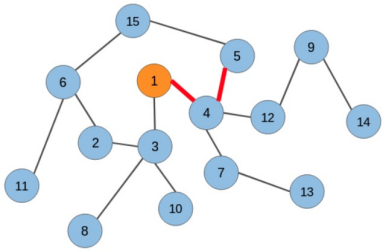


**Figure 7:** Random Walk start from Node 1 with two edges

From node 1, we could have gone to any connected node (node 3 or node 4). We randomly selected node 4. Now again from node 4, we have to randomly choose our way forward. We'll go with node 5. Now we have a sequence of 3 nodes: [node 1 – node 4 – node 5].

Let's generate another sequence, but this time from a different node: Let's select node 15 as the originating node. From nodes 5 and 6, we will randomly select node 6. Then from nodes 11 and 2, we select node 2. The new sequence is [node 15 – node 6 – node 2].

We will repeat this process for every node in the graph. This is how the Random Walk technique works.

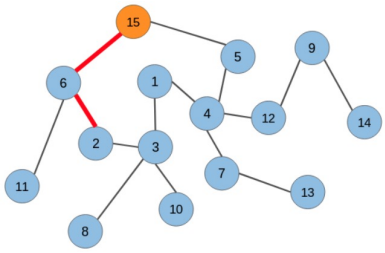


**Figure 8:** Random Walk start from Node 15 with two edges

Here the effect of k and l is important. The more k is increased, the more the network is explored since more random walks are produced. On the other hand, when l is increased, paths become longer and more distant nodes are accepted as similar nodes. This corresponds to relaxing the similarity constraint and can introduce noisy and misleading co-occurrences.

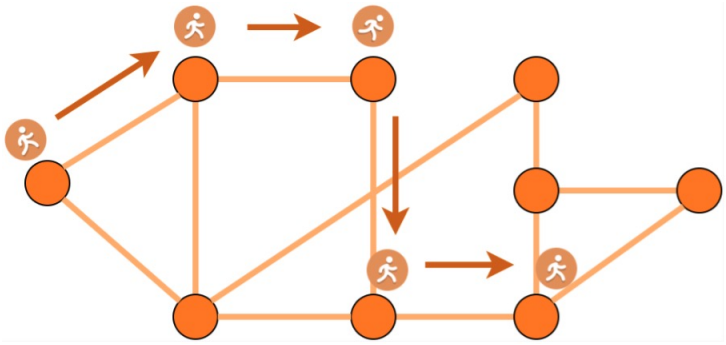

**Figure 9:** Random walk example of length 5 on a network

**0.3.2.2.2 Language model - Skip-gram** SkipGram algorithm is a popular technique used to learn word embeddings. It was introduced by Mikolov et. al. in their well-known paper on word2vec[2]. Given a corpus and a window size, SkipGram aims to maximize the similarity of word embeddings' of the words that occur in the same window. These windows are also called context in NLP. This idea is based on the assumption that:

"Words that occur in the same context, tend to have close meanings. Therefore, their embeddings should be close to each other as well."

We're going to train the neural network to do the following. Given a specific word in the middle of a sentence (the input word), look at the words nearby and pick one at random. The network is going to tell us the probability for every word in our vocabulary of being the "nearby word" that we chose. The output probabilities are going to relate to how likely it is find each vocabulary word nearby our input word. For example, if you gave the trained network the input word "Soviet", the output probabilities are going to be much higher for words like "Union" and

"Russia" than for unrelated words like "watermelon" and "kangaroo". We'll train the neural network to do this by feeding it word pairs found in our training documents. The below example shows some of the training samples (word pairs) we would take from the sentence "The quick brown fox jumps over the lazy dog." I've used a small window size of 2 just for the example. The word highlighted in blue is the input word.

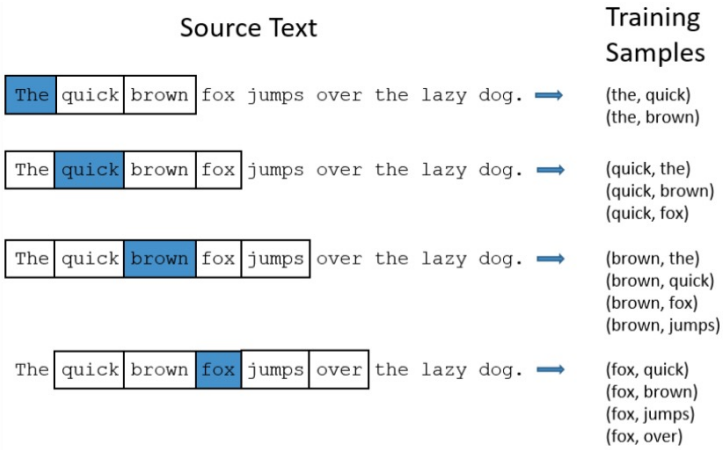


**Figure 10:** Training example of Skip-gram model

Here we can see internal architecture of Skip-gram model. Skip-gram aims to predict the context word for a given target word.

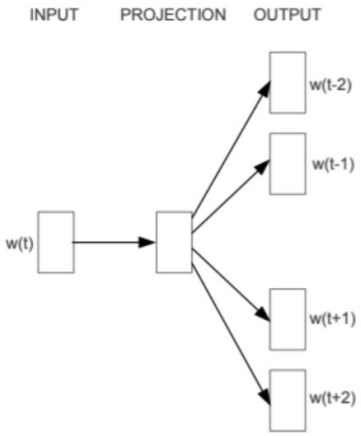


**Figure 11:** Architecture of Skip-gram model

As we can see w(t) is the target word or input given. There is one hidden layer which performs the dot product between the weight matrix and the input vector w(t). No activation function is used in the hidden layer. Now the result of the dot product at the hidden layer is passed to the output layer. Output layer computes the dot product between the output vector

of the hidden layer and the weight matrix of the output layer. Then we apply the softmax activation function to compute the probability of words appearing to be in the context of w(t) at given context location.

To adapt SkipGram to networks, we need to determine what context corresponds to in the network world. This is where the random walks come into play. We can think of each walk produced in the previous step as a context or word window in a text. Thus, we can maximize the similarities of embeddings of nodes that occur on the same walks.

"In this sense, node sequences in networks correspond to word sequences in text."

To learn the embeddings via SkipGram, we first generate random vectors of dimension d for each node. Secondly, we iterate over the set of random walks and update the node embeddings by gradient descent to maximize the probability of the neighbors of a node, given the node itself. This can be achieved by the softmax function. When all of the walks are processed, we can either continue optimization with additional passes over the same walk set or generate new walks on the network.

**0.3.2.2.3 DeepWalk Algorithm** From above 2 approaches, we can know that the DeepWalk algorothm can use language modeling approaches to learn node embeddings via leveraging the local structures in the network. With node embeddings at hand, we can use appropriate ML methods for GDL tasks such as node classification, link prediction, and community detection tasks. We can see the concrete steps of DeepWalk in following figure.

**Algorithm 1** DeepWalk($G$, $w$, $d$, $\gamma$, $t$)

**Input:** graph $G(V, E)$
window size $w$
embedding size $d$
walks per vertex $\gamma$
walk length $t$
**Output:** matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$
Initialization: Sample $\Phi$ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree $T$ from $V$
**for** $i = 0$ to $\gamma$ **do**
  $\mathcal{O}$ = Shuffle($V$)
  **for each** $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram($\Phi$, $\mathcal{W}_{v_i}$, $w$)
  **end for**
**end for**

**Figure 12:** Steps of DeepWalk

The random walk generator firstly takes a graph G and samples uniformly a random vertex vi as the root of the random walk Wvi. A walk samples uniformly from the neighbors of the last

vertex visited until the maximum length (t) is reached. After generating node-sequences, we need to feed them to a skip-gram model to get node embedding vectors. This entire process is how DeepWalk algorithm works.

By combining both truncated random walks and neural language models we formulate a method which satisfies all of our desired properties. This method generates representations of social networks that are low-dimensional, and exist in a continuous vector space. Its representations encode latent forms of community membership, and because the method outputs useful intermediate representations, it can adapt to changing network topology.

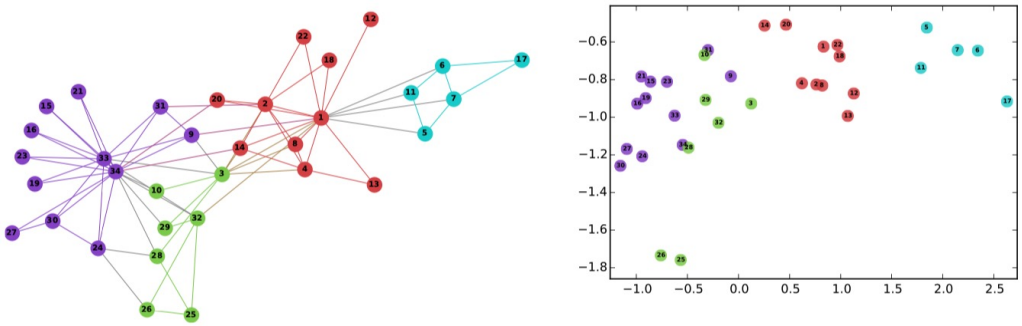


**Figure 13:** DeepWalk embedding representation

From the above figure which show us the representation of an undirected graph by DeepWalk, we can fiund that it learns a latent space representation of social interactions in Rd. The learned representation encodes community structure so it can be easily exploited by standard classification methods. Note the correspondence between community structure in the left input graph and the right embedding representation. Vertex colors represent a modularity-based clustering of the input graph.

#### 0.3.2.3 Additional Graphical Features

From the section 0.4.1.3 Graph Analysis, we have already done some statistical analysis and exploration on dataset and got some graphical attributes of our used data, such as degree distribution of User and Movie nodes, which give us degrees of each user and movie nodes. Also we did community detection which cluster users and movies according to the data features. Therefore there are fixed cluster for each users and movies node it belongs to.

As a result, we can adopt all these graphic attributes of our data as graphic features. Then we can input all these graphic features into our neural network based recommender system with both User and Movie nodes embedding vectors. Then we can treat this neural network based recommender system model as a multi-label classification problem, which can return us the prediction whether a movie will be rated or like by specified user or not.

Therefore, the total graphic features we added to the neural network are as follow:

- Degree for all User and Movie nodes
- Degree centrality for all User and Movie nodes
- Clustering coefficients for all User and Movie nodes
- Preferential attachment for all User and Movie nodes
- Community (cluster) for all User and Movie nodes

#### 0.3.2.4 Construction of Neural Network

After we got the embedding vector representation for each User and Movie nodes and all graphic attributes vectors as feature vectors, we can construt a Multiple Layer Perceptron (MLP) neural network. And the we set it as a binary classification model, which will predict whether the input User will rate the input Movie or not as kind of link prediction problem.

And the detailed MLP network architecture and hyper-parameters setting up are as follow: We build a two-layer fully connected network with ReLU activation function. And the batch size we set as 100, learning rate is 0.005, number of epochs is 150.

#### 0.3.2.5 Result Analysis

Here we could have a look at the user and movie nodes embedding vectors distribution. We could find that some user-user or user-movie or movie-movie have quite near distance which means they have direct connection or short distance connection in raw data.

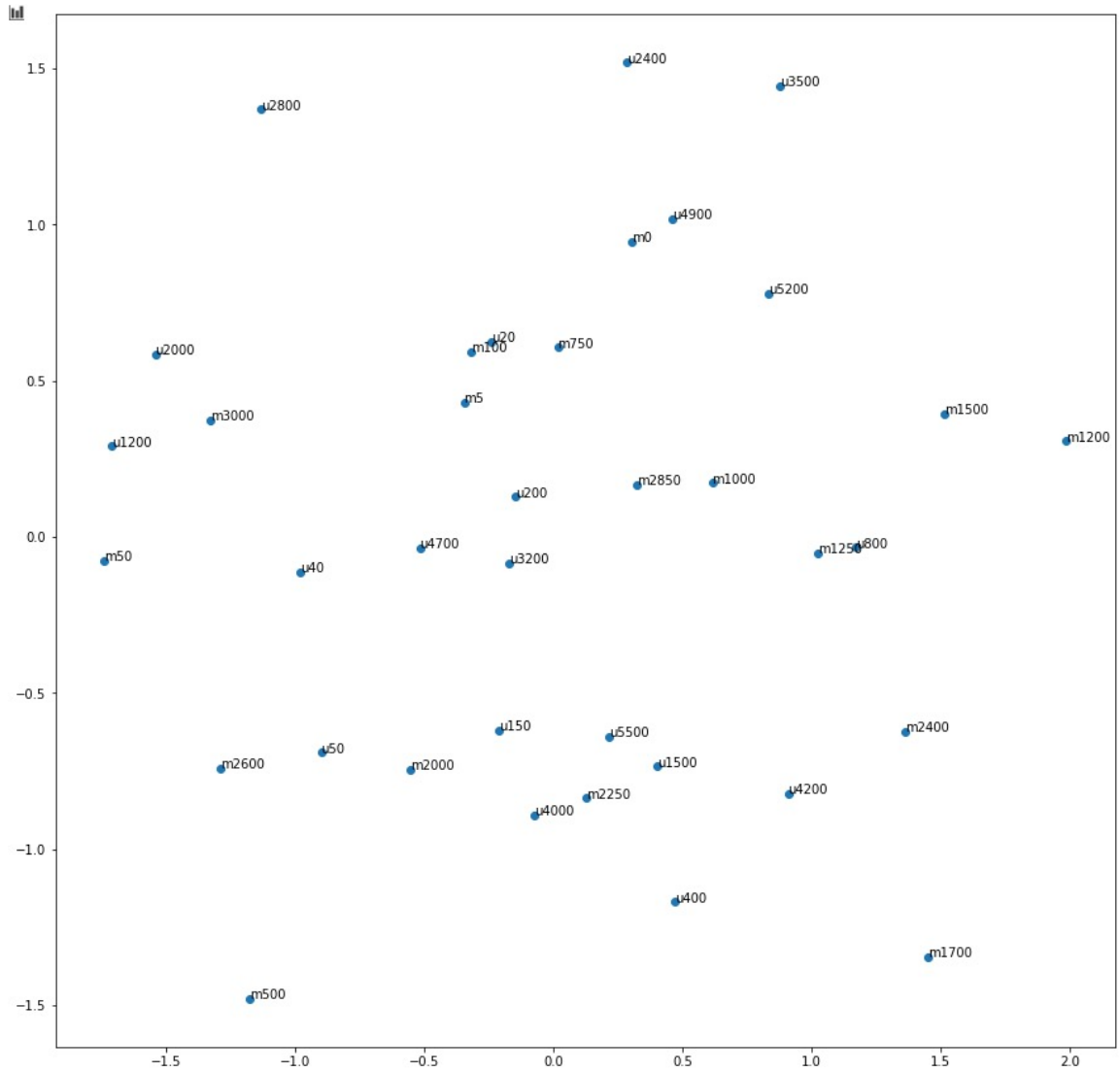


**Figure 14:** Embeddings by DeepWalk

And from the training loss and accuracy figure, we could find that the best link prediction accuracy in 20 percent of raw data achieve almost 86%. And the final prediction accuracy in test data achieve almost 84%.

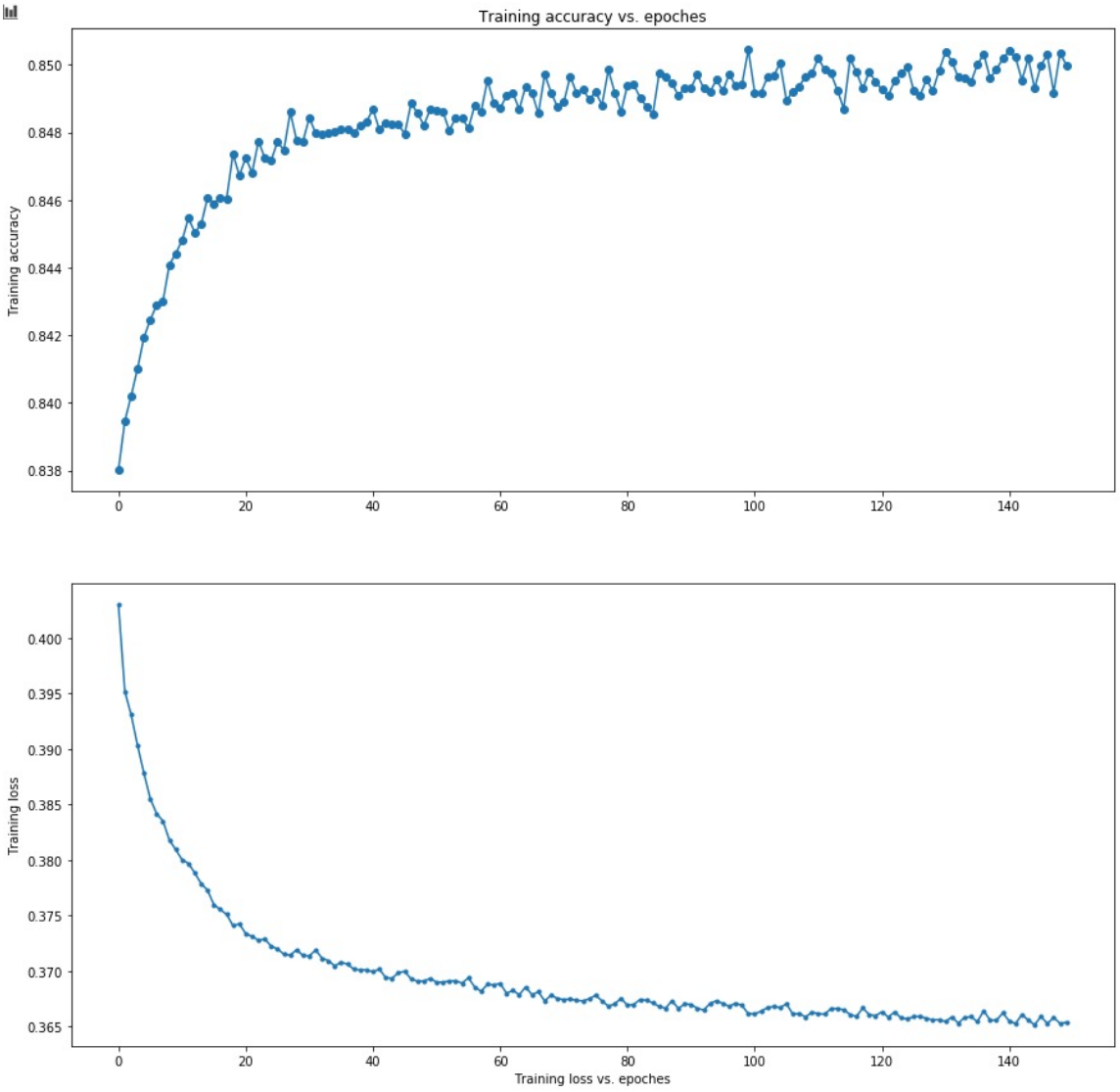


**Figure 15:** Loss and Accuracy

## 0.4 Conclusion

Assume a network with node features and a node classification task. With traditional ML techniques, we can learn feature to label mappings by assuming that each instance is independent of each other. Clearly, this is an invalid assumption for network structured data. Thanks to DeepWalk, we can reflect neighborhood relations and local structures in the network to node representations. Since Skip-gram tries to maximize similarities of neighboring nodes, nodes are sharing information between each other. This means that we enrich our existing feature set with node embeddings that were learned in a self-supervised manner. Having learned new representations for nodes, we can now use classification models for our task. And with the results achieved from the graph analysis, we can also add more graph attributes as additional features to our constructed MLP network classifier, which consequently give us quite acceptable prediction result.

# References

[1] Bryan Perozzi et al. (2014) DeepWalk: Online Learning of Social Representations.

[2] Göksu Tüysüzoğlu and Zerrin Işık. (2018) A Hybrid Movie Recommendation System Using Graph-Based Approach.

[3] Rıza Özçelik (2019) An Intuitive Explanation of DeepWalk