國立政治大學資訊科學系
Department of Computer Science
National Chengchi University

碩士論文
Master's Thesis

以使用者音樂聆聽記錄於音樂歌單推薦之研究

Learning User Music Listening Logs for Music Playlist Recommendation

研 究 生：楊淳堯
指導教授：蔡銘峰

中華民國 一百零七 年 二 月
February 2018

107
碩士論文

以使用者音樂聆聽記錄於音樂歌單推薦之研究

政治大學資訊科學系

楊淳堯

以使用者音樂聆聽記錄於音樂歌單推薦之研究

Learning User Music Listening Logs for Music Playlist Recommendation

研 究 生：楊淳堯　　Student：Chun-Yao Yang
指導教授：蔡銘峰　　Advisor：Ming-Feng Tsai

國立政治大學

資訊科學系

碩士論文

A Thesis
submitted to Department of Computer Science
National Chengchi University
in partial fulfillment of the Requirements
for the degree of
Master
in
Computer Science

中華民國 一百零七 年 二 月

February 2018

國立政治大學
National Chengchi University

致謝

很榮幸在政大就學期間能跟隨銘峰老師進行研究，在與老師學習的過程中總能感受到老師在學術研究上的熱忱，以及在對問題、細節上的謹慎處理態度，能有這個機會從旁學習並在獨立研究上接受指導，才有我今天這般的研究成果。此外，能在老師的推薦下與 KKBOX 研究團隊合作，對我而言是個十足的肯定。向志明學長一同研究學習的過程中，讓我對音樂產業有更深入的瞭解，並從中汲取了將機器學習應用在推薦領域裡的經驗。從業界、學術不同角色上去審視解決辦法，這樣的觀點切換經驗幫助我在思考上更彈性且實際。最後但也同樣重要且讓人開心的是，在與銘峰老師、釧茹老師、志明學長和致群的共同努力下，我們的研究順利地在 2016 年於 ACM RecSys 上和大家分享。感謝這一切得來不易的機會，讓這些從未想像的事情，一個個美好的發生在生活裡。

能有這樣的經歷，感謝父母在背後全力的支持我所做的決定，讓我在政大的靜謐環境下，更加心無旁騖地去學習。也謝謝實驗室優秀的夥伴們，嘉祐、至偉、致群、育文一同在研究上奮鬥和打氣，讓這條孤獨的路走的更加暖心。在此要格外感謝育文和郁惠，除了在研究上的相互砥礪外，在臺北的生活幸好有妳們的指引，使我在外地求學的過程更加順遂。而研究最後階段的英文撰寫上，謝謝 Jay 和 Peter 從旁給予提點，幫助我最後能達成這樣的嘗試與挑戰，除此之外，我也很享受與你們交流的過程，因爲背景上的不同，讓我學習從自身領域的觀點中抽出，反覆省思並嘗試用不同的方式去解釋自己的研究，也感謝你們尊重、耐心地聆聽，並從各自的專業領域中給予這篇研究寶貴的看法。

謝謝在人工智慧領域奮鬥的同仁們所打下的基石，讓我能在機器學習的研究中，透過直覺地類比思考去激發不同的學習想法，並從中找尋有意思的方式去研究問題，學習用自身的經歷去尋思自己是透過怎樣的模式和外在環境接觸，而外界的訊息刺激又是如何改變自己的行爲策略，我喜歡透過這樣的切身經歷去觸類旁通。最後期許自己能精進學習，運用系統、科學嚴謹的角度去解釋自然界中多變的樣貌，去瞭解這世界，並將所領略的心得應用在社會上並分享這份喜悅給身旁和喜好的人。

楊淳堯

國立政治大學資訊科學系

February 2018

中文摘要

音樂歌單是由一組多首不同元素、風格的音樂所組成的，它包含了編輯者的個人品味以及因應主題、目的性產生而成。我們可以透過樂曲的律動、節奏、歌曲的主題精神，進而編輯一個相應契合的系列歌曲。當今的音樂收聽市場主要是在網路串流平台上進行隨時、隨地的聆聽，主要的平台有 Spotify、Apple Music 以及 KKBOX。各家業者不單只是提供使用者歌曲的搜索、單曲的聆聽，更提供訂閱專業歌單編輯者的歌單訂閱服務，甚至是讓一般的使用者參與歌單自訂編輯的過程。然而如何在有限的時間內針對使用者的聆聽習慣去介紹平台上豐富的音樂資源是個很大的挑戰。上述的過程我們稱之爲推薦，而當前的音樂推薦研究大多是在對使用者進行相關歌曲的推薦，鮮少能進一步在更抽象層次上的歌單上進行推薦。這邊我們就此一推薦應用提供嵌入式向量表示法學習模型，在有著使用者、歌曲、歌單的異質性社交網路上，對使用者進行歌單的推薦。爲了能有效的學習出歌單推薦的模型，我們更將使用者、歌單和歌曲的異質性圖形重組成二分圖（bipartite graph），並在此圖形的邊上賦予不等的權重，此一權重是基於使用者隱式反饋獲得的。接著再透過隨機漫步（random walk），根據邊上的權值進行路徑的抽樣選取，最後再將路徑上經過的節點進行嵌入式向量表示法的學習。我們使用歐幾里德距離計算各節點表示法的鄰近關係，再將與使用者較爲相關的歌單推薦給使用者。實驗驗證的部分，我們蒐集 KKBOX 兩年份的資料進行模型訓練並進行推薦，並將推薦的結果與使用者所喜愛的歌單進行準確度（Precision）評估，結果證實所得到的推薦效果較一般熱門歌單的推薦來的好，且爲更具個人化的歌單推薦。

Abstract

Music playlist is crafted with a series of songs, in which the playlist creator has controlled over the vibe, tempo, theme, and all the ebbs and flows that come within the playlist. To provide a personalization service to users and discover suitable playlists among lots of data, we need an effective way to achieve this goal. In this paper, we modify a representation learning method for learning the representation of a playlist of songs, and then use the representation for recommending playlists to users. While there have been some well-known methods that can model the preference between users and songs, little has been done in the literature to recommend music playlists. In light of this, we apply DeepWalk, LINE and HPE to a user-song-playlist network. To better encode the network structure, we separate user, song, and playlist nodes into two different sets, which are grouped by the user and playlist set and song as the other one. In the bipartite graph, the user and playlist node are connected to their joint songs. By adopting random walks on the constructed graph, we can embed users and playlists via the common information between each other. Therefore, users can discover their favorite playlists through the learned representations. After the embedding process, we then use the learned representations to perform playlist recommendation task. Experiments conducted on a real-world dataset showed that these embedding methods have a better performance than the popularity baseline. In addition, the embedding method learns the informative representations and brings out the personal recommendation results.

國立政治大學
National Chengchi University

Contents

致謝 **3**

中文摘要 **4**

Abstract **5**

1 Introduction **1**

2 Related Work **5**
2.1 Word Embedding 5
2.2 Social Network Representation 6
2.3 Preserving Network Structure 6

3 Methodology **9**
3.1 Music Dataset and Creating the Bipartite Graph 9
3.2 DeepWalk 11
3.3 Large-Scale Information Network Embedding 12
3.4 Heterogeneous Preference Embedding 13

4 Experimental Results **17**
4.1 Experimental Settings 17
4.1.1 Dataset and Ground Truth 17
4.1.2 Similarity Calculation 18
4.1.3 Evaluation Metrics 19
4.2 Experimental Results 20
4.2.1 DeepWalk 20
4.2.2 LINE 21
4.2.3 HPE 22
4.3 Case Study 23

5 Conclusions **29**

Bibliography **31**

國立政治大學
National Chengchi University

List of Figures

3.1 Music social network 10
3.2 The bipartite graph for preference embedding 11
3.3 Extract a random walk from social network with the specified window size and walking steps 12
3.4 An example of first-order and second-order proximity 13
3.5 Neural network for learning social network representations 14
3.6 Neural network architecture 15

4.1 Playlist: A collection of songs 18
4.2 Comparison between DeepWalk, LINE and HPE on precision rate 22
4.3 UserA listening behavior 24
4.4 UserB listening behavior 26

國立政治大學
National Chengchi University

List of Tables

4.1 Dataset statistics 17
4.2 Genre statistics 17
4.3 DeepWalk parameters definition 20
4.4 Precision at 20 of DeepWalk with different parameter settings 20
4.5 Precision at 20 of LINE with different parameter settings 21
4.6 Precision at 20 of HPE with different parameters 23
4.7 MAP at 20 with different embedding methods 23
4.8 User listening statistic 24
4.9 UserA label reference 25
4.10 Playlist: Recommendation playlist 25
4.11 UserB label reference 26
4.12 User performance 27

國立政治大學
National Chengchi University

Chapter 1

Introduction

Music streaming services provide various ways for users to explore music they like, such as through creating and sharing music playlists. To deliver songs to users more efficiently, lots of music platforms deploy the recommender system to achieve this goal. For music recommendations, the explicit method for recording users' tastes is to let users input their preferences, such as through liked or disliked songs. Afterwards, we can apply well known recommendation techniques like content-based filtering or collaborative filtering [1, 7] approach to predict what a user may enjoy. In current literature, much has been studied about how to model the preference of users and songs for an effective recommender system [6, 13, 14]. However, little has been studied how to recommend a combined set of items (i.e., playlists) based on user preference logs on individual items (i.e., songs). Therefore, when we want to recommend playlists to the user without any pre-existing user preference toward playlist, the above conventional methods obviously become inadequate. In light of this, we proposed to use Heterogeneous Preference Embedding (HPE) approach [3] based on their past listening songs and construct a user-song-playlist bipartite graph to recommend music playlists.

The idea behind the embedding technique in HPE is to compress the surrounding information of an object into its vector representation. By adopting HPE method, we can encode the entities' relations into the vector space. The first embedding concept comes out from natural language processing, and the famous word embedding method, word2vec, was proposed by Mikolov [9, 10] in 2013. It is a method for language modeling and feature learning where words from vocabulary are mapped into vector space. The embedding of an object contains its neighborhood information. The more similar behavior of the object, the higher chance we get the similar vector representation. And if we get more related information about the object, it would be more likely that we derive meaningful representation of the object. To apply this concept to our music social network, we can encode user's context information such as his/her listening songs into the continuous vector

space. Likewise, we can embed playlist's context information into the same dimensional vector space.

To analyze user's behavior logs, we used the general data structure, graph, for information management and used it to model user's activities, including its sophisticated structures and their interactions. A few years earlier, Perozzi et al. (2014) proposed an unsupervised feature learning method called DeepWalk [12]. It was used to analyze social network in embedding method for the first time. DeepWalk learns the network representation of graph, which encodes vertices related neighbors and community membership into a continuous vector space. Since then, DeepWalk has become the state-of-the-art embedding method. Unfortunately, it suffers in efficiency when network vertices number grows to millions. Hence, there is another method, LINE:Large-scale Information Network Embedding [15], which enhances efficiency of the embedding learning and mimic the node's structure more effectively. LINE not only observes the relation between the connected neighbor but also models the second-order proximity of each vertex on the graph. Furthermore, it can assign different weighted values to the graph edges to express different preference strength. Inspired by these ideas, the HPE method was developed for music recommendation by learning the representations of users' preference for items. HPE takes the edge sampling to boost up the embedding learning process, and it takes the random walk method to comprehensively compute the local community structure information.

A playlist is a set of songs [4] which can be edited for personal collections or for some specific occassions. It is a high level of abstract concept which not only represents personal preference but also presents ones mood in the moment. Based on the HPE method, we proposed a preference graph over three types of node (i.e., users, songs, and playlists) for embedding. In order to enhance the embedding learning on music social network, we arranged our preference graph into bipartite graph which user and playlist nodes are on the same side and the rest of nodes, songs, are on the other side. Our evaluation showed such constructed way of graph can get the informative representations for each node, because the same type of information is encoded into the vector space.

In our experiments, a real-world music streaming dataset containing 50,000 users, 400,000 songs, and 130,000 playlists is employed to verify the effectiveness of the proposed method. By using the indirect preference information to recommend playlists to users, we got the following results, the network embedding based methods can beat the baseline of popularity; in addition, the proposed HPE method can bear comparison with the two state-of-the-art graph embedding techniques, DeepWalk and LINE.

The remainder of this thesis is organized as follows. In Chapter 2, we review some related works on the embedding techniques and how we apply it to our desired task. In Chapter 3, we investigate the playlist recommendation work based on the HPE model and

use the bipartite graph to better learn vertices representations. In Chapter 4, we present the real-world data sources we used and evaluate different embedding methods and finally in Chapter 5 we summarize our work.

國立政治大學
National Chengchi University

Chapter 2

Related Work

Over the past few decades, deep learning has received a tremendous progress in computer vision and speech recognition. Due to its capability in solving complex tasks while providing good results, researchers have applied this method on some difficult applications, such as natural language processing. To explore our music social network, we investigate social structure through network and graph theory and surveyed some embedding methods which are popular in the natural language processing field to derive network representations.

In this section, we describe the word representation method, skip-gram with negative sampling, the random sampling of an unweighted graph proposed in DeepWalk and the weighted edge sampling claimed in LINE.

2.1 Word Embedding

There has been recent interest in the work of word embeddings [8, 11, 16]. In 2007, Bullinaria and Levy [2] presented the representation of word meanings from word co-occurrence statistics, which extract semantic information and encode it to vector space. In 2013, Mikolov et al. improved the efficiency of training word embedding. They adopted the skip-gram model to learn high-quality distributed vector representations and used several methods such as hierarchical softmax and negative sampling to speed up the training process.

In Word2vec, the vector representation of word carries semantic meanings. It encodes word's semantic relationships to the word itself based on conditional probability of observing the actual context words given the input target word. To learn the weights of neurons between the neural network layers, Word2vec uses the backpropagation to update the skip-gram model parameters by going through context-target word pairs generated from a training dataset. Additionally, the negative sampling algorithm reduces the number of

samples to accelerate the training process yet retaining the accuracy of the representations. Therefore, we want to use the concept of vector embeddings to represent users and items. Due to the similarity of the representations, we can recommend the appropriate items to the user.

2.2 Social Network Representation

To extend the usage of vector representation of objects, Perozzi et al. developed DeepWalk to learn the embedding of vertices in a network. The basic idea of vertices embedding is to preserve the structure of neighborhood and community membership. Therefore, DeepWalk encodes the linkage information of each vertex. Vertices are mapped in a continuous vector space with a relatively lower dimensions. In order to learn latent features effectively, DeepWalk models a stream of random walks as the equivalent of sentences. It generalizes a natural language model to capture the relationship between vertices.

Using graph structure to represent social relations is an intuitive way to describe the network. It is a better way to learning the network based on our observation of the interactions between vertices. Hence, we can treat users, songs, and playlists as nodes, and personal behavior such as inferred from listening logs can be seen as an edge.

In the truncated random walk, we can pick a strating vertex arbitrarily. We then randomly select the next vertex from all the connected vertices. This next vertex becomes the new starting vetex and the process is repeated until we get enough vertices which are preset numbers in the beginning. Finally, we use these node sequences to extract the network structure and learn representations for vertices from random walks. Therefore, DeepWalk provides a bright future for the social representations that each node in the social network can be represented as a vector.

2.3 Preserving Network Structure

DeepWalk converts unweighted graphs into collections of linear sequences by sampling several random paths. In reality however, the edges in the graph could be a direction or a weighted value. Therefore, Tang et al. investigated the novel network embedding method, LINE. It is capable of learning various types of graphs including undirected, directed, and weighted graphs. LINE preserves both local and global network structures. Here, the local network structure focus on each pair of connected vertices and the global network structures mainly capture pairs of vertices which shares the same connected vertices even though there is no direct relation between them. To speed up the training rate, LINE uses edge sampling and samples edges according to their weights. In the mean time, it could

be possible to deal with a larger graph containing millions of vertices.

To serve millions of users, songs and playlists, LINE would be the practical way to overcome problem of scale. Besides, each user has his/her own preference toward songs, so it is reasonable to learn the weighted network embedding. Consequently, we took the weighted graph into consideration and apply the edge sampling algorithm to reduce the training time.

國立政治大學
National Chengchi University

Chapter 3

Methodology

As we mentioned before, it is rare to recommend a playlist to a user based on user's listening songs. Furthermore, our music social network is composed by different types of entity, so to recommend a playlist to a user in such heterogeneous graph is different from the traditional recommendation task. In order to capture the non-linear relationships in our user-song-playlist graph, we apply deep learning technique to our music social network. After we get the latent vector of vertices in our music social network, we can calculate their similarity on representations for recommendation task. We will describe the dataset we collected, the way that we build the music social network, and the methods we apply to playlist recommendation such as DeepWalk, LINE, and HPE are in the following sections.

3.1 Music Dataset and Creating the Bipartite Graph

We gathered our dataset from KKBOX which is a well-known music streaming company. KKBOX provides a platform for millions of user to enjoy and shares music with each other. The company employs music experts whose job is to share their insight about music and create playlists with their expertise. Also, there are lots of playlists made up by users, each playlist has its own flavor showing personal taste and different intent, such as for Christmas party, Birthday party or studying occasion (e.g. Figure 4.1), and these playlists are our recommendation items to users.

In our dataset, each user may have lots of listening logs of the songs but without having any preference recordings of the playlists. In order to recommend playlists to users in such situation, we used their listening records toward songs and related playlist information like what songs are wrapped in it to build the music social network, shown in Fig. 3.1.

This network is constructed with three types of node: the users, songs, and playlists. We treat them like two groups of node, one is grouping by the user and playlist nodes,

Figure 3.1: Music social network

and the other contains song nodes only. In addition, these nodes are connected by the undirected edge and each edge has its own numerical value which can be seen as the preference strength. For example, if the UserA had listened to the SongB C times, then the edge's weight between these two nodes would be assigned as the value C, which represented the UserA's preference to the SongB.

According to our collection dataset, there are two types of edge, one is user-song, and the other is playlist-song connection. Definitions are as follows:

1. The user-song connection: a user is connected to a song if the user listens to the song more than three times. This connection is mainly to record the user preference over songs.

2. The playlist-song connection: a playlist is connected to a song if the playlist contains the song, which records the relations between songs and playlists.

Based on the above definition, we then reconstruct our music social network to the bipartite graph, where both user and playlist type of nodes are connected with song. Therefore, we can utilize this characteristic and use embedding method to compress the neighborhood information into the continuous vector space. The graph is shown in Fig. 3.2.

Before we apply the following embedding methods to our music social network, we transform the network into bipartite graph first and then do the playlist recommendation on such heterogeneous graph.

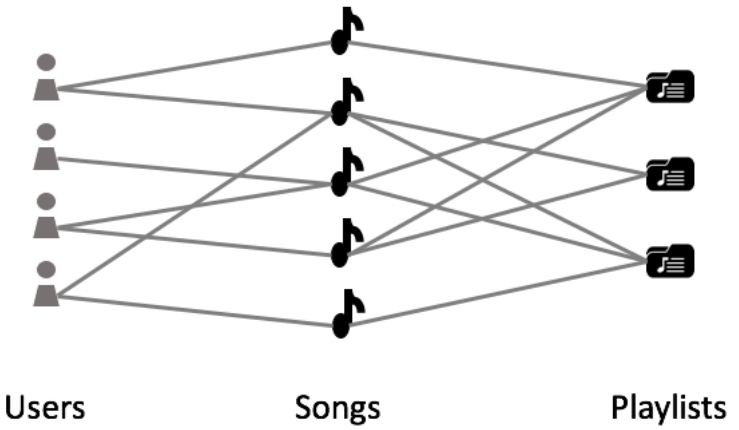

Figure 3.2: The bipartite graph for preference embedding

3.2 DeepWalk

The DeepWalk model focuses on the social network representation learning. DeepWalk adopts an embedding method through organizing the music social network onto latent representations. The model puts nodes on the graph into the same dimension, mapping the interaction between nodes into the same vector space. There are two different ways to model the relationship between nodes: one is continuous bag-of-word model (CBOW), the other is skip-gram model (SG).

CBOW defines the optimization problem as:

$$minimize \quad p(v_O|v_{I,1}, \cdots, v_{I,C}) \tag{3.1}$$

On the other hand, SG defines the optimization problem as:

$$minimize \quad p(v_{O,1}, \cdots, v_{O,C}|v_I) \tag{3.2}$$

where C is the number of nodes in the context, $v_{I,1}, \cdots, v_{I,C}$ are the input vectors in the neighbors, and $v_{O,1}, \cdots, v_{O,C}$ are the output vectors in the neighbors.

Both models are used in word embedding. The difference between them is CBOW tries to predict the context words by center word, while SG tries to predict the center word by context words. DeepWalk generalizes the skip-gram method in graphs by sampling streams of short random walks from each vertex, which can be seen as sentences in language modeling as shown in Fig. 3.3. By extracting the pairs of nodes from the random walk and using it as our training data, we can iteratively train our neural network and derive the network representation.

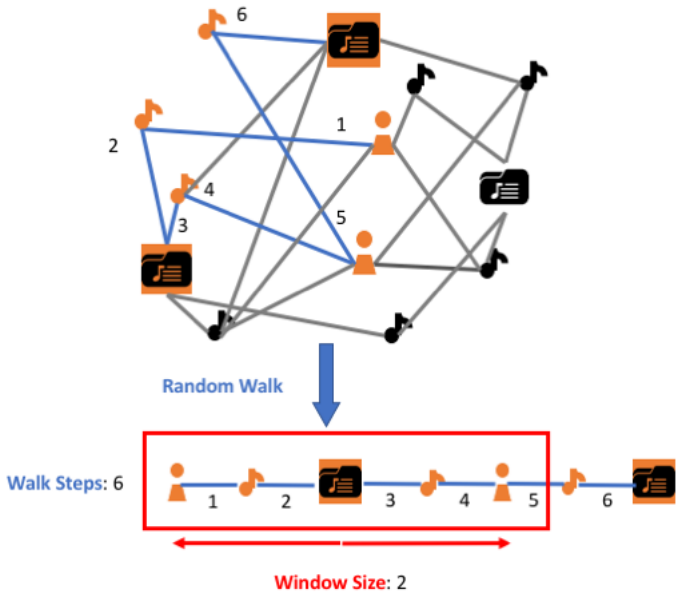

Figure 3.3: Extract a random walk from social network with the specified window size and walking steps

3.3 Large-Scale Information Network Embedding

In order to deal with larger network embedding, LINE may be used instead of DeepWalk. LINE provides a scalable way to process real world information networks. It also describes network structure in two different perspectives which are better for preserving the structure information.

Referring to network structure, the pairwise proximity between vertices can be considered as first-order proximity and second-order proximity. The following is a brief example about these two types of proximity orders. In Fig. 3.4, u1 and s5 has a high weight value connection as shown by the higher first-order proximity. On the other hand, p1 and u1 do not have a direct connection. Instead, they share the same common songs: s1, s3, and s4. Therefore, they have a high second-order proximity and should have closer representations.

To model the first-order proximity between the pair of nodes $\mathbf{v_i}$ and $\mathbf{v_j}$, the joint probability of $\mathbf{v_i}$ and $\mathbf{v_j}$ can be defined as follow:

$$p(v_i, v_j) = \frac{1}{1 + \exp(-\mathbf{u_i^T} \cdot \mathbf{u_j})} \tag{3.3}$$

where v_i, v_j are vertices, and $\mathbf{u_i}, \mathbf{u_j}$ are vector representations of vertices. The more similar of two vertices' latent representations, the higher value of their joint probability. It indicates that these two vertices have a closer relation because there is higher weight on the edge between the two vertices.

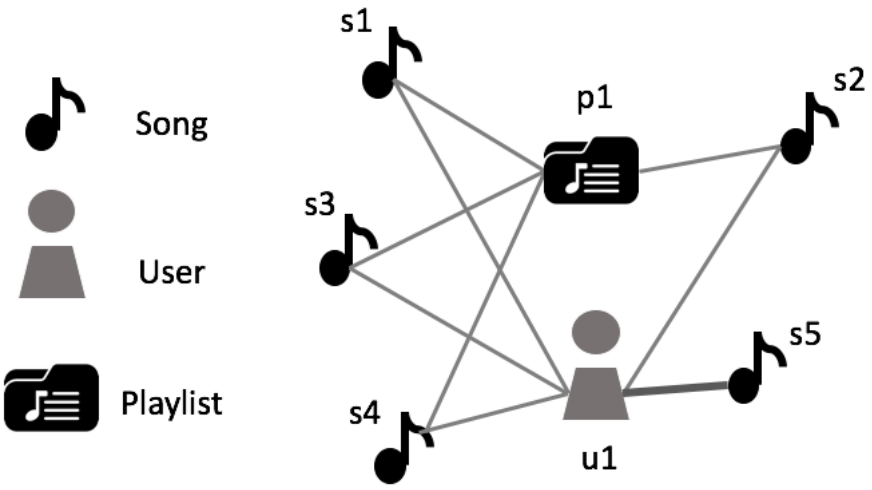

Figure 3.4: An example of first-order and second-order proximity

The second-order proximity model measures the similarity between pairwise vertices and their neighborhood network structure. The intuition behind this is similar to J.R.Firth's quote: "You shall know a word by the company it keeps" in text corpora [5]. To demonstrate the similarity of the node, it is derived as follows:

$$p(v_j|v_i) = \frac{\exp(\mathbf{u_j'^T} \cdot \mathbf{u_i})}{\sum_{k=1}^{|V|} \exp(\mathbf{u_k'^T} \cdot \mathbf{u_i})} \tag{3.4}$$

where $\mathbf{u_i}$ is the target vertex embedding, and $\mathbf{u_j'}$ is the specific context vertex embedding.

After training first-order proximity and second-order proximity model separately, both embeddings are concatenated as the node vector representations. By preserving the second-order proximity and also considering first-order proximity, information about the interaction between nodes and their shared neighbors can be learned simultaneously and encoded into the latent representation.

3.4 Heterogeneous Preference Embedding

Motivated by creating a learning representation through a neural network, HPE combines advantages from the previous methods. However, it was proposed to solve the query-based recommendation in the first place. Therefore, to apply HPE to our recommendation task, we transform our music social network into bipartite graph in the beginning. HPE takes several random walks through the graph to capture the social network structure. During the walk, it adds in local information such as the strength of the connection between

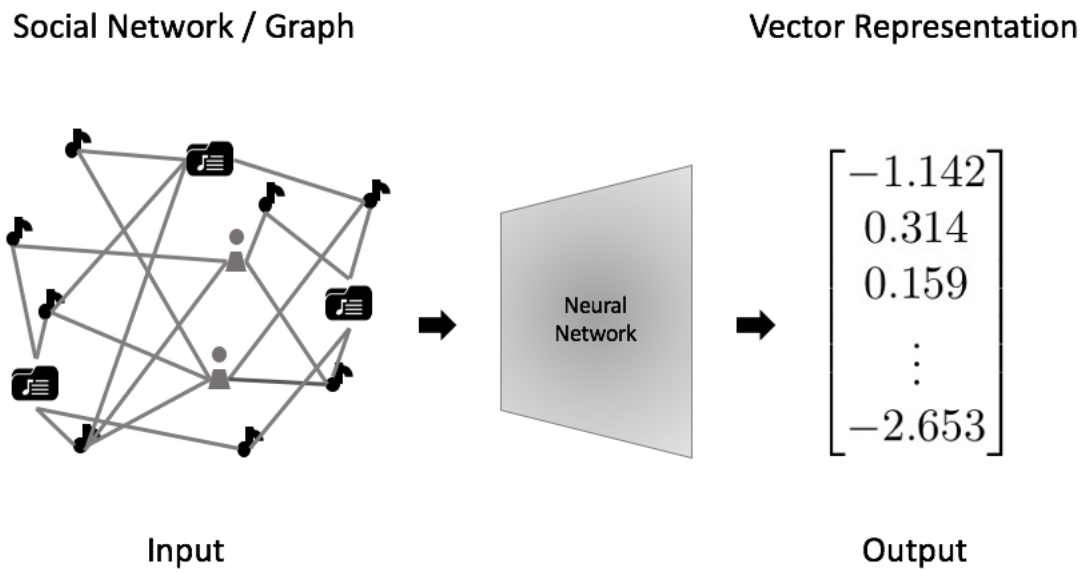

Figure 3.5: Neural network for learning social network representations

nodes from every step of the walk. In addition to preserve the graph structure information, we additionally train the model by giving pairs of related vertices in our bipartite graph.

Our goal is to project vertices' vector into the same dimensional space; therefore we would set the n-dimension vector representations for respective vertices, and also we would define what the dimension of hidden layer would be to get the better embedding. In order to derive the vertices' embedding, HPE generates the number of random walks from the graph, and uses neural network to learn representations.

Fig. 3.5 shows the concept of learning latent representations and the details on neural network is explained below in Fig. 3.6.

In HPE model, the neural network can be separated by input layer, output layer and one hidden layer in the middle, and each layer is fully connected. We will define the projection process between the input layer and hidden layer below. Each node in the input layer has its own weights toward units in the hidden layer. These weights are represented as respective vertex parameter vectors, and we can describe them as the following:

$$\mathbf{h_i} = \mathbf{W^T x_i} \tag{3.5}$$

where $\mathbf{x_i}$ is the i-th node vector, $\mathbf{h_i}$ is the corresponding latent representation, and $\mathbf{W}$ is the V $\times$ N matrix. Each row in the matrix W is the weight to the unit in the hidden layer of the vertex.

The second phase of the neural network between the hidden layer and the output layer is trying to model the multinomial distribution of the vertices. In this step, the latent representation is decoded back to the original dimensional space, with each hidden

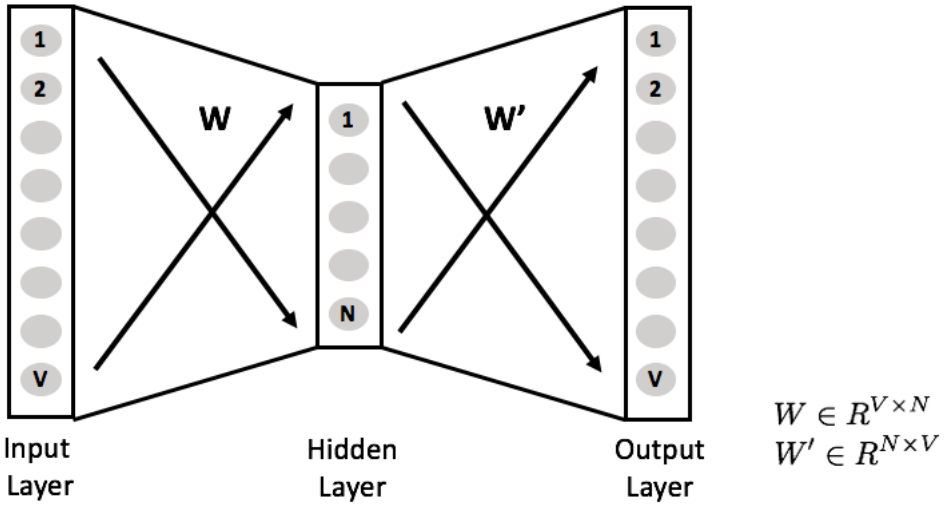

Figure 3.6: Neural network architecture

embedding vector having its own weights to the output one-hot vector. The parameter vector can be derived as following:

$$\mathbf{u_i} = \mathbf{W}'^{\mathbf{T}}\mathbf{h_i} \tag{3.6}$$

where $\mathbf{u_i}$ is the i-th node vector and $\mathbf{W}'$ is the $N \times V$ matrix. Then, for each vertex, we can use softmax to obtain the posterior distribution of vertices, which can be written as:

$$p(v_j|v_i) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u_{j'})} \tag{3.7}$$

where $\mathbf{u_j}$ is the score for each node in the graph. To maximize the probability of observing output vertex given the input vertex with regard to the weight, we can derive proper parameter vectors by Stochastic Gradient Descent (SGD) during the backpropagation step in the neural network training. During the training process, we will get two weight matrices, $\mathbf{W}$ and $\mathbf{W}'$. The former one is derived from the input to hidden layer and the latter is derived from the hidden layer to output layer. We can utilize these embeddings in our recommendation task.

In our proposed bipartite graph, the user and playlist nodes are both connected with songs. And their context neighbors can be seen as the semantic information relative to the target node. Therefore, if only the pairs with direct connections are sampled, the same type of information would be encoded into the dimensional space. The objective function of embedding process for a user or playlist can be expressed as follows:

$$O_v = \begin{cases} -\sum_{s \in Pref(u)} \log p(s|\mathbf{u}) & if \quad v \in U \\ -\sum_{s \in Plist(p)} \log p(s|\mathbf{p}) & if \quad v \in P. \end{cases} \tag{3.8}$$

where Pref(u) is the user's preference in songs and Plist(p) is the set of songs in the playlist.

After we trained the model and iteratively updated each node's vector representation throughout the sampled pairs of nodes in the preference graph, we shall derive the vector representation of each node. To achieve our goal in recommending playlists to users, we measure the preference relation between user and playlist nodes by the similarity metrics of their latent representations, such as cosine distance and euclidean distance. In general, we train our model to get the W matrix, with each row representing n-dimensional vector of each node in the graph. We then use these embeddings to calculate their relative neighbors.

Chapter 4

Experimental Results

In the last chapter, we introduced the network embedding methods, DeepWalk, LINE and HPE, which encodes network structure information into individual vector representations. In this chapter, we first take a look at our dataset and create the user-song-playlist graph. We then learn this social network representation through embedding methods. Finally, we compare the performance between these methods and use case studies.

4.1 Experimental Settings

4.1.1 Dataset and Ground Truth

In our experiments, the dataset consists of fifty thousand users, one million songs and close to three million playlists with a total of 478,509,126 user-to-song listening logs, and it was collected from the KKBOX music streaming service. The detail statistics are shown in Table 4.1 and Table 4.2.

	User	Song	Playlist
Number	50,282	1,251,071	2,971,424

Table 4.1: Dataset statistics

Genre	Number
Chinese	158,256
English	40,220
Taiwanese	5,429
Japanese	9,457
Soundtrack	39,789
Rock n Roll	55,958

Table 4.2: Genre statistics

Unfortunately, the dataset we have doesn't record personal favorite songs or playlist subscription logs, so we need to define the ground truth and clarify what the favorite meaning in the context.

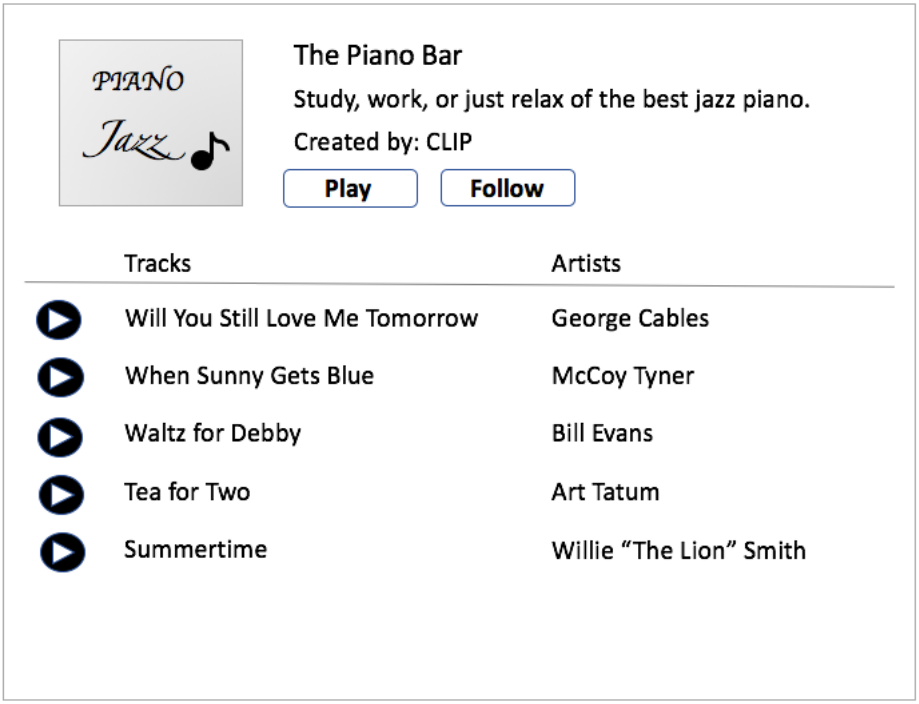

Figure 4.1: Playlist: A collection of songs

A song is assumed to be a user's favorite if he/she listens to it more than three times, and each time he/she had almost played the song from start to end. The ground truth of the chosen playlists is determined by the standard where the number of songs a user loved in a playlist are over 70% of the total songs.

For the purpose of retaining the appropriate listening logs, we only considered the listening logs where playtime is over 90% of the song time. Listening records not meeting this requirement were discarded. The filtering result remains 51,122,547 listening logs (11% of the initial listening records) among 1,008,461 songs and 50,271 users.

For evaluation, we split the dataset for each user according to the following 50/50 rule: keeping full listening logs for 50% as training data and the rest of listening logs as testing data. Besides, to oppose to the embedding methods, we select the popular playlists as our baseline. And the popular playlists are generated on the basis of users' preference.

4.1.2 Similarity Calculation

Each node in the user-song-playlist graph has its own vector representation. To achieve our goal in recommending user playlists, we retrieved the input vector from the trained neural network. If two different items have very similar local structures, then we can derive similar parameters from the network. For example, if the user has specific genre or artist preferences, and then the composed playlists would have the similar vectors as the user. The way we evaluate these similar items is to apply Euclidean distance to these

vectors:

$$d(x, y) = \sqrt{\sum_{d \in dim} x_d^2 - y_d^2} \tag{4.1}$$

where x, y represent different items separately, and d is the dimension of the vector.

The closer distance between the user and the playlist, the higher priority of the playlist would be recommended.

4.1.3 Evaluation Metrics

To access the effectiveness of our recommender system, we employed several metrics to evaluate the performance of algorithms in recommendation, precision (hit rate), mean average precision at n ($MAP@n$) and recall.

We denote the precision at n as $P(n)$ which is the fraction of the recommendation playlists and the given positive integer number n. It can be computed as the following formula:

$$P(n) = \frac{|R(n) \cap Pref(u)|}{n} \tag{4.2}$$

where $R(n)$ is the top n recommendation playlists and $Pref(u)$ is the user's preference in playlists. Furthermore, the average precision of the user ($AP(u)$) is denoted as:

$$AP(u) = \frac{1}{n} \sum_{k=1}^{n} P(k) \tag{4.3}$$

and the mean of the average precision score for the top n result (MAP@n) can be computed by:

$$MAP@n = \frac{1}{|U|} \sum_{u \in U} AP(u) \tag{4.4}$$

where $|U|$ is the total number of users. The higher of the MAP@n score, the better recommendation result.

The other performance metric is recall. Recall is the fraction of preference playlists that have been recommended over the total amount of preference playlists. The formula is defined as:

$$Recall = \frac{|R(n) \cap Pref(u)|}{|Pref(u)|} \tag{4.5}$$

High recall means that most of playlists user loved are recommended.

4.2 Experimental Results

In this section, we compare the recommendation results between DeepWalk, LINE, and HPE methods. Also, we separately discuss the consequence of their parameter settings.

4.2.1 DeepWalk

We created the bipartite graph to describe the relationships between the nodes in this user-song-playlist network. Then we input the edge list like, user-song, playlist-song, as our training data. In addition, we only consider one hidden layer in our model. Before training this model, we have to initiate several parameters to start the training process. The following are the parameters we used in our experiments.

Parameters	Definition
window size (wsize)	The neighbor's number around the center node. See Fig. 3.3
walking wteps (wstep)	Number of steps walking from the starting node. See Fig. 3.3
walking times (wtime)	Number of random walks on each node.
dimension (dim)	The neuron's number in the hidden layer.

Table 4.3: DeepWalk parameters definition

In oreder to evaluate how changes to the parameterization of this method affects its performance, we had fixed the window size and the walking steps to the values, wsize=5, wstep=6, which is choosen heuristically. We then varied the number of neurons in the embedding layer, and the number of walking times started per node, to determine their impacts on the recommendation task.

Walk Times(k)	Dim=64	Dim=128	Dim=256	Dim=512
1699(0.1)	0.131653	0.233496	0.309483	0.302355
935(0.3)	0.127857	0.223235	0.295636	0.290624
610(0.5)	0.117862	0.206414	0.27606	0.27765
407(0.7)	0.111141	0.18107	0.250692	0.264103
267(0.9)	0.10195	0.154225	0.214481	0.24156

Table 4.4: Precision at 20 of DeepWalk with different parameter settings

Table 4.4 shows the effects of such impacts to the model. And from this table, we can observe two interesting situations. The first is that as walking times increase, we get the higher hit rates in each dimension circumstance. As for selecting the number of the

walking times, we can derive it from the user listening logs. The walking times is equal to the top k percent of the user's least times of listening song, and the k is 1, 10, 30, 50, 70, 90 as shown in the Table 4.4.

The second is that the performance of the recommendation task rises up as the number of neurons in the hidden layer increases. The number of neurons could be seen as the number of latent features for the node. In our bipartite graph of the user-song-playlist network, we tried to encode the song information into the user and the playlist in the same dimensional space. In our experiments, we set up the highest dimension to 512, which approximates to the essential features that is mentioned in the Music Genome Project. The Music Genome Project[1] uses over 450 attributes, which are the essence of music at the most fundamental level, to describe each song. Therefore, setting the appropriate dimension would help us encode sufficient information and get better performance on our task.

4.2.2 LINE

To embed the local structure information effectively, we used the LINE model to preserve first-order proximity and second-order proximity between the vertices. This model not only preserves network structure information, but also retains edge information which is assigned as a different numerical value. For example, the edge weight can be explicitly used to quantify the user's interest.

In this model, we investigated the performance with respect to the parameter dimension.

Sample Times	Dim=64	Dim=128	Dim=256	Dim=512
1,000	0.119317	0.129924	0.136724	0.137504
3,000	0.136117	0.150536	0.164196	0.173081
5,000	0.131234	0.147328	0.166585	0.178067
10,000	0.119171	0.142782	0.167774	0.178445
30,000	0.111514	0.145627	0.165697	0.165595

Table 4.5: Precision at 20 of LINE with different parameter settings

Table 4.5 reports the performance of the LINE model. We can see performance drops when the sample time becomes too large, but the higher precision with the larger dimension is consistent with the results on DeepWalk method.

However, the precision rate in LINE cannot compete with DeepWalk in our dataset as you can see in the Figure 4.2.

[1] https://en.wikipedia.org/wiki/Music_Genome_Project

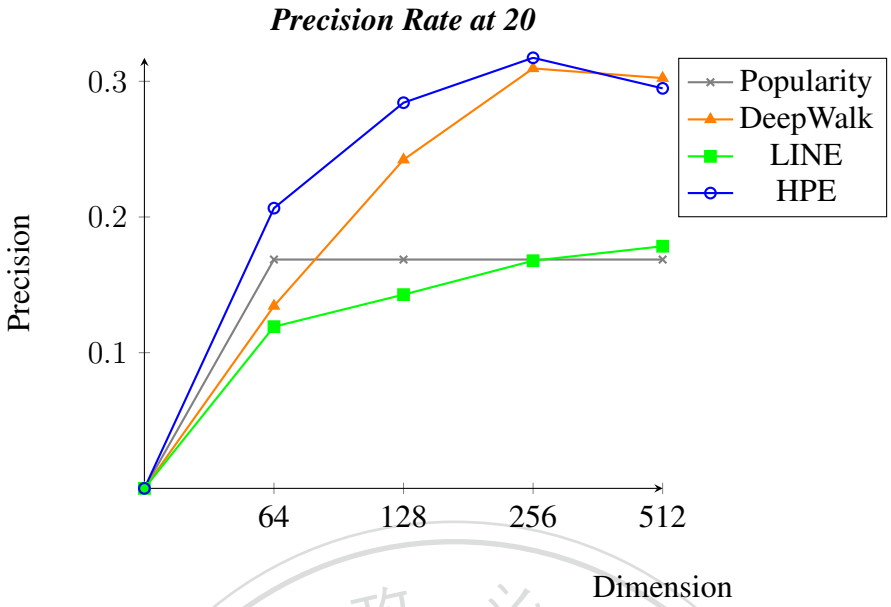

Figure 4.2: Comparison between DeepWalk, LINE and HPE on precision rate

The reason why the local structure learning doesn't work so well in our experiments is because we don't have the explicit user-playlist subscription information in our dataset. Since the user and playlist vertices don't have the direct connection, we couldn't take the advantage of the first-order proximity structure learning in LINE.

Therefore, we could only take the second-order proximity into consideration, and try to capture the nodes' similar local structures. There is still another issue that the user vertices edge's degree is at different scale with the playlist vertices'. Each user might have hundreds of user-song edges. On the other hand, there is only tens of the playlist-song edges for each playlist node. Compared to user node's embedding, the playlist node's embedding might be more precise to specific genres.

Despite DeepWalk's precision rate outcompeting LINE's, the latter takes less training process time, and also has the good performance for the recommendation tasks. To accelerate the training process, and improve the effectiveness of the stochastic gradient descent, LINE uses the edge sampling method to reduce the learning time.

4.2.3 HPE

In this method, we set up the number of walk steps to 6, and the related parameter settings is described in the Table 4.6. The table also shows the performance of the playlist recommendation task.

Besides, we compare HPE model with the other network embedding methods for our dataset. As shown in Fig. 4.2, the experiment results reflect that the HPE model embeds a better network representation in our bipartite graph. And also, we can see that there

Sample Times	Dim=64	Dim=128	Dim=256	Dim=512
30,000	0.206507	0.284222	0.317316	0.294788
10,000	0.177745	0.23117	0.265098	0.270714
5,000	0.152062	0.186453	0.214954	0.222994
3,000	0.139876	0.163291	0.182021	0.194649
1,000	0.13059	0.149434	0.165154	0.174612

Table 4.6: Precision at 20 of HPE with different parameters

are some common properties from these models. As the number of neurons in the hidden layer increases, more latent features are possible. At the same time, sufficient samples will make our vertex representations more robust.

The HPE model adopts weighted edge random walk and uses edge sampling to quicken the convergence. Taking the weighted edge into account, helps us encode user interests more accurately, especially if we don't have user-song preference information. Sampling from multiple random walks on each vertex also helps us get the more indirect information from the network. From these walks, the user nodes can walk through related playlist nodes by their common songs, using edge sampling to sample from these weighted edge helps us encode similar relations between the nodes. Table 4.7 shows the HPE model outperforming the DeepWalk and LINE models.

Methods	Dim=64	Dim=128	Dim=256	Dim=512
DeepWalk	0.28	0.36	0.43	0.45
LINE	0.33	0.33	0.34	0.35
HPE	0.37	0.44	0.48	0.48

Table 4.7: MAP at 20 with different embedding methods

Overall, the HPE method benefits from these two advantage processes, edge sampling and random walk. These really provide us a better and more efficient way to embed our bipartite graph into the more robust representations.

4.3 Case Study

In this section, we will discuss some case studies and compare different playlist recommendation results on each case. To help us intuitively check our recommendation outcomes, we will show user's listening statistic and others related information.

After training the embedding model, we should get the similar playlist representation

to the user who have like tastes with that playlist. We would also like to project the different users with alike listening behaviors into similar vector space. Here is the result that demonstrates these embedding method outcomes. We will briefly introduce UserA (u16011847) and UserB (u96013022) first, and then analyze their recommendation outcomes. Below Table. 4.8 are their listening statistic information.

Statistic	Number of Times
Listening Records	16,703
Songs	251
Avg. Listening Times	66
Max. Listening Times	185
Min. Listening Times	3

(a) UserA

Statistic	Number of Times
Listening Records	19,559
Songs	898
Avg. Listening Times	21
Max. Listening Times	166
Min. Listening Times	3

(b) UserB

Table 4.8: User listening statistic

In the Figure 4.3, we can see that UserA has strong preferences to Korean songs, and based on his listening behaviors, we can recommend a similar genre playlist to him, like Table 4.10 where the playlist is mainly composed of Korean songs.

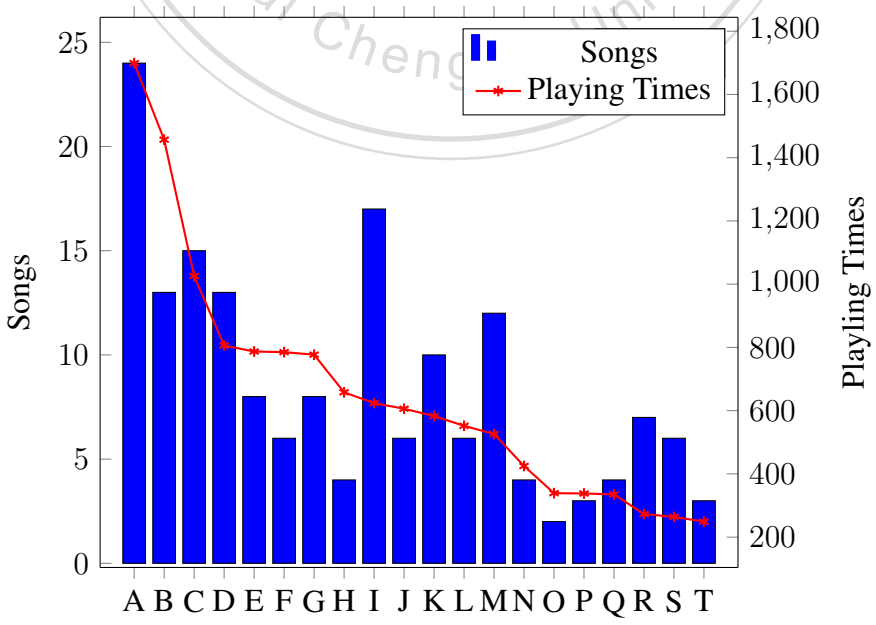

Figure 4.3: UserA listening behavior

Labels	Artist
A	SUPER JUNIOR
B	SUPER JUNIOR DONGHAE & EUNHYUK
C	EXO
D	BIGBANG
E	皮諾丘 電視原聲帶
F	沒關係 是愛情啊 電視原聲帶 Volume 1
G	EXID
H	G-DRAGON
I	Various Artists
J	繼承者們電視原聲帶
K	JUNG YONG HWA
L	BIGBANG TAEYANG
M	BoA
N	My Love From the Star
O	主君的太陽 電視原聲帶
P	Girls' Generation (少女時代)
Q	2NE1
R	KYUHYUN
S	miss A
T	AOA

Table 4.9: UserA label reference

Song	Artist	Genre
Miniskirt	AOA	Korean
NoNoNo	Apink	Korean
Pinocchio - ROY KIM	皮諾丘 電視原聲帶	Korean
Up & Down	EXID	Korean
BANG BANG BANG	BIGBANG	Korean
CALL ME BABY	EXO	Korean
Devil	SUPER JUNIOR	Korean
Growing pains	SUPER JUNIOR DONGHAE & EUNHYUK	Korean
12:30	BEAST	Korean
最佳的幸運	沒關係 是愛情啊 電視原聲帶 Volume 1	Korean

Table 4.10: Playlist: Recommendation playlist

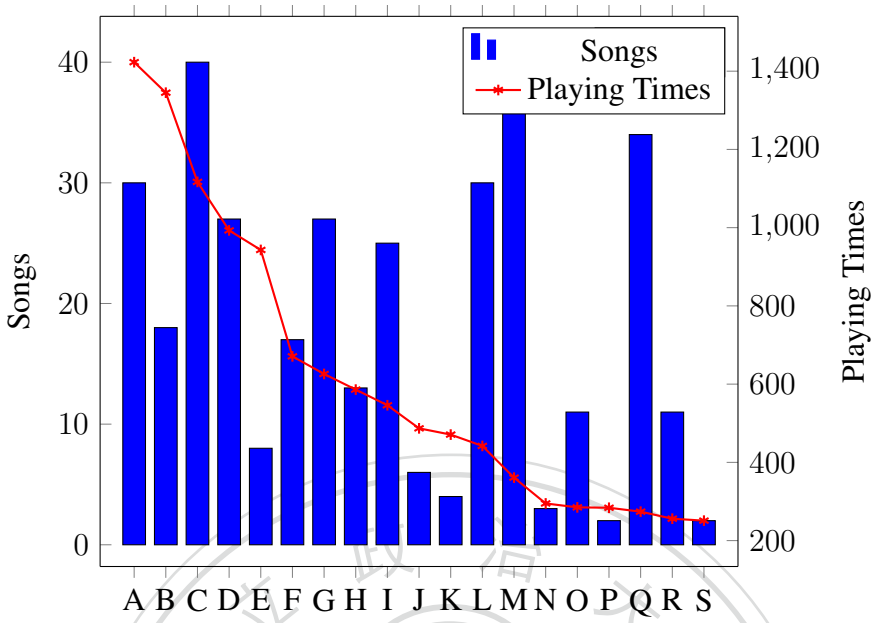

Figure 4.4: UserB listening behavior

Labels	Artist
A	G-DRAGON
B	SUPER JUNIOR DONGHAE & EUNHYUK
C	SUPER JUNIOR
D	Lily Allen
E	EXID
F	EXO
G	Various Artists
H	BIGBANG
I	Meghan Trainor
J	miss A
K	沒關係 是愛情啊 電視原聲帶 Volume 1
L	Calvin Harris
M	2NE1
N	BIGBANG TAEYANG
O	Maroon 5
P	My Love From the Star
Q	Colbie Caillat
R	Min Chae
S	主君的太陽 電視原聲帶

Table 4.11: UserB label reference

Evaluation	DeepWalk	LINE	HPE
Precision@20	0.55	0.3	0.85
Recall@20	0.1	0.05	0.15

(a) UserA

Evaluation	DeepWalk	LINE	HPE
Precision@20	0.05	0.35	0.7
Recall@20	0.003	0.023	0.046

(b) UserB

Table 4.12: User performance

On the other hand, we can also discover the similar preference user, like UserB, in the high dimensional space. As you can see in the Figure 4.4, UserB and UserA have alike preference on Korean songs and similar artists, so the playlist (Table 4.10) should be recommended to the UserB, too. And as we expected, the playlist (Table 4.10) is recommended to UserA and UserB, which means that the embedding method will actually encode the similar listening behavior into the similar representation.

Figure 4.4 shows that UserB has additional flavor on English song too, such as Lily Allen, Meghan Trainor, Calvin Harris, and Maroon 5, but the total English song playing times just account for the small part of his listening records. Therefore, it is more likely to recommend the Korean style playlist to the user. However, if we use the DeepWalk which does not consider the weighted edge in the training model, then it cannot capture the user's taste precisely. And here is the experiment result that shows DeepWalk's performance is worse than HPE as you can see in the Table 4.12 (b). In general, the embedding methods could compress the user's preference into the representation, and recommend the related playlists to user.

國立政治大學
National Chengchi University

Chapter 5

Conclusions

In this paper, we use embedding methods to encode the user-song-playlist networks, and apply the node's learning vector to our playlist recommendation task. In our network, it is constructed from multiple types of node, like user, song, and playlist, and we propose to use the bipartite graph to describe such heterogeneous graphs. To sufficiently utilize the social information in the network, we adopt the embedding method, a deep learning technique, to encode the network structure information into the vector space. There are some embedding methods which are proposed to learn the latent representations for classification tasks, and the basic concepts behind these methods are exploited by statistical models. The state-of-the-art methods like DeepWalk and LINE are mostly applied in the homogeneous graph. In order to apply DeepWalk, LINE, and HPE method to the heterogeneous graph, we reconstruct our heterogeneous network by decomposing our user, playlist and song nodes into separate sets, and then employ these methods on our bipartite graph. Experimental results show that HPE method outperforms the others. Considering weighted edges and using random walks as sampling method in our bipartite graph help us learn the informative representation. Besides, by walking through our bipartite graph, both direct and indirect relations would be explored on the walks, and that may yield improvements to our recommendation task. Overall, by using the embedding method, it provide a personalization service to a user and giving more exposures to the content provider (i.e.,music playlist creator).

In the future, we could investigate the different impacts on the multiple types of edge, and also study how to adjust the weight on each edge depending on every step of the walk. In addition, if we have the user-playlist subscription information, we can preserve the first-proximity and enhance the vector representation. Therefore, we can encode the more personalization information into the vectors, and derive the more diversity recommendation results.

國立政治大學
National Chengchi University

Bibliography

[1] G. Adomavicius and A. Tuzhilin. *Context-Aware Recommender Systems*, pages 217–253. Springer US, 2011.

[2] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: a computational study. *Behavior Research Methods*, 39 3:510–26, 2007.

[3] C.-M. Chen, M.-F. Tsai, Y.-C. Lin, and Y.-H. Yang. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 79–82. ACM, 2016.

[4] K. Choi, G. Fazekas, and M. B. Sandler. Understanding music playlists. *CoRR*, abs/1511.07004, 2015.

[5] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, Jan 2004.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008*, pages 263–272, 2008.

[8] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150. Association for Computational Linguistics, 2011.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[11] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, 2013.

[12] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710. ACM, 2014.

[13] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57:1–57:22, May 2012.

[14] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 713–719. ACM, 2005.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, Switzerland, 2015. ACM.

[16] W. Y. Zou, R. Socher, D. M. Cer, and C. D. Manning. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, 2013.