Heterogeneous Information Network and Deep Learning based Music Recommender System

Xinyu Gao
xinyugao@umich.edu

Kailin Tang
tangkl@umich.edu

Ye Wang
umwangye@umich.edu

Siyuan Xie
xiesiyu@umich.edu

ABSTRACT

In recent years, music recommender systems have gained extensive attention from both academia and industry. However, the existing music recommender systems still have problems in hybridizing useful heterogeneous types of data. To tackle this problem, in this work, we propose a novel recommendation framework based on Heterogeneous Information Network (HIN). We represent users, songs, content features and their rich relationships as a structured HIN. Then we use a meta-path based approach to characterize the semantic relationships between users and songs. Each meta-path within the HIN is used to formulate one category of preference over user-song pairs. In order to accurately predict the preferences between users and songs, we apply machine learning techniques to different meta-paths which consider to be high-level interactions between meta-path features. Besides, to further explore the content feature of songs, we perform song acoustic feature analysis and clustering which provide more relationships in HIN. Promising experimental results demonstrate that our purposed music recommender framework outperforms other traditional recommendation methods.

KEYWORDS

Recommender Systems; Heterogeneous Information Network; Acoustic Feature Analysis; Deep Learning

1 INTRODUCTION

Music recommender systems have experienced explosive growth in recent years due to the rapid development of the music industry and rising accessibility provided by various music streaming services. Although many distinct music recommender systems have been applied in daily use, oftentimes they still fail to hybridize heterogeneous information well. Therefore, optimizing the music recommender system becomes increasingly significant in both research and industrial field.

One of the most widely used models in recommender systems is Heterogeneous Information Network (HIN)[1], which is proposed to be a general representation of heterogeneous relationship in real-life graph. At the beginning, HIN was used to deal with search and similarity measurement problems, where the query and search entity are assumed to be the same. Later, it was extended to heterogeneous entity recommendation problems, e.g. $User \rightarrow Item$. To deal with the rich semantics in HIN, meta-paths[2] are developed to represent different relationships between entities in HIN. For example, the traditional collaborative filtering can be implemented via a single meta-path $User \rightarrow Song \rightarrow User \rightarrow Song$, and traditional content-based recommendation can be implemented via a single meta-path $User \rightarrow Song \rightarrow Content \rightarrow Song$. As we can see, traditional recommendation methods with heterogeneous information can be merged under the same framework, which is more informative thus motivates us to assemble different meta-paths for better recommendation.

In [3] the authors implemented SVM on different meta-path features to optimize malware classification in Android. The authors of [4] did matrix factorization on meta-path commuting matrix and applied factorization machine on multiple concatenated latent factors considering the interaction between features. Machine learning methods should be considered when hybridizing multiple meta-path features. It's intuitive to think of linear regression while it does not take into account correlations between features and thus loses important information. We decide to choose multi-layer perceptron regression as it considers high-level interactions between features and fully uses the given information.

One of the challenges in music recommender systems is acoustic feature analysis. Acoustic feature is a digital record of the all the existed feature that a song has. Rather than other features such as tags and artist, acoustic feature is a more original and comprehensive record of what exactly the song sounds like. Oftentimes people prefer some certain songs not only because they prefer its theme, artist or the category, but also because of some deep features embedded in the rhythm which does not belong to any specific tags. Therefore, acoustic feature has huge potential in extracting different kinds of features and may more profoundly reveal a person's preference. Therefore, how to utilize acoustic features for better recommendation is crucial in music recommender systems.

2 PROBLEM DEFINITION

Since existing music recommender systems frequently produce unsatisfactory recommendations, we decide to utilize heterogeneous information to optimize music recommendation. The problems are as follows: 1. How to extract most descriptive acoustic features useful for music recommendation. 2. Given the user listening history and content information of each song, how to effectively utilize the heterogeneous information under same framework. 3. How to optimize top-N-song recommendation for each user based on the high-level interaction of heterogeneous information we have.

3 DATA

The dataset we used is the Million Song Dataset (MSD), a collection of audio features and meta-data for a million contemporary popular music tracks[5]. MSD also contains a cluster of complementary datasets, including the lyrics provided by *musiXmatch*, user-defined tags provided by *Last.fm* and user taste profile history provided by *EchoNest*.

3.1 User Listening History

User listening history comes from the taste profile subset, which contains more than 48 million triplets (user, song, count) involving more than 1,000,000 distinct users and 384,546 unique songs in MSD. As the user-item matrix is very sparse, the non-zero entries are only 0.01% of all entries. The statistics about the listening history data is shown in Table1[6].

	min	max	median	mean
songs per user	10	4400	27	47.46
users per song	1	110479	13	125.80

Table 1: Statistics of user listening data.

As shown above, many songs are listened by just a few users and many users only listen to a few songs. What's more, MSD is biased since some songs have more than 100k users while other songs may only have 1 user. Therefore, in order to get rid of noise to reduce the probability of overfitting the data, we need to preprocess the biased and sparse dataset before we do further analysis.

3.2 Music Meta-Data

For given songs, MSD provides audio features and meta-data, so features with significant contribution to the similarity calculation between songs can be used for content-based recommendation algorithms.

One of the most significant features is the acoustic feature, which specifies distinctive attributes for each song. The main acoustic features defined by the *Echo Nest Analyze API*[7] are loudness, pitch and timbre, among which we choose timbre as the most descriptive one.

Timbre is the quality of a musical note or sound that distinguishes different types of musical instruments, or voices, independent of pitch and loudness. Typically the overall timbral characteristic of a song is represented by local MFCC vectors. As shown in many literatures, Mel Frequency Cepstral Coefficients (MFCCs), which are a compact representation of the spectral envelope of a short audio frame, are often used in acoustic feature analysis. However, MSD represents timbre in a different way. The actual timbre is described as a linear combination of 12 basis functions, which characterizes distinct features of the timbre, weighted by the coefficient values: timbre = $c_1 \times b_1 + c_2 \times b_2 + \ldots + c_{12} \times b_{12}$, where c_1 to c_{12} represent the 12 coefficients and b_1 to b_{12} represent the 12 basis functions.

The timbre feature in Echo Nest Analyzer is a vector that includes 12 unbounded values roughly centered around 0. And the MSD stores each the timbre information in a $12 \times n$ matrix, where each column stores 12 coefficients of 12 basis functions, and n is the length of the song measured in seconds.

The interpretation of timbre is shown in Figure.1[7].

3.3 Data Preprocessing

As discussed above, the original MSD dataset is sparse and biased. Therefore, we need to do data preprocessing to reduce the chance of overfitting. Hence, We do random sampling over the huge dataset by following rules: first of all, we do random sampling from 384,546 unique songs in the user listening history and filter out songs whose number of listeners is less than 150. Then we do random sampling for users and filter out the users who listen less than 100 songs. After data preprocessing, we have 14,516 users and 10,748 songs with 1,894,113 user-song pairs. The preprocessed user listening data is shown in table 2.

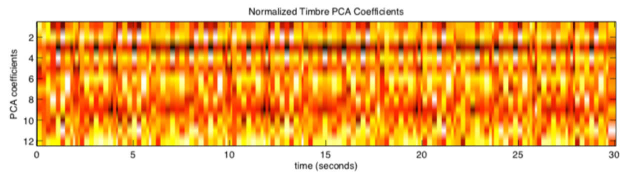

Figure 1: Normalized timbre coefficients along time[7]

	min	max	mean
songs per user	91	599	130
users per song	3	3308	176

Table 2: Statistics of preprocessed user listening data.

Later, we split the preprocessed user listening data into two parts: we choose 80% user listening data as training set and the remaining 20% as test set. As for the test set, we choose 50% as visible validation set and remaining 50% as hidden test set. The reason why we split the test set into visible set and hidden set is that if we hide all test set of 20% user listening data, then we cannot model the collaborative preference, thus our problem degenerates into a cold start problem which will make our job more difficult.

After we finish the data preprocessing step, we can build a user-song matrix to model the preference between the user-song pair. As mentioned above, this preference is represented by the listing count for each user-song pair. However, since different user has different behavior, which means the average listening count for different users is very likely to be unbalanced. In order to deal with this problem and model the user-song preference from listening history properly, it is necessary to normalize the count. We calculate the mean listening count for each user and then for each song that the user listened, we subtract the mean and then let the result go through a sigmoid function, which will map the result into [0, 1]. A score for a user-item pair closer to 1 means that the user preference to that song is higher.

$$\overline{r}_{u,i} = sigmoid(r_{u,i} - \mu_u)$$

where $\overline{r}_{u,i}$ denotes the balanced rating for user u and song i, $r_{u,i}$ denotes unbalanced rating for user u and song i, and $\mu_u$denote the average ratings for user u.The unobserved pair will not be balanced.

4 PROPOSED METHOD

4.1 Acoustic Feature Extraction

In most recent studies, acoustic features are not considered into the recommender systems. Instead, most of them attempted to make fully use of existed and obvious features(tag features) such as Album, Genre, Artist, etc. In our heterogeneous information network, we adopted Meta-path algorithm which takes those tag features into account and it made reasonable recommendation. However, acoustic feature of a song is an important part when we consider content-based recommender system. The underlying difference between tag features and acoustic features is that the latter is more original features of a song. It can be obtained by simply analyzing song tracks, which is the only raw data when a song comes out. Tag features, however, involve human labeling or most concrete and intact information of the song. Therefore, it is important to make recommender systems purely rely on raw data, i.e., the acoustic features. The overall idea here in to use deep learning approaches to extract the feature that represents listener's perspective of song's pure attributes. The reason is that oftentimes people prefer some certain songs not only because they prefer its theme, artist or the category, but also some deep features embedded in the rhythm which does not belong to any specific tags.

As analyzed before, the most informative one is $timbre$ as it is obtained using 12 different ways of feature extraction, which contains almost all features that represents a song in a different perspective. In order to extract meaningful feature from given acoustic features, we propose two approaches based on Neural Networks.

4.1.1 SOM Based Feature Clustering. Self Organizing Map (SOM) by Teuvo Kohonen[8][9] is a neural network trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discrete representation of the input space of the training samples, called a map, which represents clustering concept by grouping similar data together. Therefore SOM is designed to reduce data dimensions and display similarities among data. The basic idea behind SOM is to setup neurons which compete for the signal. SOM network can be visualized as Figure. 2.

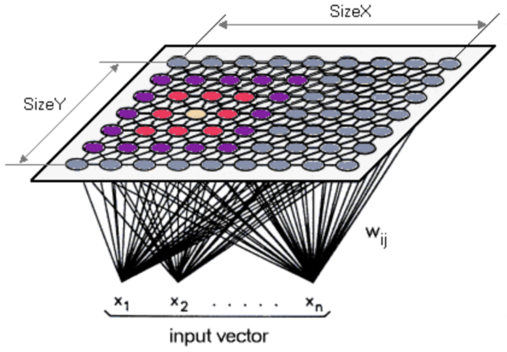

Figure 2: Visualization of SOM network[8][9]

SOM defines a mapping from the input data space spanned by $x_1 ... x_n$ onto a one- or two-dimensional array of nodes. Each node of the map is defined by a vector w_{ij} whose elements are adjusted during the training. The basic training algorithm is quite simple:

1. Select an object from the training set;

2. Find the node which is closest to the selected data (i.e. the distance between w_{ij} and the training data is a minimum);

3. Adjust the weight vectors of the closest node and the nodes around it in a way that the w_{ij} move towards the training data;

4. Repeat from step 1 for a fixed number of repetitions;

In acoustic feature clustering, as discussed in section 3.2, timbre matrix is $12 \times n$, where n is the length of the song measured in seconds. As songs in our dataset do not have the same length, the dimensions of the acoustic matrix are not the same. Since different dimensions will cause trouble in the input space of SOM, we need to do acoustic feature preprocessing. We use Principal Component Analysis(PCA) over acoustic matrix to find the first principle component of acoustic features in each song and then use SOM to do automatic clustering. We set the cluster number to be 400 and 3,600 respectively.

4.1.2 Deep Learning Based Feature Extraction. The core part here is to design a program to extract the useful and contributive part of the information in the acoustic feature that shows the person's preference towards songs. One of the main obstacles is the huge amount of data which contains thousands of songs and listeners. A possible approach here is to establish personalized criterion that output a value that represents the person's preference toward a specific song. The principle behind this is relatively simple but requires huge amount of resources as each listener needs a corresponding personalized forward propagation mechanism. Another possible method is to classify the acoustic features into categories using unsupervised learning and then use the classification result as the input of the meta-path algorithm. The potential problem of this idea is that we do classify the acoustic features, but in a way that makes no sense. What we want originally is to find and amplify the contributive part in features that represent to some extent a person's respective towards a specific song. Therefore, besides unsupervised learning, we explore alternative acoustic feature extraction methods in this section.

We also propose two alternative approaches that combine acoustic feature and listener's information. The basic underlying assumption is that if two songs are preferred by the same person, which means there should be some part embedded in the acoustic feature that contributed to this similarity. Intuitively, we want this two acoustic features to be closer spatially in another space. And then those clusters will be clearly shown and represent people's preference in a new space. The two approaches are presented as below.

The first method adopts Siamese network[10] which is presented in Figure.3. Usually the model used here is deep learning based and contains multiple layers of neural network. $Deep - Nets$ is a combination of several cascade convolutional layer, ReLu layer and Batch Normalization layer. It can be seen as feature extractor. Each time two acoustic features are fed into the Deep-Nets simultaneously, and the output for each Nets are calculated by Contrastive loss. Generally, the Siamese network aims to make the distance between two output features closer if the label in the parameter of loss function is 1. Otherwise do nothing. The two Deep-Nets share the same weights and update the weight at the same time. Algorithm.1 shows the concrete procedures in providing classification number using Siamese model.

The second method is relatively simple to understand. We will not choose to spatially relatively minimize the distance between the acoustic feature that we believe should be close according to the assumption. Instead, we want to classify the acoustic feature using a single Deep-Nets. In this model, since there is a feature mapping

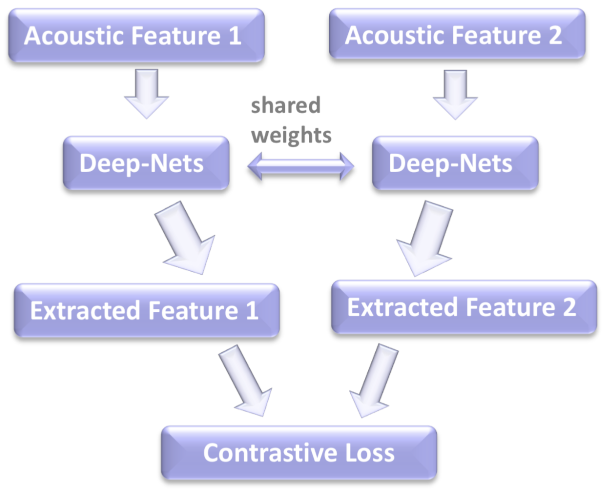

Figure 3: Siamese Network

```
Data: The listener-songId tuple and song's acoustic feature
Initialize Epoch, Hyper-parameters, Siamese Network ;
while Iter less than Epoch×Num do
    Gather acoustic features of two songs from one listener's
      play-list, or different listener's play-list;
    if The songs corresponding to acoustic features are both in
      one listener's play-list then
        Set the Contrastive Loss with label parameter 1;
        Train the Siamese network;
    else
        Set the Contrastive Loss with label parameter 0;
        Train the Siamese network;
    end
end
Feed the model with all acoustic features and gather the
  output features;
Run K-means on those features and obtain the classification
  number;
    Algorithm 1: Siamese Model based Categorization
```

from high dimension to low dimension in the classifier layer at the end of the Deep-Nets, we come up with the statement that it is highly possible that closely located feature in low dimension refers to a similar spatial location in higher dimension. To put it another way, if we set apart the output vectors, the features that fed into the classifier layer will be spatially far away among each other. So we may gather the features from the last layer before the classifier layer and use that feature as the output feature of the classification model.

Before training the classification model, we should first generate the labels for each song. The aim of this network is to mine the features that contribute to the listeners' preference. We use a boolean vector to record a song's listeners and categorize the songs by using K-means on the vectors. In this way, we could generate labels for classification network under the consideration of both

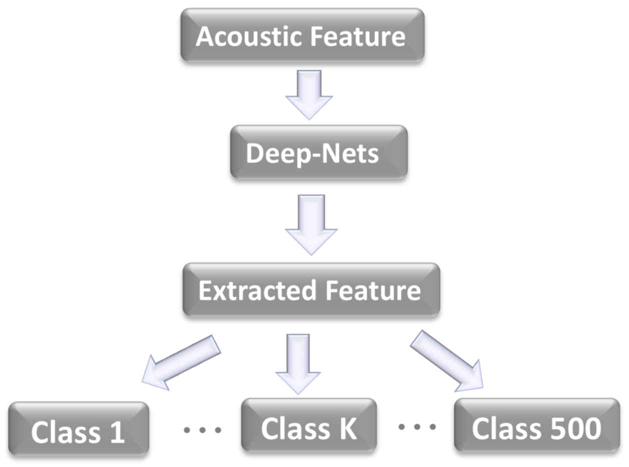

Figure 4: Classification Network

arbitrary K and listeners' preference. The classification result does work for using labels only, but the reason of doing it this way is to train the network in listeners' perspective, and prepare for the cold start problem when a new song track comes in. Algorithm.2 shows the concrete procedures in providing classification results using Classification model.

```
Data: The listener-songId tuple and song's acoustic feature
Generate listener vector for each song;
Apply K-means on listener vectors, and each song has label;
Initialize Epoch, Hyper-parameters, Classification Network;
while Iter less than Epoch×Num do
    Randomly gather acoustic features of a song;
    Train the classification network using song's label;
end
Feed the model with all acoustic features and gather the
  output classification number;
  Algorithm 2: Classification Model based Categorization
```

Both algorithms are implemented and the extraction part of the network are of the same design in both models. Experiment results are provided by adopting classification result in meta-path algorithms.

4.2 Heterogeneous Information Network Based Recommendation

Given rich information containing user listening history and content features of songs, it's necessary to adopt an efficient model to combine them so that different relationships between users and songs can be modeled under the same framework properly. Heterogeneous Information Network (HIN) is an ideal model for this use case. In this section we will introduce how we build our HIN using our data and how we make music recommendation using meta-paths within the HIN.

Definition 4.1 [1] **Heterogeneous Information Network(HIN)** is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the entity type mapping $\phi : \mathcal{V} \to \mathcal{A}$

and a relationship type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{V}$ denotes the entity set and $\mathcal{E}$ denotes the link set, $\mathcal{A}$ denotes the entity type set and $\mathcal{R}$ denotes the relationship type set, and the number of entity types $|\mathcal{A}| > 1$ or the number of relationship types $|\mathcal{R}| > 1$. The network schema for network $\mathcal{R}$, denoted as $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, is a graph with nodes as entity types from A and edges as relationship types from $\mathcal{R}$.

Heterogeneous Information Network provides a high-level abstraction of different relationships between entities. The HIN we constructed for music recommendation involves two entities: user, song and five relationship: user-song, song-artist, song-album, song-acoustic feature and song-genre shown as Figure.5.

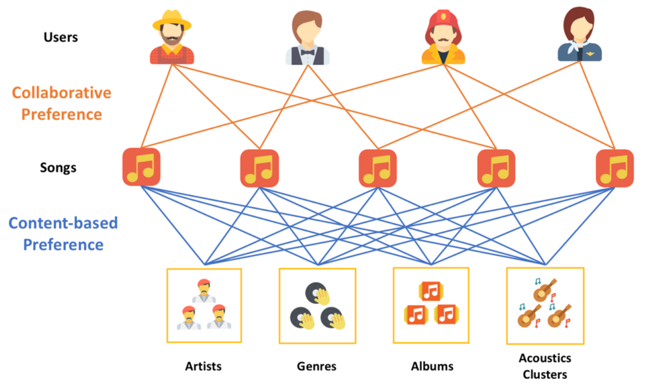

Figure 5: HIN for music recommendation

HIN enables us to model different relationships between entities. In order to formulate the semantics of high-order preferences between users and songs, we define several meta-paths within the HIN we constructed.

Definition 4.2 [2] **meta-path** $\mathcal{P}$ is a path defined on the graph of network schema $\mathcal{T}_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$, and is denoted in the form in the form of $\mathcal{A}_1 \xrightarrow{\mathcal{R}_1} \mathcal{A}_2 \xrightarrow{\mathcal{R}_2} \ldots \xrightarrow{\mathcal{R}_L} \mathcal{A}_{L+1}$, which defines a composite relation $\mathcal{R} = \mathcal{R}_1 \cdot \mathcal{R}_2 \cdot \ldots \cdot \mathcal{R}_L$ between types $\mathcal{A}_1$ and $\mathcal{A}_{L+1}$, where $\cdot$ denotes relationship composition operator, and L is the length of $\mathcal{P}$

A typical meta-path in collaborative filtering music recommendation is $User \xrightarrow{listens} Song \xrightarrow{listens^{-1}} User \xrightarrow{listens} Song$ and a typical meta-path in content based music recommendation is $User \xrightarrow{listens} Song \xrightarrow{contains} Content \xrightarrow{contains^{-1}} Song$. If a path only connects the users and songs, such a path will model the collaborative preference between them. If a path goes through any song content features, such a path will model the content-based preference between users and songs. Since we are modelling the user-song preference for recommendation, a valid meta-path should begin with users and end with songs and should have at least 3 hops between entities. We can also add multiple relationship in one meta-path.

Since there can be multiple meta-paths that model the same preference, we will use the commuting matrix which represents the same kind of preference as a matrix.

Definition 4.3 [2] Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and its network schema $\mathcal{T}_{\mathcal{G}}$, a **commuting matrix** $M_{\mathcal{P}}$ for a meta-path $\mathcal{P} = (\mathcal{A}_1 - \mathcal{A}_2 - \ldots - \mathcal{A}_{L+1})$ is defined as $M_{\mathcal{P}} = G_{\mathcal{A}_1\mathcal{A}_2} G_{\mathcal{A}_2\mathcal{A}_3} \ldots G_{\mathcal{A}_L\mathcal{A}_{L+1}}$, where $G_{\mathcal{A}_i\mathcal{A}_j}$ is the adjacency matrix between types $\mathcal{A}_i$ and $\mathcal{A}_j$. $M_{\mathcal{P}}(i, j)$ represents the number of path instances between entities $x_i \epsilon \mathcal{A}_1$ and $y_j \epsilon \mathcal{A}_{L+1}$ under the meta-path $\mathcal{P}$

For example, the adjacency matrix between $User$ and $Song$ is $G_{u,s}$, the adjacency matrix between $Song$ and $Content$ is $G_{s,c}$, then the commuting matrix on the meta-path $User \xrightarrow{listens} Song \xrightarrow{contains} Content \xrightarrow{contains^{-1}} Song$ will be $G_{u,s} G_{s,c} G_{s,c}^T$. More complex preferences can be computed using longer meta-paths, e.g. the meta-path $User \xrightarrow{listen} Song \xrightarrow{contains} Artist \xrightarrow{contains^{-1}} Song \xrightarrow{contains} Genre \xrightarrow{contains^{-1}} Song$ contains the information of both the relationship between Song and Artist and the relationship between Song and Genre.

After we get the commuting matrices, we can make recommendation based on them. Since all meta-paths begin with users and end with songs, any commuting matrices will be a user-song matrix. Every entry in the commuting matrix will be a user-song pair preference score. The higher the score is, the stronger the user preference to that song. Based on this, for the users that we want to recommend songs to, we can recommend the songs based on the preference score with respect to that user except for the songs that the user had already listened.

4.3 Deep Learning on Meta-Paths

4.3.1 Combining Multiple Meta-Paths. The HIN hybridizes collaborative preference and content-based preference with different features in the same framework, which motivates us to implement machine learning method to assemble multiple meta-paths for better music recommendation. We can build multiple meta-paths, some paths may have a positive effect on recommendation, while others may not be helpful. Machine learning helps us automatically learn from observed history then effectively select useful meta-path features.

Before applying machine learning methods, since the commuting matrices are generated from different meta-paths, the user-song preference scores are in different scale, so that we need to normalize the scores for each user in commuting matrices by doing the following calculation:

$$\bar{r}_{u,i} = \frac{r_{u,i} - \mu_u}{\sigma_u}$$

where $\bar{r}_{u,i}$ denotes normalized rating, $r_{u,i}$ denotes original rating, μ_u denotes the mean score of user u and σ_u denotes standard derivation of user u.

Let $m_{u,i}^{(L)}$ denotes the Lth meta-path prediction of user u to item i, the prediction rule of multi-layer perceptron can be formulated as follows:

$$\widehat{r}_{u,i} = f(m_{u,i}^{(1)}, m_{u,i}^{(2)}, \ldots, m_{u,i}^{(L)} | \theta)$$

where function $f(\cdot)$ defines the multilayer perceptron, and θ is the parameters of this network. The loss function for predicting explicit rating is defined as square error:

$$L = \sum_{(u,i)} (r_{u,i} - \widehat{r}_{u,i})^2$$

where $r_{u,i}$ is the actual rating, and $\widehat{r}_{u,i}$ is the predicted ratings of user i to song j. Note that the actual rating is coming from the normalized user listening history we mentioned before.

4.3.2 Multi-Layer Perceptron on Meta-Paths. Since we already have multiple meta-paths, intuitively we can combine features from different meta-paths by grouping them together, the simplest way is the linear regression. However, simply a linear regression does not account for any interaction between features, which is insufficient for predicting. To tackle the problem, we proposed to use multi-layer perceptron algorithm with two hidden layers to learn the correlation between features. Therefore, we can give our model more flexibility and non-linearity to learn the correlation between features. More precisely, the MLP model we used is formulated as follows:

$$z_1 = [m_{u,i}^{(1)}, m_{u,i}^{(2)}, ..., m_{u,i}^{(L)}]$$

$$z_2 = \phi_2(z_1) = a_2(W_2^T z_1 + b_2)$$

$$...$$

$$z_L = \phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_L)$$

$$\widehat{y}_{u,i} = \sigma(h^T \phi_L(z_{L-1}))$$

where W_x, b_x, and a_x denote the weight matrix, bias vector, and activation function for the x-th layer's perceptron, respectively. For the activation function of MLP layers, one can freely choose sigmoid, tanh and ReLU. We choose ReLU which is more suitable and proven to be non-saturated; Moreover, it encourages sparse activations, which is suitable for sparse data and making the model less likely to be overfitted.

Tower pattern is a common solution for the design of network structure, where bottom layer is the widest and each successive layer has a small number of neurons. In addition, the first hidden layer allows for more neurons as they do not influence the output and better for adjusting. Each user-song sample's feature is a length 10 vector when we merge 10 meta-paths together, with ith meta-path contribute its corresponding user-song pair as ith feature. Therefore, we implement a two hidden-layer MLP on 10 meta-paths with 16 neurons on the first and 8 on the second layer. The network structure is illustrated as follows:

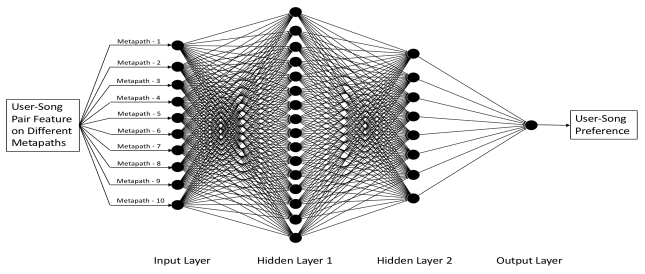

Figure 6: MLP structure

5 EXPERIMENTS

5.1 Evaluation Metrics

We evaluate the performance of our recommender system based on the hit rates (HR). More precisely, it's computing the number of overlap between the predicted songs and hidden songs for each user. One thing to mention here is that we will predict exactly the number of hidden songs for each user, for example, if the user has N hidden songs, then we will recommend top-N songs with the highest scores for that user, therefore, **hit rates = top-N precision = top-N recall**. We illustrate it more precisely in Table. 3. For a different user, the number of hidden songs will be different, as we choose 50% of listened songs for each user in the test set to be hidden. The average hit rates will be the average of each user's hit rate.

Indices	*Description*
TP	# of songs correctly predicted as like
TN	# of songs correctly predicted as dislike
FP	# of songs mistakenly predicted as like
FN	# of songs mistakenly predicted as dislike
H	# of hidden songs
N	# of songs recommended
$Precision$	$TP/(TP + FP) = TP/N$
$Recall$	$TP/(TP + FN) = TP/H$
$HitRates$	choose $N = H$, $TP/N = TP/H$

Table 3: Evaluation Metric

5.2 Single Meta-Paths Comparison in Music Recommendation

As we mentioned before, we have two entities and five relationships in our HIN and we can build meta-paths connecting two entities utilizing multiple relationships. We do experiments on single path performance for total 15 meta-path we chose and Table. 4 shows the result, where R denotes user-song adjacency matrix, T denotes song-artist adjacency matrix, E denotes song-genre adjacency matrix, M denotes song-album adjacency matrix and C denotes song-acoustic adjacency matrix.

From the experiment results we get several observations on different meta-path choosing:

1. Shorter meta-paths perform better than longer meta-paths. The longer meta-path is less readable and more information do not yield better predictions.

2. Artist and Album related meta-paths perform much better than other content features in music recommendation. Acoustic feature is helpful in content-based recommendation.

3. Genre performs badly in recommendation. The same genre contains too many songs and it is not informative in music recommendation. The category of content features should be rich and representative.

We rank meta-paths based on average hit rates and choose the top-5 and top-10 meta-paths to apply machine learning methods. The paths we choose are shown in Table. 5. We abandon meta-paths with song-genre information as it performs so poorly.

Metapaths	*Single Path Average Hit Rates*
RR^TR	20.07%
RT^TT	25.30%
RE^TE	0.82%
RM^TM	26.93%
RC^TC	6.94%
RR^TRT^TT	13.49%
RR^TRE^TE	0.46%
RR^TRM^TM	15.17%
RR^TRC^TC	1.65%
RE^TET^TT	2.11%
RE^TEM^TM	1.69%
RE^TEC^TC	0.25%
RT^TTM^TM	23.41%
RT^TTC^TC	7.79%
RM^TMC^TC	8.03%

Table 4: single meta-paths comparison

Methods	Chosen meta-paths
$Metapath(5)$	RR^TR
$Metapath(5)$	RT^TT
$Metapath(5)$	RM^TM
$Metapath(5)$	RR^TRM^TM
$Metapath(5)$	RT^TTM^TM
$Metapath(10)$	RR^TR
$Metapath(10)$	RT^TT
$Metapath(10)$	RM^TM
$Metapath(10)$	RC^TC
$Metapath(10)$	RR^TRT^TT
$Metapath(10)$	RR^TRM^TM
$Metapath(10)$	RR^TRC^TC
$Metapath(10)$	RT^TTM^TM
$Metapath(10)$	RT^TTC^TC
$Metapath(10)$	RM^TMC^TC

Table 5: Meta-path chosen to apply machine learning

5.3 Comparison with other Baseline Models in Music Recommendation

5.3.1 Baselines. To compare the performance of our model, we implement several baseline models in music recommendation. The chosen baselines are introduced as follows.

Popular based Recommendation. We recommend top-N mostly listened songs for each user, where N is the number of hidden songs for each user.

Collaborative Filtering. We implement a rank-3 approximation on user-song rating matrix and reconstruct them to predict the unobserved ratings in hidden set.

Meta-path + Matrix Factorization. For commuting matrices we obtained from each meta-path, we implement traditional matrix factorization and concatenate the user latent factors and song latent factors from the chosen meta-paths together. We implement supervised learning on the concatenated feature vectors and make the recommendation based on the prediction of user-song pair scores.

Meta-path + linear regression. We simply implement linear regression on different commuting matrices based on normalized user listening history.

Methods	*Average Hit Rates*
Random Guess	0.62%
Popular based Recommendation	7.30%
Collaborative Filtering	14.11%
$Metapaths(5) + MatrixFactorization$	12.11%
$Metapaths(10) + MatrixFactorization$	12.67%
$Metapaths(5) + LinearRegression$	28.97%
$Metapaths(10) + LinearRegression$	30.29%
$Metapaths(5) + NeuralNetwork$	29.73%
$Metapaths(10) + NeuralNetwork$	32.28%

Table 6: Comparison with baseline models

5.3.2 Comparison with Baseline Models. From Table.6, we find that meta-path based recommendation outperforms traditional collaborative filtering and matrix factorization methods. By utilizing machine learning techniques, meta-path based recommendation achieves higher hit rates. Neural Network methods work better than linear regression as it considers the correlation between different features. With more information, meta-paths(10) performs better than meta-paths(5) in recommendation. Figure.7 shows the distribution of meta-paths(10) + neural network recommendation satisfies the Gaussian distribution with mean=32.28% and standard derivation=15.02%.

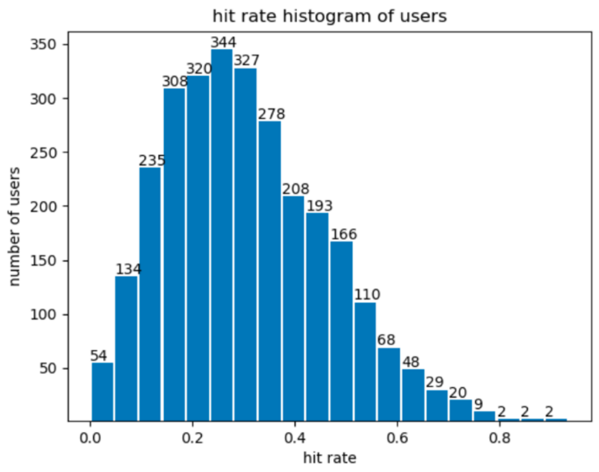

Figure 7: The distribution of Meta-path(10)+Neural Network

5.4 Deep Learning Based Feature Extraction

We run SOM, Siamese model and Classification model in our dataset.

5.4.1 Acoustic Feature Preprocessing. In the experiment, we adopt *Timbre* as acoustic feature of songs. Keep in mind that there are many kinds of acoustic features out there provided by Million Song Dataset. The reason we use *Timbre* is that it is the most informative feature. *Timbre* contains 12 different kinds of patterns of a song track which may represent a song's pattern in a different perspective and dimension.

Acoustic features were originally in HD5 format in Million Song Dataset. We first extract them from those original files into *Numpy* format with *songId* as its file name. We do this way for utilizing data and features more efficiently and directly. With regard to features that used in Siamese model and classification model, considering that the size of the features are arbitrarily set, we choose to resize it using interpolation. In our experiment, all the *Timbre* that feed to the model are resized to 1×1×12×1024. We prepare them in 4 dimensional format for the data-flow requirement in *Pytorch*

5.4.2 Model Structure. For better comparison, we adopt the same feature extractor part in each model. The concrete implementation of the model for feature extractor part is shown in Table7.

type	patch size/stride	output size	depth
Conv+BN+ReLU	3×3/1	16×12×1024	16
Conv+BN+ReLU	3×3/1	16×12×1024	16
Max pooling	1×2/	16×12×512	
Conv+BN+ReLU	3×3/1	32×12×512	32
Conv+BN+ReLU	3×3/1	32×12×512	32
Max pooling	2×2/	32×6×256	
Conv+BN+ReLU	3×3/1	32×6×256	32
Conv+BN+ReLU	3×3/1	32×6×256	32
Max pooling	2×2/	32×3×128	
Conv+BN+ReLU	3×3/1	64×3×128	64
Conv+BN+ReLU	3×3/1	64×3×128	64
Max pooling	1×2/	64×3×64	
Conv+BN+ReLU	3×3/1	128×3×64	128
Conv+BN+ReLU	3×3/1	128×3×64	128

Table 7: Model structure of feature extractor

The feature extractor part of the network that we designed is a 14 layers of $Conv - Relu - BN$ and $max - pooling$ combinations, with a fully connected layer at the end if adopting classification model, or with a contrastive loss layer at the end if adopting Siamese model. We train both of the network for 5 epochs at most, with a learning rate of 3e-5 at the first stage and apply the strategies of regularization and momentum.

5.4.3 The Result of the Model. We test the model using forward propagation. The classification results are provided for the use in meta-path algorithm. In that algorithm, only information from acoustic features are used. Generally, after testing model in multiple iteration step, the results are shown in Figure.8

From the result in Figure.8, we can see that the classification model performs much better than Siamese model. 6.94% is the best model when the number of classification is 500.

The given model has proven to be contributive to recommender system if we only consider acoustic feature. However, mostly a mature recommender system may integrate many kinds of features so that a reasonable result of prediction will present.

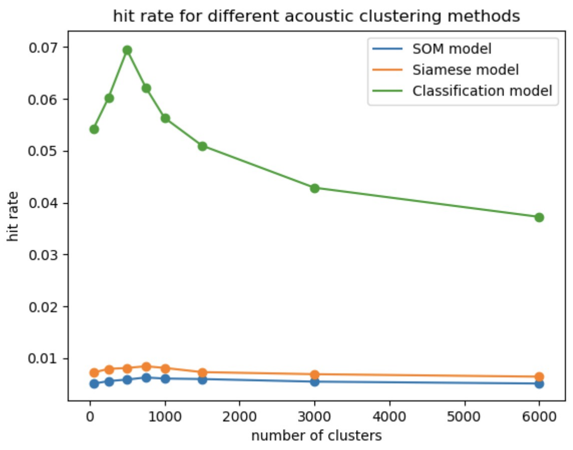

Figure 8: Prediction accuracy using pure acoustic feature information

6 RELATED WORK

6.1 Heterogeneous Information Networks

HIN is helpful to represent real-world graphs and networks. Meta-paths have been developed to solve real graph-based problems on HIN schema. Meta-paths enable traditional similarity measurement used in homogeneous graphs also available in heterogeneous graphs, like PathSim[2], and Path Constrained Random Walk[11]. The introduction of meta-paths in HIN also open up a field of other typical graph mining topics, like recommendation[12–14] and classification[15]. Shifu Hou et al.[3] constructed a HIN for Android malicious software detection, which modeled the similarities between APPs and APIs using meta-paths and applied multi-kernel learning method for malicious software detection. Huan Zhao et al.[4] introduced the concept of meta-graphs to model more complicated semantics in HIN and applied them in the recommender systems. Inspired by these work, we design a HIN based music recommender framework. In addition, their work also motivates us to test different meta-paths performance and implement machine learning algorithms on meta-paths to achieve higher performance.

6.2 Recommendation with Heterogeneous Information

HIN is suitable to combine different semantics and information in nowadays recommender systems. Before applying HIN on recommendation, many efforts have been made to merge different kinds of heterogeneous information in recommendation system. In [16] the author incorporated social relationship as a regularization term into the matrix factorization methods based the recommender systems. The author of [17] incorporated user preferences with acoustic features for music recommendation. In [18] the author proposed a probabilistic model to represent user's preferences, social network and geographical information to enhance the point-of-interests recommendation. These previous work demonstrated the importance of heterogeneous information in recommender systems, while they utilized the heterogeneous information in a disparately way.

Traditional recommendation method can be enhanced and merged into a canonical framework using HIN. Based on meta-paths, many efforts have been done to tackle recommendation problems within HIN. In [14], multiple meta-paths are used to learn user and item latent factors, which are used to rebuild similarity matrix for the recommendation afterward. In [12], users' ratings to items are utilized to build a weighted HIN and the recommendation algorithms can be applied based on the constructed HIN. However, the traditional hybridization methods for different meta-paths do not fully utilize the information between them. Machine learning will help us in this field of work.

6.3 Machine Learning in Recommendation Systems

When we consider combing different meta-paths for optimization, we need to adopt suitable machine learning model to capture the pattern within the meta-path features. Intuitively, a naive linear regression model is able to give different weights to different meta-paths. However, it does not consider the correlation between features thus lose important information. The author in [3] proposed multi-kernel learning model on different meta-paths for classification problems. Since we are facing a rating score regression problem, we decide to implement a multi-layer perceptron regression model which considers the high-level correlation between features.

7 CONCLUSIONS

7.1 Summary

To realize better music recommendation, instead of simply combining classical collaborative filtering and content-based recommendation, we proposed a novel recommendation framework that utilizes the meta-paths within heterogeneous information network. HIN creates higher level semantics thus captures different kind of preferences between entities. In order to model the content feature of songs, except the normal features like artists, albums and genres, we further explored the acoustic feature of the songs. In clustering songs, we came to a conclusion that doing multi-label classification over acoustic features works better than grouping them according to hyper dimensional distance using listeners' preference. Based on the user listening history and the content feature of songs, we built a structured heterogeneous information network and aggregate different meta-paths using machine learning techniques. From the experiments, we found that our purposed recommendation framework outperforms other traditional recommendation methods. The meta-paths combination through supervised learning yields better result than unsupervised approaches. Combination through neural network performs better than simple linear regression since the neural network can capture more correlation between user-song pairs.

7.2 What We Learned in this Project

In this project, first of all, after we had an investigation of methods for recommender system, we have a better understanding of how recommender system works as well as the classical recommendation algorithms. Since the recommender system is a popular topic in both academia and industry, these knowledges give us a solid foundation of recommender system and we might use these knowledges in the future. Besides, since our recommendation framework is based on heterogeneous graph, accompanied with what we learned in class, we grasp the crux of the heterogeneous graph and we can apply this model in other domains. Furthermore, we also improve our programming skills in doing this project, in this project, we mainly use Python to do the data pre-processing, graph modeling and system testing jobs, they enriched our experience in building complicated system involving large scale graphs. Last but not least, it cultivates our team cooperation and project management skills, we create a schedule table in the beginning of this project and we hold weekly group meeting to communicate ideas and progress. We think these skills are also very important when we get a job.

7.3 Further Work

Although we have already achieved a good hit rate currently, there is still some work can be done in the future. Since we use a relatively small dataset compared with the original dataset due to computational constraints, we might enlarge the dataset we used and also try to parallelize our method in order to achieve higher scalability. Besides, we may use some representative learning methods like metapath2vec to further explore the usage of meta-paths. In addition, we can extend our HIN based recommendation framework into other different recommendation domains to check whether it still works well.

8 DIVISION OF WORK

In this project, we formed a group of 4 people and finished this work together, the detailed work distribution is given below:

Kailin Tang is responsible for the data preprocessing part including data filtering, dataset generation and data normalization. Besides, he designed the heterogeneous information network framework for recommendation including the graph topology (entities and relationships) and meta-paths within the graphs to represent different kind of preferences. He also participated in the model testing in which he analyzed the performance of single meta-paths based recommendation.

Siyuan Xie is responsible for applying machine learning methods for different meta-path features including linear regression and multi-layer perceptron. He designed the structure and parameter of MLP model. In addition, He evaluated the performance of proposed machine learning methods and compared them with baseline models including popular based method and traditional collaborative filtering.

Ye Wang is responsible for data preprocessing which includes data format transformation and songs' feature extraction. Besides, he designed the structure and parameters of deep learning based Siamese model and classification model that used in acoustic feature extraction. He also provided classification result for model testing.

Xinyu Gao is responsible for acoustic feature analysis and acoustic feature extraction, and she applied PCA to extract first principle acoustic feature for each song and designed the SOM network and trained SOM network to cluster the filtered songs by their acoustic features. Besides, she also provided classification result for model testing afterwards.

9 ACKNOWLEDGEMENT

Thanks to Prof. Danai Koutra for providing this wonderful course, through which we learned a lot state-of-arts graph mining technique. Also, thanks to the great advice from professor with respect to our project. We would like to thank the LabROSA of the Columbia University who present the whole Million Song Dataset and also its subsidiary datasets that we used in our project. Thanks to the Echo Nest that provide the user listening history which greatly facilitate our work.

REFERENCES

[1] Y. Sun and J. Han, "Mining heterogeneous information networks: principles and methodologies," in *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2012.

[2] Y. Sun, J. Han, X. Yan, P.S.Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," in *PVLDB (2011).*

[3] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An intelligent android malware detection system based on structured heterogeneous information network," in *In KDD'17, August 13-17, 2017, Halifax, NS, Canada.*

[4] H. Zhao, A. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *In KDD'17, August 13-17, 2017, Halifax, NS, Canada.*

[5] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[6] F. Aiolli, "A preliminary study on a recommender system for the million songs dataset challenge preference learning," in *Problems and Applications in AI (PL-12)*, 2012.

[7] L. A. E. Tristan Jehan, David DesRoches, "Analyzer documentation," Janurary 2014.

[8] T. Kohonen, "Self-organization and associative memory," *Springer-Verla*, 1989.

[9] T. Kohonen, "Self-organizing maps," *Springer-Verla*, 1995.

[10] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 539–546, IEEE Computer Society, 2005.

[11] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," in *Machine Learning, 81(1): 53-67*, 2010.

[12] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, "Semantic path based personalized recommendation on weighted heterogeneous information networks.," in *In CIKM, pages 453-462*, 2015.

[13] X. Yu, X. Ren, Q. Gu, Y. Sun, and J. Han, "Collaborative filtering with entity similarity regularization in heterogeneous information networks," in *IJCAI HINA, 27*, 2013.

[14] X. Yu, X. Ren, Y. Sun, B. S. Q. Gu, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *In WSDM, pages 283-292*, 2014.

[15] P. Bangcharoensap, T. Murata, H. Kobayashi, and N. Shimizu, "Transductive classification on heterogeneous information networks with edge betweenness-based normalization," in *In WSDM, pages 437-446*, 2016.

[16] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *In WSDM, pages 287-296*, 2011.

[17] T. L. Bo Shao, Dingding Wang and M. Ogihara, "Music recommendation based on acoustic features and user access patterns," in *IEEE Transactions on Audio,Speech, and Language Processing*, 2009.

[18] M. Ye, P. Yin, W. C. Lee, and D. L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *In SIGIR, pages 325-334*, 2011.