WSDM - KKBox Music Recommendation Challenge

Kumari, Sangeeta
The Ohio State University
Columbus, OH
kumari.14@osu.edu

Ghai, Piyush
The Ohio State University
Columbus, OH
ghai.8@osu.edu

Mahajan, Pravar D.
The Ohio State University
Columbus, OH
mahajan.89@osu.edu

ABSTRACT

The 11th International Conference on Web Search and Data Mining (WSDM 2018)[1] is organizing a music recommendation competition in conjunction with KKBox - Asia's leading music streaming service. This paper presents a Deep Learning based approach to tackle the music recommendation problem on KKBox's dataset. [2]

1 INTRODUCTION

Gone are the days when people would be forced to listen to a given set of songs based on whatever a Radio Jockey played on a Radio station. Now in the days of unlimited streaming services, personalized algorithms are the key to keep the users hooked to your music services.

WSDM - KKBox Music Recommendation Challenge is an online competition hosted on Kaggle. As per the competition page [1], KKBox currently uses a collaborative filtering (CF) approach but is looking for new approaches to generate a better recommendation for individual users and effectively handle new users and songs. The task for the competition is to predict whether the same user would listen to a song again or not; the challenge being insufficient history for some users or/and songs, also referred to as Cold Start problem. Content consumption services such as Netflix, Spotify, KKBox etc. have to deal with cold start problems and most of the existing solutions for the same assumes the use of CF or presence of lyrics and audio clips of songs [5]. Though the nature of the problem is apt for boosting algorithms, we propose Deep Learning approaches for the task, while handling cold start problem (i.e., the problem of new users and/or new songs) in a novel way. All our model implementations were done in Keras, using Tensorflow as a backend.

The rest of the paper is organized as follows: Section II of this paper describes the given dataset and provides a exploratory dataset analysis on it. Section III describes the data preprocessing steps performed on the dataset. Section IV discusses the engineered features which we used in some of our models including the one that gave the best accuracy. Section V presents the various models that we have tried. The results and analysis of the models is summarized in Section VI. Finally, in Section VII, we give our approach to handle Cold-start problem.

[1] https://www.kaggle.com/c/kkbox-music-recommendation-challenge

2 DATA & EXPLORATORY DATA ANALYSIS (EDA)

2.1 About the dataset

The dataset presented by Kaggle consists of a training dataset of **7377418** records. The test dataset consists of **2556790**. Apart from this, they have provided a **songs** table, which consists of metadata for the songs, such as their artists, genre, composers, language, name of the song, ISRC code for the song. There is also a provided **members** table, which contains a user's demographic information such as age, city, gender, registered_via (the source from which they subscribed to KKBox) and also their subscription creation and expiry dates on KKBox. The training and test dataset consist of msno (user descriptor), song_id (song descriptor) and some additional information about the source from which the song was played (for example, was the song played from the user's playlist or from online radio?) The fields in various tables presented in the dataset are in Table 1. From the dataset, the challenge becomes clear, which is to predict whether a user will listen to a given song again within a 30 day time window.

The ISRC presented in the songs table is an International Standard Recording Code, which can theoretically can be used as an identity of a song. However, there are certain caveats such as ISRC generated from providers have not been officially verified, therefore the information in ISRC, such as country code and reference year, can be misleading/incorrect. Also, multiple songs could share one ISRC since a single recording could be re-published several times.

Table 1: Fields in the dataset

Table Name	Fields
train	msno, song_id, song play source, target
test	msno, song_id, song play source
songs	song_id, genre, artists, composers, lyricist, language
members	msno, city, bd, gender, registered_via, subscription age
song_extra_info	song_id, song name, ISRC code

2.2 EDA

The training dataset has a pretty even distribution of classes. In the training and test data combined, there are **419868** songs and **34404** users. Interestingly, the songs table provided consists of over **2.3M** songs. The code for our EDA can be found at [2]

[2] https://github.com/pravarmahajan/WSDM-Challenge/blob/master/EDA.ipynb

	song_id	song_length	genre_ids	\
0	CXoTN1eb7AI+DntdU1vbcwGRV4SCIDxZu+YD8JP8r4E=	247640	465	
1	o0kFgae9QtnYgRkVPqLJwa05zIhRlUjfF7O1tDw0ZDU=	197328	444	
2	DwVvVurfpuz+XPuFvucclVQEyPqcpUkHR0ne1RQzPs0=	231781	465	
3	dKMBWoZySCdxSkihKG+Vf47nc18N9q4m58+b4e7dSSE=	273554	465	
4	W3bqWd3T+VeHFzHAUfARgW9AvVRaF4N5Yzm4Mr6Eo/o=	140329	726	

	artist_name	composer	lyricist	language	
0	張信哲 (Jeff Chang)	董貞	何啟弘	3.0	
1	BLACKPINK TEDDY| FUTURE BOUNCE| Bekuh BOOM	TEDDY	31.0		
2	SUPER JUNIOR	NaN	NaN	31.0	
3	S.H.E	湯小康	徐世珍	3.0	
4	貴族精選	Traditional	Traditional	52.0	

Figure 1: Snapshot of the songs table

2.2.1 Members Table. Next up, we looked at the members table. The user information is a complete mess, as a lot of users have their *bd (birth date)* missing. The same is inferred for the gender attribute as well. The cities attribute seems to be a categorical attribute. The distribution of cities is quite skewed, with a large number of cities labeled as 0, probably a default option for the users?
The additional metadata information about the source from where the user played a song is also categorical and not much inference can be drawn by plainly looking at the column values. The subscription age details suggest a lot of users registered with the app in *2016*, while a small proportion of users registered before 2016.

2.2.2 Songs Table. The songs table is even more messed up than the users table. The figure 1 represents a snapshot of the data present in the songs table.

From the figure 1 we can see that artist_name , composers, lyricists can have multiple values in a single column, separated by delimiters such as **'|', '/', '&', 'and', ';'** etc. The names are also represented in different languages such as Mandarin, English, Korean with some names accompanied by their translation in another language within round braces.

The genre is a categorical attribute, where a song can also have multiple genres separated by a '|'. Language is also a categorical attribute, with a lot of missing values filled in as default -1. The language frequencies are presented in table 2.

Table 2: Language Distribution

Language Value	Frequency
-1.0	639467
3.0	106295
10.0	15482
17.0	92518
24.0	41744
31.0	39201
38.0	2385
45.0	14435
52.0	1336694
59.0	8098

Although we cannot infer the actual language mapping from the numbers, but a closer inspection of language value **52** led us to believe that it could possibly be English, since all the artist, composer and lyricist names were in plain and clean English.
The polluted nature of the dataset made it a very challenging and tedious exercise to run our models, and it became imperative for us to clean the dataset first in order to proceed ahead. We next present the section on Preprocessing steps used.

3 DATA PREPROCESSING

Given the nature of this dataset, it became imperative to perform some filtering on it. Here are briefly the steps followed for preprocessing the data :

(1) We first extracted those song ids from the songs table, which are present only in the training and test dataset. These were combined to create a new **shortlisted_songs** file. The same was repeated for the members table, where we extracted members present in training and test data in a **shortlisted_members** file.
(2) From the *shortlisted_songs* table, we found that a given song can have multiple artists/composer/lyricist and/or their names can be given in a Non-English language, or a mixture of English and Non-English language, as can be seen in 1. Thus, in order to extract out the unique value from each of these attributes, we decided to de-duplicate them.
(3) For the de-duplication of the attributes mentioned above, we split them on the following : **'|', '/', '&', 'and', ';', 'feat', 'featuring', 'ft.'**. This was done for all the attributes in the songs table, except genre. The genre attribute was split only on '|' character.
(4) Next up, we undertook a mammoth task to translate all the non-English names for song names, artists, composers, lyricists. We used Google Translate's **googletrans** [3] Python package for the same.
(5) After translating all the names, we created a meta training dataset, which augmented the existing training dataset with new fields added corresponding to artist name, composer, lyricist, genre for a given song id, and member information such as age, city, subscription details for a given member id.
(6) In this meta training dataset, if a song had multiple values for artists, we created multiple rows, each row corresponding to a unique artist. Similarly, unique rows were created for other attributes with multiple values.
(7) The new meta training dataset consisted of **23M** rows, and contained sufficient contextual information for a song and a user, to feed to our models, described in the next section of this paper.

4 FEATURE ENGINEERING

4.1 User Statistics

Users who listened to songs repeatedly (target = 1) are more likely to repeat other songs also, provided they like it. On the other hand, some users show no history of repeating songs thus less likely to do so in future. Also, users who have been member of KKBox since long might show higher activity thereby proving to be a significant factor towards the target.

[3] https://pypi.python.org/pypi/googletrans

Table 3: Features from user data

Feature	Description
user_song_repeated	no. of times an user repeated songs
membership_days	no. of days an user is a member (expiration date - registration date)
registration_year	year of registration
expiration_year	year of expiration

4.2 Source Statistics

There are three different fields pertaining to source information in data namely source_system_tab, source_screen_name and source_tab. During EDA, we noticed that source information has relation with the target. A song which is played from local playlist is listened to again more than 60% of the time whereas those heard from the radio are the least (20%) likely to be listened to again. In order to capture this relation, we introduced 3 additional features.

Table 4: Features from source data

Feature	Description
source_system_repeated	no. of times a source system resulted in target=1
source_screen_repeated	no. of times a source screen resulted in target=1
source_type_repeated	no. of times a source type resulted in target=1

4.3 Song Statistics

Majority of music listeners have their favorite artist and usually like to hear to songs which are in top charts. Also, if a song has many artists or is relevant to various genres, there is a high probability of it being listened to again. In order to incorporate a song's/artist's popularity and get an idea of general liking by other users, we introduced the following features.

Table 5: Features from song data

Feature	Description
song_repeated	no. of times a song was repeated
artist_repeated	no. of times an artist was repeated
language_repeated	no. of times a language was repeated
artist_played	no. of times users listened to (once/repeated) a particular artist
song_played	no.of times users listened to a song
count_artists	no. of artists associated with a song
count_lyricists	no. of lyricists associated with a song
count_composers	no. of composers associated with a song
count_genre_ids	no. of genres relevant to the song

5 MODELS

5.1 Gradient Boosting

We tried gradient boosting in order to use it as a baseline for our model. For gradient boosting, we used an off the shelf package : Microsoft's LightGBM[4] package. Since this was to be treated as a baseline score for our other Deep Nets, not a lot of hyper parameter tuning was done on it. Despite that, this method gave a very good AUC score of **0.6552** the test dataset. The hyper parameters used in Gradient Boosting are presented in Table 6.

Table 6: Gradient Boosting Hyper Parameter values

Parameter Name	Frequency
learning_rate	0.1
metric	binary_logloss
bagging_fraction	0.95
max_bin	128
max_depth	10
num_rounds	200
metric	auc

5.2 Deep Neural Network (DNN)

5.2.1 Input Data. The model was tested with three kinds of data (all being Label Encoded): raw data, preprocessed data (mentioned in Section 3) and with engineered features (mentioned in Section 4) augmented. Table 7 compares the performance of each input data.

5.2.2 Architecture. For all the data representations, we used the same model of 4 dense layers, as presented in Figure 2

Table 7: DNN Performance

Data	Accuracy
Raw data	0.4972
Preprocessed data	0.5874
Engineered features added	0.6012

5.3 DNN with Pre-Trained Embedding

The dataset presents us some contextual information about a user, *viz.* the user demography. Similarly, for a song, the song parameters such as its artist, composer, lyricist etc are given. We believed that this information could be useful if captured in some embedding space while training our models. In order to get a better representation of users and songs, we chose to generate their respective embeddings. In all our embedding models, we chose a common recipe of using a single attribute as input and tried to predict other relevant attributes as our output, giving it a one-to-many model architecture.

[4]https://github.com/Microsoft/LightGBM

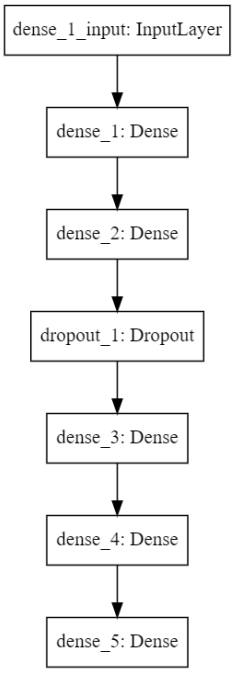

Figure 2: DNN Model

5.3.1 User Embedding. For generating the user embeddings, the model consists of a trainable embedding layer whose output dimension is 50 units. The input to the embedding layer is a single one hot encoded value of msno. The output from the embedding layer is passed onto 4 dense layers, each trying to predict the following : *user city*, *user registered via*, *user subscription registration year*, *user subscription expiry year*. Each of the output values is a categorical value which will be predicted by the embeddings model. The model is presented in Figure 3. A closer analysis of the accuracies at each dense layer, presented in Table 8, revealed that the model was memorizing two attributes (registered via & expiry year), rather than learning it. This could be due to small number of possible values to predict for these, or in general over fitting in the learning process. Since the model was trained for all the unique users, memorization seems highly likely by the model.

Table 8: User Embedding Output accuracies

Layer Name	Accuracy
dense_1	0.8253
dense_2	0.9233
dense_3	0.8300
dense_4	0.9441

5.3.2 Song Embedding. For generating song embedding, we used two different types of models. In one, we used song names as input and expected to gain something out of the name of songs and artists related to it (for example, some artists might be liked most

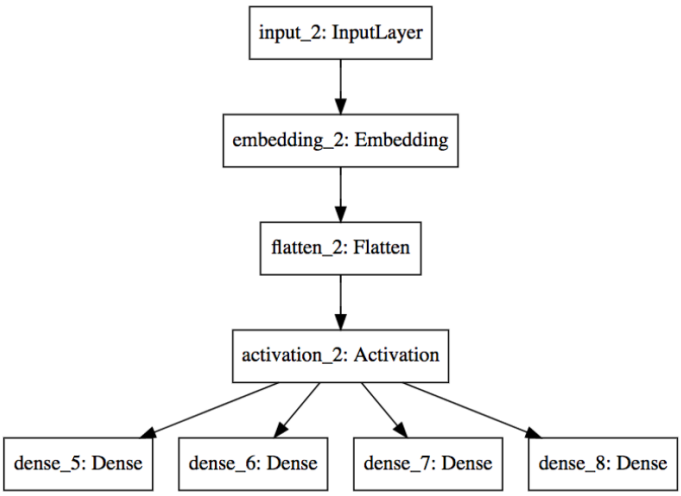

Figure 3: User Embedding Model

for their romantic numbers in which case the song name might have some similar words). Where in the first approach, we had a clear idea of what relation we are trying to exploit, in the other variation, we just tried to see what the network was learning with minimal information encoded in input i.e. song ids.

Song names The model with song names consists of a one layer LSTM of 128 hidden units, the output of which is passed onto a fully connected layer to get a 100 dimension vector used to predict artist, composer, lyricist and genre. The input to the LSTM layer is a 300 dimensional embeddings of the song names (cleaned and tokenized), where the embeddings are used from GloVe vectors.

Song ids The model based on song ids is very much similar to that for song names except the input to the model which is simply raw song ids. Therefore, instead of using an LSTM to generate hidden representation of the song, we use a simple feed forward neural network. The input to the model is a one hot representation of the song id, and the output are the song attributes, *viz.* artists, lyricist, composer and genre. The model followed was similar to the one presented in Figure 3.

5.3.3 Architecture. For the DNN with pre-trained embeddings, we tried two variations:

(1) Only pre-trained **User** & **Song** embeddings
(2) Pre-trained embeddings + other metadata present in training dataset. (Table 1).

The architectures for both these models were similar and we present the first one in Figure 5 When the metadata was added, it was concatenated as one hot encoded input with the outputs from the user and song embedding layers.

5.3.4 Training & Results. Both the variants of the models were trained for 20 epochs each. The training time was slow, **45 minutes** per epoch. The learning rates were varied between *1e-2* to *1e-4*. The

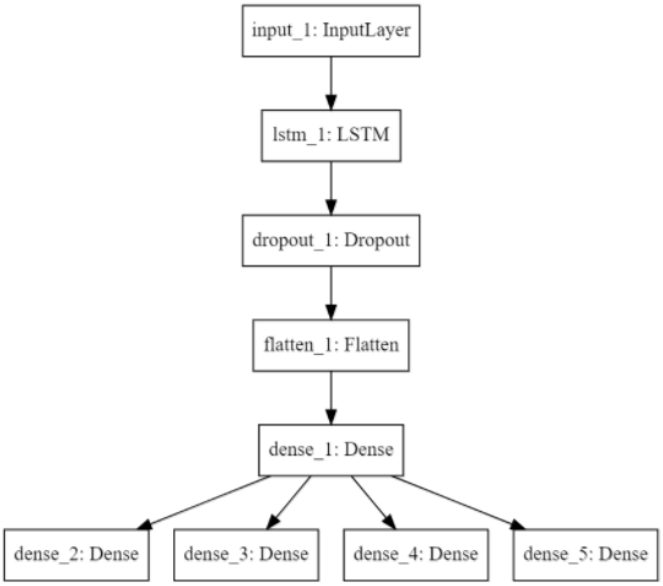

Figure 4: Song Embedding Model

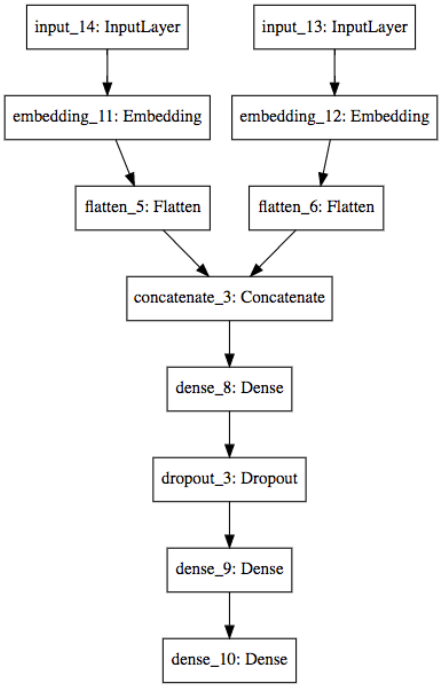

Figure 5: DNN with Pre-Trained Embeddings

batch size was kept constant as *32768*. The best results were obtained with a learning rate of **1e-4**, **ReLU activation**, **Dropout as 0.5**. The second variant of the model with metadata as additional input performs better than the first one. This is not surprising, since some of the metadata fields such as the source of playing songs (such as user's library or online) could be useful and intuitive factors in determining the outcome of repeating the same song again. The best test AUC scores for both the models are presented in Table 9.

Table 9: DNN with Pre-Trained Embeddings Results

Model	Test AUC
Only Pre-trained Embeddings	0.5423
Pre-trained Embeddings + Metadata	**0.5725**

5.4 DNN with No Pre-Trained Embeddings

We now present our best performing variant of the model. This is a variant of the Neural Collaborative Filtering (NCF) [4]. Whereas in the original NCF model, the output prediction was a rating between 0-10, our NCF treats the problem as a binary classification task 0-1. The architecture of our model is shown in 5.4. The inputs consist of user id, song id, source information, metadata about users and songs. Additionally, we engineered new features, which have been described in 4. We fed to our DNN model all the provided features in training dataset as well as the engineered features, and binary output was the target for our DNN model. Our results have been summarized in 10 and it shows a marginal improvement in Test AUC score on including the engineered features.

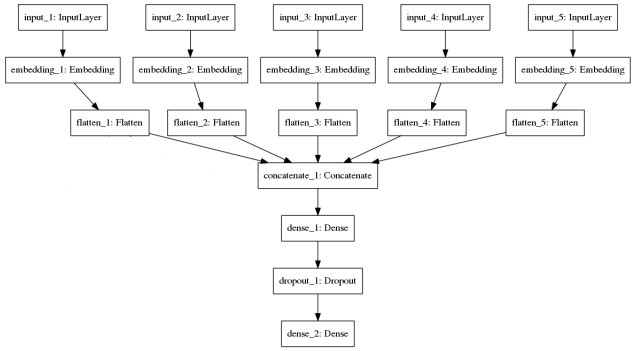

Table 10: DNN with No Pre-Trained Embeddings Results

Model	Test AUC
NCF with no engineered features	0.6603
NCF with engineered features	**0.6620**

6 COLD-START PROBLEM

When using a DNN trained with all features (given dataset and Section 4), the model randomly guesses the output since the related data for the user and(/or) song were missing while training. To handle this problem, we decided to have 3 variations of our DNN model. We divided the test set into the following 3 cold-start categories:

(1) New User but existing Song: The user was missing in train set but the song was present. To test this set of data, we used a DNN model trained with features only from Song and Source statistics.

(2) New Song but existing User: When the user was seen before in train set but the song is new, we tested the subset with DNN trained with features only from User and Source statistics.
(3) Both User and Song are New: For this case, we trained our DNN with only Source statistics. To predict the target for these cases, we took average of the results from relevant DNN (according to the case) from above and the generic DNN (as mentioned in Section 5.4)

7 CONCLUSIONS AND FURTHER WORK

We thus present an end to end deep learning based solution to the music recommendation competition. We showed different ways in which the metadata may be combined with the training inputs in order to improve the AUC score. We also showed that engineering some new features help us improve the score slightly. Given that the problem is more naturally suited to adaptive boosting techniques, our models do not achieve the best possible scores. For further work, we may consider exploring techniques of combining the results from our deep learning solution as well as these boosting techniques. It may be possible that not all of our engineered features are useful, and in fact, some of them could be hampering the performance of our model. Using techniques for feature selection, such as L1-regularization or Tree Based Feature selection might be a good way of eliminating these detrimental features, and thus improving the performance of the model.

ACKNOWLEDGMENTS

The authors would like to thank The Ohio Supercomputing Center [3] for providing us with platform for expensive computations required for training our deep learning models. We would also like to thank Kaggle for hosting the competition and KKBOX for providing the dataset. Finally, we would like to thank Professor Eric Fosler for his valuable inputs during the course of this project.

REFERENCES

[1] 2018. *WSDM 2018.* http://www.wsdm-conference.org/2018/.
[2] 2018. *WSDM KKBox Music Recommendation Challenge.* https://www.kaggle.com/c/kkbox-music-recommendation-challenge/.
[3] Ohio Supercomputer Center. 1987. Ohio Supercomputer Center. http://osc.edu/ark:/19495/f5s1ph73. (1987).
[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17).* International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 173–182. https://doi.org/10.1145/3038912.3052569
[5] Mohamed Sordo Sergio Oramas, Oriol Nieto and Xavier Serra. 2017. *A Deep Multimodal Approach for Cold-start Music Recommendation.*