Utilizing Attention Mechanism for Sequential Skip Predcition

Zhaoyue Cheng
Layer6 AI
Toronto, Ontario
joey@layer6.ai

Maksims Volkovs
Layer6 AI
Toronto, Ontario
maks@layer6.ai

ABSTRACT

Predicting whether the user is going to skip the next song is an important problem in the field of music recommendation. The prediction can be challenging due to different user preferences. In this paper, we will explore using deep learning architectures with attention mechanism to model temporal behaviour of skipping patterns. The proposed model was submitted to the ACM Spotify Sequential Skip Prediction Challenge as team learner6 and ranked fourth on the final leaderboard. The code is open sourced at: https://github.com/ZhaoyueCheng/WSDM_Spotify

KEYWORDS

Music Recommendation, Playlist Skip Prediction, Recurrent Neural Network, Attention Mechanism

1 INTRODUCTION

With the popularity of online music streaming platforms, accurately predicting if a user is going to skip songs is an important problem in this field. In online streaming platforms, users normally listening to music in sessions. Being able to infer from the past session information and determine whether users are going to skip upcoming songs in the playlist can help online streaming platforms give more personalized recommendations and better serve the user interests.

The Spotify Skip Prediction Challenge provides data [2] for user listening sessions along with various session features as well as different track features. In the test set, we are given the complete history tracks with session features for first half of the session, and tracks only for second half of the sessions. The target is to predict whether the user is going to skip the song in the second half of the sessions. The metrics is the average accuracy of the prediction of songs in the second half of the sessions.

Many works [4, 5] have show that recurrent neural networks can achieve state-of-the-art result in the field of session based recommendation. Also, recent advances [1] in the field of natural language processing have shown that attention mechanism is really powerful in sequence modelling since they reduce the effective path between elements in the sequence. As a result, we decide to combine recurrent neural networks (RNN)[6] with attention mechanism in our team's approach.

Melbourne'19, Feburar 2019, Melbourne, Australia
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In our approach, we first model the interactions within the history and predict songs with Bi-directional RNN and various attention mechanism, then we merge the history and predict hidden vectors using another layers of Bi-directional RNN and attention to model the interaction of the entire session information. Finally, we apply feed forward layers to obtain the logits for target tracks.

We choose to model history and predict interactions separately since history tracks have both session history information together with track features while predict tracks only have track features. Using a unified Bi-directional RNN to encode history and predict tracks may not be ideal since the information we have about each group is different which might result in different latent distributions. So we are using separate encoders for history and predict groups. Then we combine the information we obtained from history and predict groups in following layers to model global interactions.

This paper discusses the related work in Section 2 and data preprocessing steps we took in Section 3, the network architecture we used in Section 4 and the ensemble technique we used in Section 5. Finally, we discuss some experiment results in Section 6 and conclusions we got in Section 7.

2 RELATED WORK

Traditional techniques in recommendation systems such as collaborative filtering [7, 11] and matrix factorization [7, 9] fails to work in the field of session based recommendation since these models don't take order of actions into account. Also, most of the session based recommendation systems [3, 5, 13] are focusing on predicting the top-K items to recommend which is slightly different than the problem we are solving in this case. Our work propose a novel attention based architecture for sequential skip prediction problems.

3 DATA PREPROCESSING

The WSDM Spotify Skip Prediction dataset includes millions of user listening sessions. Within each session we separate the tracks into two parts, the first half of the tracks are the history tracks and the second half of the tracks are the predict tracks. We have tracks information for all tracks appearing in the dataset. Along with the tracks information, the dataset also provides session information. Session information is available for all tracks in the training set, but only available for the history tracks in the test set.

For all the track and session features, we use similar ways of preprocessing. For all the categorical features, we turn them into one hot embedding. For all the numerical features, we perform the following normalization strategy. First we clip the minimum and maximum value into certain ranges, then for each numerical value v, we perform $\frac{v-min}{max-min}$ to normalize the values where min and max are the minimum and maximum of the column values after clipping respectively. There are also several boolean features in the dataset, we map False to 0 and True to 1. Lastly we concatenate all the features to get the track feature embedding matrix E and session feature embedding matrix F. The dimension of E and F are 31 and 40 respectively after preprocessing.

Finally, since all session can have length from 10 to 20, so the resulting history and predict sessions can have length 5 to 10. We pad all the history and predict sessions to length 10 resulting in a total length of 20 for all session so that it's easier to batch the input for neural networks.

4 NETWORK ARCHITECTURE

Figure 1 gives an overview of the network architecture.

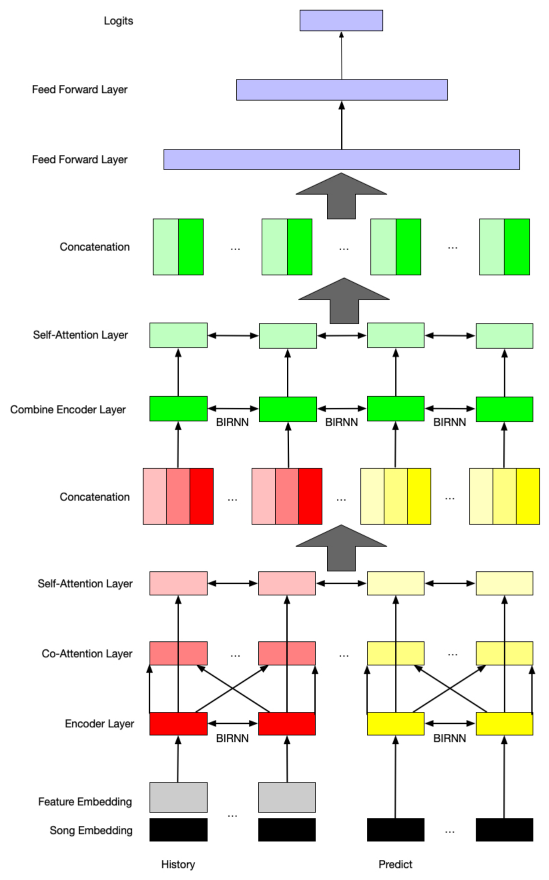

Figure 1: Network Architecture of RNN Model

4.1 History and Predict Encoder Layer

Consider History songs $H_s = \{s_t^H\}_{t=1}^{10}$ and Predict songs $P_s = \{s_t^P\}_{t=1}^{10}$. First the history and predict songs are embedded by looking up the embedding matrix E we obtained from the preprocessing step resulting in the history song embedding vector $\{e_t^H\}_{t=1}^{10}$ and the predict song embedding vector $\{e_t^P\}_{t=1}^{10}$. Additionally, for History, we also have the history feature vectors $f = \{f_t^H\}_{t=1}^{10}$ obtained from the preprocessed session feature matrix F by performing an embedding lookup.

Then we append the history feature vectors with the history song embedding vector to obtain the history representation. Since the predict songs don't have feature available, we use the predict song embedding alone as the predict representation.

We then use two seperate bi-directional RNN[10] to produce new representation $u_1^H, \ldots, u_{10}^H$ and $u_1^P, \ldots, u_{10}^P$, and the result hidden representations will share the same dimension for both history and predict songs:

$$u_t^H = BIRNN_H(u_{t-1}^H, [e_t^H, f_t^H]) \quad (1)$$

$$u_t^P = BIRNN_P(u_{t-1}^P, [e_t^P]) \quad (2)$$

We use LSTM as the choice of RNN encoder in this case. We obtain the u_t^H and u_t^P for $t \in \{1, 10\}$ as the hidden representation of session history and session predict respectively.

4.2 Self Matching Attention Layer

Vaswani et al. (2017) [14] proposed stacking self-attention modules to model long-term dependencies. Their model shows the effectiveness of self-attention in modeling dependencies and interactions between words since self-attention mechanism reduces the length of signal paths compared to RNN modules.

Though the sequence we need to model in this case is not as long as in the original work, but we still find self-attention really helpful in modeling interactions within the session history and session predict groups. Adding self matching attention layer allows us to build richer representation for both of the history song sequence and the predict song sequence.

Consider the input $\{u_t^H\}_{t=1}^{10}$ for history songs and the input $\{u_t^P\}_{t=1}^{10}$ for predict songs obtained through previous step. We perform a self matching attention on the history sequence and predict sequence respectively:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (3)$$

$$s_t^H = Attention(u_t^H, u_t^H, u_t^H) \quad (4)$$

$$s_t^P = Attention(u_t^P, u_t^P, u_t^P) \quad (5)$$

s_t^H and s_t^P for $t \in \{1, 10\}$ are the self attention representation of history and predict tracks respectively.

4.3 Co Attention Layer

Although self-attention mechanism is very powerful in modeling interactions within a history and predict session groups. We still use some mechanism to help us incorporating history into the predict sessions hidden representation and vice versa. In order to solve this problem, we match history tracks hidden representation to the predict tracks representation to get a history aware predict representation. We also map the predict tracks hidden representation to the history tracks representation to get a predict aware history representation.

These representations build on top of the original history and tracks representations with extra information, which helps the model to build a richer representation of the history and tracks features.

Consider the input $\{u_t^H\}_{t=1}^{10}$ for history songs and the input $\{u_t^P\}_{t=1}^{10}$ for predict songs. We perform a co attention layer on the history sequence and predict tracks respectively:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

$$a_t^H = Attention(u_t^H, u_t^P, u_t^P) \tag{7}$$

$$a_t^P = Attention(u_t^P, u_t^H, u_t^H) \tag{8}$$

a_t^H and a_t^P for $t \in \{1, 10\}$ are the co attention representation of history and predict tracks respectively.

4.4 Combine Encoder Layer

Now that we have obtained the hidden representation u_t^H and u_t^P, self-attention representation as well s_t^H and s_t^P as the co-attention representation a_t^H and a_t^P for history and predict tracks respectively. We need to merge the information we obtained from history tracks and predict tracks. We choose to use bi-directional RNN to model the complete interaction by concatenating the three set of hidden vectors we get from Encoder Layer, Self Matching Attention Layer as well as Co Attention Layer and feed it as input to the bi-directional RNN to get a hidden representation on the complete listening history in one session:

$$i_t^H = [u_t^H, s_t^H, a_t^H] \tag{9}$$

$$i_t^P = [u_t^P, s_t^P, a_t^P] \tag{10}$$

$$i_t = [i_t^H, i_t^P] \tag{11}$$

$$h_t = BIRNN(i_{t-1}, i_t) \tag{12}$$

We use LSTM[6] as the choice of RNN encoder in this case. We get the hidden representation on the complete session history as h_t for $t \in \{1, 20\}$

4.5 Self Matching Attention Combination Layer

Although we already modeled the interactions between groups of tracks in session history and session predict respectively in the Self Matching Attention Layer, we only performed Self Matching Attention on the songs within the groups, so it's natural to apply Self Matching Attention on the combined sequence which includes both history and predict tracks on the result from Combine Encoder Layer h_t:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{13}$$

$$s_t = Attention(h_t, h_t, h_t) \tag{14}$$

s_t for $t \in \{1, 20\}$is the self matching attention representation of the entire session tracks.

4.6 Feed Forward Layer

Then we concatenate the representation we got from the Combine Encoder Layer h_t with the result we obtained from the Self Matching Attention Combination Layer s_t. Then we get a session length × hidden dimension matrix. Then we apply feed forward layer to reduce the hidden dimension matrix to 1 to get the Logits for the classification of the predict tracks. We will get 20 logits from the final FeedForward layer including the padding l_t for $t \in \{1, 20\}$, then we take the last 10 elements in the vector and feed to the Loss Function since it corresponds to the predict tracks and its padding.

$$l_t = FeedForward([h_t, s_t]) \tag{15}$$

4.7 Loss Function

We use the average binary cross entropy loss of the targets as our final loss, since we are trying to predict skip information for all the songs in the predict session. We compute binary cross entropy loss for each of the predict tracks and take an average as the loss for one session. We get the logits corresponding to the predict songs as l_t for $t \in \{1, 10\}$. Also we have m_t from batching which is the masking for predict tracks and padding. m_t is 0 if it's a padding track and 1 if it's a predict track. And we have y_t as the target prediction.

$$Loss(l, m, y) = \sum_{t=1}^{10} (y_t * log(l_t) + (1 - y_t) * log(1 - l_t)) * m_t \tag{16}$$

5 ENSEMBLE

We train a few different deep learning approaches which are small variations of the architecture explained above. One example is that we change the encoder layer to use two stacked Bi-directional LSTM instead of just one Bi-directional LSTM layer. Another example is we change the dimension of the hidden representation used in the LSTMs.

We use a simple ensemble strategy by basically averaging the prediction logits from different approaches and this give us a small gain in the average accuracy on both validation set as well as on the leaderboard.

6 EXPERIMENTS

All experiments are conducted on a single Ubuntu Linux server with two Intel Xeon(R) E5-2620 CPUs, 256GB RAM, and Titan Xp GPU. We use a training batch size of 48 and inference batch size of 1000 which can be fit on a single Titan Xp GPU with 12GB memory. We use Adamax[8] optimizer with a constant learning rate of 0.001 and a gradient clipping value of 10. The hidden dimension we used in Bi-directional RNN is 100. We use a dropout[12] rate of 0.1 for all RNN outputs and 0.2 for Feed Forward Layer outputs. Also, we pad both the history and predict to be 10 songs since the maximum number of songs in each is 10 so that each batch of RNN can have same number of length which is easier to process later.

We conducted an simple ablation study on the architecture to see how effective is the attention mechanism. We removed all the intermediate layers and only keep one Bi-directional RNN encoder while padding all the predict features with 0. Then we feed the output of the Bi-directional RNN directly through the feed forward layers and compute the loss on its result. The results we obtained is in Table 1. We can see adding attention effectively helps the modeling of skip behaviour.

	Average Accuracy	First Prediction Accuracy
Our method	0.636	0.803
RNN baseline	0.605	0.693

Table 1: Comparison of our method with baseline approach

	Average Accuracy	First Prediction Accuracy
ekffar	0.651	0.812
DIKU-IR	0.641	0.807
mimbres	0.637	0.804
learner6	**0.636**	**0.803**
Adrem Data Lab	0.613	0.794

Table 2: Test set Leaderboard results for the Top 5 Teams

Among the 660 given training files, we use the first 600 files as training set and next 20 files as validation set. After training two entire epochs on the training set, our model can achieve 0.630 average precision on the local validation set and 0.633 on the leaderboard. Ensemble models with slightly different hyperparameters give us the final score of 0.636 on the leaderboard. The complete private test leaderboard results are shown in Table 2.

7 CONCLUSION

In this paper, we describe our end to end approach for the 2019 ACM WSDM Spotify Skip Prediction Challenge. This Challenge gives a problem of predicting whether the given songs are going to be skipped or not give a history of track and session information. We use a combination of Bi-directional RNN and Attention to model both history tracks and predict tracks interactions, then we apply Bi-directional RNN and Attention to model global interactions again after concatenating the history and predict tracks. This approach allows us to model complex interactions within the history and predict groups as well as global session interactions. Our single model achieves highly competitive results with an end-to-end deep learning architecture.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]* (Sept. 2014). http://arxiv.org/abs/1409.0473 arXiv: 1409.0473.

[2] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. 2019. The Music Streaming Sessions Dataset. In *Proceedings of the 2019 Web Conference*. ACM.

[3] Ruining He and Julian McAuley. 2016. Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Barcelona, Spain, 191–200. https://doi.org/10.1109/ICDM.2016.0030

[4] BalÃązs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18* (2018), 843–852. https://doi.org/10.1145/3269206.3271761 arXiv: 1706.03847.

[5] BalÃązs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv:1511.06939 [cs]* (Nov. 2015). http://arxiv.org/abs/1511.06939 arXiv: 1511.06939.

[6] Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[7] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, Pisa, Italy, 263–272. https://doi.org/10.1109/ICDM.2008.22

[8] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Dec. 2014). http://arxiv.org/abs/1412.6980 arXiv: 1412.6980.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[10] Tomas Mikolov, Martin KarafiÃąt, LukÃąs Burget, Jan CernockÃ¡, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web - WWW '01*. ACM Press, Hong Kong, Hong Kong, 285–295. https://doi.org/10.1145/371920.372071

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313

[13] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. *arXiv:1809.07426 [cs]* (Sept. 2018). http://arxiv.org/abs/1809.07426 arXiv: 1809.07426.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]* (June 2017). http://arxiv.org/abs/1706.03762 arXiv: 1706.03762.