Using Latent Semantics of Playlist Titles and Descriptions to Enhance Music Recommendations

Yun Hao and **J. Stephen Downie**
School of Information Sciences
University of Illinois at Urbana-Champaign
{yunhao2, jdownie}@illinois.edu

Abstract

Music playlists, either user-generated or curated by music streaming services, often come with titles and descriptions. While crucial to music recommendations, leveraging titles and descriptions is difficult due to sparsity and noise in the data. In this work, we propose to capture useful latent semantics behind playlist titles and descriptions through proper clustering of similar playlists. In particular, we clustered 20,065 playlists with both titles and descriptions into 562 groups using track vectors learned by word2vec model on over 1 million playlists. By fitting a Naive Bayes model on titles and descriptions to predict cluster membership and using the cluster membership information for music recommendations, we present a simple and promising solution to the cold-start problem in music recommendation. We believe that when combined with other sources of features such as audio and user interaction, the proposed approach would bring further enhancement to music recommendations.

1 Introduction

In the theory of Information Retrieval (IR), users formulate "queries" using natural language to express information needs to IR systems (Baeza-Yates and Ribeiro-Neto, 1999), and the query terms make up a sparse semantic space. Similarly, users on music streaming platforms, when creating new playlists, express their needs for music by providing playlist titles and descriptions. These playlist titles and descriptions also make up a highly sparse corpus, and capturing useful latent semantics from such sparse space for making music recommendations is challenging. In fact, using playlist titles for music recommendations can even worsen the performance of recommender systems (Zamani et al., 2018).

In this work, we present that through proper clustering of similar playlists, titles and descriptions can be effectively employed for making more accurate music recommendations. Specifically, two stages are included in this work: in Stage 1, track sequences in playlists are used to embed playlists and tracks into a latent embedding space using word2vec (Mikolov et al., 2013a), and agglomerative clustering is implemented on playlist embeddings to form clusters of similar playlists; in Stage 2, we fit a multinominal Naive Bayes model on words from playlist titles and descriptions to predict cluster membership and use the cluster membership information to make music recommendations. Details of the two stages are in Section 4 and Section 5, respectively. In Section 6, we evaluate the proposed recommending strategy by the task of making music recommendations given only playlist titles and descriptions, with several baseline models and strategies compared.

2 Related Work

Works have been done to capture hidden semantics from playlist titles for music recommendation. Pichl et al. (2015) formed clusters of playlist titles and interpreted each cluster as a latent music listening context for making music recommendations. The authors expanded the corpus by adding synonyms and hypernyms using WordNet (Miller, 1995) to deal with sparsity. The same authors later built on this work and formed situational clusters using selected playlist titles that contain activities and other descriptors (e.g., season, events) to improve music recommender systems (Pichl and Zangerle, 2018). One of the ACM RecSys Challenge 2018[1] tasks is to predict tracks in playlists given titles only. Approaches adopted

[1]http://www.recsyschallenge.com/2018/

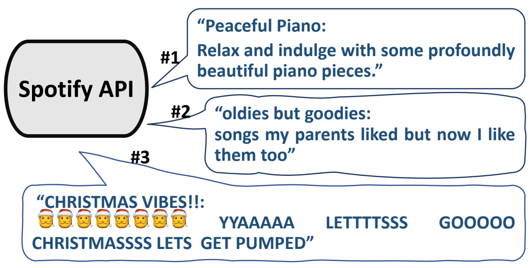
Spotify API
#1
"Peaceful Piano:
Relax and indulge with some profoundly beautiful piano pieces."
#2
"oldies but goodies:
songs my parents liked but now I like them too"
#3
"CHRISTMAS VIBES!!:
YYAAAAA LETTTTSSS GOOOOO CHRISTMASSSS LETS GET PUMPED"

Figure 1: Example playlists from the datasets

by the top performing teams include matrix factorization on (playlist, track)-title co-occurrence matrix (Volkovs et al., 2018), character-level convolutional neural network to embed playlist titles (Yang et al., 2018), and using playlist titles as queries to pseudo-documents generated for each track by concatenating all the titles of the playlists that contained a particular track (Kallumadi et al., 2018).

Starting from the intuition that interpreting playlist titles and descriptions as plain text is not effective enough, we propose to fit a language model on titles and descriptions based on some "intermediate" information so that the "intermediate" information can guide us towards a better understanding of the language behind playlist generation.

3 Data

The datasets we used include the Million Playlist Dataset (MPD) released by Spotify for ACM RecSys Challenge 2018 as well as 1,417 playlists curated by Spotify collected via Spotify API[2]. The MPD is further divided into two subsets, one with playlists with descriptions (D_1), and one with playlists without descriptions (D_2). Usage of each subset in this work will be detailed in later sections. To get more quality titles and descriptions data, we also collected 1,417 playlists curated by Spotify (D_3). Table 1 shows the summary of the datasets. Three example playlists from the datasets are shown in Figure 1. Playlist #1 is a curated playlist on Spotify, while playlist #2 and #3 are user-generated playlists that have been made public on Spotify.

4 Clustering of Playlists

4.1 Latent Representations of Playlists

Word embedding approaches, such as word2vec (Mikolov et al., 2013a) and GloVe (Pennington et al., 2014), provide an effective way to learn dense vector representations of words by leveraging word co-occurrence information. By treating playlists as the equivalent of sentences, and tracks as the equivalent of words, similar to (Kallumadi et al., 2018), we propose to apply word embedding approach to learn a dense vector representation for each of the unique track IDs and represent each playlist by aggregating its track embedding vectors. For learning the track embedding vectors, the word2vec model was chosen and the reasons are as follow: 1) with the continuous bag-of-word model of word2vec, ordering information is discarded and is more preferred in the setting of making "static" recommendations, as opposed to playlist continuation task in music listening session; 2) the linearity of the vector operations is claimed to weakly hold for the addition of several vectors by word2vec (Mikolov et al., 2013c), so aggregating track vectors should yield a meaningful representation of playlists.

[2]https://developer.spotify.com/documentation/web-api/

Dataset		Size
D_1	MPD w/ descriptions	18,760
D_2	MPD w/o descriptions	981,240
D_3	Spotify Curated Playlists	1,417

Table 1: Summary of the datasets

All the 1,001,417 playlists in the dataset were used for learning the latent representations of playlists so that the learning process can make the most of the available data. In total, over 64 million unique tracks were fed to the word2vec model, and after subsampling (Mikolov et al., 2013b) we learned 50-dimensional latent representations of 600,501 tracks.

With the learned track vectors, each playlist in D_1 and D_3 (20,177 playlists in total) is represented as the average of its track vectors. Because not all tracks in the dataset has a dense vector, there are 112 playlists whose tracks are all absent from the latent embedding space. These playlists are discarded, leaving 20,065 playlists with descriptions in the dataset.

4.2 Clusters of Similar Playlists

With latent vector representations of playlists, groups of similar playlists can be formed using clustering algorithms. Among options such as K-means clustering and modularity-based clustering (Clauset et al., 2004), we found agglomerative clustering using cosine distances normalized for

#	Size	Top 10 words with highest BiTF from the cluster
1	628	oldies,80s,goodies,classics,soul,love,school,70s,dad,60s
2	597	rap,fire,hype,litty,chill,af,bangers,gang,trap,party
3	458	throwback,throwbacks,childhood,nostalgia,disney,2000s,school,tbt,middle school,bops
4	457	rock,classic,classics,classic rock,oldies,dad,roll,70s,80s,school
5	433	edm,house,electronic,dance,dubstep,chill,trap,gaming,bass,drops

Table 2: Top 5 largest clusters of similar playlists, presented by top 10 words from each cluster

each playlist yields the best result. In total, 580 clusters are formed in the data, including 18 singletons (clusters with 1 playlist). We removed the singletons to uncover general patterns in the data, leaving 562 clusters of similar playlists. Table 2 shows a summary of the top 5 largest clusters. From the table, it can be shown that playlists in the same cluster share something similar – genre, event, mood, etc..

5 Language Modeling on Playlist Titles and Descriptions via Naive Bayes

Given the clusters of playlists, we fit a multinominal Naive Bayes model using words from playlist titles and descriptions. Naive Bayes model was chosen because it is fast and accurate enough to serve as a proper baseline for text classification, and that with multinomial Naive Bayes, each cluster can be represented as a unigram language model which allows us to get more insights into the language used in playlist generation.

Before fitting the model, a series of data cleaning and preprocessing steps such as normalizing emojis was implemented on the text data. We omit the details for brevity here.

We chose binary term frequency (BiTF) as the text feature to extract from titles and descriptions because BiTF usually works better with short and sparse text. Bigrams were also included so that frequently mentioned artist names such as "Ed Sheeran" can be preserved. We further pruned the vocabulary with a minimum term frequency of 3, which yields a vocabulary of 5,487 tokens.

5.1 Model Details

Stratified sampling was implemented to split the playlists with descriptions (i.e., D_3) into training (19,044, 95%) and test set (1,003, 5%). We then fit a Naive Bayes model with Laplace smoothing on the training set.

6 Evaluation

6.1 Experimental Setup

Evaluation is done by the task of making music recommendations given playlist titles and descriptions. The baselines to compare are word2vec word embeddings trained on the training set, pretrained GloVe word vectors on 2 billion tweets (both 50-dimensional and 200-dimensional), as well as the top-performing approach based on matrix factorization to dealing with cold-start playlists from the RecSys Challenge 2018 (vl6[3]). Naive approaches that either recommend popular tracks or random tracks are also included as baselines.

For all approaches except vl6 and the two naive approaches, one of two recommending strategies was employed according to the type of text features:

- Cluster-based: predict C potential clusters and recommend top tracks from the clusters by track frequencies weighted by normalized distances between the query playlist and the predicted clusters centers.
- Similarity-based: retrieve S similar playlists and recommend top tracks from the playlists by track frequencies weighted by normalized distances between the query playlist and the similar playlists.

We set $C = 5$ and $S = 5 \times 11 = 55$, where 11 is the median size of clusters in the training set, for fair comparison. Each model will return 500 candidates for evaluation.

F1, NDCG, R-precision, and R-artist are reported. F1 score measures the retrieval quality of the approaches while NDCG measures the ranking quality. R-artist is the same R-precision metric used for RecSys Challenge 2018 (Zamani et al., 2018), where artist matches were partially rewarded even if the predicted track was incorrect. For brevity, we only describe the standard

[3]https://github.com/layer6ai-labs/RecSys2018/

R-precision here. Let R be the set of ground truth tracks for a playlist, and T be the set of first $|R|$ tracks returned by the system. R-precision is then calculated as:

$$R\text{−}precision = \frac{|T \cap R|}{|R|} \qquad (1)$$

6.2 Results and Analysis

Table 3 summarizes the evaluation results. Clearly, our proposed cluster-based strategy yields the most satisfying result. Of all the baseline approaches, similarity-based BiTF works the best, confirming that BiTF is a very effective text feature that works well for short and sparse text.

In the following two subsections, we present two examples to illustrate how the clusters may have helped with making more accurate recommendations.

Model	$F1$@100	$F1$@500	NDCG@100	NDCG@500	R-prec	R-artist
			Cluster-based			
BiTF	**0.0735**	**0.0524**	**0.0632**	**0.0652**	**0.0692**	**0.0717**
			Similarity-based			
BiTF	0.0713	0.0463	0.0623	0.0637	0.0663	0.0692
word2vec	0.0663	0.0432	0.0578	0.0589	0.0621	0.0641
GloVe-50d	0.0453	0.0309	0.0392	0.0400	0.0421	0.0436
GloVe-200d	0.0489	0.0339	0.0428	0.0438	0.0457	0.0472
			Others			
vl6	0.0661	0.0431	0.0588	0.0601	0.0624	0.0658
Popular	0.0381	0.0351	0.0308	0.0320	0.0337	0.0350
Random	0.0002	0.0003	0.0002	0.0002	0.0002	0.0003

Table 3: Evaluation results

6.3 Neighboring Clusters

One of the reasons why the clusters of similar playlists can help with making recommendations is that the clustering is effective to group similar playlists together. Figure 2 shows the 5 nearest neighbor clusters of the query cluster "Christmas" (shown in bold). According to the top words from each clusters, all the 5 neighbors seem to be relevant to "Christmas"; thus it is very likely that tracks from the neighboring clusters are good candidates to recommend given a query playlist comes from the "Christmas" cluster.

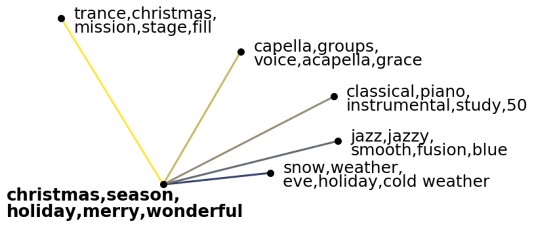

Figure 2: Neighboring clusters of "Christmas". Edge lengths indicate distances from the query cluster to its neighbors.

6.4 Candidates with Diversity

By observing the behavior of the Naive Bayes model, it is interesting to see that when no additional information is provided to a query word, the Naive Bayes can recall more diverse potential candidates, which may benefit the recommender system. For example, in Figure 3 we show the top 5 most likely clusters from which word "study" is generated. Of the 5 candidate clusters, each indicates a different "group" or "genre" and each can be relevant to the query word "study" according to different user tastes or preferences – some people may prefer classical music or movie soundtracks when study, while some may prefer electronic dance music (edm) to stay energetic. When the recommender system has no additional knowledge about the user's preference, it may be a better strategy to provide wider options for the user to choose from.

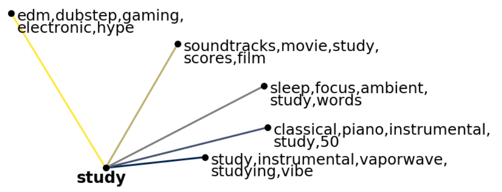

Figure 3: Top 5 clusters returned for query word "study". A shorter edge length indicates a higher probability that the query belongs to the cluster.

7 Conclusion and Future Work

In this work, we present that through proper clustering of similar playlists, titles and descriptions can be effectively employed for making more accurate music recommendations. There are several future directions to extend this work. First, it is worth exploring how the method can be combined with audio signals and user interaction data to further benefit music recommender systems. Second, other aggregation of the track embeddings for playlists than averaging can be explored for making even more accurate recommendations. Lastly, evaluating the quality of music recommendations without user feedback data may not be accurate, especially when novelty and serendipity (Schedl et al., 2014) is preferred by users. Therefore, this work can benefit from some other datasets with feedback information available as ground truth.

References

Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., USA.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E*, 70(6):066111.

Surya Kallumadi, Bhaskar Mitra, and Tereza Iofciu. 2018. A line in the sand: Recommendation or ad-hoc retrieval? *arXiv preprint arXiv:1807.08061*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Martin Pichl and Eva Zangerle. 2018. Latent Feature Combination for Multi-Context Music Recommendation. In *2018 International Conference on Content-Based Multimedia Indexing (CBMI)*, pages 1–6.

Martin Pichl, Eva Zangerle, and Günther Specht. 2015. Towards a Context-Aware Music Recommendation Approach: What is Hidden in the Playlist Name? In *15th IEEE International Conference on Data Mining Workshops (ICDM 2015)*, ICDM 15, pages 1360–1365, Atlantic City. IEEE.

Markus Schedl, Emilia Gómez Gutiérrez, and Julián Urbano. 2014. Music information retrieval: Recent developments and applications. *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261*.

Maksims Volkovs, Himanshu Rai, Zhaoyue Cheng, Ga Wu, Yichao Lu, and Scott Sanner. 2018. Two-stage model for automatic playlist continuation at scale. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–6.

Hojin Yang, Yoonki Jeong, Minjin Choi, and Jongwuk Lee. 2018. Mmcf: Multimodal collaborative filtering for automatic playlist continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018*, pages 1–6.

Hamed Zamani, Markus Schedl, Paul Lamere, and Ching-Wei Chen. 2018. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. *arXiv preprint arXiv:1810.01520*.