MULTI-TASK LEARNING OF GRAPH-BASED INDUCTIVE REPRESENTATIONS OF MUSIC CONTENT

Antonia Saravanou[1†] **Federico Tomasi**[2] **Rishabh Mehrotra**[2] **Mounia Lalmas**[2]
[1] University of Athens, [2] Spotify
antoniasar@di.uoa.gr, {federicot,rishabhm,mounial}@spotify.com

ABSTRACT

Music streaming platforms rely heavily on learning meaningful representations of tracks to surface apt recommendations to users in a number of different use cases. In this work, we consider the task of learning music track representations by leveraging three rich heterogeneous sources of information: (i) organizational information (e.g., playlist co-occurrence), (ii) content information (e.g., audio and acoustics), and (iii) music stylistics (e.g., genre). We advocate for a multi-task formulation of graph representation learning, and propose MUSIG: MUlti-task Sampling and Inductive learning on Graphs. MUSIG allows us to derive generalized track representations that combine the benefits offered by (i) the inductive graph based framework, which generates embeddings by sampling and aggregating features from a node's local neighborhood, as well as, (ii) multi-task training of aggregation functions, which ensures the learnt functions perform well on a number of important tasks. We present large scale empirical results for track recommendation for the playlist completion task, and compare different classes of representation learning approaches, including collaborative filtering, word2vec and node embeddings, as well as graph embedding approaches. Our results demonstrate that considering content information (i.e., audio and acoustic features) is useful and that multi-task supervision helps learn better representations.

1. INTRODUCTION

Recent advancements in recommendation technology [1–7] have fueled music listening on on-demand music streaming apps (e.g., Spotify, Pandora, Apple Music). Playlists form the backbone of how music is consumed, with users relying on curated or user generated playlists to discover and consume music from a massive pool of millions of songs. Personalization models built for selection of tracks, generation of playlists and subsequent recommendation of playlists to users, rely heavily on representing tracks in a meaningful way, to best capture the various intricacies and differences across musical tracks.

When learning track representations, one can leverage various types of heterogeneous information encoded in music data to learn track representations that benefit downstream tasks of music recommendation: (i) *organizational information*: tracks organized into playlists; (ii) *content information*: audio and acoustic features extracted from tracks; and (iii) *musical stylistics*: musical domain characteristics like music genres. Further, we note that such representations are used by system designers for many different downstream tasks, e.g., track recommendation for playlist completion, ranking tracks within a playlist and suggesting tracks in sequential sessions (i.e., track radios). Unfortunately, the learnt representations are often ill-suited for such downstream tasks, because of mismatch between the original learning and downstream task. Instead, training the representation learning system on multiple, complementary tasks would enable learning richer representations, allowing for an increased adoption of the representations for a variety of newer downstream tasks, which is important in an industrial setting.

Motivated by the above aspects, we propose a MUlti-task based Sampling and Inductive Graph learning approach (MUSIG) for learning track representations, that combines information from heterogeneous sources and benefits from supervision signals from a number of tasks. Instead of training a distinct embedding vector for each node, following recent advancements in graph based learning [8], we train a set of aggregator functions. These functions aggregate information from different nodes in the local neighborhood, based on various settings of search depth, and are trained via pairwise multi-task supervision. For each pair of nodes, we consider three tasks: (i) playlist co-occurrence, (ii) genre prediction, and (iii) regression of audio and acoustic properties of the tracks.

Furthermore, the trained aggregator functions afford the inductive ability to the model. Indeed, we can generate embeddings for unseen nodes by applying the learned aggregation functions. Finally, jointly leveraging organizational, content and stylistics information helps us cover individual track level information (e.g., audio/acoustic features) as well as information from across various groupings of track such as music stylistics based grouping (e.g., genres) and user consumption based grouping (e.g., playlists).

We present a case-study on music recommendations

[†]This work was done while the first author was an intern at Spotify.

and conduct large scale analysis to compare different techniques across several qualitative and quantitative measures. We make a number of contributions, including algorithmic and qualitative insights of music data at scale. Our findings suggest that extracting audio and acoustic features from music content is useful, and the addition of such content attributes better drives the representation of the tracks, especially when large amount of consumption data is not available, e.g., when launching in new markets. Furthermore, we show that training the model on multiple tasks results in performance improvements and enables us to learn more generalizable representations. We contend that our findings have implications on the design and development of representation learning approaches not only for music, but also for other types of data.

2. RELATED WORK

Music representation learning. Recent works on music representation learning rely on deep neural networks to generate music embeddings, using various groups of features: sequence of notes [9], music signals [10], pitch sequences, temporal dependencies [11, 12], and artist features [13]. Using music signals/notes from a track to generate track embeddings has shown various degrees of success, and research in this area is still ongoing. In this work, we use similar features and compare how graph representation learning models can incorporate them.

Word2Vec style embeddings. Text representations have been extensively studied in the last years. Word embeddings refer to low-dimensional real-valued vector representations for each term in the input vocabulary. Word2Vec [14] and GloVe [15] are two well-known word embedding algorithms that learn embedding vectors based on the idea that similar words appear in similar contexts. Word embeddings have been applied in various contexts, such as item recommendations [16–18], music recommendations [19] and query modeling and expansion [20, 21]. Word2Vec predicts an item based on the context (e.g., in e-commerce applications, items frequently bought together). We also use Word2Vec to generate track embeddings based on the idea that tracks appearing together in a playlist should be closer in the embedding space. However, these approaches are limited as they only consider the sequence in which the items appear, and could not include additional information on the actual content.

Graph based embeddings. In recent years, variations of Word2Vec working on graph structured data were developed. Examples include Deepwalk [22] and Node2Vec [23], which generate random walks in a specified neighborhood of the target node, to compute the node embedding. Significant advancements of learning on graph structures for recommendation applications include GraphSAGE [8], PinSAGE [24], PinnerSAGE [25], IntentGC [26], MEIRec [27]. Most of these methods are based on Graph Convolutional Networks (GCNs) [28], which combine the graph information from the neighborhood of a node (graph structure) and node features (content information) in the creation of the embeddings. Graph-based embeddings are especially useful when nodes and edges have different types [29–32]. Graph representation learning exploit the structure and the features of the data. However, the embeddings are learnt by optimizing the model on a single task, which makes the embeddings not easily generalizable to additional downstream tasks.

Table 1: Track features (all values are normalized to be between 0–1).

Feature	Description
Genre	One-hot encoded vector of the top-50 popular genres
Popularity	2-dimensional vector with the global and the region popularity
Audio	42-dimensional vector that includes: *danceability*, *energy*, *liveness*, *acousticness*, *loudness*, *tempo*, *instrumentalness*, *valence*, etc
Acoustic	8-dimensional vector, corresponding to audio characteristics

3. MULTI-TASK GRAPH EMBEDDINGS

Recommender systems rely heavily on learning meaningful representations of users and content, to offer personalized recommendations piquing users' interest. With an explicit focus on streaming music platforms, we briefly discuss few important characteristics of representation learning and describe our proposed method, MUSIG, for node representation learning with multi-task supervision.

3.1 Music Graph Data

We work with data consisting of track and playlist information from Spotify, a large music streaming platform. Tracks are organized into *playlists*. Playlists provide information on how users *organize* their music. We represent playlist-track information as a graph and create a (weighted) homogeneous graph containing all tracks in our dataset. Let the graph be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes $\mathcal{V}$ be the tracks and edges $\mathcal{E}$ be the connections between tracks co-appearing in the same playlist. We set the weight of an edge to be the number of distinct playlists in which the connected tracks co-appear. We keep all edges with weight ≥ 10. Our graph contains 5.2M edges and 15.9K nodes, from a collection of 95K playlists (albums and movies).

Following the approach outlined in [33], we extract a number of features from each track (Table 1), and refer to them as *content features*. While detailed description of this process is beyond the scope of current work, we briefly summarize the methods employed to extract these features. Following the approach outlined in [33], we extract various content features from the music recording of the track, including acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, valence and tempo, etc., which we refer to as *audio features*. Furthermore, we train a deep neural model on the music recording of each track (via 30-second windows) for a binary classification task of playlist co-occurrence and we use the last layer projected to 8 dimensions as the *acoustic features* of the track.

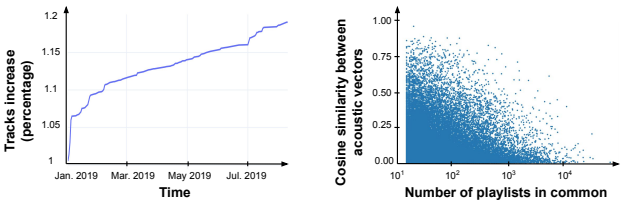

Figure 1: (Left) Percentage of tracks per day in our sample. In 2019, there is an average 0.09% increase of the catalog every day, resulting in a 19% increase from the start of the year. (Right) Correlation between number of common playlists and acoustic similarity between a random sample of 50K pairs of tracks. The actual correlation is low (-0.029) hence the graph structure (playlists) and additional features (acoustic) are complementary.

3.2 Desired Characteristics

We identify few desirable characteristics for representation learning approaches and motivate their need for inclusion with brief supporting analysis. First, many industrial applications, especially in the music domain, require representations to be promptly available for new tracks. Figure 1 (left) plots the increase in new content on a daily basis; hundreds of tracks are added every day and learning their representations as early as possible is crucial for production machine learning systems.

Second, a good representation learning approach should leverage all available information, including both playlist co-occurrence and content features. Figure 1 (right) presents the relationship between playlist co-occurrence and acoustic features between randomly selected track pairs. For each pair, we compute the similarity between tracks using their acoustic features, and plot those against the percentage of playlists in which the two tracks co-occur. We observe very low correlation between the two modalities (-0.029), and low density of the scatter-plot in the high similarity co-occurrence region, which highlights that these modalities capture different information.

Finally, the learnt track representations are employed in a number of use cases across multiple product features. Usually, methods that compute representations are optimized for a specific task. We hypothesize that training the representation learning modules on multiple tasks would enable learning generic representations which would help in a wide variety of downstream recommendation tasks.

3.3 MUSIG Overview

The key idea behind MUSIG is multi-task supervision of neighborhood aggregator functions that aggregate information from a node's neighborhood. The idea of exploiting node's neighborhood has shown to provide state-of-the-art results [8, 34]. MUSIG adopts a multi-task based learning of aggregator functions, which enables it to learn parameters of the functions based on feedback from multiple, complimentary tasks. Specifically, the algorithmic computations performed by MUSIG are divided into two key steps: (i) **Neighborhood Aggregator Step**, which generates embeddings by aggregating information from different nodes in multiple hops away from a given node (based on search depth), and (ii) **Multi-Task Supervision Step**,

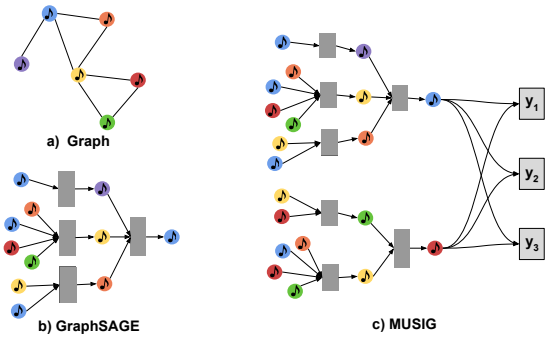

Figure 2: (a) Input graph, each node represents a different track and it appears with a different color. (b) GraphSAGE model, consecutively aggregating the *blue* node's neighbors, to generate the node embedding. (c) Our multi-task model, MUSIG, computing *blue* and *red* nodes' embeddings while optimizing for three different loss functions and tasks.

which trains the parameters of the aggregation functions by jointly predicting multiple tasks, and back-propagates the combined losses to the aggregator function parameters.

3.4 Neighborhood Aggregator Step

Unlike traditional representation learning approaches, which train a specific embedding for each item in an end-to-end neural model, MUSIG relies on local neighbourhood information and learns aggregator functions that can digest local information to obtain a representation of any given node (Figure 2).

For any depth d, the aggregator function recursively aggregates information from all nodes in the d-depth neighborhood of a node, and uses a set of weight matrices $\mathbf{W}^k$, $\forall k \in \{1, \ldots, K\}$ to propagate information between layers arising for each depth. At each iteration, or search depth, the nodes aggregate information from their local neighbors, and as this process iterates, the nodes incrementally gain more information from further reaches of the graph.

We follow an iterative approach to aggregate information. First, each node $v \in \mathcal{V}$ aggregates the representations of the nodes in its neighborhood $\mathcal{N}(v)$, $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$, into a single vector $\mathbf{h}_{\mathcal{N}(v)}^{k-1}$:

$$\mathbf{h}_{\mathcal{N}(v)}^{k-1} \leftarrow \textsc{Aggregate}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \quad (1)$$

The representations at step k depend on the representations generated at the previous step $k-1$, with representations at $k = 0$ being encoded by the default node features provided to the graph. By incorporating node features in the learning algorithm, the model simultaneously learns the topological structure of each node neighborhood as well as the distribution of the node features in the neighborhood. To extract all adjacent nodes, we uniformly sample a fixed-size set of neighbors and thereby keep the computational footprint of each batch fixed. Following [34] we use a permutation invariant aggregator function which implements the mean operator, taking the element-wise mean of the vectors in $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$:

$$\textsc{Aggregate}_k(\mathbf{h}_v^{k-1}) \leftarrow \textsc{Mean}(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \quad (2)$$

We then concatenate the node's current representation, with the aggregated neighborhood vector. This concatenated vector is fed through a fully connected layer with

nonlinear activation function, which transforms the representations to be used at the next step of the algorithm (i.e., $\mathbf{h}_v^k, \forall v \in V$):

$$\mathbf{h}_v^k \leftarrow \sigma\Big(\textsc{Concat}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\Big) \quad (3)$$

After all iterations, the final representations output at depth K is denoted as $\mathbf{s}_v \equiv \mathbf{h}_{\mathcal{V}}^K$, which is used in the next step during training.

3.5 Multi-Task Supervision

The performance of representations learnt by MUSIG model relies heavily on how well the aggregator functions are trained and what tasks they are trained on. Most graph-based representation learning approaches are trained only on link prediction tasks, and hence learn function parameters only to do well on link prediction task. We hypothesize that training these parameters in a multi-task setting would make the parameters (and in turn, output representations) generalizable across multiple downstream applications. Learning under multi-task supervision offers various benefits, including imparting inductive bias via auxiliary tasks, which cause the model to prefer hypotheses that explain more than one task. This improves generalization by sharing the domain information between complimentary tasks, which is achieved by using a shared representation to learn multiple tasks — what is learned from one task can help learn other tasks.

Considering the output from the previous step, we denote by s_i a track from the input space S and a collection of task spaces $\{Y^t\}_{t\in[T]}$. To train the aggregator functions in a multi-task learning setup, we consider large sample of i.i.d. data points $\{\langle s_i, s_{j\in I(i)}\rangle, y_i^1, y_i^2 ..., y_i^T\}$, where $\langle s_i, s_{j\in I(i)}\rangle$ represents a pair of nodes (i.e., tracks) with $s_{j\in I(i)}$ being a track derived either from neighborhood of $s_i, \mathcal{N}(s_i)$, or negatively sampled from elsewhere. T is the number of tasks, N is the number of such node pairs sampled, and y_i^t is the label of the t-th task for the i-th track pair. Essentially, for each pair of tracks sampled from the graph, we consider labels obtained via different tasks. We further consider a parametric hypothesis class per task as $f^t(\langle s_i, s_{j\in I(i)}\rangle; \theta^{sh}) : S \rightarrow Y^t$, such that the parameters (θ^{sh}) are shared between tasks. We also consider task-specific loss functions $\mathcal{L}^t(\cdot,\cdot) : S^t \times S^t \rightarrow R^+$.

We employ an empirical risk minimization formulation of multi-task learning, and minimize the loss function:

$$\min_{\theta^{sh}} \sum_{t=1}^{T} c^t \mathcal{L}^t(\theta^{sh}) \quad (4)$$

for some static or dynamically computed weights c^t per task. In essence, losses from all tasks are combined into a single surrogate task via linear weighted scalarization, with each task having c^t weight. We perform grid search over the space of the parameters to estimate the final set of task weights used to report results. $\mathcal{L}^t(\theta^{sh})$ is the empirical loss of the task t, defined as:

$$\mathcal{L}^t(\theta^{sh}) \triangleq \frac{1}{N} \sum_i \mathcal{L}(f^t(\langle s_i, s_{j\in I(i)}\rangle; \theta^{sh}), y_i^t) \quad (5)$$

We apply a multi-task supervision based loss function to the output representations, $\mathbf{z}_u, \forall u \in \mathcal{V}$, and tune the weight matrices $\mathbf{W}^k$, $\forall k \in \{1, \ldots, K\}$, and parameters of the aggregator functions via stochastic gradient descent.

3.5.1 Identifying Supervision Tasks

The multi-task supervision of the model encourages nodes to have representations that help them solve all tasks for which the model is trained on. Our choice of supervision tasks is guided by our focus on leveraging the heterogeneous information encoded in music data, specifically, around three types of information (i) *organizational information*: tracks organized into playlists, (ii) *content information*: audio and acoustic features extracted from tracks, and, (iii) *musical stylistics*: musical domain characteristics like music genres. To have a representative set of tasks to train the model on, we select one task from each of these three categories of information:

1. **Playlist prediction:** binary classification task, where we predict whether or not the two tracks co-occur in same playlists. This task encapsulates the organization structure embedded in music playlists: two tracks sharing a playlist would make their representations similar to each other.

2. **Genre prediction:** binary classification task, where we predict whether two tracks belong to the same genre. Genres are useful for the categorization, and training on them ensures that tracks with the same genre have similar representations.

3. **Acoustic or audio similarity prediction:** regression task, where we encourage the embeddings to capture similarities in the music content space. We define track similarity as the inner product between acoustic or audio vectors and, thus, force the learnt space to encode music characteristics. This training task enforces representations to rediscover audio and acoustic distances between the tracks.

To better highlight that these tasks contribute heterogeneous information during representation learning, we compute the label correlation across them, and observe that the playlist prediction task has very little correlation with the other two tasks, and that the genre and acoustic similarity prediction tasks do share some commonality, but still differ enough (Figure 1). We use binary cross-entropy loss for the classification task and RMS loss for the regression task. Overall, the final loss function combines the losses from each of the three tasks as:

$$\mathcal{L}_{\text{Final}} = c_1\mathcal{L}_{\text{Playlist}} + c_2\mathcal{L}_{\text{Genre}} + c_3\mathcal{L}_{\text{AudioDist}}. \quad (6)$$

Importantly, unlike previous approaches, the representations s_i that are fed into this loss function are generated from the features contained within a node's local neighborhood, rather than training a unique embedding for each node (via an embedding look-up).

4. EXPERIMENTAL SETUP

To study the quality of the track embeddings, we conduct an empirical evaluation using data from Spotify, [1] a popular music streaming service.

[1] https://www.spotify.com/

Table 2: Results of the comparison methods using the 50% of playlist's tracks as seed track list.

Method	Features	HR	MRR	Prec@1	Prec@5	Prec@10	NDCG@1	NDCG@5	NDCG@10
CF	-	0.2715	**0.1350**	0.0119	0.0136	0.0119	0.0610	0.0890	0.0921
TRACK2VEC	-	0.5616	0.0174	0.0123	0.0218	0.0225	0.0123	0.0520	0.0813
NODE2VEC	-	0.4739	0.0128	0.0001	0.0005	0.0025	0.0001	0.0008	0.0065
TRACK2VEC	GENRE	0.5017	0.0137	0.0126	0.0133	0.0139	0.0126	0.0378	0.0575
TRACK2VEC	POPULARITY	0.5029	0.0138	0.0125	0.0140	0.0137	0.0125	0.0388	0.0564
TRACK2VEC	AUDIO	0.5020	0.0133	0.0156	0.0132	0.0128	0.0156	0.0368	0.0539
TRACK2VEC	ACOUSTIC	0.5014	0.0137	0.0145	0.0145	0.0140	0.0145	0.0425	0.0610
GRAPHSAGE	GENRE	0.4981	0.0132	0.0038	0.0129	0.0135	0.0038	0.0318	0.0499
GRAPHSAGE	POPULARITY	0.5038	0.0149	0.0239	0.0219	0.0180	0.0239	0.0624	0.0808
GRAPHSAGE	AUDIO	0.5914	0.0162	0.0143	0.0155	0.0154	0.0143	0.0420	0.0615
GRAPHSAGE	ACOUSTIC	0.5382	0.0163	0.0164	0.0217	0.0186	0.0164	0.0594	0.0795
MUSIG	GENRE	0.7077	0.0359	0.0278	0.0362	0.0393	0.0278	0.0724	0.1150
MUSIG	POPULARITY	**0.7505**	0.0358	0.0024	0.0193	0.0327	0.0024	0.0312	0.0776
MUSIG	AUDIO	0.7203	0.0324	**0.0661**	**0.0494**	**0.0415**	**0.0661**	**0.1240**	**0.1614**
MUSIG	ACOUSTIC	0.7305	0.0308	0.0412	0.0429	0.0372	0.0412	0.1047	0.1404

4.1 Downstream Task: Playlist Completion

Playlists are the backbone of how music content is consumed, with over one-third consumption resulting from user-generated playlists. [2] To assist users in selecting music for their playlists from the massive music catalog of million tracks, platforms rely on track recommendation services for playlist completion. Good track representations are crucial for the playlist completion task to be effective. Given a number of tracks in a playlist, our goal is to recommend related tracks. We construct this experiment in an offline fashion. We randomly select 10K playlists that have at least 40 tracks. We keep the top $x\%$ tracks to be the seedlist and we calculate the average embedding from those. Then, we mix the bottom $(100 - x)\%$ tracks with the same number of tracks from a random pool and we call them candidate tracks C. We calculate the *cosine similarity* of all pairs (z_s, z_c), where z_s is the average seedlist embedding and z_c is the embedding of candidate $c \in C$, and we rank them in descending order. Finally, we recommend the top $(100 - x)\%$ ranked tracks.

4.2 Baselines

We compare the proposed MUSIG with representative models from the three different classes of representation learning approaches: collaborative filtering, word2vec based models and graph embedding based model.

1. Collaborative Filtering. We compare with a collaborative filtering matrix factorization method trained with WARP loss [35], which aims at maximizing the rank of positive examples by repeatedly sampling negative ones.

2. Track2Vec. We compare a Word2Vec-based model, which aims at tracks co-occurring in the same playlists. We also concatenate the normalized features of Table 1 to the final embeddings of the model for fair comparison (TRACK2VEC-FEATURE).

3. Node2Vec. This approach takes into account random walks in the neighborhood of the node to create embeddings. Tracks connected by links in the graph are encouraged to be closer in the embedding space.

4. GraphSAGE. We compare with state-of-the-art graph representation learning approach: GraphSage [8]. This is a node representation model that produces embeddings based on the structure (i.e., node neighborhood), as well as the feature vector.

5. GraphSAGE-Feature. To investigate the performance of the model when adding features, we run the GraphSage model with all four different groups of features. To tune the model, we use the same parameters as before.

Our MUSIG/MUSIG-Feature model. Since the proposed MUSIG model affords multiple supervision, it is trained on genre prediction and audio/acoustic feature similarity tasks in addition to playlist co-occurrence task. We modulated the balance between the three tasks by empirically selecting the best performing triple (c_1, c_2, c_3) of Equation (6) across the following set: $\{(1, 1, 1), (1, 0, 0), (0.7, 0.1, 0.2), (0.4, 0.2, 0.4)\}$, to evaluate different properties of the single loss functions (i.e., when all losses weight the same; when only the first is non-zero, which is equivalent to a single task GraphSAGE).

4.3 Evaluation Metrics

We use four metrics to compare the approaches on the playlist completion task. Firstly, we include the standard versions for Precision at k (P@k) and Normalized Discounted Cumulative Gain at k (NDCG@k). We define *hit rate* (HR) as the fraction of tracks that were ranked in the top K candidates for a specific playlist P. This metric directly measures the probability that the track recommendations are the correct ones. In our experiments, K dynamically changes based on the size of the playlist, defined as:

$$K = \frac{100 - \text{size of the seedlist tracks}}{\text{actual size of the playlist}} \quad (7)$$

We also use (scaled) mean reciprocal rank (MRR), which takes into account the rank of the track u among recommended tracks for playlist P, defined as in [24]:

$$MRR = \frac{1}{n} \sum_{u \in P} \frac{1}{|R_u/10|}$$

where n is the number of all pairs and R_u is the rank of the track u among all recommended tracks for playlist P.

4.4 Comparison across approaches

We compare all representation learning approaches (CF, TRACK2VEC, NODE2VEC, GRAPHSAGE) to our ap-

[2] https://www.businessofapps.com/data/spotify-statistics/

proach, MUSIG, on the playlist completion task. In Table 2, we show the performance and we highlight the highest results. All results are calculated using the 50% of the playlist as seedlist. In Section 4.6, we discuss more on the performance of the models using all features. MUSIG trained on the Music Graph using the multi-task training and including the POPULARITY in the node attributes, outperforms all other models. More specifically, we observe that MUSIG–POPULARITY achieves the best HR=75.05%, while the best performance from any of the comparison models is achieved by GRAPHSAGE-AUDIO with an HR=59.14%. All results were tested for statistical significance and proven significant ($p \ll 0.01$).

4.5 Impact of training on multiple tasks

MUSIG improves the hit-rate score of the best existing model by 27%. This is an important indicator that embeddings generated by optimizing on multiple tasks are able to significantly improve the performance of the downstream task of playlist completion. Furthermore, this indicates that imparting the representations to perform well on genre classification and acoustic/audio distance similarity tasks enriches them further, thereby helping them achieve improved performance. We leave for future work further validation of other tasks for training, and the impact on other downstream tasks.

4.6 Impact of Features

We extensively investigate the importance of leveraging content features while learning embeddings. We select the best performing track (TRACK2VEC) and node embedding (GRAPHSAGE) models, and we evaluate the performance of these models and MUSIG using different groups of features (GENRE, POPULARITY, AUDIO, ACOUSTIC). For fair comparison, in TRACK2VEC we use aggregations of features and track embeddings. Table 2 shows the comparison between the three models, when trained with different features as node attributes.

First, we observe that TRACK2VEC achieves best performance when trained *without* the content features, which was expected since the model is designed to leverage only *organization* information. Second, we observe that in MUSIG the GENRE [3], AUDIO and ACOUSTIC features achieve lower hit rate scores, when compared to the POPULARITY features. An explanation is that all three features are already included as tasks in MUSIG, while popularity enriches further the learning phase. Intuitively, popularity does have a relationship to content, as it is related to more "mainstream" or "alternative" track types. However, in all groups of content features, our model outperforms all other models when trained with the same content features. The improvements of our model for each group of content features are: GENRE: 42%, POPULARITY: 49%, AUDIO: 22%, and, ACOUSTIC: 36% compared to the second best model. This highlights that the information contained by the audio and the acoustic features extracted from music recordings is indeed informative, and leveraging them while learning embeddings is useful.

[3] An explanation for the limited performance offered by genre could be our restriction to the most 50 popular genres (Table 1).

Table 3: Interplay between regression task and node features.

Regr. Task	Features	HR	MRR	Prec@10	NDCG@10
AUDIO	ACOUSTIC	**0.7305**	0.0308	0.0372	0.1404
AUDIO	AUDIO	0.5518	0.0186	0.0164	0.0581
ACOUSTIC	ACOUSTIC	0.7203	**0.0324**	**0.0415**	**0.1614**
ACOUSTIC	AUDIO	0.4339	0.0134	0.0085	0.0295

We also evaluate the models for the playlist completion task when using *all* four feature categories as node attributes. For MUSIG, we only add the POPULARITY feature in the node attributes, as information from the rest of the available features is already used in the tasks during the training. The best performing model is MUSIG, which achieves *HR*=0.75. The second best performance, GRAPHSAGE with all features as node attributes; achieves *HR*=0.50. This highlights that the multi-task learning in our MUSIG model achieves better results in all cases, for each feature separately but also in combination. This further motivates the use of multi-task learning methods for node classification tasks.

4.7 Variations in Multi-task Learning

To leverage music audio and acoustic properties, the proposed model not only considers them as node features, but also as regression tasks in the multi-task setting. We investigate the interplay between using such content information, and we compare multi-task models trained on both audio and acoustic features and tasks, results shown in Table 3. Among all combinations of tasks and features, using AUDIO similarity as the regression task with ACOUSTIC features as node features gives the best hit rate, while the reverse model, i.e., ACOUSTIC similarity task with AUDIO features, outperforms other combinations for all other metrics. AUDIO features capture rich information about music content. This further motivates the usefulness of hybrid representation learning approaches, that combines playlist organization information with content information.

5. CONCLUSION

We propose a multi-task graph-based learning model for music recommendation. Our method learns the track representations based on *content* features and *structural* graph neighborhoods, while the multi-task training is aggregating multiple functions and ensuring that representations are learnt based on supervision from multiple training tasks. The inductive aspect of MUSIG helps in reducing the wait time to surface new, fresh content in front of users from days to few hours, while the multi-task supervision enables the use of these representations for tasks that could not directly benefit from playlist co-occurrence only. Empirical results demonstrate the benefits of our method, wherein we show the value of the multi-task over the single-task learning. Furthermore, we show that extracting content features (such as audio or acoustic) improves the performance in existing methods, achieving the best improvements when those features are used in the multi-task setting.

6. REFERENCES

[1] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li, "Personalized recommendation via parameter-free contextual bandits," in *SIGIR '15*. ACM, 2015.

[2] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause, "Explore-exploit in top-n recommender systems via gaussian processes," in *RecSys '14*. ACM, 2014.

[3] E. Christakopoulou and G. Karypis, "Local item-item models for top-n recommendation," in *RecSys '16*. ACM, 2016.

[4] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *SIGKDD*. ACM, 2016.

[5] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, 2019.

[6] R. Katarya and O. P. Verma, "Efficient music recommender system using context graph and particle swarm," *Multimedia Tools and Applications*, 2018.

[7] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems with Applications*, 2017.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS '17*, 2017.

[9] R. de Volk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *ISMIR '18*, 2018.

[10] Y.-J. Luo and L. Su, "Learning domain-adaptive latent representations of music signals using variational autoencoders." in *ISMIR '18*, 2018.

[11] S. Lattner, M. Grachten, and G. Widmer, "Learning transposition-invariant interval features from symbolic music and audio," 2018.

[12] ——, "A predictive model for music based on learned interval representations," 2018.

[13] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," *arXiv:1710.06648*, 2017.

[14] T. Mikolov, I. Sutskever, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[15] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation. 2014," 2018.

[16] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," in *RecSys '18*. ACM, 2018.

[17] N. Ben-Lhachemi and E. H. Nfaoui, "Using tweets embeddings for hashtag recommendation in twitter," *Procedia Comput. Sci.*, 2018.

[18] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *ACM SIGKDD*. ACM, 2015.

[19] Z. Cheng, J. Shen, L. Zhu, M. S. Kankanhalli, and L. Nie, "Exploiting music play sequence for music recommendation." in *IJCAI*, 2017.

[20] P. D. Bruza and D. Song, "Inferring query models by computing information flow," in *CIKM '02*. ACM, 2002.

[21] F. Diaz, B. Mitra, and N. Craswell, "Query expansion with locally-trained word embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*. ACM, 2014.

[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD '19*. ACM, 2016.

[24] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*. ACM, 2018.

[25] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[26] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: A scalable graph convolution framework fusing heterogeneous information for recommendation," in *KDD '19*. ACM, 2019.

[27] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. ACM, 2019.

[28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[29] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[30] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[31] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, and H. Wang, "Dynamic heterogeneous graph embedding using hierarchical attentions," *Advances in Information Retrieval*, vol. 12036, p. 425, 2020.

[32] H. Xue, L. Yang, V. Rajan, W. Jiang, Y. Wei, and Y. Lin, "Multiplex bipartite network embedding using dual hypergraph convolutional networks," in *Proceedings of the Web Conference 2021*, 2021, pp. 1649–1660.

[33] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv:1709.05584*, 2017.

[35] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.