TISMIR

Artist Similarity for Everyone: A Graph Neural Network Approach

RESEARCH ARTICLE

FILIP KORZENIOWSKI
SERGIO ORAMAS
FABIEN GOUYON

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

Artist similarity plays an important role in organizing, understanding, and subsequently, facilitating discovery in large collections of music. In this paper, we present a hybrid approach to computing similarity between artists using graph neural networks trained with triplet loss. The novelty of using a graph neural network architecture is to combine the topology of a graph of artist connections with content features to embed artists into a vector space that encodes similarity. Additionally, we propose a simple and effective regularization method—*connection dropout*—which aims at improving results for long-tail artists, for which few existing connections are known.

To evaluate the proposed method, we use two datasets: the open OLGA dataset, which contains artist similarities from AllMusic, together with content features from AcousticBrainz, and a larger, proprietary dataset. We find that using graph neural networks yields superior overall results compared to state-of-the-art methods.

Beyond the overall evaluation, we investigate the effectiveness of the proposed model for long-tail artists. Such artists may benefit less from graph-based methods, since they typically have few known connections. We show that the proposed regularization approach clearly improves the performance for long-tail artists, without negatively affecting results for well-connected ones; it computes high-quality embeddings and good similarity scores for everyone.

CORRESPONDING AUTHOR:
Filip Korzeniowski
Pandora/SiriusXM, Oakland, USA
research@superfuzz.me

KEYWORDS:
Graph Neural Networks; Artist Similarity; Long-Tail; Deep Learning

TO CITE THIS ARTICLE:
Korzeniowski, F., Oramas, S., and Gouyon, F. (2022). Artist Similarity for Everyone: A Graph Neural Network Approach. *Transactions of the International Society for Music Information Retrieval*, 5(1), 129–140. DOI: https://doi.org/10.5334/tismir.143

1. INTRODUCTION

Music similarity has sparked interest early in the Music Information Retrieval community, (Aucouturier and Pachet, 2002; Ellis et al., 2002) and has since then become a central concept for music discovery and recommendation in commercial music streaming services.

There is however no consensual notion of *ground truth* for music similarity, as several viewpoints are relevant (Ellis et al., 2002). For instance, music similarity can be considered at several levels of granularity; musical items of interest can be musical phrases, tracks, artists, genres, to name a few. Furthermore, the perception of similarity between two musical items can focus either on (1) comparing *descriptive* (or *content-based*) aspects, such as the melody, harmony, timbre (in acoustic or symbolic form), or (2) *relational* (sometimes called *cultural*) aspects, such as listening patterns in user-item data, frequent co-occurrences of items in playlists, web pages, et cetera.

In this paper—which is an extended version of our previous work (Korzeniowski et al., 2021)—, we focus on *artist*-level similarity, and formulate the problem as a *retrieval* task: given an artist, we want to retrieve the most similar artists, where the ground truth for similarity is *cultural*. More specifically, artist similarity is defined by music experts in some experiments, and by the "wisdom of the crowd" in other experiments.

In this sense, we aim at bridging the semantic gap (Celma and Serra, 2008) between content (music) and context (culture). Connecting these two disparate views is crucial for music recommendation: the user's perception of similarity is driven by cultural aspects, but reliable context-related data (such as ratings) is available only for established artists; for the undiscovered long tail, we may only have content-based features. Thus, if we want to build a similarity model for everyone—popular, upcoming and niche artists—, we need our system to consider both content and context. Neglecting context, we would miss the cultural perspective of listeners; neglecting content, our model would only work well for the selected few.

2. RELATED WORK

A variety of methods have been devised for computing artist similarity, from the use of audio descriptors to measure similarity (Pohle et al., 2009), to leveraging text sources by measuring artist similarity as a document similarity task (Schedl et al., 2014). A significant effort has been dedicated to the study of graphs that interconnect musical entities with semantic relations as a proxy to compute artist similarity. For instance, Celma and Serra (2008) combine user profiles, music descriptions and audio features in a domain-specific ontology to compute artist similarity, whereas Oramas et al. (2015) extract semantic graphs of artists from artist biographies.

Other approaches use deep neural networks to learn artist embeddings from heterogeneous data sources and then compute similarity in the resulting embedding space (McFee and Lanckriet, 2009). Furthermore, metric learning approaches trained with triplet loss have been applied to learn the embedding space where similarity is computed (Lee et al., 2020a; Doras et al., 2020; Lee et al., 2020b; Park et al., 2018; Yesiler et al., 2020; Dorfer et al., 2017). While these models work well in objective tasks (e.g. genre classification, artist and song version identification), they do not consider cultural aspects of similarity.

Most recently, graph neural networks (GNNs) successfully improved upon metric-learning-based approaches: Salha-Galvan et al. (2021) train a GNN for directed link prediction between artists, considering both artist similarity and popularity, with a focus on cold-start artists with no known connections; our previous work (Korzeniowski et al., 2021), on the other hand, used a metric-learning objective to learn a GNN for artist similarity, and provided an overall evaluation for all artists, long-tail or not. Compared to older graph embedding methods such as node2vec (Grover and Leskovec, 2016), GNNs easily leverage both graph structure and node features.

Our artist similarity model thus combines graph approaches and embedding approaches using GNNs. The proposed model, described in detail in Section 3, uses content-based features (audio descriptors, or musicological attributes) together with explicit similarity relations between artists made by human experts (or extracted from listener feedback). These relations are represented in a graph of artists; the topology of this graph thus reflects the contextual aspects of artist similarity. The proposed graph neural network is trained using triplet loss to learn a function that embeds artists using both *content features* and *graph connections*. In this embedding space, similar artists are close to each other, while dissimilar ones are further apart.

We use two datasets (described in-depth in Section 4) to evaluate our approach: the OLGA dataset, which is collected from publicly available sources, comprising 17,673 artists; and a larger, proprietary dataset, consisting of 136,731 artists. Our experiment setup—metrics, models, data partitioning, etc.—is detailed in Section 5.

Beyond overall results, we take a deeper look at the model's performance on long-tail artists. Both are presented in Section 6. In contrast to Salha-Galvan et al. (2021), we consider not only artists with no known connections, but evaluate the change in performance

at different levels of known connectivity. We do find that performance suffers for artists with fewer known connections. To impede this effect, we devise a simple and effective training method—which we call *connection dropout*—that drastically mitigates this problem. This deeper evaluation, and the presentation of a novel method which improves results, constitute the main extension of our previous work on this topic (Korzeniowski et al., 2021).

3. MODELLING

The goal of an artist similarity model is to define a function $s(a, b)$ that estimates the similarity of two artists—i.e., yields a large number if artist a is considered similar to artist b, and small number if not.

Many content-based methods for similarity estimation have been developed in the last decades of MIR research (see Section 2). The field has closely followed the state-of-the-art in machine learning research, with general improvements coming from the latter translating well into improvements in the former. Acknowledging this fact, we select our baselines based on the most recent developments: Siamese neural networks trained with variants of the triplet loss (Doras et al., 2020; Lee et al., 2020b; Park et al., 2018; Yesiler et al., 2020; Dorfer et al., 2017). Building and training this type of model falls under the umbrella of *metric learning.*

3.1 METRIC LEARNING

The fundamental idea of metric learning is to learn a projection $\mathbf{y}_v = f(\mathbf{x}_v)$ of the input features $\mathbf{x}_v$ of an item v into a new vector space; this vector space should be structured in a way such that the distances between points reflect the task at hand. In our case, we want similar artists to be close together in this space, and dissimilar artists far away.

There is an abundance of methods that embed items into a vector space, many rooted in statistics, that have been applied to music similarity (Slaney et al., 2008). In this paper, we use a neural network for this purpose. The idea of using neural networks to embed similar items close to each other in an embedding space was pioneered by Bromley et al. (1993), with several improvements developed in the following decades. Most notably, the contrastive learning objective—where two items are compared to each other as a training signal—was replaced by the *triplet loss* (Hoffer and Ailon, 2015; Wang et al., 2014). Here, we observe three items simultaneously: the *anchor* item $\mathbf{x}_a$ is compared to a *positive* sample $\mathbf{x}_p$ and a *negative* sample $\mathbf{x}_n$. With the following loss formulation, the network is trained to pull the positive close to the anchor, while pushing the negative further away from it:

$$\mathcal{L}(t) = \left[d(\mathbf{y}_a, \mathbf{y}_p) - d(\mathbf{y}_a, \mathbf{y}_n) + \Delta\right]^+,$$

where t denotes the triplet $(\mathbf{y}_a, \mathbf{y}_p, \mathbf{y}_n)$, $d(\cdot)$ is a distance function (usually Euclidean or cosine), Δ is the maximum margin enforced by the loss, and $[\cdot]^+$ is the ramp function.

As mentioned before, state-of-the-art music similarity models are almost exclusively based on learning deep neural networks using the triplet loss. We thus adopt this method as our baseline model, which will serve as a comparison point to the graph neural network we propose in the following sections.

3.2 GRAPH NEURAL NETWORKS

A set of artists and their known similarity relations can be seen as a graph, where the artists represent the nodes, and the similarity relations their (undirected) connections. Graph methods thus naturally lend themselves to model the artist similarity problem (Oramas et al., 2015). A particular set of graph-based models that has been gaining traction recently are *graph neural networks* (GNNs), specifically *convolutional* GNNs. Pioneered by Bruna et al. (2014), convolutional GNNs have become increasingly popular for modelling different tasks that can be interpreted as graphs. We refer the interested reader to Wu et al. (2021) for a comprehensive and historical overview of GNNs. For brevity, we will focus on the one specific model our work is based on—the GraphSAGE model introduced by Hamilton et al. (2017) and refined by Ying et al. (2018)—and use the term GNNs for convolutional GNNs.

3.2.1 Model Overview

The GNN we use in this paper comprises two parts: first, a block of *graph convolutions* (GC) processes each node's features and combines them with the features of adjacent nodes; then, another block of fully connected layers projects the resulting feature representation into the target embedding space. See Figure 1 for an overview.

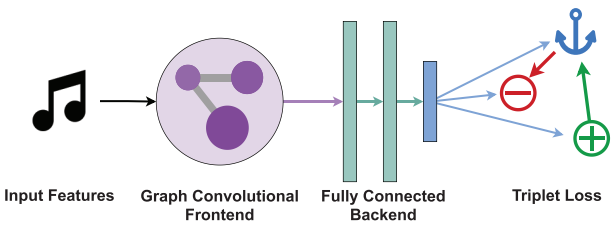

Figure 1 Overview of the graph neural network we use in this paper. First, the input features $\mathbf{x}_v$ are passed through a *front-end* of graph convolution layers (see Section 3.2.2 for details); then, the output of the front-end is passed through a traditional deep neural network *back-end* to compute the final embeddings $\mathbf{y}_v$ of artist nodes. Based on these embeddings, we use the triplet loss to train the network to project similar artists (positive, green) closer to the anchor, and dissimilar ones (negative, red) further away.

We train the model using the triplet loss, in an identical setup as the baseline model. Viewing the proposed GNN from this angle, the only difference of the GNN from a standard embedding network is the additional *Graph Convolutional Frontend*. In other words, if we remove all graph convolution layers, we arrive at our baseline model, a fully connected Deep Neural Network (DNN).

3.2.2 Graph Convolutions

The graph convolution algorithm, as defined by Hamilton et al. (2017); Ying et al. (2018), features two operations which are not found in classic neural networks: a *neighborhood function* $\mathcal{N}(\cdot)$, which yields the set of neighbors of a given node; and an *aggregation function*, which computes a vector-valued aggregation of a set of input vectors.

As a neighborhood function, most models use guided or uniform sub-sampling of the graph structure (Oh et al., 2019; Hamilton et al., 2017; Ying et al., 2018). This limits the number of neighbors to be processed for each node, and is often necessary to adhere to computational limits. As aggregation functions, models commonly apply pooling operators, LSTM networks, or (weighted) point-wise averages (Hamilton et al., 2017).

In this work, we take a simple approach, and use point-wise weighted averaging to aggregate neighbor representations, and select the strongest 25 connections as neighbors. If weights are not available, we use the simple average of random (but fixed) 25 connections. This enables us to use a single sparse dot-product with an adjacency matrix to select and aggregate neighborhood embeddings. Note that this is not the full adjacency matrix of the complete graph, as we select only the parts of the graph which are necessary for computing embeddings for the nodes in a mini-batch.

Algorithm 1 describes the inner workings of the graph convolution block of our model. Here, the matrix $\mathsf{X} \in \mathbb{R}^{D \times V}$ stores the D-dimensional features of all V nodes, the symmetric sparse matrix $\mathsf{A} \in \mathbb{R}^{V \times V}$ defines the connectivity of the graph, and $\mathcal{N}(v)$ is a neighborhood function which returns all connected nodes of a given node v (here, all non-zero elements in the v^{th} row of $\mathbf{A}$).

To compute the output of a graph convolution layer for a node, we need to know its neighbors. Therefore, to compute the embeddings for a mini-batch of nodes $\mathcal{V}$, we need to know which nodes are in their joint neighborhood. Thus, before the actual processing, we first need to trace the graph to find the node features necessary to compute the embeddings of the nodes in the mini-batch. This is shown in Figure 2, and formalized in lines 1–4 of Algorithm 1.

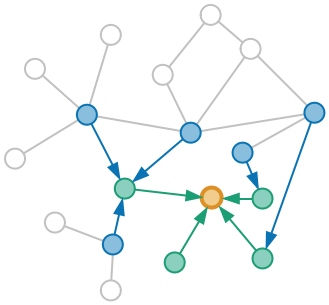

Figure 2 Tracing the graph to find the necessary input nodes for embedding the target node (orange). Each graph convolution layer requires tracing one step in the graph. Here, we show the trace for a stack of two such layers. To compute the embedding of the target node in the last layer, we need the representations from the previous layer of itself and its neighbors (green). In turn, to compute these representations, we need to expand the neighborhood by one additional step in the preceding GC layer (blue). Thus, the features of all colored nodes must be fed to the first graph convolution layer.

At the core of each graph convolution layer $k \in [1 \ldots K]$ there are two non-linear projections, parameterized by projection matrices $\mathbf{Q}_k \in \mathbb{R}^{H_{Q_k} \times D}$ and $\mathbf{W}_k \in \mathbb{R}^{H_{W_k} \times (H_{Q_k} + D)}$, and a point-wise non-linear activation function σ, in our case, the Exponential Linear Unit function (ELU). Here, H_{Q_k} and H_{W_k} are the output dimensions of the respective projections. The last output, $\mathbf{X}_K \in \mathbb{R}^{H_{W_K} \times V}$, holds the l_2-normalized representations of each node in the mini-batch in its columns. It is fed into the following fully connected layers, which then compute the output embedding $\mathbf{y}_v$ of a node. Finally, these embeddings are used to compute the triplet loss and back-propagate it through the GNN.

3.2.3 Connection Dropout

As we observed in our experiments (see Section 5), the GNNs learned to overly rely on the graph topology. This is because—given enough GC layers—graph topology trumps features when it comes to predicting similarity (as we will see in Section 5). To alleviate this issue, we introduce a tweak during training: each time we consult the neighborhood of a node k, we return a randomly sampled subset of its neighbors. This is achieved by dropping each connection to k with a given probability p. Concretely, in Algorithm 1, line 7, we randomly set each element $\mathbf{A}_k$ to 0 with probability p. During evaluation, we do not drop any connections, and use the allowed maximum of 25. As we will see when discussing our results, this method greatly improves the GNN's performance on artists with few known connections.

Connection Dropout can be seen as sub-sampling the neighborhoods in the graph. Sub-sampling has been previously used in GNNs, but for a different purpose: to condense neighborhoods and to control the computational burden. Indeed, Ying et al. (2018) finds

Algorithm 1: GRAPH CONVOLUTION BLOCK

Input : Node input features $\mathbf{X}$.
Sparse connectivity matrix $\mathbf{A}$.
Nodes in mini-batch $\mathcal{V} \subset [1 \dots V]$.
Number of layers K.
Weight matrices $\mathbf{Q}_k, \mathbf{W}_k, \forall k \in [1 \dots K]$.
Activation function σ.
Output: Node output representation $\mathbf{X}_K$

▷ Trace back input nodes for each layer.
$\mathcal{V}_K \leftarrow \mathcal{V}$;
for $k = K - 1 \dots 0$ **do**
  $\mathcal{V}_k \leftarrow \bigcup_{v \in \mathcal{V}_{k+1}} \mathcal{N}(v)$;
end
▷ Select input features for first layer. We use $\mathbf{M}[r, c]$ to denote selecting r rows and c columns from a matrix $\mathbf{M}$.
$\mathbf{X}_0 = \mathbf{X}[\cdot, \mathcal{V}_0]$;
for $k = 1 \dots K$ **do**
  $\mathbf{A}_k = \mathbf{A}[\mathcal{V}_{k-1}, \mathcal{V}_k]$;
  $\mathbf{N}_k = \sigma(\mathbf{Q}_k \cdot \mathbf{X}_{k-1}) \cdot \mathbf{A}_k$;
  $\mathbf{X}_k \leftarrow \sigma\left(\mathbf{W}_k \cdot \begin{bmatrix} \mathbf{N}_k \\ \mathbf{X}_{k-1}[\cdot, \mathcal{V}_k] \end{bmatrix}\right)$;
  ▷ l_2-normalize embeddings of each output node.
  $\mathbf{X}_k \leftarrow \left[\frac{\mathbf{x}_v}{\|\mathbf{x}_v\|_2} \mid v \in \mathcal{V}_k\right]$;
end
return $\mathbf{X}_K$

importance-weighted, dense neighborhoods using short random walks; these random walks were carried out until convergence criteria determined that the neighborhoods are stable enough (Eksombatchai et al., 2018). This is in stark contrast to our method, which randomly destabilizes and sparsifies neighborhood structures on purpose to achieve better generalization. Future work could aim at combining these two purposes; it is however out of scope of this work.

4. DATASETS

Many published studies on the topic of artist similarity are limited by data: datasets including artists, their similarity relations, and their features comprise at most hundreds to a few thousand artists. In addition, the quality of the ground truth provided is often based on 3rd party APIs with unknown similarity methods like the last.fm API, rather than based on data curated by human experts.

For instance, Oramas et al. (2015) provides two datasets, one with ~2k artists and similarity based on last.fm relations, and another with only 268 artists, but based on relations curated by human experts. Schedl et al. (2014) use a dataset of 1,677 artists based on last.fm similarity relations for evaluation. Also, the dataset used in the Audio Music Similarity and Retrieval (AMS) MIREX task, which was manually curated, contains data about only 602 artists. Others, like Lee et al. (2020a), use tag data shared among tracks or artists as a proxy for similarity estimation—which can be considered as a weak signal of similarity—and use a small set of 879 human-labeled triplets for evaluation.

Due to all these issues regarding existing datasets, we compiled a new dataset, the OLGA Dataset, which we describe in the following.

4.1 THE OLGA DATASET

For the OLGA ("**O**h, what a **L**arge **G**raph of **A**rtists") dataset, we bring together content-based low-level features from AcousticBrainz (Porter et al., 2015), and similarity relations from AllMusic, as curated by their music editors. Assembling the data works as follows:

1. Select a common pool of artists based on the unique artists in the Million Song Dataset (Bertin-Mahieux et al., 2011).
2. Map the available MusicBrainz IDs of the artists to AllMusic IDs using mapping available from MusicBrainz.
3. For each artist, obtain the list of "related" artists from AllMusic; this data can be licensed and accessed on their website. Use only related artists who can be mapped back to MusicBrainz.
4. Using MusicBrainz, select up to 25 tracks for each artist using their API, and collect the low-level features of the tracks from AcousticBrainz.
5. Compute the track feature centroid of each artist.

In total, the dataset comprises 17,673 artists connected by 101,029 similarity relations. On average, each artist is connected to 11.43 other artists. The quartiles are at 3, 7, and 16 connections per artist. The lower 10% of artists have only one connection, the top 10% have at least 27.

While the dataset size is still small compared to industrial catalog sizes, it is significantly bigger than other datasets available for this task. Its size and available features permits to apply more data-driven machine learning methods to the problem of artist similarity.[1]

For our experiments, we partition the artists following an 80/10/10 split into 14,139 training, 1767 validation, and 1767 test artists.

4.2 PROPRIETARY DATASET

We also use a larger proprietary dataset to demonstrate the scalability of our approach. Here, explicit feedback from listeners of a music streaming service is used to define whether two artists are similar or not: we derive similarity connections based on the co-occurrence of positive feedback for two artists.

For artist features, we use the centroid of an artist's track features. These track features are *musicological* attributes annotated by experts, and comprise hundreds of content-based characteristics such as "amount of electric guitar", or "prevalence of groove".

In total, this dataset consists of 136,731 artists connected by 3,277,677 similarity relations. The number of connections per artist is a top-heavy distribution with a few artists sharing most of the connections: the top

10% are each connected to more than 134 others, while the bottom 10% to only one. The quartiles are at 2, 5, and 48 connections per artist.

We follow the same partition strategy as for the OLGA dataset, which results in 109,383 training, 13,674 validation, and 13,674 test artists.

5. EXPERIMENTS

Our experiments aim to evaluate how well the embeddings produced by our model capture artist similarity. To this end, we set up a ranking scenario: given an artist, we collect its K nearest neighbors sorted by ascending distance, and evaluate the quality of this ranking. To quantify this, we use normalized discounted cumulative gain (Järvelin and Kekäläinen, 2002) with a high cut-off at $K = 200$ ("NDCG@200"). We prefer this metric over others, because it was shown that at high cut-off values, it provides better discriminative power, as well as robustness to sparsity bias (and, to a moderate degree, popularity bias) (Valcarce et al., 2018). Formally, given an artist a with an ideal list of similar artists $\mathbf{s}$ (sorted by relevance), the NDCG_K of a predicted list of similar artists $\hat{\mathbf{s}}$ is defined as:

$$\mathrm{NDCG}_K(a,\hat{\mathbf{s}},\mathbf{s}) = \frac{\sum_{k=1}^{K} g(\hat{s}_k,a)d(k)}{\sum_{k=1}^{K} g(s_k,a)d(k)},$$

where $g(\cdot, a)$, the *gain*, is 1 if an artist is indeed similar to a, and 0 otherwise, and $d(k) = \log_2^{-1}(k+1)$ the *discounting* factor, weights top rankings higher than the tail of the list.

In the following, we first explain the models, their training details, the features, and the evaluation data used in our experiments. Then, we show, compare and analyze the results.

5.1 MODELS

As explained in Section 3.2.1, a GNN with no graph convolution layers is identical to our baseline model (i.e. a DNN trained using triplet loss). This allows us to fix hyper-parameters between the baseline and the proposed GNN, and isolate the effect of adding graph convolutions to the model. For each dataset, we thus train and evaluate four models with 0 to 3 graph convolution layers.

The other hyper-parameters remain fixed: each layer in the graph convolutional front-end consists of 256 ELUs (Clevert et al., 2016); the back-end comprises two layers of 256 ELUs each, and one linear output layer with 100 dimensions; we train the networks using the ADAM optimizer (Kingma and Ba, 2015) with a linear learning-rate warm-up (Ma and Yarats, 2021) for the first epoch, and following a cosine learning rate decay (Loshchilov and Hutter, 2017) for the remaining 49 epochs (in contrast to Loshchilov and Hutter (2017), we do not use warm-restarts); for selecting triplets, we apply distance-weighted sampling (Wu et al., 2017), and use a margin of $\Delta = 0.2$ in the loss; finally, as distance measure, we use Euclidean distance between l_2-normalized embeddings.

We are able to train the largest model with 3 graph convolution layers within 2 hours on the proprietary dataset, and under 5 minutes on OLGA, using a Tesla P100 GPU and 8 CPU threads for data loading, which includes tracing the graph to find the relevant neighborhood as explained in Section 3.2.2.

5.2 FEATURES

We build artist-level features by averaging track-level features of the artist's tracks. Depending on the dataset, we have different types of features at hand.

In the OLGA dataset, we have low-level audio features extracted by the AcousticBrainz project using the Essentia library. These features represent track-level statistics about the loudness, dynamics and spectral shape of the signal, but they also include more abstract descriptors of rhythm and tonal information, such as BPM and the average pitch class profile.[2]

Although AcousticBrainz also provides high-level features such as mood and genre predictions, we refrain from using them. The reason is twofold: first, they are derived from the low-level features themselves, and as such, do not provide complementary information; second, as stated on the AcousticBrainz website itself, the high-level features may be subject to change if and when the models predicting them are changed, re-trained or improved.

We select all numeric features and pre-process them as follows: we apply element-wise standardization, discard features with missing values, and flatten all numbers into a single vector of 2613 elements.

In the proprietary dataset, we use numeric musicological descriptors annotated by experts (for example, "the nasality of the singing voice"). We apply the same pre-processing for these, resulting in a total of 170 values.

Using two different types of content features gives us the opportunity to evaluate the utility of our graph model under different circumstances, or more precisely, features of different quality and signal-to-noise ratio. The low-level audio-based features available in the OLGA dataset are undoubtedly noisier and less specific than the high-level musical descriptors manually annotated by experts, which are available in the proprietary dataset. Experimenting with both permits us to gauge the effect of using the graph topology for different data representations.

In addition, we also train models with *random vectors* as features. For each artist, we uniformly sample a random vector of the same dimension as the real features, and keep it constant throughout training and testing. This

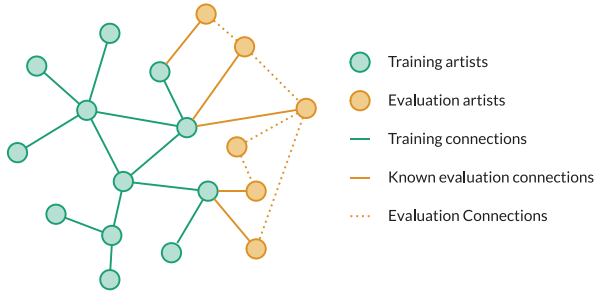

Figure 3 Artist nodes and their connections used for training (green) and evaluation (orange). During training, only green nodes and connections are used. When evaluating, we extend the graph with the orange nodes, but only add connections between validation and training artists. Connections among evaluation artists (dotted orange) remain hidden. We then compute the embeddings of all evaluation artists, and evaluate based on the hidden evaluation connections.

way, we can differentiate between the performance of the real features and the performance of using the graph topology in the model: the results of a model with no graph convolutions is only due to the features, while the results of a model with graph convolutions but random features is only due to the use of the graph topology.

5.3 EVALUATION DATA

As described in Section 4, we partition artists into a training, validation and test set. When evaluating on the validation or test sets, we only consider artists from these sets as candidates and potential true positives. Specifically, let $\mathcal{V}_{\text{eval}}$ be the set of evaluation artists; we only compute embeddings for these, and retrieve nearest neighbors from this set, and only consider ground truth similarity connections within $\mathcal{V}_{\text{eval}}$.

This notion is more nuanced in the case of GNNs. Here, we want to exploit the *known artist graph topology* (i.e.,which artists are connected to each other) when computing the embeddings. To this end, we use all connections between artists in $\mathcal{V}_{\text{train}}$(the training set) *and* connections between artists in $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{eval}}$. This process is outlined in Figure 3.

Note that this does not leak information between train and evaluation sets; the features of evaluation artists have not been seen during training, and connections within the evaluation set—these are the ones we want to predict—remain hidden.

5.4 EMULATING LONG-TAIL ARTISTS

Overall evaluations portray a model's performance from a birds-eye view. Beyond that, we are interested in the performance of our model for the segment of long-tail artists. Such artists usually have few known connections, which not only limits the information a GNN is able to leverage, but also limits our capability to evaluate how well the GNN is able to leverage existing information. Since ground truth for these artists is sparse, retrieved lists of similar artists can contain relevant items for which we do not know that they are relevant; we cannot quantitatively distinguish a list of bad recommendations from a list of good recommendations of which we do not know that they are indeed good.

To circumvent this problem, we collect a subset of well-connected artists for which we will then artificially sparsify *known evaluation connections* (i.e.,connections between validation artists and training artists, see Figure 3). This will enable us to emulate a long-tail artist for which the GNN cannot use a dense neighborhood to compute embeddings. At the same time, we retain the ability to quantify the quality of retrieved similar artists, since we have a lot of unseen evaluation connections at hand. In particular, we will sweep the number of known evaluation connections from zero to 25 (the maximum number of connections), and inspect the results for each degree of connectivity.

Depending on the dataset, we use different criteria to select these artists. Since each dataset differs in size and connection density, parameters that work for one would not work for the other. For the proprietary dataset, which is large and densely connected, we use artists with at least 25 connections to the training graph (known evaluation connections), and 50 unseen connections ("evaluation connections" in Figure 3); this results in 207 artists. For the OLGA dataset, we also require 25 known evaluation connections, but are satisfied with at least 5 unseen evaluation connections; this gives us 44 artists to look at.

6. RESULTS AND DISCUSSION

We will first discuss the overall results in the following section. Then, we will use the subsets of artists selected in Section 5.4 to evaluate the sensitivity of our model to decaying connectivity, as observed with less popular artists.

6.1 OVERALL EVALUATION

Table 1 compares the baseline model with the proposed GNN, trained without connection dropout. We can see that the GNN easily out-performs the DNN. It achieves

DATASET	FEATURES	DNN	GNN
OLGA	Random	0.02	0.45
	AcousticBrainz	0.24	0.55
Proprietary	Random	0.00	0.52
	Musicological	0.44	0.57

Table 1 NDCG@200 for the baseline (DNN) and the proposed model with 3 graph convolution layers (GNN), using features or random vectors as input. The GNN with real features as input gives the best results. Most strikingly, the GNN with random features—using only the known graph topology—out-performs the baseline DNN with informative features.

an NDCG@200 of 0.55 vs. 0.24 on the OLGA dataset, and 0.57 vs. 0.44 on the proprietary dataset. The table also demonstrates that the graph topology is more predictive of artist similarity than content-based features: the GNN, using random features, achieves better results than a DNN using informative features for both datasets (0.45 vs. 0.24 on OLGA, and 0.52 vs. 0.44 on the proprietary dataset).

Additionally, the results indicate—perhaps to little surprise—that low-level audio features in the OLGA dataset are less informative than manually annotated high-level features in the proprietary dataset. Although the proprietary dataset poses a more difficult challenge due to the much larger number of candidates (14k vs. 1.8k), the DNN—which can only use the features—improves more over the random baseline in the proprietary dataset (+0.44), compared to the improvement (+0.22) on OLGA. These are only indications; for a definitive analysis, we would need to use the exact same features in both datasets.

Similarly, we could argue that the topology in the proprietary dataset seems more coherent than in the OLGA dataset. We can judge this by observing the performance gain obtained by a GNN with random features—which can only leverage the graph topology to find similar artists—compared to a completely random baseline (random features without GC layers). In the proprietary dataset, this performance gain is +0.52, while in the OLGA dataset, only +0.43. Again, while this is not a definitive analysis (other factors may play a role), it indicates that the large amounts of user feedback used to generate ground truth in the proprietary dataset give stable and high-quality similarity connections.

Figure 4 depicts the results for each model and feature set depending on the number of graph convolution layers used. (Recall that a GNN with 0 graph convolutions corresponds to the baseline DNN.) In the OLGA dataset, we see the scores increase with every added layer. This effect is less pronounced in the proprietary dataset, where adding graph convolutions does help significantly, but results plateau after the first graph convolution layer. We believe this is due to the quality and informativeness of the features: the low-level features in the OLGA dataset provide less information about artist similarity than high-level expertly annotated musicological attributes in the proprietary dataset. Therefore, exploiting contextual information through graph convolutions results in more uplift in the OLGA dataset than in the proprietary one.

Looking at the scores obtained using random features (where the model depends solely on exploiting the graph topology), we observe two remarkable results. First, whereas one graph convolution layer suffices to outperform the feature-based baseline in the OLGA dataset (0.28 vs. 0.24), using only one GC layer does not produce meaningful results (0.05) in the proprietary dataset. We believe this is due to the different sizes of the respective test sets: 14k in the proprietary dataset, while only 1.8k in

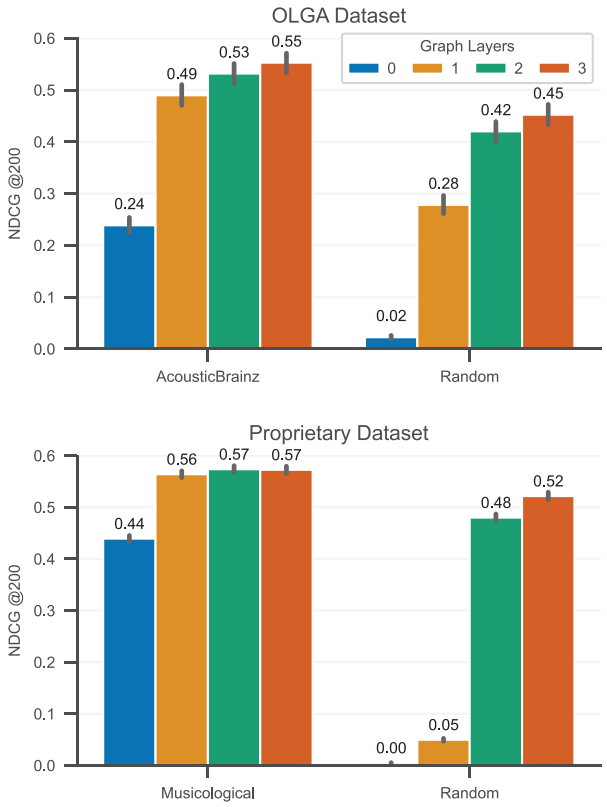

Figure 4 Results on the OLGA (top) and the proprietary (bottom) dataset with different numbers of graph convolution layers, using either the given features (left) or random vectors as features (right). Error bars indicate 95% confidence intervals computed using bootstrapping.

OLGA. Using only a very local context seems to be enough to meaningfully organize the artists in a smaller dataset.

Second, most performance gains are obtained with two GC layers, while adding the third GC layer pushes the results to a much lesser degree. Our explanation for this effect is that most similar artists are connected through at least one other, common artist. In other words, most artists form similarity cliques with at least two other artists. Within these cliques, in which every artist is connected to all others, missing connections are easily retrieved by no more than 2 graph convolutions.

In fact, in the OLGA dataset, ~71% of all cliques fulfill this requirement. This means that, for any hidden similarity link in the data, in 71% of cases, the true similar artist is within 2 steps in the graph—which corresponds to using two GC layers.

6.2 EVALUATION OF LONG-TAIL ARTISTS

Let us now focus on the results specific for long-tail artists. As explained in Section 5.4, we will not use actual long-tail artists for this, since data sparsity prevents a solid evaluation. Instead, we emulate the long-tail condition by removing known connections of well-connected artists, while keeping all their unseen evaluation connections. From the OLGA dataset, we collected 44 artists with at least 25 known connections and at least 5 unseen ones; for the proprietary dataset,

we found 207 artists with at least 25 known connections, and at least 50 unseen ones (the proprietary dataset is larger and more densely connected).

We train the largest models with 3 graph convolution layers using varying connection dropout probabilities: 0.0, 0.25, 0.5, 0.75, 0.95 and 0.99; a connection dropout probability of 0.0 corresponds to the baseline GNN model with no connection dropout. Once these models are trained, we use them to evaluate the resulting artist embeddings in different connectivity settings: we sweep the *known evaluation connections* between 25 and zero (the cold-start scenario) for each evaluation artist, dropping the weakest connection at each step. To reiterate, we do not re-train the models; we only manipulate the connectivity of validation artists when computing artist embeddings.

Figure 5 shows the results. We see that for the baseline model (blue, no connection dropout), results degrade significantly with decreasing connectivity. We observe this effect—though with different intensity—on both datasets. Indeed, the baseline model with no connection dropout performs poorly for cold-start artists: it needs 2 known connections in the OLGA dataset to be on-par with a simple DNN without GC layers (and even 5 in the proprietary one).

We also observe how connection dropout greatly reduces that degradation, *without negatively impacting results for well-connected artists*: in the OLGA dataset, we can use very high dropout rates such as 0.95 to achieve better-than-baseline results for cold-start artists without significantly sacrificing results for others; in the proprietary dataset, we achieve this with a lower dropout probability of 0.75. The optimal ratio of connection dropout clearly depends on the dataset, and is a hyper-parameter to be tuned. However, values of 0.5 or 0.75 seem to be good starting points.

Using connection dropout achieves better results for sparsely connected artists because it prevents the GNN from relying too much on the graph connectivity when computing the embedding. To substantiate this claim, we examine the stability of artist embeddings while manually removing known connections, using the same subset of artists as before. We consider the embedding computed using all 25 known connections to be the true embedding of an artist. We then remove known connections one by one, compute a new artist embedding at each level of connectivity, and calculate the cosine distance of these embeddings to the true embedding.

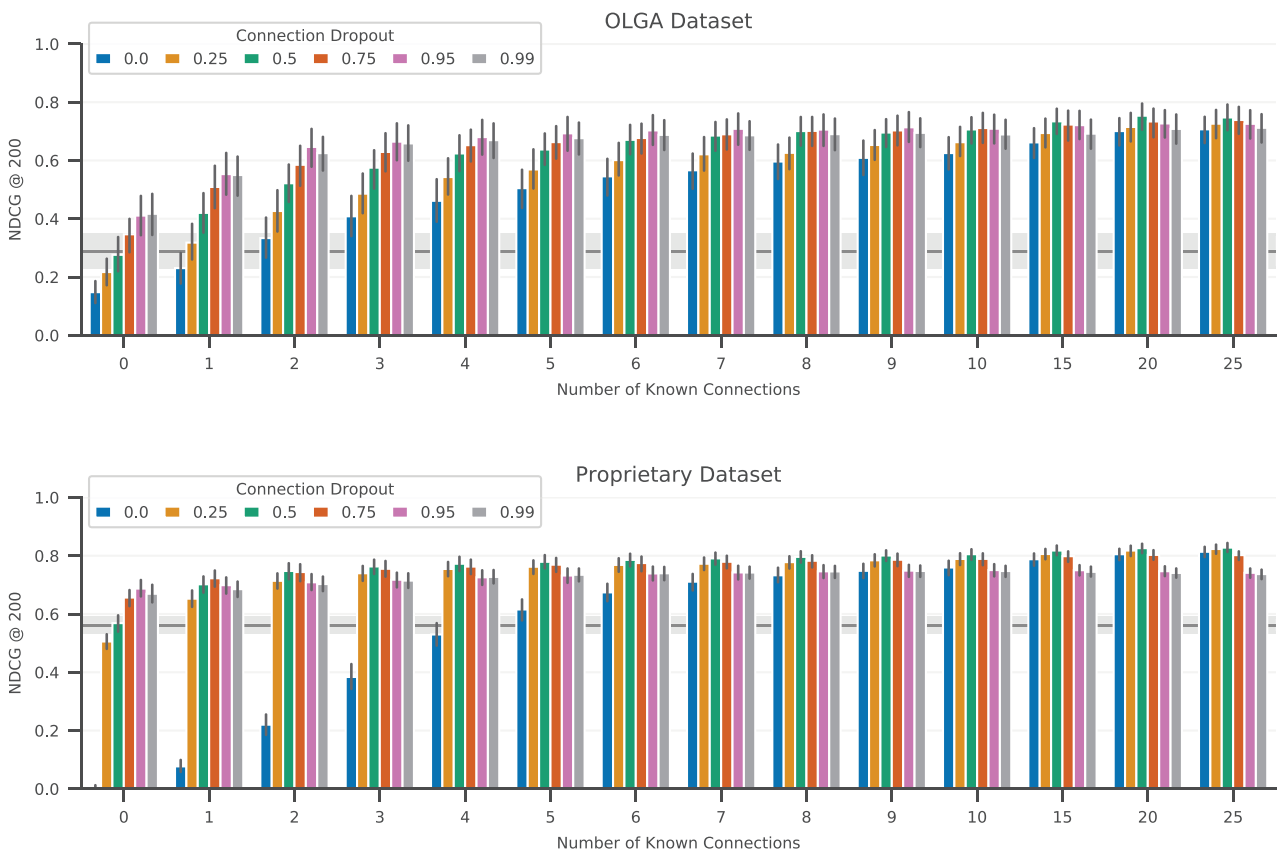

Figure 5 Evaluation of the long-tail performance of a 3-GC-layer model on the OLGA dataset (top) and the proprietary dataset (bottom). The different bars represent models trained with different probabilities of connection dropout. The gray line in the background represents the baseline model with no graph convolution layers, with the shaded area indicating the 95% confidence interval. We see that for the standard model (blue, no connection dropout), performance degrades with fewer connections. Introducing connection dropout significantly reduces this effect.

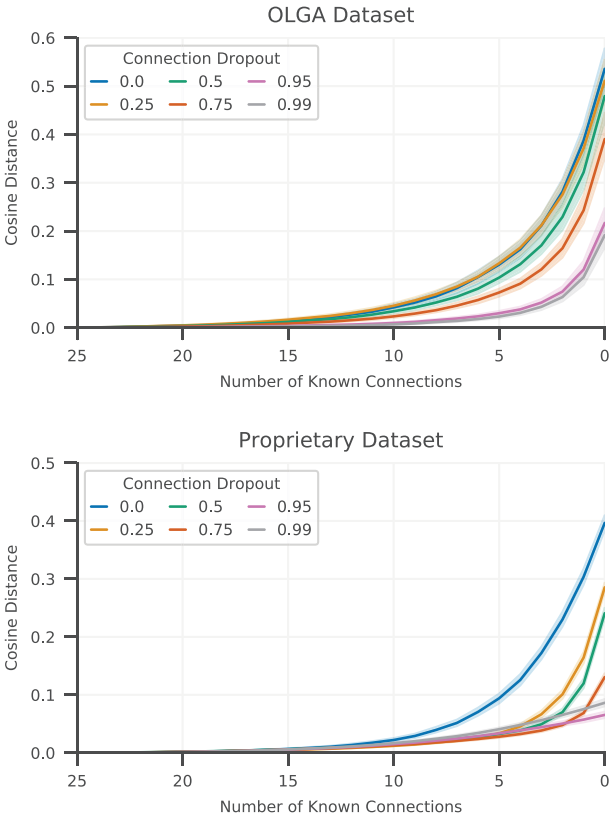

Figure 6 Cosine distance between embeddings computed using reduced connectivity and the "true" embedding (computed using all 25 known connections). Without connection dropout, the GNNs learn to rely too much on the graph connectivity to compute the artist embedding: the distance between an embedding computed using fewer connections and the "true" embedding grows quickly. With connection dropout, we can strongly curb this effect.

The results are shown in Figure 6: the distance stays close to zero at first, but degrades quickly after we dropped a certain number of connections. We also see how connection dropout greatly reduces this effect. In the OLGA dataset, the cosine distance between the embedding using no connections and the one using all 25 connections is more than 0.5 on average when no connection dropout is used; this number drops to less than 0.2 if we employ a connection dropout rate of 0.99. Similar effects can be observed for the proprietary dataset.

7. SUMMARY AND FUTURE WORK

In this paper, we described a hybrid approach to computing artist similarity, which uses graph neural networks to combine content-based features with explicit relations between artists.

To evaluate our approach, we assembled a dataset with 17,673 artists, their features, and their similarity relations. Additionally, we used a much larger proprietary dataset to show the scalability of our method. The results showed that leveraging known connections between artists can be more effective for understanding their similarity than high-quality features, and that combining both gives the best results.

The introduction of *Connection Dropout* in training was shown to be effective in decreasing the model's reliance on the number of known artist connections, which was detrimental for sparsely connected long-tail artists. The proposed method significantly improves results for such artists without negatively affecting densely connected ones.

Our work is a first step towards models that directly use known relations between musical entities, like tracks, albums, artists, or even genres. Future work could investigate how to employ multi-modality in this context; for example, we could build a multi-modal graph by using connections between different types of entities (e.g. tracks, albums, artists), or different types of connections between the same entities (e.g. artist collaborations, band memberships). Another avenue of research could focus on collecting and using better and/or higher-level features for the OLGA dataset. This would provide a better judgement of the importance of feature quality in the proposed model.

NOTES

1 The procedure to assemble the dataset, including relevant metadata, is available on https://gitlab.com/fdlm/olga/.

2 The exact list of low-level features we use is available at https://gitlab.com/fdlm/olga/.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Filip Korzeniowski orcid.org/0000-0002-7287-8053
Pandora/SiriusXM, Oakland, USA

Sergio Oramas orcid.org/0000-0002-8028-2890
Pandora/SiriusXM, Oakland, USA

Fabien Gouyon
Pandora/SiriusXM, Oakland, USA

REFERENCES

Aucouturier, J.-J. and **Pachet, F.** (2002). Music similarity measures: What's the use? In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France.

Bertin-Mahieux, T., Ellis, D. P. W., Whitman, B., and **Lamere, P.** (2011). The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA.

Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and **Shah, R.** (1993). Signature verification using a "Siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems (NIPS)*, San Francisco, USA. DOI: https://doi.org/10.1142/S0218001493000339

Bruna, J., Zaremba, W., Szlam, A., and **LeCun, Y.** (2014). Spectral networks and locally connected networks on graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada.

Celma, Ò. and **Serra, X.** (2008). FOAFing the music: Bridging the semantic gap in music recommendation. *Journal of Web Semantics*, 6(4):250–256. DOI: https://doi.org/10.1016/j.websem.2008.09.004

Clevert, D.-A., Unterthiner, T., and **Hochreiter, S.** (2016). Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico.

Doras, G., Yesiler, F., Serra, J., Gomez, E., and **Peeters, G.** (2020). Combining musical features for cover detection. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Online.

Dorfer, M., Arzt, A., and **Widmer, G.** (2017). Learning audio-sheet music correspondences for score identification and offline alignment. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China. DOI: https://doi.org/10.5334/tismir.12

Eksombatchai, C., Jindal, P., Liu, J. Z., Liu, Y., Sharma, R., Sugnet, C., Ulrich, M., and **Leskovec, J.** (2018). Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference (WWW)*, Lyon, France. DOI: https://doi.org/10.1145/3178876.3186183

Ellis, D. P. W., Whitman, B., Berenzweig, A., and **Lawrence, S.** (2002). The quest for ground truth in musical artist similarity. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France.

Grover, A. and **Leskovec, J.** (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 855–864, New York, USA. DOI: https://doi.org/10.1145/2939672.2939754

Hamilton, W. L., Ying, R., and **Leskovec, J.** (2017). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, Long Beach, USA.

Hoffer, E. and **Ailon, N.** (2015). Deep metric learning using triplet network. In Feragen, A., Pelillo, M., and Loog, M., editors, *Similarity-Based Pattern Recognition (SIMBAD)*, Copenhagen, Denmark. DOI: https://doi.org/10.1007/978-3-319-24261-3_7

Järvelin, K. and **Kekäläinen, J.** (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446. DOI: https://doi.org/10.1145/582415.582418

Kingma, D. P. and **Ba, J.** (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA.

Korzeniowski, F., Oramas, S., and **Gouyon, F.** (2021). Artist similarity with graph neural networks. In *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online.

Lee, J., Bryan, N. J., Salamon, J., Jin, Z., and **Nam, J.** (2020a). Disentangled multidimensional metric learning for music similarity. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain. DOI: https://doi.org/10.1109/ICASSP40776.2020.9053442

Lee, J., Bryan, N. J., Salamon, J., Jin, Z., and **Nam, J.** (2020b). Metric learning vs classification for disentangled music representation learning. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Online.

Loshchilov, I. and **Hutter, F.** (2017). SGDR: Stochastic gradient descent with warm restarts. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France.

Ma, J. and **Yarats, D.** (2021). On the adequacy of unturned warmup for adaptive optimization. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*, Online. DOI: https://doi.org/10.1609/aaai.v35i10.17069

McFee, B. and **Lanckriet, G. R. G.** (2009). Heterogeneous embedding for subjective artist similarity. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan.

Oh, J., Cho, K., and **Bruna, J.** (2019). Advancing GraphSAGE with a data-driven node sampling. In *Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds*, New Orleans, USA.

Oramas, S., Sordo, M., Espinosa-Anke, L., and **Serra, X.** (2015). A semantic-based approach for artist similarity. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain.

Park, J., Lee, J., Park, J., Ha, J.-W., and **Nam, J.** (2018). Representation learning of music using artist labels. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France.

Pohle, T., Schnitzer, D., Schedl, M., Knees, P., and **Widmer, G.** (2009). On rhythm and general music similarity. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan.

Porter, A., Bogdanov, D., Kaye, R., Tsukanov, R., and **Serra, X.** (2015). AcousticBrainz: A community platform for gathering music information obtained from audio. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Malaga, Spain.

Salha-Galvan, G., Hennequin, R., Chapus, B., Tran, V.-A., and **Vazirgiannis, M.** (2021). Cold start similar artists ranking with gravity-inspired graph autoencoders. In *Proceedings of the 15th ACM Conference on Recommender Systems*

(RECSYS), Amsterdam, Netherlands. DOI: https://doi.org/10.1145/3460231.3474252

Schedl, M., Hauger, D., and **Urbano, J.** (2014). Harvesting microblogs for contextual music similarity estimation: A co-occurrence-based framework. *Multimedia Systems*, 20(6):693–705. DOI: https://doi.org/10.1007/s00530-013-0321-5

Slaney, M., Weinberger, K. Q., and **White, W.** (2008). Learning a metric for music similarity. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, USA.

Valcarce, D., Bellogín, A., Parapar, J., and **Castells, P.** (2018). On the robustness and discriminative power of information retrieval metrics for top-N recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RECSYS)*, Vancouver, Canada. DOI: https://doi.org/10.1145/3240323.3240347

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and **Wu, Y.** (2014). Learning fine-grained image similarity with deep ranking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, USA. DOI: https://doi.org/10.1109/CVPR.2014.180

Wu, C.-Y., Manmatha, R., Smola, A. J., and **Krahenbuhl, P.** (2017). Sampling matters in deep embedding learning. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Venice, Italy. DOI: https://doi.org/10.1109/ICCV.2017.309

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and **Yu, P. S.** (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24. DOI: https://doi.org/10.1109/TNNLS.2020.2978386

Yesiler, F., Serra, J., and **Gomez, E.** (2020). Accurate and scalable version identification using musicallymotivated embeddings. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain. DOI: https://doi.org/10.1109/ICASSP40776.2020.9053793

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and **Leskovec, J.** (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, London, United Kingdom. DOI: https://doi.org/10.1145/3219819.3219890

TO CITE THIS ARTICLE:
Korzeniowski, F., Oramas, S., and Gouyon, F. (2022). Artist Similarity for Everyone: A Graph Neural Network Approach. *Transactions of the International Society for Music Information Retrieval*, 5(1), 129–140. DOI: https://doi.org/10.5334/tismir.143

Submitted: 13 June 2022 **Accepted:** 12 September 2022 **Published:** 27 October 2022