EasyChair Preprint

№ 8872

Exploring Graph Representation of Chorales

Somnuk Phon-Amnuaisuk

September 24, 2022

Exploring Graph Representation of Chorales

No Author Given

No Institute Given

Abstract. This work explores uncharted areas overlapping music, graph theory, and machine learning. An embedding representation of a node, in a weighted undirected graph $\mathcal{G}$, is a representation that captures the meaning of nodes in an embedding space. In this work, 383 Bach chorales were compiled and represented as a graph. Two application cases were investigated in this paper (i) learning node embedding representation using *Continuous Bag of Words (CBOW), skip-gram*, and *node2vec* algorithms, and (ii) learning node labels from neighboring nodes based on a collective classification approach. The results of this exploratory study ascertains many salient features of the graph-based representation approach applicable to music applications.

Keywords: Graph representation · Chorales · Learning node embedding · Node2Vec · Collective classification

1 Introduction

Knowledge representation is one of the fundamental ingredients in intelligent systems since inference processes operate on the representations that the system uses. In the past decade, interests in graph representation learning have surged in correlation with advances in computational power and deep learning techniques. In this paper, we investigate the graph representation learning of chorales. We construct monopartite graphs where each node in the graph represents a chorale composition and an edge linking between any two nodes represents the similarity between the two chorales.

In this work, the *node2vec* method [1] is applied to learn low-dimensional embedding representation of nodes in a chorale graph. The attempts at a novel application of graph representation learning in the chorales domain entails many challenges. Hence, methodology and appropriate data must be prepared from scratch. Thanks to Music21 [2], the process of chorales data preparation has been simplified substantially. We prepare harmonic progressions from three hundred and eighty-three (383) chorales retrieved from the Bach chorales corpus available in Music21. A chorale graph is created based on similarities between harmonic sequences of chorales. A low dimensional-embedding representation that preserves the original neighborhood structure of the graph is learned using the following three techniques: continuous bag of words (*CBOW*), *skip-gram*, and *node2vec*. Both CBOW and skip-gram are available from the *word2vec* [3] class in *Gensim* library [4]).

Two application cases were presented in this paper (i) learning node embedding representation and then suggesting similar chorales based on similarities in the embedding space, and (ii) learning node labels from neighboring nodes using the collective classification approach. The rest of the paper is organized as follows: Section 2 discusses the background and some related works; Section 3 discusses our approach and gives the details of the techniques behind it; Section 4 provides a critical evaluation and discussion of the empirical results; and the final section presents the conclusion and further research.

2 Graph Representation Learning of Chorales

Representing music as symbols on staves to signify pitches, durations, dynamics, and various intended sonorities such as staccato, legato, crescendo, etc., proves to be an effective means of representation. This representation system has been widely adopted by music industries and academia. The western music notation system was designed for humans to read, reason and realize those notations into sound.

With the progressive development of electronic musical instruments in the 1960s, performance from one keyboard instrument can be transmitted to other electronic instruments to enrich the sonic texture by combining various sound sources. The transmission of digital performance data has been standardized to the MIDI protocol which captures performance information such as pitch, duration, and pitch bending. MIDI was designed for the transferring of performance data between electronic music instruments.

2.1 Representing Music for Analysis and Reasoning

Traditional music notation is a native language used by music professionals. Musicologists reason and discuss music concepts based on this language. To facilitate automated reasoning of music notations using computers, researchers have developed various representation systems based on the point of view from which the problem is being examined e.g. MIDI, **kern, Humdrum [5].

Two important criteria in music representation systems: *expressive completeness* and *structural generality* were discussed in [6]. Expressive completeness refers to the range of raw musical data that can be represented, and structural generality refers to the range of higher-level structures that can be represented and manipulated using the representation system.

Since traditional music notations have been developed to abstract sound streams, the representation system for the purpose of music analysis commonly represents the basic building blocks of pitches, durations and facilitates functionalities on them for the derivation of more complex knowledge from these building blocks [7, 8]. For example:

Basic types	Degree : $\{1, 2, 3, 4, 5, 6, 7\}$
	Accidental : $\{\natural, \sharp, \times, \flat, \flat\flat\}$
	Octave : $\{1..8\}$
	Pitch : $\langle$ Degree, Accidental, Octave $\rangle$
Basic operations	add_{dd} : Duration $\times$ Duration $\rightarrow$ Duration
	add_{td} : Time $\times$ Duration $\rightarrow$ Time
	sub_{tt} : Time $\times$ Time $\rightarrow$ Duration
	sub_{dd} : Duration $\times$ Duration $\rightarrow$ Duration

In this work, we resort to Music21, a python-based toolkit that facilitates knowledge representation and scores well in both expressive completeness, and structural generality criteria discussed above. Music21 has been developed since 2008 and was inspired by, notably, Humdrum and musicXML [9].

Apart from viewing music as a stream of note events (pitches and durations). Musicologists also explore other representation schemes which could facilitate reasoning about deep structure, for example, *Generative Theory of Tonal Music (GTTM)* [10]; computational models of tonality [11]; reasoning about the similarity between pitch intervals using a geometric representation of pitch and harmony: key finding algorithm [12], chord progression sequences [13], and perceived harmonic relations [14].

2.2 Representing Music as a Graph

Describing domain knowledge using graphs by abstracting knowledge into nodes, and edges (e.g., social networks, communication networks, molecule or protein structures, visual scene graph) offers additional benefits from information inherited or derived from its structures as they present information which would otherwise be unavailable in traditional non-graph representation approaches.

A musical dice game *(Musikalisches Würfelspiel)* is one of the formal compositional processes invented since the 1700s [15] (page 4). This process can be formulated as a multi-graph where each node represents a measure (i.e. bar) of pre-composed materials linking to various nodes. This graph can be employed to generate a new piece of music by traversing the graph in accordance with imposed constraints.

In [16], a graph structure is employed to represent music scores where terminal nodes describe melodic contents, the internal node represents its incremental generalization and the edge represents a relationship between nodes. The graph structure in [16] is employed to calculate melodic similarities for content-based music retrieval tasks. The design of the graph can be in various forms e.g., monopartite graphs (single class of nodes), multipartite graphs, simple graphs, and multigraphs. The design is application-dependent. In [17], a graph neural network is designed for music score data and modeling expressive piano performance.

3 Materials and Methods

3.1 Compute Harmonic Sequences

Music21 provides various modules for musicologists to analyze music. *Chordify* is one of the modules provided in Music21. Chordify analyzes polyphonic notes and reduce them to a chord. These chords can be notated using a Roman numeral in a functional harmony fashion. For example, I and $I6$ denotes the tonic chord in its root and first inversion, respectively. $V7$ denotes the dominant seventh chord and *viio* denotes the diminished leading chord, respectively.

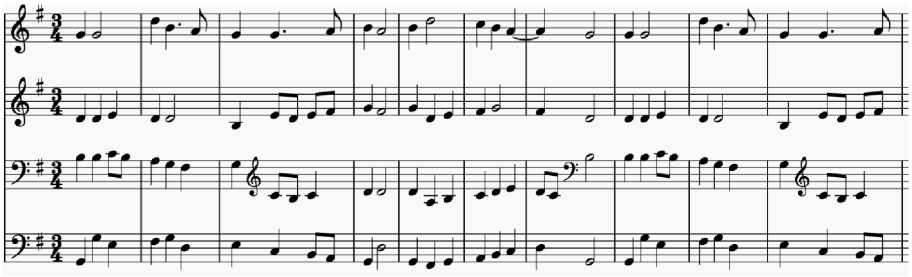


Fig. 1. The first ten measures (bars) from chorale BWV269.

We retrieved 383 chorales from the core corpus available in Music21[1] and wrote a Python script to extract four voices (i.e., soprano, alto, tenor and bass) then infer chordal sequences of the four part chorales. Figure 1 shows the first ten (10) measures from chorale BWV269. Table 2 illustrates representative examples of harmonic sequences extracted from chorales ID BWV269, BWV86.6, BWV178.7 and BWV185.6, respectively.

Table 1. Examples of harmonic sequences extracted from Chorales

BWV	Harmonic sequence	Cadence
86.6	[I, V6, iii43, iv74, V7, I, bVII6, iiiϕ7b53, I, ...]	[I, V, v5, II6, V, I]
148.6	[i, ii743, i6, ii, viio#63, i, i#7, iv6, iio64, ...]	[iv6, i64, iiϕ65, V, V75#3, I]
178.7	[i, III6, v7, bVI, iio64, III6, bVII64, III, i, ...]	[iv6, i43, iiϕ65, i54, V75#3, I]
185.6	[V6, V65, i532, i, ii754, V75#3, i, ii42, i, i, ...]	[iio6, III7, i752, i54, V75#3, I]
269	[I, I, IV6, vi, V6, I, iii6, V, vi, IV, I752, ...]	[vi, vi42, ii65, V, V7, I]
355	[I, I6, ii65, V, I, I42, #ivo6, V, V, V42, I6, ...]	[I6, ii65, ii7, V, V7, I]

[1] At presence, there are 387 unique Bach chorales in the Music21 corpus, we only use 383 chorales in this work since some chorales formats are not compatible with our conversion script.

3.2 Compute Harmonic Word Embedding Representation

Recent progress in natural language processing (NLP) promotes the concept of word embedding representation which is a distributed vectorized representation of a word w, $f(w) \mapsto \mathcal{R}^d$. Word embedding representation could encode the meanings of words where words that are close together in the representation space also share similar meanings or related concepts. For example, the word *ice* should be closer to the word *cold* rather than the word *hot*.

Music chords abstract polyphonic sound into symbols (i.e., the extracted harmonic sequences of chorales). These symbols can be considered as a set of words describing the harmonic vocabulary of chorales. Here, the *word2vec* model in Gensim [4] was employed to learn the embedding representation of chords (i.e., words) in chorales. The embedding representations were used to compute similarities between chorale nodes when constructing a chorale graph.

Algorithm 1 Graph construction and Similarity computation

Let $Comp$ be a set of chorale compositions.
Let Seq_u and Seq_v be any two chord sequences having the same length n.
Let c_i, c_j be chords in the sequence and i, j denote index $1...n$.
Let $w2v(c_i, c_j) \mapsto \mathcal{R}$ be a function that computes similarity between chords.
The similarity value is weighted by attention score $e^{-|i-j|}$ emphasizing
a chord pair with the same position $f(w) \mapsto \mathcal{R}^d$. index.

procedure *GraphConstruction* ($Comp$)
    forall $(u, v) \in \{Comp \times Comp\}$ and $u \neq v$
        if $\mathcal{S}$ (u, v) complies to constraints ξ
        then add nodes u, v to graph $\mathcal{G}$
            and add edge (u, v) with weight $\mathcal{S}$ (u, v)
    return $\mathcal{G}$

procedure $\mathcal{S}$ (u, v)
    extract Seq_u, Seq_v from composition u and v
    $similarity = 0$
    for c_i in Seq_u
        for c_j in Seq_v
            $similarity \mathrel{+}= w2v(c_i, c_j) * e^{-|i-j|}$
        end for
    end for
    return $similarity$

3.3 Constructing Chorale Graphs

Let a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ denotes a set of vertices and each vertex (node) represent a chorale composition; and $\mathcal{E}$ denotes a set of un-ordered pairs $(u, v) \in \{\mathcal{V} \times \mathcal{V}\}$. A graph $\mathcal{G}$ is constructed by connecting any two chorales that satisfies the required constraints ξ imposed by a similarity function $\mathcal{S}$. e.g., $\mathcal{S}\ (u, v) > \xi$.

In this paper, a graph was constructed based on the similarity of the harmonic progression in the intro, and the final cadence between two chorales. The similarity $\mathcal{S}$ between two harmonic progressions (chord sequences taken from chorales u and v) were estimated using the Algorithm 1.

3.4 Learning Node Embedding of Chorales

Word embedding representation from the *word2vec* model is a latent variable learned by training a shallow feed-forward neural network to learn relationships between target and context words. Training samples for *word2vec* model are prepared from the desired text corpus. The same tactic used in *word2vec* to learn word embedding representation can be applied to learn relationships among nodes in a graph. The words are replaced with nodes, and training samples are prepared by sampling nodes from the graph.

Learning node embedding yields a compact representation of nodes in a graph. A node embedding representation $f : u \mapsto \mathcal{R}^d$ maps nodes u and v in a graph to vectors $f(u) \rightarrow \mathbf{z}_u$ and $f(v) \rightarrow \mathbf{z}_v$ where $\mathbf{z}_u, \mathbf{z}_v$ still preserves the original topological structure of graph $\mathcal{G}$ in the embedding space. The more compact representation in the embedding space is effective in computing similarity using $\mathbf{z}_u$ and $\mathbf{z}_v$. The function f is unknown but can be approximated by maximizing the log likelihood of observing neighbouring node of u given the embedding $\mathbf{z}_u$. This can be written down as the objective function below:

$$max_f \sum_{u \in \mathcal{V}} logP(N_r(u)|\mathbf{z}_u) \tag{1}$$

where $N_r(u)$ denotes a neighbourhood function. Let us define neighbouring nodes of u as nodes in the path walking from u using a random walk policy r.

Collect Training Samples using Biased Random Walk Starting from node u, all nodes in the same path starting from u were considered as sharing some similarities to u since it was within a predefined walking steps. Readers may foresee that the graph could be traversed in many fashions, by exploring neighboring nodes or wandering deeper and further away from the starting points. Here, the bias random walk was employed with two hyper-parameters: return parameter (p) and in-out parameter (q), follows [1].

In brief, if the current node was v, all edges to all neighboring nodes $N(v)$ were weighted according to parameters p and q. Let u be the previous node, and w be the next possible node, where $u \in N(v) \wedge w \in N(v)$. The edge (u, v) was assigned with the weight of 1, the edges (v, w) were assigned with the weight $1/p$

if $w \in N(u)$ else the weight $1/q$ was assigned to the edges. Finally edge weights were normalized, and the values conditioned their chance of being sampled as the next node.

Setting $p < 1$ encourages local exploration while setting $q < 1$ encourages global exploration since a lower value of p will increase probability of exploring nodes neighboring both previous node u and the current node v, while setting $q < 1$ encourages the next node to be further away from u (see Figure 2).

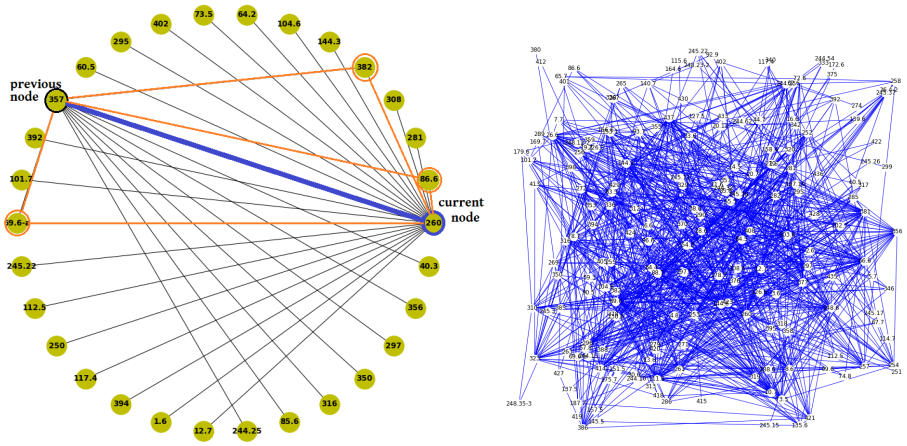


Fig. 2. Left pane: node 260 is the current node and all nodes connected to it are possible next nodes $n \in N(260)$. The edges (260, n) are assigned with the weight either 1, $1/p$, or $1/q$ (see text for detailed explanation). Right pane: an instance of a chorale graph used in our study.

One could collect nodes from many traversed paths starting from each node in the graph. Walk data collection was controlled by the number of steps in each walk and the number of repeated walks for each node. These walk data samples constituted a dataset for training *node2vec* models using CBOW and skip-gram approaches. These walks were also used to prepare negative sampling training data. In a negative sampling approach [3], any node pairs in the same path would be collected as positive (target, context) examples and node pairs not from the same path would be collected as negative examples.

4 Empirical Results

Two application cases are discussed in this section: (i) learning node embedding representation and (ii) learning node labels from neighbouring nodes. Both applications are based on the following assumptions, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: (i) $similarity(u, v) \approx \mathbf{z}_u \mathbf{z}_v$, for $u, v \in \mathcal{V}$; (ii) Node without label can be inferred

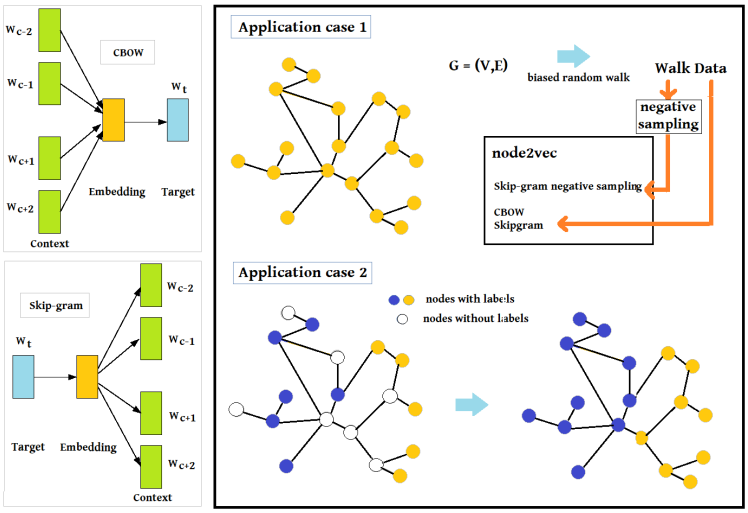


Fig. 3. Left column: architecture of CBOW and skip-gram; Right column: a graphical summary of application case 1 and 2.

from labels from neighbouring nodes. Figure 3 provides a graphical summary of our empirical study.

4.1 Application Case 1: Learning Node Embedding Representation

A chorale graph, consisting of 194 nodes 861 edges, and average node degree of 8.88, was created using similarities among the first six chords (as discussed in section 3.3). The construction of this graph can be conceptually understood as, starting with a complete graph, then removing edges that do not fulfil specified constraints from the chorale graph, and finally remove all isolated nodes.

Three node embedding algorithms were explored, the first model was the *node2vec* trained using the negative sampling tactic [1]. Negative sampling is a form of noise contrastive estimation (NCE) [18] that approximates the log probability of $P(w|\mathbf{z}_v)$. Instead of normalizing with all nodes in the graph i.e., $\sum_{w \in N_r(v)} log(P(w|\mathbf{z}_v))$, it is normalized using only k random negative samples i.e., $\sum_{i=1}^{k} log(\sigma(\mathbf{z}_v^T, \mathbf{z}_{n_i}))$. The negative $\mathbf{z}_{n_i}$ examples are randomly sampling over relevant nodes in the graph with a bias towards nodes with higher degree values.

The other two node embedding models were trained using two popular approaches CBOW and skip-gram from *word2vec* class [3] provided in the Gensim [4]. Algorithm 2 outlines the concepts of the methods used in this work.

Analyzing the Learned Node Embedding We evaluated the quality of learned node embedding from the three models by querying them for chorales

Algorithm 2 Learning node embedding using *node2vec_SGNS*, SG and CBOW

Let $Comp$ be a set of chorale compositions.
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} \subseteq Comp$ and $\mathcal{E} \subseteq \{\mathcal{V} \times \mathcal{V}\}$.
Let τ_u be an n-step random walk traversal starting from node u.

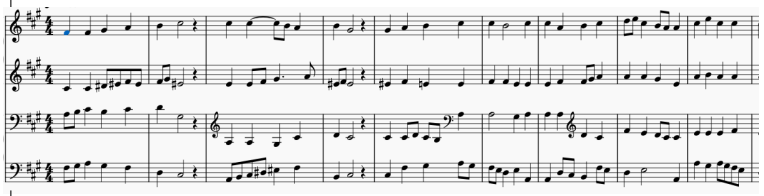
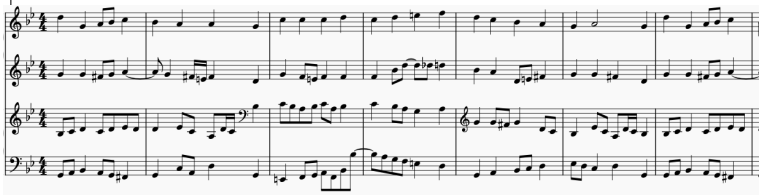
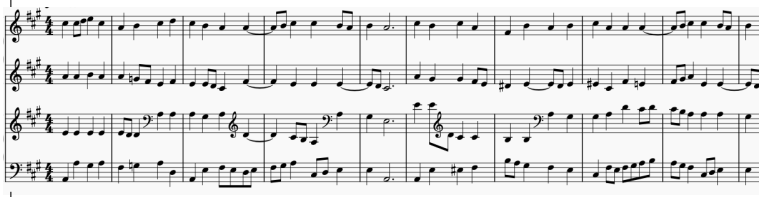
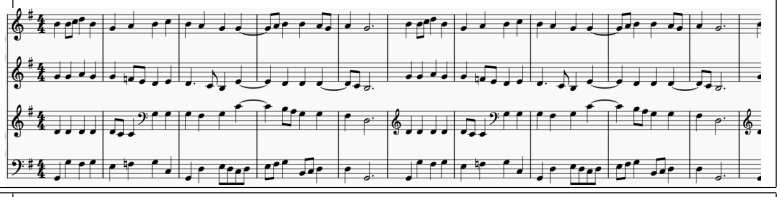
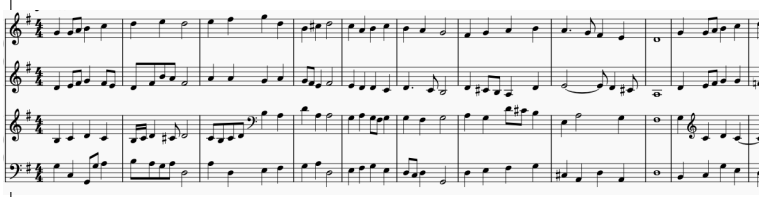
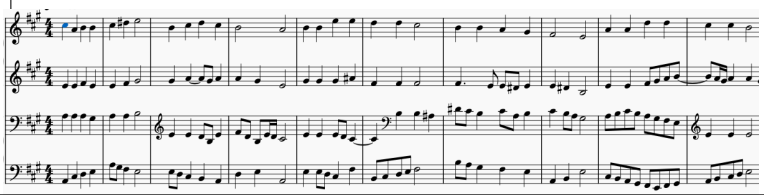
node2vec with skip-gram negative sampling
Let (u, w, lab) denotes target u and context w. If $lab = 1$ (positive example) then $u \in \tau_u$ and $w \in \tau_u$, if $lab = 0$ (negative example) then $u \in \tau_u$ and $w \notin \tau_u$.
Let D be the training data composed of positive and negative samples.
Let $\mathbf{Z} \in \mathcal{R}^{d \times |\mathcal{V}|}$ denotes an encoder matrix.
Let $u, w \in \mathcal{I}^{|\mathcal{V}|}$ denotes one hot encoding of node u and w, respectively.

procedure *node2vec_SGNS* (D)
  $initialize(\mathbf{Z})$ to random values $\mathbf{Z} \in \mathcal{R}^{d \times |\mathcal{V}|}$
  repeat epochs
    forall $(u, w, lab) \in D$
      $\mathbf{z}_u = \mathbf{Z} \cdot u$
      $\mathbf{z}_w = \mathbf{Z} \cdot w$
      $P(w|\mathbf{z}_u) = \frac{exp(\mathbf{z}_u^T \mathbf{z_w})}{\sum_{n \in \mathcal{V}} exp(\mathbf{z}_u^T \mathbf{z_n})}$
      $\mathcal{L} = -\sum_{u \in \mathcal{V}} \sum_{w \in N_r(u)} log(P(w|\mathbf{z}_u)$
        $\approx -\sum_{u \in \mathcal{V}} \sum_{w \in N_r(u)} [log(\sigma(\mathbf{z}_u^T, \mathbf{z}_w)) - \sum_{i=1}^{k} log(\sigma(\mathbf{z}_u^T, \mathbf{z}_{n_i}))]$
      $\mathbf{z}_w \leftarrow \mathbf{z}_w - \eta \frac{\partial \mathcal{L}^{(u)}}{\partial \mathbf{z}_w}$

Learning node embedding with CBOW, and skip-gram
Let D be the training data composed of walk $\tau_u, \forall u \in \mathcal{V}$
Let $c \in C$ be node in the walk τ_u, C denotes context nodes $C \subseteq \tau_u$

procedure *CBOW_SG* $(D, method)$
  $initialize(\mathbf{Z})$ to random values $\mathbf{Z} \in \mathcal{R}^{d \times |\mathcal{V}|}$
  repeat epochs
    forall $u \in \tau_u, C \subseteq \tau_u$ and $\tau_u \in D$
      if method **is** CBOW
        Given $P(u|C, \mathbf{Z}^C)$ where the expectation $\mathbf{Z} = \frac{1}{|C|} \mathbf{Z}_c^C \ \forall \ c \in C$
        $P(u|c, \mathbf{Z}) = \frac{exp(\mathbf{z}_c^T \mathbf{z}_u)}{\sum_{n \in V} exp(\mathbf{z}_c^T \mathbf{z}_n)}$
        $\mathcal{L} = -\sum_{c \in C} \sum_{u \in \mathcal{V}} log(P(u|c, \mathbf{Z})$
      if method **is** skip-gram
        $P(C|u, \mathbf{Z}) = \prod_{c \in C} P(c|u, \mathbf{Z})$
        $P(c|u, \mathbf{Z}) = \frac{exp(\mathbf{z}_u^T \mathbf{z}_c)}{\sum_{n \in V} exp(\mathbf{z}_u^T \mathbf{z}_n)}$
        $\mathcal{L} = -\sum_{c \in C} \sum_{u \in \mathcal{V}} log(P(c|u, \mathbf{Z})$

Table 2. Three examples of similar chorales suggested by the models, BWV148.6-BWV316, BWV253-BWV414, and BWV318-BWV355.

BWV	Excerpts from chorales
148.6	
316	
253	
414	
318	
355	

similar to a given chorale u. Since $similarity(u, v) \approx \mathbf{z}_u \mathbf{z}_v$, it was expected that $\mathbf{z}_v$ should represent the chorale v which was similar to chorale u according to the same similarity measures used for constructing the graph.

Hence, the three models i.e., skip-gram with negative sampling (SGNS), skip-gram (SG), and continuous bag-of-words (CBOW) were queried with all nodes in the graph. For each queried node, the ten most similar nodes were listed out. Common nodes between three possible model pairs SGNS-SG, SGNS-CBOW and SG-CBOW were counted. Ten-counts mean both models agreed perfectly, while zero-counts mean no matching node was found.

We report the mean values of common nodes as well as the mean similarity values of the top ten similar nodes suggested by each model. The similarity metric is based on the similarity of functional harmony as discussed in section 3.3. From Table 3, the model SG and CBOW agree well with each other, with an average of 6.88 common nodes for every 10 nodes. The model SGNS appears to suggest a different set of nodes, with an average of 3.79 and 3.57 common nodes for SGNS-SG and SGNS-CBOW, respectively.

Upon examining the mean similarity of the three model pairs, all pairs show comparable mean values. This means that even if SGNS suggests a different set of similar nodes (as those suggested by CBOW and SG), the similarity measures from all pairs are comparable. Finally, three sets of similar chorales (six chorales) according to the models are shown in Figure 2 for readers to evaluate.

Table 3. Summary of similarity performances evaluated using common similar nodes between models (top three rows), and average similarity of all suggested nodes (bottom three rows).

Mean common nodes between			Remarks
SGNS-SG	SGNS-CBOW	SG-CBOW	
3.79	3.57	6.88	p=1, q=1
3.76	3.45	7.14	p=0.7, q=1
3.67	3.51	6.94	p=1, q=0.7
Mean node similarity from			
SGNS	SG	CBOW	
8.22	8.82	8.79	p=1, q=1
8.20	8.80	8.78	p=0.7, q=1
8.27	8.87	8.78	p=1, q=0.7

4.2 Application Case 2: Learning Node Labels from Neighbours

Four chorale graphs used in this task were prepared using four different arbitrary similarity threshold values. Hence the four graphs had different number of nodes, edges, and average node degrees (see Table 4). All edges were labeled with similarity measures according our descriptions in section 3.3. Each chorale node in the graph was labeled as either *major* or *minor* mode.

For this application case, the four graphs were initialized then their node labels were randomly removed using the following rates 10%, 30%, 50%, 70%, and 90% from each graphs[2]. This was to emulate the common scenario of missing labels, partially labeled dataset. We investigated the collective classification approach which aggregate information from neighboring nodes to infer the missing labels. The aggregation of information was computed using the Algorithm 3.

Algorithm 3 Collective classification

Let W be a weighted adjacency matrix of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.Here weights denote similarity between nodes.
Let label of node u depends on its neighbouring labels: $P(y_u) = P(y_u|N(u))$
Let $y \in \{0, 0.5, 1\}$ denotes labels for class 0, 1 and 0.5 denotes no label.

procedure CollectiveClassification ($\mathcal{G}$)
  repeat iterations
    forall u in G
      $P(y_u) = \frac{1}{\sum_{(u,v)\in\mathcal{E}} W_{u,v}} \sum_{(u,v)\in\mathcal{E}} W_{u,v} P(y_v)$
    end for
    if $P(y_u) >$ class 1 threshold **then** $P(y_u) = 1$
    if $P(y_u) <$ class 0 threshold **then** $P(y_u) = 0$

Table 4. Summary of final accuracies after five iterations. Impressive improvements are observed from small chorale graphs with lower number of nodes, edges and average node degree. This is because a smaller chorale graph is constructed with a stronger similarity constraint (in our case) and therefore reinforce the *homophily* and *influence* concepts in the graph network.

Accuracy					Graph info.		
% missing labels					num.	num.	avr.
10	30	50	70	90	nodes	edges	degree
99.5	98.1	96.7	94.3	87.0	79	112	2.8
99.5	97.9	96.1	93.5	82.3	194	861	8.8
98.8	95.9	93.6	87.6	72.2	365	14556	79.7
96.3	89.6	79.2	67.1	56.2	383	73153	382

The underlying concepts here are *homophily* and *influence* in social network where characteristics of individuals within the same social group tend to corre-

[2] Four different graphs and each with different missing node label rates. Hence, there are 20 experiments, see Table 4.

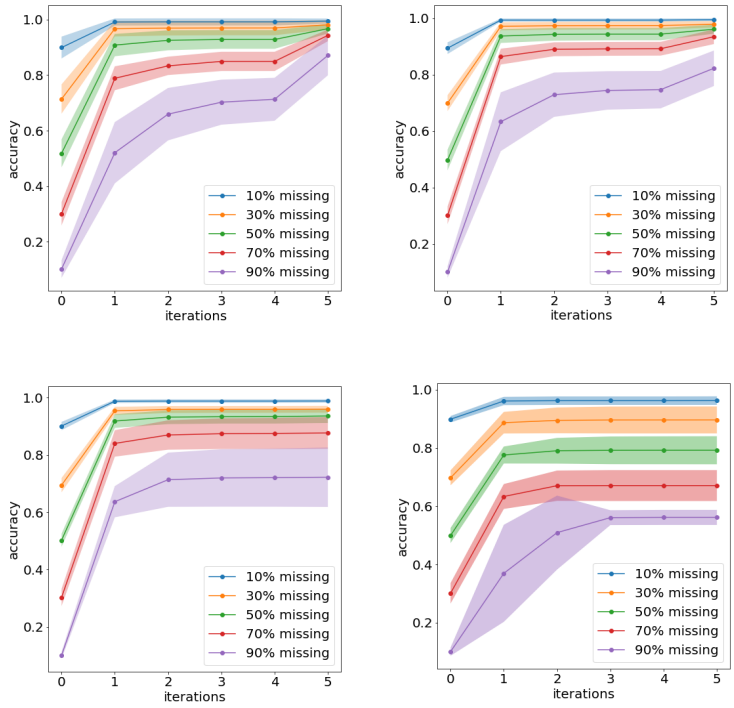


Fig. 4. Summary of results from four graphs with different rates of missing node labels after five iterations of the collective classification process.

late well. Table 4 summarizes the accuracy of all 20 outcomes. Each outcome is averaged from 30 runs and their standard deviation is represented by their respective shaded areas. Figure 4 shows the plot of nodes that were correctly labeled after five iterations of message aggregation from neighbouring nodes.

5 Conclusion & Future Direction

This work explores graph representation of chorales. Three hundred and eighty three Bach chorales were prepared using Music21. Each node in the graph represents a chorale composition and each edge that connects two chorales was weighted with the similarity between them. Two application cases were explored in this study, (i) learning node embedding mapping nodes in a chorale graph to an embedded space where three algorithms were explored : node2vec, CBOW and skip-gram.; (ii) learning chorale mode labels from neighboring nodes using collective classification.

In the first application case, the results show that node2vec (trained using negative samplings) seems to suggest a different set of similar nodes from those suggested from CBOW and SG. However, the similarity measures appear comparable. This implies that the approach is applicable to various music information retrieval tasks. In the second application case, the missing labels can be classified correctly with a high accuracy rate. The severely missing labels case such as 90% missing labels (10% accuracy) could see the correct classification at 56%-87%. This is an increment of 46%-77% from 10% correct labels at the initial stage.

In future works, we will explore various graph neural network designs for other task such as query by humming, genre classification, music synthesis, etc.

Acknowledgments We would like to thank the GSR office for their partial financial support given to this research.

References

1. Grover, A., and Leskovec, J.: node2vec: Scalable feature learning for networks. arXiv:1607.006533v1 (2016)
2. Cuthbert, M., Ariza, C.: music21: A toolkit for computer-aided musicology and symbolic music data. In: Proceedings of the International Symposium on Music Information Retrieval, pp. 63742, (2010)
3. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
4. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45-50 (2010)
5. Huron, D.: Design principles in computer-based music representation, In: Marsden, A. and Pople, A. (eds.) Computer Representations and Model in Music. pp. 5-40, Academic Press. (1992)
6. Smaill, A., Wiggins, G. and Harris, M.: Hierarchical music representation for composition and analysis. Computers and the Humanities **27**(1): 7-17 (1993)
7. West, R. and Howell, P. and Cross, I.: Musical structure and knowledge representation. In: Howell, P., West, R., Cross, I. (eds.) Representing Musical Structure, chapter 1, pp. 1-30, Academic Press, (1991)
8. Courtot, F.: Logical representation and induction for computer assisted composition. In: M. Balaban, K. Ebcioglu, and 0. Laske, (eds.) Understanding Music with AI: Perspectives on music cognition, chapter 7, pp. 157-181. The AAAI Press/The MIT Press. (1992)
9. Good, M.: MusicXML for notation and analysis. In: Hewlett, W.B., Selfridge-Field, E. (eds.) The Virtual Score: Representation, Retrieval, Restoration. Computing in Musicology 12, pp. 113-124, (2001)
10. Lerdahl, F. and Jackendoff, R.: A Generative Theory of Tonal Music. The MIT Press. (1983)
11. Chew, E.: Mathematical and Computational Modeling of Tonality: Theory and Applications. Springer. (2014)
12. Longuet-Higgins, H. and Steedman, M.: On interpreting Bach. Machine Intelligence, 6, 221-241. (1971)

13. Holland, S.: Artificial Intelligence, Education and Music: The use of Artificial Intelligence to encourage and facilitate music composition by novices. Ph.D. thesis, The Open University, Milton Keynes. (1989)
14. Cambouropoulos, E.: The harmonic musical surface and two novel chord representation schemes. In: Meredith, D. (eds.) Computational Music Analysis. Springer, Cham. pp. 31-56, (2016)
15. Cope, D.: Virtual Music: Computer Synthesis of Musical Style. MIT Press, Oxford (2001)
16. Orio, N. and Rodà, A.: A measure of melodic similarity based on a graph representation of the music structure. In: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009) pp. 543-548. (2009)
17. Jeong, D., Kwon, T., Kim, Y., and Nam, J.: Graph neural network for music score data and modelling expressive piano performance. In: Proceedings of the 36th International Conference on Machine Learning. PMLR 97:3060-3070. (2019)
18. Goldberg, Y., and Levy, O.: word2vec Explained: Deriving Mikolov et al.'s negative sampling word-embedding method. arXiv:1402.3722v1 (2014)