Exploring chord embedding spaces between musical composers and eras

Authors: Adina Ciubotaru, Sergei Kliavinek, Marcus Gruneau
Mentors: Dr. Fabian Moss, Dr. Andrew McLeod
Hosting lab: Digital musicology, music cognition and language
EPFL, Lausanne, Switzerland

Abstract—**In recent years, natural language processing algorithms have become more sophisticated and powerful due to the development of deep neural networks. In this paper we have decided to exploit these powerful tools in order to generate an embedding space of musical chords using two sets of annotated corpora, containing musical scores from different eras. The resulting embedding spaces were visualized in different ways and show that the trained models managed to catch fundamental similarities between different chords. The possibility of predicting subsequent chords in context using various word representation models and a recurrent neural network was also investigated. It is shown that even using a small neural network, it is possible to achieve good results.**

I. Introduction

Natural language processing (NLP) [5] is a sub-field of computer science, where the objective is to program and build a model such that computers can understand and interpret natural language to perform data analytical tasks. Natural language can be anything from regular speech to singing as well as come in the form of musical notation, such as a set of chords used to build up a musical score. However, only recently large labeled corpora combining score and harmonic annotations have been made available for research. The Annotated Beethoven Corpus (ABC) [8] represents a data set of harmonic analyses of all Beethoven's string quartets. Since its publication, the data set was enriched with chord annotations from the pieces of 23 more composers, that lived during the *common practice period*[1]. By manipulating these corpora, one can apply NLP techniques in order to create a model that can be interpreted by a computer. A previous approach in this sense involved a study on the structure of the chord lexicon [7] using the initial form of the data set ABC. This model can be trained using machine learning methods in order to find similarities between the different elements (chords) in the data set by generating an embedding space, where each chord is mapped to a vector. The resulting embedding spaces can be compared in different ways, for example by looking at the similarities of the neighborhoods for each embedded vector per embedding space. In the following sections we will describe how we pre-processed the data, applied different ML-based models and the interpretation of our results.

[1]Between the years 1600-1910 (Baroque, Classical and Romantic era)

II. Data

A. Understanding the data

This project makes use of two data sets that were pre-processed in different ways, originating from the enriched ABC data set. They contain the same series of chord progressions from 24 composers, summing up for a total of approximately 68000 chords, similar to words in a text. The corpora are split by composer into separate files, where each row represents a progression of chords. The progressions can be distinguished by the mode of their global key (minor or major), which means that an entire musical piece can span over several rows if its global key changes.

In the first data set which we denote as A, each chord in a progression has the format ROMAN_NUMERAL:MODE. An example of a progression where the global mode is major would be: "MAJOR; I:MAJ, IV:MAJ, II:MIN". The vocabulary consists of 81 unique chords in major and 77 unique chords in minor.

The second data set which we denote as B, is closer to the original annotations from the ABC since it treats applied chords (chords that prepare or imply another chord [7]) differently. While data set A reduces applied chords to single chords, data set B preserves the function of the applied chord and the tonicized chord, as can be seen in the third chord in this example: "MAJOR; I, ii, V/IV". Here the major chords are uppercase and the minor ones are lowercase. In this data set there are 321 unique chords in major and 317 unique chords in minor.

An augmented or diminished chord in data set A can be marked as "II:AUG" respectively "II:DIM", while in data set B we find "II+" respectively "iio". Characters "#" and "b" mark sharp and flat chords in both data sets.

B. Data pre-processing

In order to use Word2Vec (Subsection III-A), we replaced all characters in each chord that can't be part of a token (":","#","+","/") when using a simple pre-process tokenizer. After training the models we reversed the process and mapped each chord back to its normal representation for visualization purposes.

We decided to further split each data set into two subsets based on the global key of each progression because of the natural difference in sound between the two. [1]

For analytical purposes, we decided to split the minor and major subsets in three ways to obtain different inputs for the machine learning models which are introduced in (Subsection III-A): No splitting, treating each composer separately and grouping composers based on which era they lived during (Baroque, Classical or Romantic).

III. Models and Methods

In this section we describe the different models and methods that were applied to generate our results and why we chose them.

A. Word2Vec

One of the most popular word embedding libraries that exists is called Word2Vec [6] (W2V), which uses neural networks in order to learn relationships between each word (chord in our case). We chose to use the Gensim[2] implementation because we found that it was well documented and simple to include in our Python machine learning pipeline.

The parameters used when training the W2V model (for the subsets that include all composers or eras) were $size = 20$, $min\_count = 50$, $window = 1$ and $sg = 0$ (default). The $size$ parameter adjusts the dimension of the embedding space. This was manually tuned to 20 which seemed to give us a good spread between the vectors in the embedding space. The documentation states that "Bigger size values require more training data, but can lead to better (more accurate) models. Reasonable values are in the tens to hundreds.". This is important because our vocabulary for data set A is small (in the hundreds) compared to a large language corpora such as the English language which can contain up to hundreds of thousands of words. The $window$ parameter is according to the documentation the "maximum distance between the current and predicted word within a sentence." This parameter was set to 1 because we assume that a chord depends mostly on its immediate neighbor. The min_count parameter filters out any chords which have a lower frequency than the set value. We chose 50 because it allowed us to keep the most common chords while filtering out the noise generated by rare chords. We used a smaller value (15) for the min_count parameter when the subsets with separate composers were trained because some of them have a small corpus. The sg parameter determines if W2V will use a Skip-gram (SG) or continuous bag of words (CBOW) model. The main difference between these two is the way they are trained: While the neural network used by the CBOW implementation tries to predict a word given a context of words, SG tries to predict the context given single words. We decided to use the (default) CBOW model because the generated embedding spaces were almost identical when compared to the ones created when using a SG model.

There are several ways to compare and visualise the embedding space generated by each model. A common method is to project the entire embedding space onto a 2D plane using FastICA [3]. Even though the resulting plot can be difficult to interpret for our purposes, the idea is that the more similar two chords are, the closer they should be on the projected space.

B. LSTM training

Another interesting topic is the prediction of the next chord given a set of previous chords. This is important from several points of view:

- It allow us to compare our Word2Vec model with the classical one-hot vector representation.
- It allows us to determine if there are any patterns in the different works of composers or any stable combinations of chords.
- With the capability of predicting the next chord it is possible to create a recommendation system, which could allow musicians to use it for generating new chord sequences.

To solve the problem of predicting the next chord, a neural network of the LSTM type [2] (Long short-term memory) can be used. This is a subspecies of recurrent neural networks, which is actively used in the tasks of predicting and analyzing time series and recognizing human speech. The main advantage of such a network is that the "forgetting" of information is regulated, not by the usual activation function, but by special gates that allow storing information for a long time. As a result, it is possible to get a dependence on several previous chords. This requires a large amount of data due to the complexity of the model.

We chose to use a simple architecture, in which the number of input and output neurons is equal to the dimension of embedding space containing a single 100 neuron hidden layer. The neural network was trained for 10 epochs.

During the training phase, three different embedding spaces generated by Word2Vec with a dimension of 20, 77 and 100 were used. As a reference, a one-hot vector representation with 77 dimensions was also used. The predictions based on the 2–5 previous chords were analyzed: If the next chord was the nearest neighbor of the predicted one, the prediction was scored as successful, otherwise—as unsuccessful.

IV. Results

A. Word2Vec

We plotted the minor and major progressions separately using all models resulted from the grouping possibilities

[2]https://radimrehurek.com/gensim/

described in Subsection II-B. Each chord point was scaled proportionally to its log-frequency in the corpus making the common chords stand out in the plots.

In Figure 1 the chords that are in the same key as the tonic (I:MAJ, II:MIN, III:MIN, IV:MAJ, V:MAJ, VI:MIN, VII:DIM) and which are also the most frequent, are close together except VII:DIM and III:MIN. Common chords are usually grouped together, as we noticed in most of the FastICA plots regardless of the subset used for training the models. This can be explained by the fact that common chords also occur together in progressions.

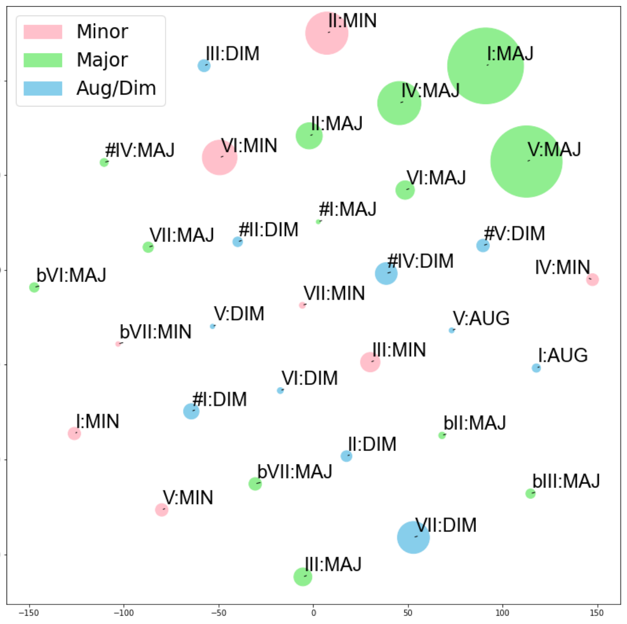

Figure 1. FastICA projection of the embedding space (major context) using dataset A.

Other than studying the chords for each individual model we also compared the different models to each other. To do this we used the similarity function from the Word2Vec gensim module which returns a list containing the top n most similar chords given a single one, along with their similarity scores, which is based on the cosine similarity (because of the high dimension of the embedding space). Each comparison involved either several major models or minor ones. We compared all data sets (using the models that were trained on all the composers), all eras and all composers to each other. To get a better overview we created a clustered heat map using the similarity matrix between each chord in an era and compared the resulting plots.

Dividing the data sets by eras, we notice the same patterns repeating through each period, namely the common chords in the major models (I:MAJ, II:MIN, IV:MAJ, V:MAJ, VII:DIM) being the most similar and sticking together as in the left cluster formed in Figure 2.

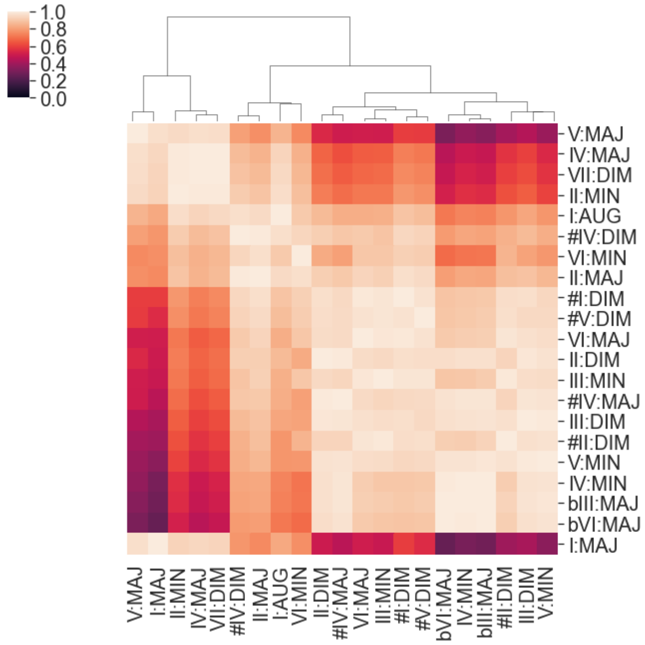

Figure 2. Clustered heatmap based on the similarity between the chords of the classical major model

Another approach we adopted was to apply the K-means clustering [4] algorithm over the different embedding spaces generated from the models which were trained across all composers. Since our vocabulary is relatively small, we could iterate over a narrow range of values for the K parameter in order to manually choose the one that produced the most meaningful split.

Figure 3 is the fastICA projection of data set B in minor context. It supports the clustering obtained by K-means on the same data set, for example the cluster formed around the point $(-0.1, 0.0)$ which maps to each chord in cluster 3, or the cluster which is concentrated around the point $(0.2, -0.05)$ which maps to cluster 5.

Figure 4 shows the distance between each pair of clusters. We can assign a cluster a label signifying its repetitive tonicized chord: cluster 1 – v, cluster 3 – III, cluster 6 – iv, cluster 7 – VII, cluster 8 – VI, cluster 9 – bII. Clusters 2 and 4 can't be categorized in this way since they do not signify any repetitive tonicized chord. Cluster 5 contains the common chords in minor which are also displayed in the right side of Figure 3. This result is meaningful because we observe that most clusters contain applied chords with the same tonicized chord.

In Figure 4 clusters 1, 3 and 7 are grouped together and separated from all the others, which makes sense in the minor key, because they are related to the relative major key. The left side of Figure 3 presents the same separation for the tonicized chords: III – at the top, VII – in the middle and v – at the bottom.

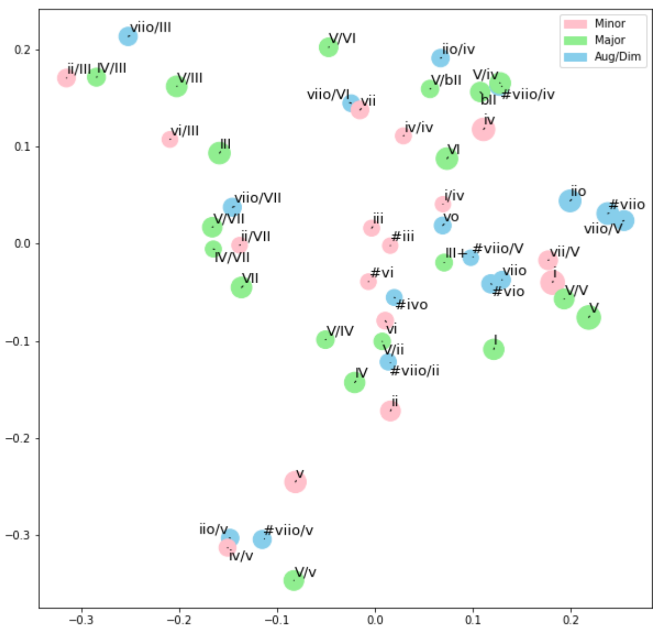

Figure 3. FastICA projection of the embedding space (minor context) using dataset B.

- Cluster 1: v, iio/v, iv/v, V/v, #viio/v
- Cluster 2: VI, #vio, I, viio, vo, i/iv, #viio/V, III+
- Cluster 3: III, viio/III, ii/III, V/III, vi/III, IV/III
- Cluster 4: IV, V/IV, #viio/ii, V/ii, ii, #iii, vi, iii, #vi, #ivo
- Cluster 5: i, iio, V, #viio, viio/V, V/V, vii/V
- Cluster 6: iv, #viio/iv, iio/iv, iv/iv, V/iv
- Cluster 7: V/VII, viio/VII, VII, ii/VII, IV/VII
- Cluster 8: viio/VI, V/VI, vii
- Cluster 9: bII, V/bII

B. *LSTM results*

During the study, three models were considered:

- Word2Vec input with Word2Vec output
- Word2Vec input with one-hot vector output
- One-hot vector input with one-hot vector output

In all three cases we found that there exists a dependence between a chord and the previous chords. The prediction accuracy was between 35-40 % (for the W2V in/out model), which we believe is successful in the presence of several dozen different chords. Since this was applied across multiple composers of different eras we come to the conclusion that there exists stable combinations of chords for each era.

Another interesting result is that the prediction results seem to be weakly dependent on the length of the previous sequence of chords. They were only slightly different when comparing the predictions for 2, 3 or 4 of the previous

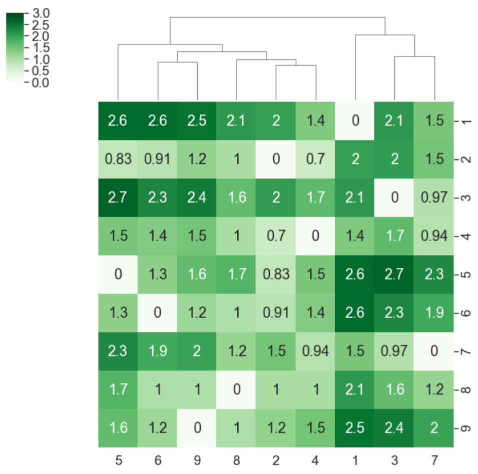

Figure 4. Clustered heatmap based on the distance between the clusters formed on the minor model.

chords (the difference fluctuating within 5 percent). Stable dependencies are clearly observed only for pairs of chords.

As a final step, we compared the LSTM model when trained using a Word2Vec embedding vs. a one-hot vector encoding. A one-hot vector model is simple and it is interesting for us to compare it and see if we over complicated things by using W2V. The best result using the Word2Vec embedding was about 35-40 %, which is almost double performance when comparing it with the result of the one-hot model, which was less than 20 % (and higher than predicting a one-hot vector from Word2Vec vectors $\approx$ 20%). These results can be explained by Word2Vec having a connection between the relative position of the chords and their similarity (in a musical sense). Therefore, gradient descent would make more sense with this model than with a one-hot vector representation, in which our chord arrangement is fairly random. The result of predicting using a Word2Vec embedding is weakly dependent on the dimension of the chord representation space which causes it to fluctuate between 35-40%.

V. Summary

Our main objective was to explore how well the different proposed models would fit to the initial data set of annotated chord corpora and if we could find some interesting patterns, since most of the learning we applied is unsupervised. Using the proposed Word2Vec model to generate a word embedding space of chords, we discovered that it did cluster meaningful chords with each other. This was easy to visualize using K-means clustering, as similar chords were put in the same groups. We also experimented with training a LSTM model to predict the following chord given a sequence of previous ones, which gave us an accuracy of around 40%.

Acknowledgements

We thank Dr. Fabian Moss, Dr. Andrew McLeod and the Digital and Cognitive Musicology Lab at EPFL for providing the dataset, supervising the project and advising us throughout.

References

[1] Edward Aldwell and Carl Schachter. *Harmony and Voice Leading. Vol. 2*. 1979.

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[3] A. Hyvärinen and E. Oja. "Independent component analysis: algorithms and applications". In: *Neural Networks* 13.4 (2000), pp. 411–430. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(00)00026-5. URL: http://www.sciencedirect.com/science/article/pii/S0893608000000265.

[4] Stuart P. Lloyd. "Least squares quantization in pcm". In: *IEEE Transactions on Information Theory* 28 (1982), pp. 129–137.

[5] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press, 1999. URL: http://nlp.stanford.edu/fsnlp/.

[6] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[7] Fabian C. Moss et al. "Statistical characteristics of tonal harmony: A corpus study of Beethoven's string quartets". In: *PLOS ONE* 14.6 (June 2019), pp. 1–16. DOI: 10.1371/journal.pone.0217242. URL: https://doi.org/10.1371/journal.pone.0217242.

[8] Markus Neuwirth et al. "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets". In: *Frontiers in Digital Humanities* 5 (2018), p. 16. ISSN: 2297-2668. DOI: 10.3389/fdigh.2018.00016. URL: https://www.frontiersin.org/article/10.3389/fdigh.2018.00016.