Enchordings - Harmony Embeddings

André Ghattas, Beatriz Borges, Irene Petlacalco Barrios
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—In the present, Machine Learning (ML) techniques are being used in digital musicology. As stated in [1], an increasing number of digital music resources are available. However, digital musicology still lacks large labeled corpora combining score and harmonic notations. A data set called The Annotated Beethoven Corpus (ABC)[2] was elaborated to address this gap. With the elaboration of this data set, the question of how to represent it in a machine-learning-friendly manner arises, as most techniques usually take as input numerical data. This project's aim was thus to employ ML methods to compute a meaningful numerical representation of the harmony entities, chords, of the ABC data set. Specifically, we analysed various possible representations of harmonic entities, as well as several possible dimensional spaces in which to embed them. Finally, we conclude that a skip-gram model, for intermediate chord representations, with a vectorial representation dimension of 3, was the best technique to represent harmony chords.**

I. Introduction

Lately, big data techniques have become popular in the field of music theory[1]. One example is using tonal and harmonic relationships in music for exploring musical properties and comparisons across pieces. Another example is the use of deep learning models for chord sequence generation[3]. All these applications need a reliable source of data and an accurate way of representing it so that ML models can learn from it.

In the Natural Language Processing (NLP) field, one precise way of representing text is by using word embeddings, which are semantic vector space models that represent each word with a real-valued vector[4]. Continuous bags of Words (CBoWs), skip-grams, dimensionally-reduced co-occurrence matrices, neural networks, and other various ML techniques have been useful to build robust word embeddings.

The question that this report aims to address is if the ML techniques used for word embeddings could be helpful to represent music, particularly tonal harmony. To approach this task, a large data set of expert-generated harmonic labels which was provided by the Digital and Cognitive Musicology Laboratory at EPFL is used[1].

In this report, the steps for building a tonal harmony embedding are described. First, a brief data exploration is performed. Then, the embeddings are computed by using ML techniques, using the Python `Keras` library[5].

Lastly, an extensive visualization- and clustering-based analysis for determining whether the resulting embeddings can meaningfully represent harmony chords is done, with recourse to several `sklearn` methods. The visualization methods used include t-SNE, Locally Linear Embedding and PCA. The unsupervised clustering methods used include K-means clustering[6], DBSCAN and spectral clustering. Finally, an expert assisted clustering approach, consisting of labeling data points such that similar ones share the same cluster, is taken.

II. Data Exploration

The aforementioned data set, *The Annotated Beethoven Corpus (ABC)*, consists of harmonic analysis of all Beethoven string quartets. The harmonic analysis are encoded in a human and machine-readable format (MuseScore XML). The data set has 28,095 observations and 19 features. Each of these features aims at representing one aspect of the harmony at any given point of all of the sixteen string quartet pieces. A brief description of some of the variables used in this project can be found in table I.

TABLE I
Layout of a Subset of variables in the ABC data set[1]

Feature	Description
key	indicates the global key of the piece
pedal	indicates the beginning of a pedal tone
root	denotes the root of the chord relative to the current key
chord form	denotes the form of a chord
figure bass	allows the inference of the bass note of the chord
changes	additional changes to the chord
pedal end	indicates the end of a pedal
phrase end	annotates the end of a phrase

The data contain more than 27,500 unique chords. Approximately 50% of them appear only once, making for a heavy tailed distribution, overall. Only 0.1% of the chords appear more than 160 times. The 10 most frequent chords can be seen in Figure 1

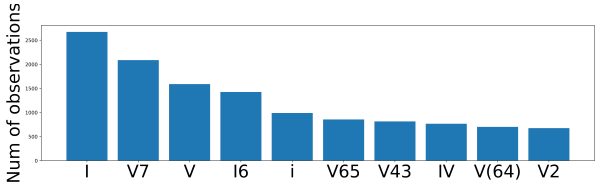

Fig. 1. Top 10 most frequent chords in the ABC data set

This data set provides a starting point for processing different representations of chords. Such representations are defined by their feature selection choices. For this project three different possible representations were used.
The first one, called *Full Representation (FR)*, is a concatenation of all of the features: *key, pedal, root*, which is the only required part of any chord symbol, *chord form, figure*

bass, changes, pedal end, and *phrase end*[1]. It has more than 27,598 unique symbols.
The second representation, *Root Representation (RR)*, is simply the *root* (including sharp and flat symbols) feature of any given chord. This representation has 41 unique symbols. An example of a root representation's sentence can be observed in figure 2.

```
['I', 'V', 'I', 'V', 'I', 'vi', 'IV']
```

Fig. 2. Root representation chord sentence

The last representation, called *Intermediate Representation (IR)*, is defined by concatenating the *chord form, figure base, changes, relative root* and *root* features. This representation has 1,116 unique symbols.

In the following sections, chord embeddings will be trained for each of the previously described representations.

III. Data Preprocessing

A. Typical NLP Pipeline

As previously mentioned, ML models take as input numerical data frames. In NLP, one main source of data is text files. A manner of transforming this kind of data into a machine-learning-friendly format is by computing word embeddings. Such embeddings have to map each word to vectors of real numbers while trying to retain all the information that the word provided, including semantics and similarities within the text [4].
Before training embeddings, the text files need to be pre-processed as not all its parts offer meaningful information. One commonly used pre-processing pipeline includes first the removal of punctuation, special characters, and stop words. Then, each word is lemmatized (or, alternatively, stemmed), changing each word into its root, in order to reduce inflectional forms [7], and the amount of meaningless information kept. Lastly, the pipeline tokenizes each sentence in the text.

B. Harmony-directed Pipeline

The previous preprocessing pipeline is not transferable to chords. Firstly, if special characters were to be removed, essential things about chords would be removed as well. For instance, the double backslash represents the ending of a phrase at that chord. Another key difference is that chords cannot be stemmed or lemmatized. Put a different way, it can be argued that the chord representations are already pre-processed, as apart from conscious feature selection, no other simplifications to the data can be made.
The provided data set includes a set of features containing the harmony for several string quarters. In order to capture the meaning (and consequent similarity) of each chord, sentences were created from the data. The method used was to split each chord sentence whenever a movement of a piece ended, (that is, when the 'measure' feature resets to value 1). A list of token sequences was the result. Thereupon, a dictionary for the integer mapping of the different tokens was created, allowing the encoding of a chord sentence into an integer sequence. This method was applied for the three music representations described in section II.

IV. Machine learning methods

In this section, several Machine learning models employed to generate and visualize embeddings are described. Methods like CBoW, skip-grams, and GloVe are well-known methods for embeddings generation. PCA, t-SNE, and LLE are methods for reducing the dimensionality of usually high-dimensional data sets, for example for visualizion purposes. Both types of methods can be usefully applied to our own harmony dataset. However, there is a difference between the two groups of techniques. One key difference is that the models trained for chord embeddings generation try to predict a determined chord's neighbor while reducing the data's dimension, while visualization methods like PCA and t-SNE are more suited towards visualizing data. In this section, the ML methods used for this project are briefly described.

A. Training embeddings

1) Skip-grams: This technique was originally created for word embeddings. As opposed to an N-gram model, which treats words as atomic units, the skip-gram method takes into account the notion of similarity between words. This model was created to predict words within a specific range, or context, before and after a target word. Its training objective is to find word representations that are useful for predicting the surrounding words in a sentence or document. The model is a Neural Network with an input layer, an embedding layer, and an output layer. At the input layer, each word is encoded with a number from the words' previously determined dictionary. The input layer is then projected to the embedding layer, which consists of different shared nodes for all the words. Finally, each node in the embedding layer projects the weight to the output layer, where the probability distribution over all the words in the vocabulary is computed by using a `Softmax` activation function[8]. A simple diagram of the different layers previously described can be found in Figure 3

B. Visualizing embeddings

1) Principal Components Analysis (PCA): PCA is a sequence of projections of the data, mutually uncorrelated and ordered in variance. [10]. This unsupervised method reduces the dimensionality of a matrix X by rotating it such that the rotated features are statistically uncorrelated [11]. The previous is achieved by constructing the *singular value decomposition* of X. One of the most common applications of PCA is visualizing high-dimensional data sets.

2) t-SNE: Part of the visualization class called `manifold learning` algorithms, it allows much more complex mappings and often provides better visualizations. The method tries to find a two-dimensional representation of the data that preserves distances between points as best as possible. The algorithm to achieve this is the following. It

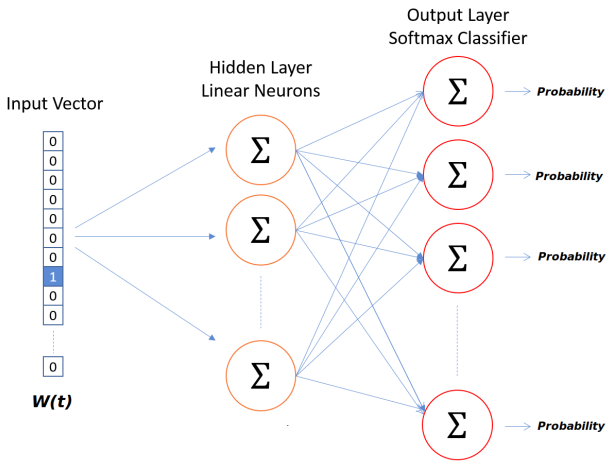

Fig. 3. Skip gram Neural Network layers [9]

starts out by giving a random two-dimension representation for each data point. It then tries to make points that are close in the original features space close in the two dimensional space and points that are far apart in the original feature space farther apart. This method is used for exploratory data analysis but is rarely used if the final goal is supervised learning [11].

3) Locally Linear Embedding (LLE): Also part of the `manifold learning` class, it seeks a lower-dimensional projection of the data which preserves distances between local neighbors. When the number of neighbors is greater than the number of input dimensions, the matrix defining each local neighborhood is rank-deficient. A way to address this situation is by applying an arbitrary regularization parameter [12].

C. Finding structure

1) Clustering: In order to try to unearth some structure in the embeddings we obtained, we applied various clustering algorithms to them, such as K-Means, DBSCAN and Spectral Clustering. The results had visual meaning as they showed structure in two and three dimensional space. However, this structure did not provide any semantic interpretation. This led us to plot our embeddings in two and three dimensions, and instead to pseudo-cluster, through colors, features with semantic meaning. As such, we assigned similar colors to data points with similar properties (e.g. frequency and log frequency in corpus, major/minor chords, root of the chord).

V. Training Enchordings

As mentioned before, the goal of this project is to compute a meaningful numerical representation of harmony chords. As stated in section IV, skip-gram models are useful for word embeddings. This model has proved to be useful for chord embeddings as well. Moreover, because of the model's nature, the resultant numerical representation will have the notion of similarity between harmony entities, allowing the grouping of similar chords.

For the three harmony representations introduced in section II, a skip-gram model with a window of size two was trained, meaning that the model tries to predict two chords before and two chords after the target chord. The model uses a `sparse categorical crossentropy` loss function, an `accuracy` score, and an `Adam` optimizer. To give more importance to future chords, a non-uniformly weighted loss function was employed.

The model has some hyper-parameters to be tuned, like the learning rate, the dimension of the embeddings, and the weights for the loss function. Using cross-validation and our advisors' expertise in the subject, a learning rate of 5×10^{-4} and several dimensions such as $2, 3, 4, 16, 32, 64, 128, 256$ were tested.

During training, the number of epochs was deliberately chosen so as to allow over-fitting. This is because the model corresponding to the best validation accuracy ends as the only one kept.

VI. Results

The loss curve in Fig. 4 offers insights into the model training. We can see that the loss for the validation and the training sets decreases until we reach the third epoch. The training loss continues to decrease after that whereas the validation loss increases. Hence, we can conclude that at this point our model starts to over-fit and the the optimal model is the one at the third epoch.

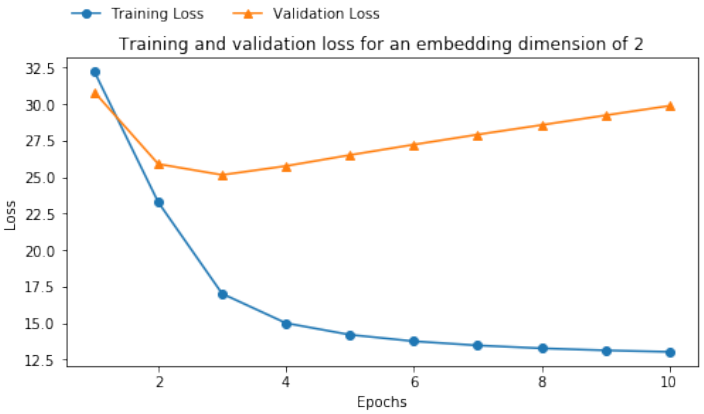

Fig. 4. Example of loss curve (embedding dimmension of size 2)

After testing the different embedding sizes on each of the representations, the best results across all of the ten model's epochs are summarized in Table II. A bold format was used to display the best scores for each music representation.

Using three as embedding size gives the minimum loss for the three representations. It also provides the maximum accuracy score for the root and intermediate representations. Although a better accuracy score is given by 64 number of dimensions for the full representation, we choose three dimensions for the three representations.

TABLE II
BEST RESULTS FOR EACH OF THE REPRESENTATIONS AND EMBEDDING SIZES TESTED

Dim/MR	Loss			Accuracy		
	FR	IR	RR	FR	IR	RR
2	**48.69**	**25.14**	**10.71**	0	**0.31**	0.33
3	48.70	25.39	10.94	0	**0.31**	**0.45**
4	48.72	25.41	10.86	0	**0.31**	0.35
16	48.77	28.51	10.90	0	**0.31**	0.37
32	49.05	29.71	11.13	0	**0.31**	0.37
64	49.49	30.83	11.35	**1e-3**	0.30	0.36
128	49.64	32.16	11.67	2e-3	0.30	0.36
256	49.87	33.36	12.03	2e-3	0.29	0.37

Though Table II presents interesting information, it must be highlighted that the columns cannot be directly compared. This is due to the fact that they tackle distinctly different tasks - while all of them are skip-gram models, the vocabulary sizes differ by at least one order of magnitude across any two given representations.

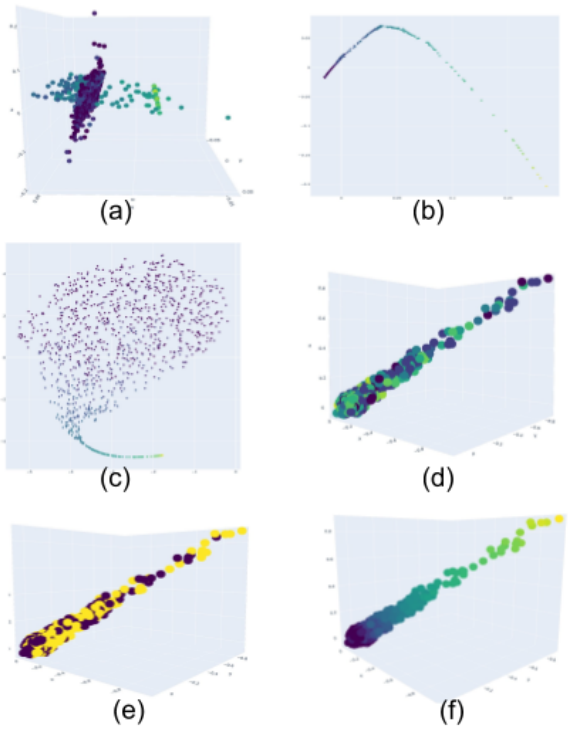

Fig. 5. Plots obtained (embedding dimension of size 3)

In Fig. 5, (a) and (b) are LLE visualizations of the embeddings and (c) is a t-SNE visualization of the embeddings. The embeddings were made for the intermediate representation of the chords, which we obtain at the end of our pipeline for embedding dimension 3. We can clearly see that the data points form a structure as they are not uniformly distributed over the plot. However, this structure does not imply a semantic interpretation which is why we made the three other plots in Fig. 5.

The coloring in the last 3 images, (d), (e) and (f), reflect the clustering by common chord root, common major or minor nature of the chords, or by their (logarithmic) frequency. They were plotted in three dimensions directly as their embedding dimension is of size 3. As we can see, though it's hard to find structure with the first two colorings, the one based on frequency correlates surprising well with the embeddings themselves. This provides us with an important insight - the structure found by our embedding model highly values the frequency with which any given chord features on the corpus.

VII. CONCLUSIONS

In this project, several machine learning models were trained for chord embeddings generation and visualization. In the end, a skip-gram model, for intermediate chord representations, with a vectorial representation dimension of 3, was the best technique to represent harmony chords. The numerical representation obtained reflects, most of all, their frequency in the corpus, as chords with similar number of appearances are grouped more closely together in the representation space.

VIII. FUTURE WORK

We used the subset accuracy score as an error metric not only to establish a first baseline but also because it can be interpreted as an objective function to find the mode of the joint probability of label sets y (given instances x). However, F1-measure maximization is often preferred to subset accuracy maximization because it is less susceptible to very large numbers of label combinations and imbalanced label distributions[13].

A further extension to our our project could also be generative algorithms implementation. These would consist mainly of Recurrent Neural Networks-based architectures (RNN's) for generating music.

IX. ACKNOWLEDGEMENTS

We thank our supervisors for their rich knowledge, indispensable advice and continued guidance and expertise throughout this project.

REFERENCES

[1] F. C. M. Markus Neuwirth, Daniel Harasim and M. Rohrmeier, "The annotated beethoven corpus (abc): A dataset of harmonic analyses of all beethoven string quartets," July 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fdigh.2018.00016/full

[2] ——, "The annotated beethoven corpus (abc): A dataset of harmonic analyses of all beethoven string quartets." [Online]. Available: https://github.com/DCMLab/ABC

[3] D. H. Ching-Hua Chuan, Kat Agres, "From context to concept: exploring semantic relationships in music with word2vec," 2018. [Online]. Available: https://doi.org/10.1007/s00521-018-3923-1

[4] Wikipedia, "Word embedding." [Online]. Available: https://en.wikipedia.org/wiki/Word_embedding

[5] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/keras-team/keras

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[7] P. R. Christopher D. Manning and H. Schütze, "Stemming and lemmatization," 2008. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[8] G. C. J. D. Tomas Mikolov, Kai Chen, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[9] Medium, "Word2vec (skip-gram model) explained." [Online]. Available: https://medium.com/datadriveninvestor/word2vec-skip-gram-model-explained-383fa6ddc4ae

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[11] A. M. Sarah Guido, *Introduction to Machine Learning with Python*. O'Reilly Media, 2016. [Online]. Available: http://shop.oreilly.com/product/0636920030515.do

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Manifold learning." [Online]. Available: https://scikit-learn.org/stable/modules/manifold.html

[13] H. J. K. Jinseok Nam, Eneldo Loza Mencía and J. Fürnkranz1, "Maximizing subset accuracy with recurrent neural networks in multi-label classification," 2017. [Online]. Available: http://papers.nips.cc/paper/7125-maximizing-subset-accuracy-with-recurrent-neural-networks-in-multi-label-classification.pdf