Using Compound Word Transformer for Classical Counterpoint Generation: Application, Evaluation, Prospects

Artur Dobija and Eres Ferro Bastian
Leiden University
Leiden
a.j.dobija@umail.leidenuniv.nl;eres.ferro.bastian@umail.leidenuniv.nl

Abstract

This paper explores music generation techniques based on Natural Language Processing (NLP) and focusing on a stylistically restricted context (Renaissance counterpoint of the composer Giovanni Pierluigi da Palestrina). The Compound Word Transformer model was used as the NLP-based music generation technique and trained with a portion of Palestrina's musical Masses database that was converted from `kern** to `MIDI** format. Inference was made to the model in order to generate new `MIDI** files, evaluated using using pitch histograms, pairwise cross-validation relative metrics, and transition matrix comparisons. The results indicate that the model has the capability to capture the style and has the potential for further enhancements in consideration of human-centered evaluation metrics. The code for this research is available on GitHub. https://github.com/ArturJD96/Leiden_ML4NLP_2023-4

1 Introduction

Language and music are both akin to their own kind of symbolic representations. Thus, the way we can approach language in machine learning should be applicable to music as well. However, the representation of music in digital domain is not so standardized as this of a language.

In this instance, we will investigate how much an existing model architecture can perform when we train it in a very different musical genre. We focus on the style of Renaissance composer Giovanni Pierluigi da Palestrina (ca 1525-1594). The style of his choir pieces characterized by independent voice-leading and rather careful use of chromaticism.

The model we use is a compound word transformer [1]. After training with database consisting of Palestrina's musical Masses, the inferred model produces new scores in the `MIDI` format. We compare the resulting files to the original artworks and investigate the techniques of such system's evaluation.

There are multiple previous works which focus on music generation. Several of them are inspired by machine learning techniques originating in the field of Natural Language Processing, like vector feature extraction using TFIDF [2], Full-Song Music over Dynamic Directed Hypergraphs [3], MuseBERT [4] and MusicBert [5]. However, most of the recent works have focused only on pop and classical music genres. Therefore, our research aims to fill the gap between application of NLP-inspired music generation techniques and older repertoire existing primarily as a symbolic representation (scores).

The code for our research is available at github[1]

2 Methodology

We investigate the prospects of applying NLP-specific machine learning techniques to the domain of stylistically restricted generative music. We focused on the Renaissance counterpoint style – a field that has been already approached from the machine learning perspective [6] [7] [8] [9] [10] and with promising NLP-specific implementations prospects [2] [7] [3].

We adapted Compound Word Transformer ([1]) which is a Transformer decoder architecture that sep-

[1]Github repo.

arates tokens including relevant information to the symbolic music representation such as pitch, duration, velocity or dynamics and placement along the time grid (onset time). Being attention-based [11], it is the most advanced and inexpensive model that we are able to run locally that utilizes an approach that could still be regarded as novel.

The authors propose the use of various token *types* corresponding to the score symbolic representation units (e.g. note pitch, duration). They develop a new tokenization method (musical *compound word*) to already existing `MIDI` music tokenization techniques (`MIDI-like` [12] and `REMI` [13]) and report a significant gain in the terms of performance without any quality sacrifice.

The compound word model was then trained by us with over 1200 `MIDI`ed Palestrina scores files (from `music21` corpus) for 100 epochs with 128 dimension size, 12 layers, 8 attention heads, 8 batch sizes and learning rate of 1e-4. Finally, we inferenced the model in order to extract predictions into set of `MIDI` files (using again Hsiao's model inference architecture). With the amount of data and the current configuration, we were able to generate a model with 0.19 loss after 2 hours of training.

For the analysis, we used two musically informed objective metrics from Yang [14]: pairwise cross-validation relative metric extracting the Euclidean distance of the pitch count among Palestrina-intra and Model-intra sets and feature transition matrix comparison.

3 Data

Corpus. In this research, we employ a digital edition of polyphonic masses by Giovanni Pierluigi da Palestrina based on [15], encoded into `**kern` format by John Miller and accessed as a part of `music21` Python package (as a parsed corpus consisting of `Score` objects). The database contains score representation of over 100 masses, where each mass movement's section is stored in a separate `**kern` file.

Corrupted files. Unfortunately, the corpus is partly corrupted and we had to filter out miscoded files (for details about the database and the mentioned issues, see footnotes in [8, p. 6]).

Cropping. During the prototyping phase of model training, we noticed that the produced inferenced output is very sparse when it comes to note density, i.e. the generated midi scores consisted of few notes separated by long lasting rests. We intuitively concluded that it was due to having very long scores together with short ones. Thus, we decided to crop the scores that are longer than 408 ♩ (30% of the database), successfully fixing the issue.

Parsing. The original architecture presented in [1] was based not on symbolic music representation files, but on collected audio files (1,748 pieces). Authors discuss the process of transcription into `MIDI` symbolic music representation format, as well as dealing with life-performance nature of data requiring further processing: note beat position estimation (*synchronization*), time grid resolution reduction (*quantization*) and distinguishing lead melody from its accompaniment (*analysis*). The resulting `MIDI` files consist of one instrument track with 480 tick resolution.

```
 tempo: 0:   IGN
        1:   no change
        int: tempo
 chord: 0:   IGN
        1:   no change
        str: chord types
bar-beat: 0:   IGN
        int: beat position (1...16)
        int: bar (bar)
  type: 0:   eos
        1:   metrical
        2:   note
duration: 0:   IGN
        int: length
 pitch: 0:   IGN
        int: pitch
velocity: 0:   IGN
        int: velocity
```

Figure 1: a pseudocode representation of event from [1] `corpus2events.py` script.

As our database already consists of symbolic music representations of music scores that are quantified by nature and does not have to deal with micro-timings typical for live performance, we did not have to execute any of the above mentioned procedures. It also allowed us to reduce midi resolution to 4 ticks per ♩ (as the smallest possible note value encountered in Palestrina is likely a four time shorter 𝅘𝅥𝅯).

In order to get `MIDI` files compatible with Hsiao's software, we applied `music21`'s `.flatten()` method to all the scores (collapsing into a single voice all the parts of a polyphonic piece). Having done this, we parsed those flattened scores to MIDI using `music21` internal parser resulting in one-track `MIDI`.

Vectorization. We reused the [1] architecture to create compound word vectors and separate data into train and validation sets. The code was reused *as is*. Authors' pipeline requires: (1) parsing `MIDI` into "corpus" (a `music21`-like score representation in the form of a dictionary containing notes, chords, tempos, labels and metadata), (2) extracting "events" from each "corpus", (3) creating dictionary of tok-

enized "events" ("words") and — finally — (4) parsing them into `numpy`'s arrays.

4 Experiments

After 100 training epochs, we generated 61 `MIDI` files and assessed them by ear as representative enough for the researched music style (i.e. the resulting music was mostly diatonic and presenting similar rhythmic movement, see later discussion Fig. 5).

Fig. 2 and 3) contain pitch histogram. The resulting data adheres to the Palestrina's choir ambitus (span of the lowest and highest pitches). For example, the present pitches on both of the histograms present a typical situation of the low register not containing chromatically altered pitches and a very low presence or complete absence of certain chromatically altered pitches in all the registers (i.e. D-flat)

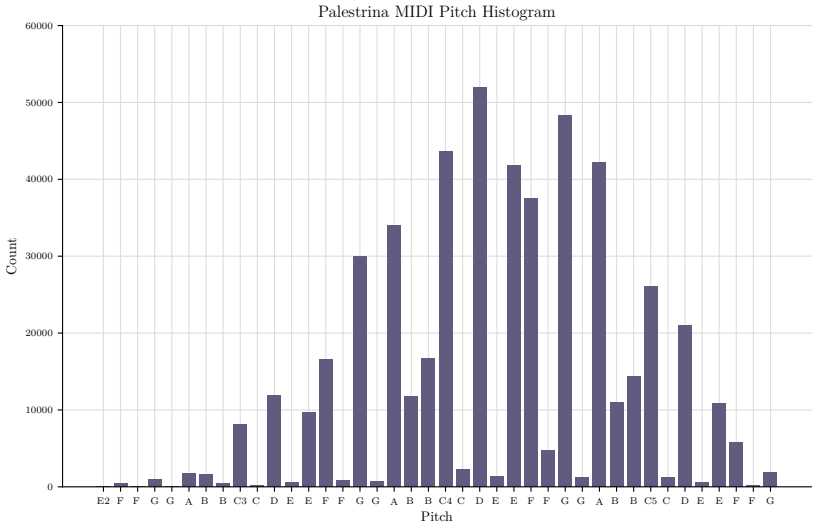

Figure 2: Histogram of pitches in Palestrina corpus (cropped and parsed to `MIDI`.

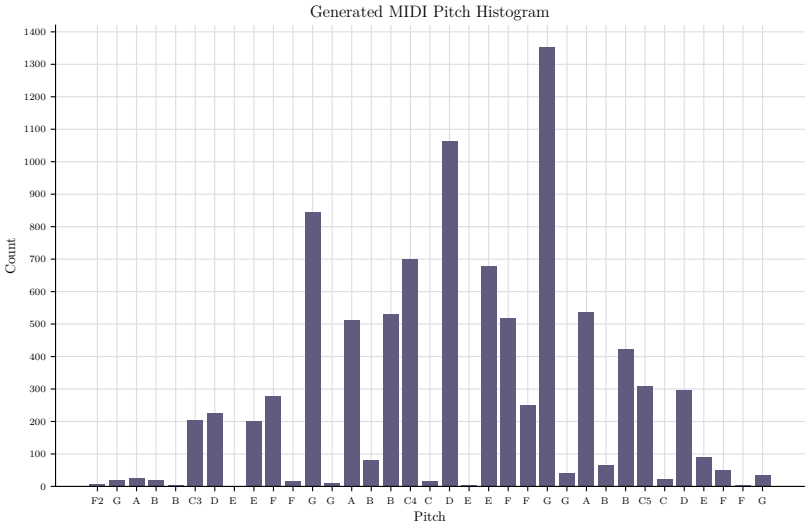

Figure 3: Histogram of pitches in generated `MIDI` files

To investigate the quality of the model, we employed two approaches proposed by [14] for this task: a pairwise cross-validation relative metric and transition matrix comparison.

Cross-validation. Given our two sets of `MIDI` files (extracted from the original Palestrina corpus and transformer's generated output), we extracted the Euclidean distance of the pitch count in order to calculate the pairwise cross-validation. Here, the *Palestrina-intra* and *Model-intra* set distances are cross-validations that are computed within one set of data, whereas the *Model-Palestrina* set distance is a cross-validation of each sample compared with all samples from the other set. According to Fig. 4, the pitch in the *Model-intra* set has the highest density but a sharp decline in the euclidean distance, whereas the *Palestrina-Intra* is lower in density and its euclidean distance is longer (not as long as *Model-Palestrina* set). Therefore, inter has the lowest density but has the longest euclidean distance which indicates strong dissimilarity with the source data.

In the *Model-Intra* set, there are also very interesting short spikes of density, easing at the edge of the euclidean distance. As the model generated data has by far fade-out like endings, this may indicate separability from the source data, when the model checks the next data point and the density comes back up as the resulting experiment data is now similar again with the source data.

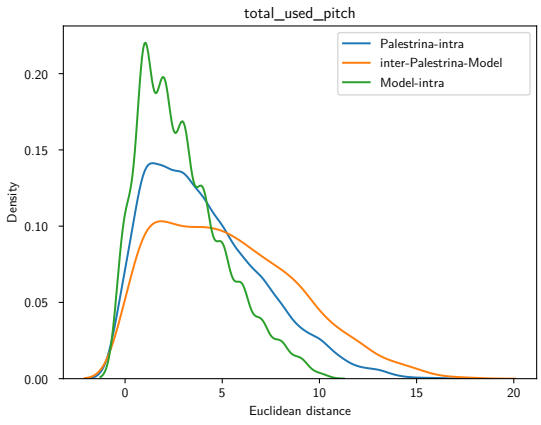

Figure 4: Pitch Count for Palestrina-Intra (blue), Model-Intra (green), and inter-Model-Palestrina (yellow)

Transition Matrices. [14] originally proposed comparison of feature transition matrices between datasets of different genres (Folk and Jazz). Feature matrices are heatmap histograms displaying how much a class within feature transits to another class. Authors employed pitch class (PCTM) and duration ("next length, NLTM) transition matrices.

We reproduce this method with regard to our Palestrina and model-generated `MIDI` databases. We re-implemented pitch and duration feature extractors using `music21` and then normalized the result to the highest value on the heat map.

Based on the heat map's shape, we can notice prominent similarities, the main differences lying rather in transition distributions. Regarding PCTM, we can see that a repetition of note G is overly prominent in the model-generated `MIDI` scores. The original database's heat map points not only to different pitch class note transitions (G to D, C to G, D to A), but also displays much higher variety of other pitch transitions, their intensity increasing when traversing the heat map from right-bottom to the upper-left corner (model's heat map doesn't differentiate here much). It is worth noting that the distribution of the non-diatonic notes is similar, however the note B♭ is much less present in the model examples when compared to the scores of Palestrina.

The produced heat maps of NLTM (next-length transition matrix) shows similarity with the overall rhythmic contour, 𝅗𝅥 𝅗𝅥 consequence being the most common for both datasets. Visible differences occur in semibreves (id: 1) transition (much more prominent in the originals) and occurrence of the slower and faster rhythmic motion (the "ghost" squares prominent for the upper right and lower left sectors for Palestrina and his model correspondingly).

5 Conclusion

Compound Word Transformer model [1] is still a highly useful tool despite the shortcomings of having a small dataset. Despite being a transformer decoder architecture, the model was also relatively inexpensive to train without sabotaging the quality of the output. Pitch histograms, pairwise cross-validation relative metric and transition matrix comparison, supports the model as approaching the style of the source data (available Palestrina's music encoded in `MIDI`). The highlights of the resulting inference of the distribution of diatonic notes and the overall rhythmic contour is the feature most alike to the original corpus.

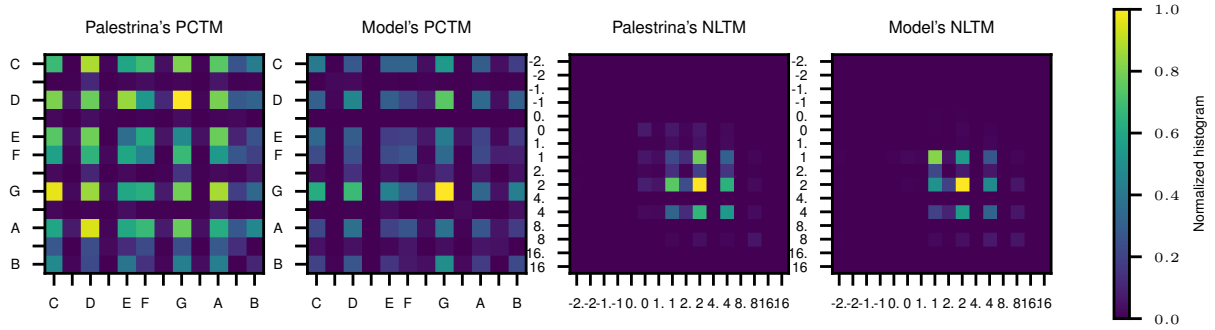

Figure 5: Comparison of PCTM (two left) and NLTM (two right) heat maps for Palestrina and model-generated `MIDI` based on [14]. The pitch classes are represented by the note names (only diatonic are listed) and the duration by the `lilypond`'s duration notation (4 being ♩, 3 being ♩., 2 being 𝅗𝅥 etc.).

6 Future Work

Due to the time constraints, we were not able to train the data on multiple models and compare the resulting systems with Yang's evaluation metrics. Nor were we able to compare multiple data within the same model. Furthermore, the corpus containing masses by Giovanni Pierluigi da Palestrina, being encoded in data-rich `**kern` format allowing for further musical contextualization, has been "downgraded" to less rich `MIDI` data format (with one track only due to preserving compatibility with [1]'s compound word model), losing track of individual voices (and thus the most idiomatic part of the repertoire). A properly satisfying usage of compound word architecture would require parsing features typical for Renaissance counterpoint (instead of piano pop live accompaniments that this architecture was optimized for).

Another issue is lack of any measurements proving effectively that attention layers were proven important for our result or that the Transformer kept track of any relevant musicological concept. They are methods to investigate layer activation when provided sufficient example data for a predefined musicological concept, as presented in [16], that can be used to query for any relevance between concept and network internals.

Therefore, for future projects it is possible to conduct further experimentation that involves the above mentioned remarks and see how the field of music generation can benefit from it. For instance, at the moment Hsiao's decoder transformer model only preserves note's pitch, duration, information about the chord the note belongs to, velocity, and time onset. It would be interesting as further experiment to enhance this decoder to also store voice information and examine the resulting model.

Lastly, we would be interested in using more human-centered evaluation methods when assessing model's inferences. Musical Turing test, where audience assess performances of the original and machine generated snippets can count as such a method, that does not need to adhere strictly to music professionals' knowledge [17]. Within more expert environment, the HER metric proposed by [9] can be a more data-rich resource, where the vector space distances between original (generated) sample and its modified version adjusted by the human evaluator (to make it sound more "human") are calculated.

7 Acknowledgments

The training data we used are part of the open-source projects (`music21`, `Humdrum` [18], `Elvis`, `SIMSSA` [19] etc) and are not extracted in any way from potentially copyrighted material.

References

[1] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, *Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs*, arXiv:2101.02402 [cs, eess], Jan. 2021. DOI: 10.48550/arXiv.2101.02402.

[2] P. Mennen, "Pattern recognition and machine learning based on musical information,"

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, *et al.*, "Music transformer: Generating music with long-term structure," *arXiv preprint arXiv:1809.04281*, 2018.

[4] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, *MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training*, arXiv:2106.05630 [cs, eess], Jun. 2021. DOI: 10.48550/arXiv.2106.05630.

[5] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, *Musicbert: Symbolic music understanding with large-scale pre-training*, 2021.

[6] K. Adiloglu and F. N. Alpaslan, "A machine learning approach to two-voice counterpoint composition," *Knowledge-Based Systems*, vol. 20, no. 3, pp. 300–309, Apr. 2007, ISSN: 09507051. DOI: 10.1016/j.knosys.2006.04.018.

[7] E. P. Nichols, S. Kalonaris, G. Micchi, and A. Aljanaki, *Modeling baroque two-part counterpoint with neural machine translation*, Jan. 19, 2022.

[8] C. Arthur, "Vicentino versus palestrina: A computational investigation of voice leading across changing vocal densities," *Journal of New Music Research*, vol. 50, no. 1, pp. 74–101, Jan. 1, 2021, ISSN: 0929-8215, 1744-5027. DOI: 10.1080/09298215.2021.1877729.

[9] S. Kalonaris, T. McLachlan, and A. Aljanaki, "Computational Linguistics Metrics for the Evaluation of Two-Part Counterpoint Generated with Neural Machine Translation," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, S. Oramas, L. Espinosa-Anke, E. Epure, *et al.*, Eds., Online: Association for Computational Linguistics, 2020, pp. 43–48.

[10] M. Farbood and B. Schoner, "Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains," Jan. 2001.

[11] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention Is All You Need*, arXiv:1706.03762 [cs], Aug. 2023. DOI: 10.48550/arXiv.1706.03762.

[12] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *CoRR*, vol. abs/1808.03715, 2018.

[13] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," ser. MM '20, Seattle, WA, USA: Association for Computing Machinery, 2020, pp. 1180–1188, ISBN: 9781450379885. DOI: 10.1145/3394171.3413671.

[14] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," en, *Neural Computing and Applications*, vol. 32, no. 9, 2020, ISSN: 0941-0643. DOI: 10.1007/s00521-018-3849-7.

[15] G. da Palestrina, R. Casimiri, K. Jeppesen, and L. Bianchi, *Le opere complete* (Le opere complete t. 30). Fratelli Scalera, 1961.

[16] F. Foscarin, K. Hoedt, V. Praher, A. Flexer, and G. Widmer, *Concept-based techniques for "musicologist-friendly" explanations in a deep music classifier*, Aug. 2022.

[17] E. Belgum, C. Roads, J. Chadabe, T. Tobenfeld, and L. Spiegel, "A turing test for "musical intelligence"?" *Computer Music Journal*, vol. 12, p. 7, Dec. 1989. DOI: 10.2307/3680146.

[18] D. Huron, "Music information processing using the humdrum toolkit: Concepts, examples, and lessons," *Computer Music Journal*, vol. 26, no. 2, pp. 11–26, 2002, ISSN: 01489267, 15315169.

[19] I. Fujinaga, A. Hankinson, and J. Cumming, "Introduction to simssa (single interface for music score searching and analysis)," Sep. 2014, pp. 1–3. DOI: 10.1145/2660168.2660184.

2