The Chordinator: Modeling Music Harmony By Implementing Transformer Networks and Token Strategies

David Dalmazzo, Ken Déguernel, Bob L. T. Sturm

The Chordinator: Modeling Music Harmony By Implementing Transformer Networks and Token Strategies*

David Dalmazzo[1][0000−0002−3262−4091], Ken Déguernel[2][0000−0001−7919−3463], and Bob L. T. Sturm[1][0000−0003−2549−6367]

[1]KTH-Royal Institute of Technology, Stockholm, Sweden
[2]Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
{dalmazzo,bobs}@kth.se*, ken.deguernel@cnrs.fr

Abstract. This paper compares two tokenization strategies for modeling chord progressions using the encoder transformer architecture trained with a large dataset of chord progressions in a variety of styles. The first strategy includes a tokenization method treating all different chords as unique elements, which results in a vocabulary of 5202 independent tokens. The second strategy expresses the chords as a dynamic tuple describing **root**, **nature** (e.g., major, minor, diminished, etc.), and **extensions** (e.g., additions or alterations), producing a specific vocabulary of 59 tokens related to chords and 75 tokens for style, bars, form, and format. In the second approach, MIDI embeddings are added into the positional embedding layer of the transformer architecture, with an array of eight values related to the notes forming the chords. We propose a trigram analysis addition to the dataset to compare the generated chord progressions with the training dataset, which reveals common progressions and the extent to which a sequence is duplicated. We analyze progressions generated by the models comparing HITS@k metrics and human evaluation of 10 participants, rating the plausibility of the progressions as potential music compositions from a musical perspective. The second model reported lower validation loss, better metrics, and more musical consistency in the suggested progressions.

Keywords: Chord progressions · Transformer Neural Networks · Music Generation.

1 Introduction

The domain of music computing has seen a variety of applications of state-of-the-art Natural Language Processing (NLP) models or neural networks, such as music generated from symbolic music notation [8, 3], sound synthesis [12, 4], or

* This paper is an outcome of: MUSAiC, a project that has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (Grant agreement No. 864189); and a Margarita Salas Grant, UPF, Barcelona, Spain

text-to-sound models such as MusicLM [1]. Nevertheless, compared to melodic sequences, the modeling of harmonic progressions has seen comparatively little research. Hence, we are interested in modeling chords from symbolic music notation by presenting an approach to encode chord progressions (CP). We designed a system to suggest a possible next chord after a CP, a complete CP for a specific bar length, or a parallel plausible CP, where the human user is the leading composer.

When encoding harmony information and chord vocabulary, there is no unified method in the literature. In Western modern popular music, the basic cell of the chord is the triad as a stack of thirds, containing the scale's 1st, 3rd, and 5th notes. The triad has four possible natures: major, minor, augmented, or diminished. A chord can also be expressed in tetrads, adding the 7th scale degree (e.g., Cmaj7: C, E, G, B), having a richer range of possible natures (e.g., major 7th, minor 7th, half-diminished, diminished, dominant, minor-major, among others). Non-triad-based chords are also in the vocabulary, for instance, a stack of fourths or fifths or *suspended* chords where the 3rd is substituted by the 2nd or 4th. Chords can also be extended further by stacking even more thirds, adding scale degrees 9th, 11th, or 13th (e.g. Dmaj7♯11: D, F♯, A, C♯, G). Moreover, chords can have alterations that modify one or several notes and subtractions that eliminate one or several notes. If we consider chord inversions, the range of vocabulary is considerably extended. Hence, modeling symbolic music notation by implementing current NLP architectures to encode form, structure, and harmonic information without losing vocabulary richness is one of the current challenges in the field.

The following three points give the contribution of the proposed system:

- A dataset with 70,812 CP from several popular music styles.
- A tokenization method to encode chord symbols without reducing its vocabulary.
- A positional encoding strategy with domain-specific embeddings of the transformer network to better suit spatial music information.

The following section reviews work applying machine learning to modeling and generating chord progressions. Section 3 presents our application of the transformer architecture to this problem. Section 4 presents the results of our models and analyzes them from a few different perspectives. We conclude with a look at the future development of our system and its integration into a pipeline for music creation. The supplementary material and related code to the publication can be found on GitHub.[1]

2 Literature Review

The year 2016 is perhaps the starting point of neural network approaches to model music information, in particular, chord progressions using symbolic music

[1] https://github.com/Dazzid/theChordinator

notation. *ChordRipple* [7] presented a Chord2Vec encoder by adapting Word2Vec using datasets derived from the Bach chorales and The Rolling Stone Top 200 songs corpus. The chord vocabulary includes triads, tetrads, and inversions. The authors evaluated the application with music students in a creative exercise and found benefits, but more precise stylistic development was still needed.

A text-based machine-learning model for chord progressions is proposed by Choi et al. [6]. The authors train two Long Short-Term Memory (LSTM) networks using textual representations of chord progression, e.g., "`C:7 E:min`". One LSTM has a vocabulary of 39 symbols corresponding to the characters which combined create chords, e.g., "`E:min`" consists of five characters in this vocabulary. The other LSTM has a vocabulary of 1,259 chord symbols. In this case, "`E:min`" is one vocabulary element. The authors train both systems by using 2,486 scores from The Realbooks and The Fakebooks. They find both models generate progressions that are plausible within jazz styles.

Transformers [13] have shown an improvement over the previous neural networks, enhancing the long-term dependencies by implementing the attentional mechanism as a parallelizable architecture. Most research in the context is focused on adapting or expanding it to specific tasks.

In 2020, Wu and Yang [14] introduced *Jazz Transformer*, a generative composition system including chords, melodies, and form. The model is trained using the Weimar Jazz Database [11], which contains 456 jazz standards, including styles such as Swing, Bebop, Cool, Hard Bob, and Fusion. The data format is MIDI. The authors suggested a set of objective measurements included in the *MusDr*[2] framework that reveals the deficiencies of machine-generated music, such as unpredictable pitch class usage, inconsistent grooving patterns, and chord progressions, or the lack of recurring structures. These metrics demonstrate the limitations of the transformer architecture as a music generator and provide practical quantitative standards for evaluating the efficacy of future automated music composition efforts. Currently, model assessment largely depends on human assessment, thus concluding that besides the training loss and evaluation rating numbers, the music generated still needs to be revised by experts.

The transformer, by default, is not fully suitable as a neural network to encode chord progressions by only providing symbolic music notation as it misses the configuration of voicings and chord construction; thus, in 2021, different publications suggest updating the positional embedding layer of the transformer architecture in order to provide the network with more precise information. Chen et al. [5] propose an encoder-based transformer model to classify chords and provide information on harmonic progressions in the classical music domain. The dataset used is the BPS-FH dataset and the Bach Preludes. They improve the capacity of transformers to encode harmonic features by providing extra information by adding a relative positional encoding, adding a layer of contextual information with the relative function of the chord in the chord sequence.

MusicBERT, by Zeng et al. [15] is a symbolic music understanding model based on the BERT: Pre-training of Deep Bidirectional Transformers for Lan-

[2] https://github.com/slSeanWU/MusDr

guage Understanding architecture, to provide melody completion, accompaniment suggestion, and style classification and recall. They use the Million MIDI Dataset (MMD), which contains 436.631 samples[3], from different styles such as Rock, Rap, Latin, Classic, or Jazz. The data format as a symbolic music notation is shaped as bar-level masking using MIDI. The authors propose adding *musical note embeddings* to the positional embeddings, including time signature, tempo, bar, position, instrument, pitch, duration, and velocity.

MrBert, by Li and Sung [9] is an automatic music generator that creates melodies and rhythm and then creates chord progressions based on the melody. The masked language model is trained with the OpenEWLD dataset (MusicXML), which is translated to music events using the Python library *music21*. The system is based on parallel transformer encoders: one encoding melodies, and the other its rhythmic pattern. Authors argue that chords are generated after the melodies, using a Seq2Seq generation that changes chords when a new symbol outside the chord pattern appears. HITS@k metrics are used to evaluate the compositions. The chord vocabulary is restricted to triads.

Li and Sung [10] propose an encoder-decoder (Seq2Seq) architecture to generate chord progressions given a melody. The data format is music XML. The authors compare three architectures: a) the bidirectional long short-term memory (BLSTM), b) the bidirectional encoder representation from transformers (BERT), and c) the generative pre-trained transformer (GPT2). They evaluated the models using HITS@k [2]. The proposed methodology uses a pre-trained encoder that takes song melodies and couples their relations with the chord progression. The decoder receives the same melodic material and produces suggested chord matchings. The authors argue that the model does not need music theory inference as it relies on the patterns reflected in the symbolic music notation; however, it is not proven the model learns harmonic principles. The transformer is trained using the OpenEWLD dataset and Enhanced Wikifonia Leadsheet Dataset (EWLD), which contain 502 samples in musicXML format, including melodies and chords. 382 chord types were extracted as a vocabulary. Examples of generated music or chord progressions are not available.

3 The Chordinator

We compare two strategies to model chord progressions, maintaining the full range of vocabulary available in the music corpus. All approaches are based on the transformer encoder, changing the token strategy. We also propose complementary embeddings to the transformer to encode chord information by adding a new level of relative embeddings using an array of defined MIDI values added to the positional and tokens embeddings layer. In the second tokenization strategy, we include a style token reference, providing the transformer with contextual information about the style and form.

[3] https://github.com/jeffreyjohnens/MetaMIDIDataset

3.1 Database

The database was formed by exporting 4,300 CP included in the *iReal Pro* application (as songs). The data provides metadata (composer, song name, key, etc.) and information regarding the form, tempo, and chords. The database includes styles such as Jazz, Samba, Ballad, etc. (see the full list in Figure 1). We filter the CP, keeping only those in 4/4 time signature (around 90% of the dataset), to keep the sequences consistent. We apply data augmentation by transposing all CP into the 18 enharmonic keys, creating 70,812 in total. Furthermore, we prepare the data stream to train a GPT-2[4] transformer architecture with a fixed-length sequence of 512-time steps in the first model and 1024 in the second model. The $< pad >$ token is used to fulfill the defined length.

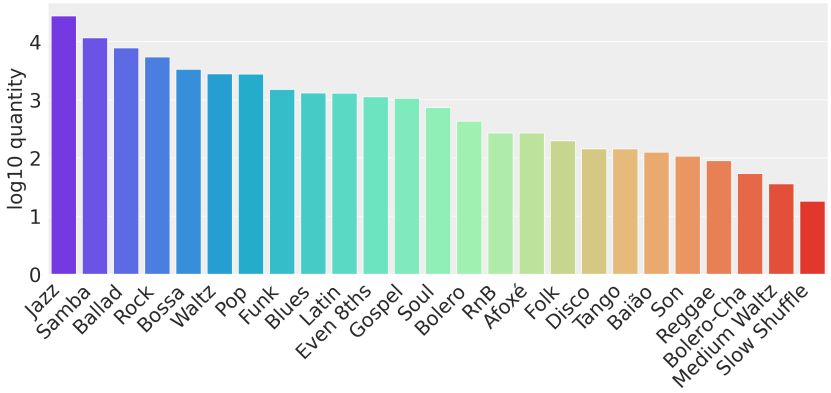


Fig. 1. List of musical styles in the *iReal Pro* dataset. Values are presented in the log10 ratio.

3.2 Model Strategies

Our proposal is based on two main strategies to train a GPT-2-based transformer encoder architecture with chord progressions.

- **Model 1**: It has the entire token vocabulary. All chords are unique elements, producing a vocabulary of size 5,202 unique chords/tokens. The chords are quantized in time, creating a chord per quarter note. When the suggestions are generated, four consecutive chords are equivalent to an entire 4/4 bar.
- **Model 2**: We split the chord units into subsections, simplifying the tokenization range without reducing the vocabulary in the corpus. Starting the chord element with an identifier token '.' (dot), followed by the **root**, **nature**, and **extensions** as shown in Table 1. For instance, '**Dm7 add 9**',

[4] https://github.com/karpathy/minGPT

is split into **'.', 'D', 'm7', 'add 9'**. This creates a 1D array of dynamic lengths depending on how much information the chord has. Slash chords are also split, following the same structure. For example, '**Cmaj7 / Bb**' results in a sequence **'.', 'C', 'maj7', '/', 'Bb'**. The second root in slash chords becomes the actual root reference. This strategy reduces the vocabulary to a more specific format with 59 Tokens: Song formatting (¡start¿, ¡and¿, ¡pad¿, '.'), roots ('A' ''A♯' 'Ab' 'B' 'Bb' 'C' 'C♯' 'Cb' 'D' 'D♯' 'Db' 'E' 'Eb' 'F' 'F♯' 'G' 'G♯' 'Gb'), natures ('dim', 'm', 'm6', 'maj7', 'o7', 'sus', 'power'), extensions (additions, subtractions, and alterations), and sharp ('/'). In the cases where the chord is notated with only one note, that chord is considered a major triad.

- **Extended Model 2**: We extend Model 2 in two ways. First, we add a context token at the beginning of the samples with a style specification, meaning we start with a token (¡style¿) followed by the actual style (such as jazz, pop, blues, samba, etc.). Second, we update the transformer model, adding a MIDI Embedding relative position array into the positional embedding layer. Based on the second strategy, when we have chords divided into root, nature, and extension, the MIDI array is composed of an array of eight numbers where the first is the MIDI root reference, the next three values are used to describe the nature, and extensions could use the next value. The longest sequence of the MIDI array for a chord is seven MIDI values; hence, the eighth MIDI value in the array is an identifier (number 127) to express that there is a slash token.

Table 1. Chord Tokens Format

Chord	Start	Root	Nature	Ext.	Slash	Root_2
C	.	C				
Dm7 add 9	.	D	m7	add 9		
Bbmaj7 ♯11	.	Bb	maj7	♯11		
C♯o7	.	C♯	o7			
Ebmaj7/D	.	Eb	maj7		/	D

3.3 MIDI Embeddings

MIDI Embeddings (ME) are extracted from the symbolic chord representation using the Python libraries *music21* (symbol to notes translation) and *librosa* (notes to MIDI translation). We decided to maintain the chord pitches in only a two-octave range. During training and generation, ME is paired with token inputs, providing only information about the current time-step in the sequence; therefore, when only one note is presented in a specific chord, by default, the ME reads it as a Major chord (triad). When a second token appears next to the original note, the first token is read as a root note (only one MIDI note is then

placed), and the nature is updated, adding its related notes to the MIDI array. The same logic is applied to extension and sharp chord, where each step in the sequence updates the ME information, having the complete chord MIDI array only in the last time step of the chord tuple, as shown in Figure 2. We avoid information leaking following this format.

The formula to update the ME into the positional embedding in the transformer network is defined as follows:

$$midi\_emb = Linear(Emb(midi\_vocab, dm)(m).view(m.size(0), m.size(1), -1), n_{embd})$$

Positional embeddings are added to the input embeddings before the transformer encoder is fed. The positional embedding layer enables the model to distribute information spatially in order to learn specific patterns. The ME takes information from MIDI and adjusts it into an embedding tensor to align with the required architecture format. This, then, is summed with the standard positional encoding layer. It is composed as follows:

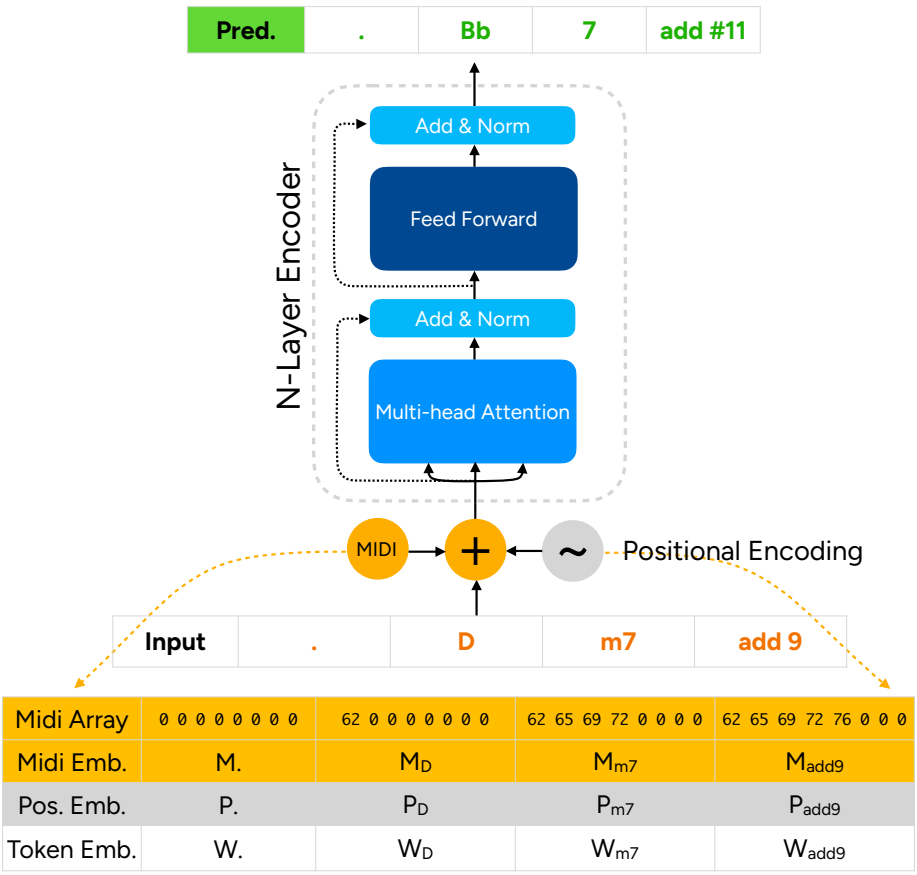
1. **Embedding MIDI Values**:
   - ***'Emb(midi_vocab, dm)(m)'***: This defines an embedding layer where *'midi_vocab'* is the total number of unique MIDI events and *'dm'* is the dimension of the embeddings. *'(m)'* is the time-step in the chord tuple sequence.
   - ***'.view(m.size(0), m.size(1), -1)'***: The tensor matrix is reshaped to adjust its format to the standard positional embedding. Here, *'m.size(0)'* represents the batch size, while '*m.size(1)'* represents the sequence length. '-1' means that the size in the last dimension is computed so that the total size remains constant.
2. **Linear Transformation**: The reshaped data is passed through a Linear layer. The size of the output data is adapted to match a tensor of shape $(batch\_size, sequence\_length, n_{embd})$. It is then summed with the positional encoding layer.

3.4 Training The Models

The models are trained on 4 GPUS NVIDIA GeForce RTX 3090 (24576MiB). While we tested many configurations, this manuscript only reports the best versions of both strategies, defined as Model 1 and Model 2.

Model 1 is trained with 120 Epochs, 6 attention heads, 6 layers, 192 embeddings sizes, 64 batch sizes, 192 embeddings, number of workers 6, and a learning rate of 3e-5. Model 1 is prone to overfit after 120 epochs; hence, these configurations are chosen to maintain a small network that adapts to the size of the vocabulary and dataset. Extra attention heads tend to increase the overfitting tendency and do not necessarily lead to better performance.

Model 2 is trained with 250 epochs without showing overfiting tendencies (it could be trained with more epochs). Thus, it has an increment in 8 attention



Midi Array	0 0 0 0 0 0 0 0	62 0 0 0 0 0 0 0	62 65 69 72 0 0 0 0	62 65 69 72 76 0 0 0
Midi Emb.	M.	M_D	M_{m7}	M_{add9}
Pos. Emb.	P.	P_D	P_{m7}	P_{add9}
Token Emb.	W.	W_D	W_{m7}	W_{add9}

Fig. 2. MIDI Embeddings: Formatted as an array of eight values for each time-step of the chord tuple. In Model 2, the chord information is split into tuples, and the input layer processes it as a dynamic sequence. The input sequence is translated into tokens embeddings, positional embeddings, and midi embeddings; all those layers share the same tensor matrix shape, and they are summed inside the positional encoding layer, which is passed to the multi-head attention.

heads and 8 layers. The other features are maintained as they are: 192 embeddings sizes, 64 batch sizes, 192 embeddings, number of workers 8, a learning rate of 3e-5, and a MIDI vocabulary size of 128, which refers to the size of the possible MIDI values. (see table 2).

3.5 Metrics

HITS@k (k = 1, 3, and 5) [15] is used to evaluate the predictions in both models. HITS@k calculates the proportion of the correct answer given by the candidates, denoted by the letter k. It was calculated as follows:

Table 2. Transformer Format

Configuration	M1: Full Tokens	M2: MIDI Emb
Epochs	120	250
Heads	6	8
Layers	6	8
Embedding	192	192
Batch size	64	64
Learning rate	3e-5	3e-5
Workers	8	8
MIDI_vocab	-	128

$$HITS@k = \frac{1}{n}\sum_{i=1}^{n} I(rank_i \leq k) \tag{1}$$

where,

- $HITS@k$: This represents the HITS at k metric.
- n: Total number of instances or items being evaluated.
- $\sum_{i=1}^{n}$: This denotes the sum over all instances.
- $I(rank_i \leq k)$: An indicator function that evaluates to 1 if the rank of instance i is less than or equal to k, and 0 otherwise.

Table 3 compares the metrics of Model 1 and 2.

Table 3. Metrics

	Train Loss	Val. Loss	HITS@1	HITS@3	HITS@5
Model 1	0.3699	0.4108	0.7383	0.8294	0.8556
Model 2	0.04982	0.03645	0.9138	0.9707	0.984

4 Results

4.1 Generated Chord Progressions

We generated 20 CP per model in one unique iteration loop where no selection of the CP was performed, and they were shown to the evaluators as they were; no voicing, arrangement, or editing was performed. All CP were constrained with a length of 8 to 24 chords, producing a collection of 40 randomized CP. The Model behind the generated CPs are not identifiable by the evaluators. Ten participants, with an average age of 36 (s.d. +/- 8), reported using a 1 to 10 scale the plausibility of the chord progressions as a starting point of a conceivable music piece: 1 implies no plausibility as a sequence, and 10 is very plausible to

start a potential composition. The overall median score received was **Model 1**: 5.7; **Model 2**: 6.7, as shown in Figure 3. All suggestions begin with a root token, which can be any scale note; nevertheless, Model 2 includes a random number defining the style from the first six most common styles in the dataset (Jazz, Samba, Ballad, Pop, Bossa, Rock). To generate a sound sample, we implemented the Python library *librosa* and *music21* to generate the chords in MIDI format using a standard Piano. Figure 4 shows examples of generated CP with form tokens in Model 2.

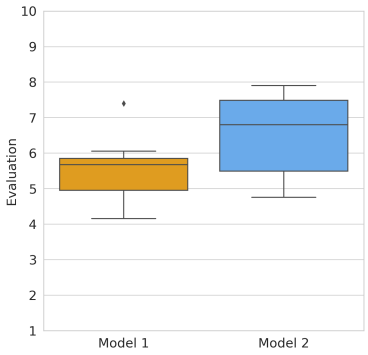


Fig. 3. Boxplot of human evaluation of Model 1 (orange) and Model 2 (blue).

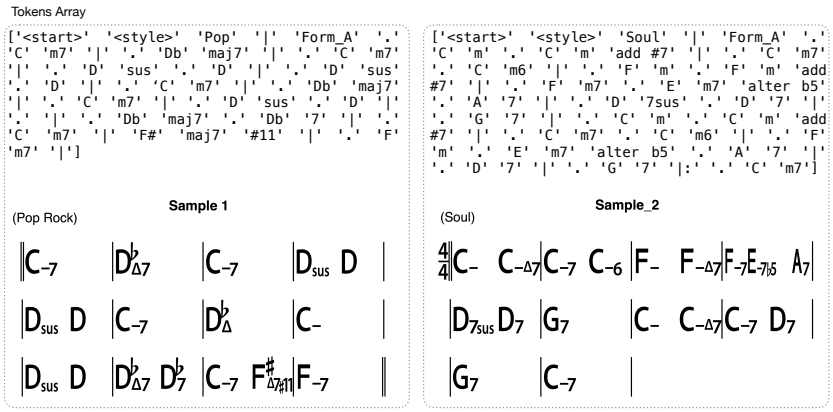


Fig. 4. Two samples from Model 2 using Pop and Soul context token. The tokens array provides information about chords and form, using the token '|' as a bar identification.

4.2 Trigrams

A supplementary dataset comprises three chord cells, including information on the song and composer. Those cells are called trigrams, which provide information on the suggested progressions to check for original versions and sources.

Trigrams are formed by creating groups of subsequent three chords, moving the windowed observation one time step forward. A sequence with Em7, A7, Dmaj7, Ab7 add ♯11, Gm7, are then packed as (Em7, A7, Dmaj7), (A7, Dmaj7, Ab7 add ♯11) and (Dmaj7, Ab7 add ♯11, Gm7). That means four chords are equal if two consecutive trigrams from a suggested progression are found in a particular song. Hence, we can define a threshold of matching trigrams to determine duplication. The trigram analysis can be executed with a list of suggested CP or immediately after a generation. In this case, analyzing 40 CP, the report is too extensive to include in the manuscript. A list of the most 20 trigrams cells found is shown in Figure 8. The report includes a tag for **Sequence**, which is the CP analyzed; **Location**, which reports the chord position in the sequence; **Trigram**, which is the progressions cell; and a list of **Composer** and **Song** where it was found.

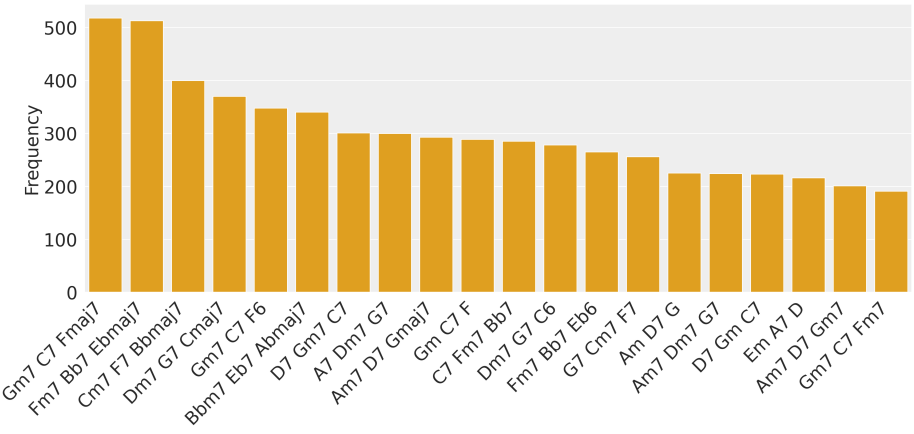


Fig. 5. Most used trigrams found in the dataset. Most progressions are sequences of II-V-I in major and minor functional formation.

Both Models show similar trigram repetitions found in the dataset. However, it does not mean the whole progression is copied; it only provides information about the original trigram cell location. However, it is possible to determine if a whole progression is, in fact, replicated from the training data. No generated CP reported more than two trigram cells from the same song. In Figure 7, some of the most common trigrams found are shown, and Model 1 is more prone to duplicate cells.

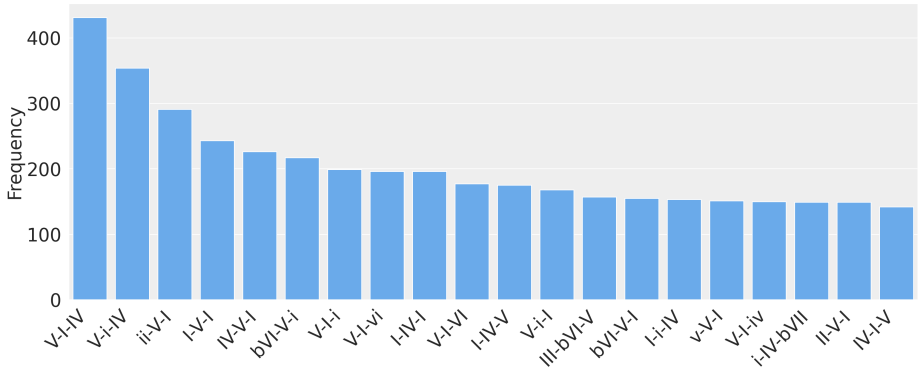


Fig. 6. Most common harmonic functions found in the dataset.

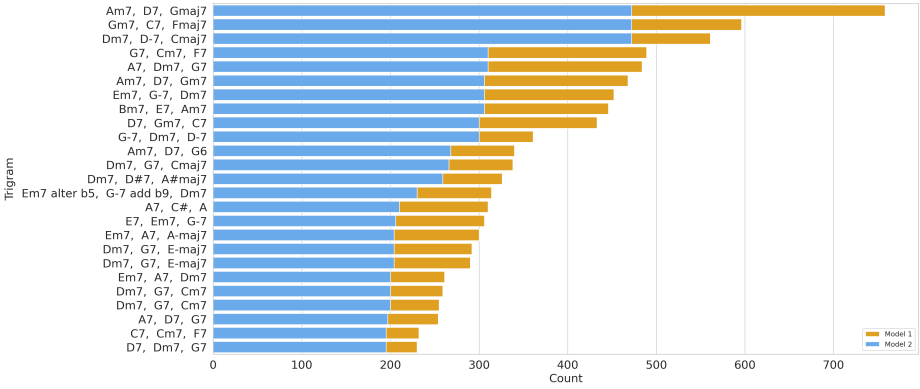


Fig. 7. Model 1 (orange) and Model 2 (blue) report trigrams cells also found in the dataset. However, Model 1 is more prone to replicate common trigram cells found in the dataset

Based on trigrams analysis, we observed that sequences based on ii-V-I progressions and their different versions in major and minor are prominent in the dataset (see Figure 5). However, the ii-V-I is not the most used progression, as shown in Figure 6. The most used progressions are combinations of IV-V-I motions and some permutations, commonly audible in the *Blues*, *Rock*, *Pop*, or *Cinematic Music*.

5 Discussion and Conclusion

We have presented two tokenization strategies with an extension of the positional embeddings (MIDI Embeddings) for Model 2. We treat the chord vocabulary with a specific methodology differentiating the root, nature, and extensions, thereby reducing tokens without diminishing the chord vocabulary and

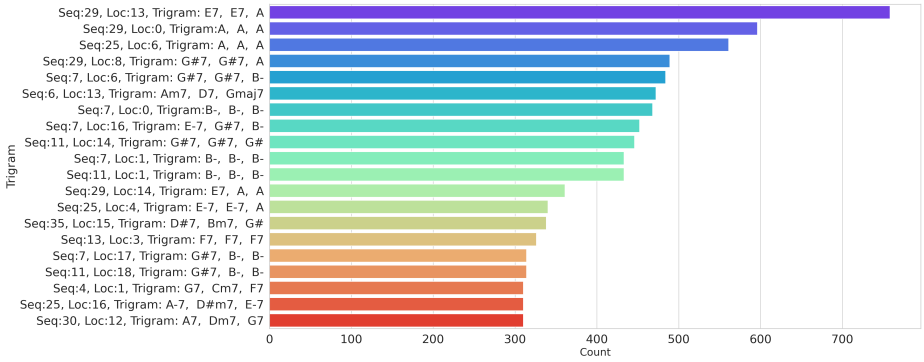


Fig. 8. Trigram analysis provides information about trigram cells found in the dataset. It contains the sequence number; we have 40 CP suggested in this case. The Sequence tab will point to that suggested CP origin. The Location tab provides information on the chord position within the progression; the Trigram tag is the chord cell, and the Count tag expresses how many times it was found in the dataset. The Figure contains 20 of the most trigram cells found

its richness. We trained each model on a large database of chord progressions and used it to generate new suggested progressions or chord completion. Our database of 70,812 songs has diverse music styles in popular music. Our evaluation of sequences generated by these models shows that reducing the vocabulary and adding contextual information improves the encoding process of the chord vocabulary. Thus, adding MIDI notes for chord constructions to the Positional Embeddings layer has helped the transformer network stabilize the learning process, allowing more training epochs without overfitting; it also helped to report better HITS@K and validation metrics (see Figure 3.

Our system contributes to a growing literature on modeling harmonic progressions as a system that can be used as a composer assistant. Future work will explore the usefulness of this system with users performing musical tasks to address further development and usability. We will also explore ways of generating melodies from chord progressions and vice versa. This will be implemented as an accessible web application for others to study.

As expressed in [14], and also based on the outputs generated by Model 1, the vanilla encoder transformer architecture (GPT-2) is not an ideal network for encoding music information from only providing symbolic music notation. Even though it can generate plausible textual tokens, the musicality is still unclear. From the perspective that chord construction and voicing formats are hidden from symbolic music notation and that decision is the responsibility of the musician's knowledge, it is still necessary to propose new versions of related neural networks and test architectural modifications to find better strategies to encodes all musical information related to chord progressions. Therefore, our next model is based on multi-hot encoders (2 octaves) exemplifying the notes in the piano from an expert performer playing chord progressions and an encoder-decoder

architecture to suggest, on the one hand, the symbolic music notation but, on the other, the voicings for more musical outputs.

The human evaluation reported a median score of 5.7 for Model 1 and 6.7 for Model 2. Even though there is a difference in appreciation of Model 2 quality, the low score could be directly affected by the test format shown to the participants. Raw MIDI piano sounds were chosen to evaluate the chord progressions, and no voicing was added to avoid human completion from expert performers. The CP were played at 120BPM; in that term, the suggested CPs are expressed in a raw MIDI version that can be highly improved by musicians' knowledge in terms of performance and interpretation and also in sound quality. However, HIT@K metrics reported a difference where Model 2 is much more accurate, particularly in the validation loss. The interpretation of its musicality based on NLP metrics is still open for discussion.

As a further development, to enhance Model 2, when translating symbolic music notation into MIDI chords, we will add an array of plausible voicings repertoire per chord and a distance calculation of note proximities to suggest more natural voicing. By default, the Python library *music21* is unsuitable for this purpose as it reported errors, particularly with slash chord formatting and notes related to suspended chords.

Model 2 reported a tendency over Model 1 to generate more consistent progression in musical terms. The explanation is given by the modeling of the contextual token, which provides a general view of the music style; therefore, Model 1 mixes styles and formats, while Model 2 is more prone to maintain some degree of consistency. For instance, when *Blues* is given, the form and chords are common to the style. Also, Model 1 chooses chord tonalities wrongly formatted in terms of notation; in other words, it mixes enharmonic, e.g., when Ebmaj7 makes sense in the context, it suggests D♯maj7. Further work is also needed to adjust this duality in names as it is relevant to be fixed in a musical context.

Model 2 also suggests form and structures; further development will incorporate that suggestion into the web application.

For future work, we will add a melody generator that is conditioned on a CP and vice versa, where a melody conditions the generation of a CP. A web application as an interface to play with the model and generate suggestions is under development; it will open its usability to a broader audience. We will perform further human evaluations by expert musicians to test the impact and usability of the system.

References

1. Agostinelli, A., Denk, T.I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., et al.: Musiclm: Generating music from text. arXiv preprint arXiv:2301.11325 (2023)
2. Ali, M., Berrendorf, M., Hoyt, C.T., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., Lehmann, J.: Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. IEEE

Transactions on Pattern Analysis and Machine Intelligence **44**(12), 8825–8845 (2021)
3. Briot, J.P., Hadjeres, G., Pachet, F.: Deep Learning Techniques for Music Generation. Springer (2019)
4. Caillon, A., Esling, P.: Rave: A variational autoencoder for fast and high-quality neural audio synthesis. ArXiv e-prints (2021)
5. Chen, T.P., Su, L.: Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models. Transactions of the International Society for Music Information Retrieval **4**(1) (2021)
6. Choi, K., Fazekas, G., Sandler, M.: Text-based lstm networks for automatic music composition. In: Proc. 1st Conference on Computer Simulation of Musical Creativity. Huddersfield, UK (2016)
7. Huang, C.Z.A., Duvenaud, D., Gajos, K.Z.: Chordripple: Recommending chords to help novice composers go beyond the ordinary. In: Proceedings of the 21st international conference on intelligent user interfaces. pp. 241–250 (2016)
8. Huang, C.Z.A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., Dai, A., Hoffman, M., Eck, D.: Music transformer: Generating music with long-term structure (2018). arXiv preprint arXiv:1809.04281 (2018)
9. Li, S., Sung, Y.: Mrbert: Pre-training of melody and rhythm for automatic music generation. Mathematics **11**(4), 798 (2023)
10. Li, S., Sung, Y.: Transformer-based seq2seq model for chord progression generation. Mathematics **11**(5), 1111 (2023)
11. Pfleiderer, M., Frieler, K., Abeßer, J., Zaddach, W.G., Burkhart, B. (eds.): Inside the Jazzomat - New Perspectives for Jazz Research. Schott Campus (2017)
12. van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: WaveNet: A Generative Model for Raw Audio. ArXiv e-prints (1609.03499) (Sep 2016)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
14. Wu, S.L., Yang, Y.H.: The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures. arXiv preprint arXiv:2008.01307 (2020)
15. Zeng, M., Tan, X., Wang, R., Ju, Z., Qin, T., Liu, T.Y.: Musicbert: Symbolic music understanding with large-scale pre-training. arXiv preprint arXiv:2106.05630 (2021)