The Beat Goes On: Symbolic Music Generation with Text Controls

Stanford CS224N Custom Project

Javokhir Arifov
Department of Linguistics
Stanford University
javokhir@stanford.edu

Philip Baillargeon
Department of Computer Science
Stanford University
pabaill@stanford.edu

Nathanael Cadicamo
Department of Symbolic Systems
Stanford University
cadicamo@stanford.edu

Abstract

Previous approaches to text-to-audio generation have focused on using text descriptions to generate entire pieces of music. These approaches to music generation are not tailored to the needs of actual musical artists, who require high fidelity and control in the creative process. Recently, a new line of research has emerged that aids augmentation and assistive functions in symbolic music creation (e.g., MIDI). This line of research has identified several targeted functions such as transition building and harmony generation as areas in which AI assistance would be the most helpful. We build off this research area by augmenting a music infilling model with natural language input, allowing artists to supplement their creative process. For our project, we have developed a pipeline through which to train and condition the performance of an Anticipatory Music Transformer (AMT) on semantic tokens in various ways, including prepending semantic embeddings onto training sequences and using a k-means approach to bundle instructions to specific clusters of prompts. We then conduct an analysis of performance with respect to different sources of semantic information and these different tokenization schemes, finding that using coarse historical information after minimal pretraining was the most fruitful to generate reasonable music with expressive controls.

1 Key Information to include

- Mentor: Bessie Zhang
- External Mentor: John Thickstun (jthickstun@stanford.edu)
- Sharing project: CS129 (Applied Machine Learning)

2 Introduction

Often attributed to *Detroit Free Press* writer Martin Mull, so goes the famous quote:

> "Writing about music is like dancing about architecture" (Brackett, 2023).

While intrigued by the possibility of architectural dance, this project will instead be an investigation of text controls in symbolic music generation. We are motivated by frequent criticisms in the AI music community which highlight a discontinuity between the needs of music composers, who work with discretized symbolic music in the form of MIDI files, and available music generation tools, which

use the properties of waveforms to immediately generate WAV or MP3 output (e.g., (Agostinelli et al., 2023), (Copet et al.)). In order to assist music composers with their creative process, we will be modifying the Anticipatory Music Transformer (AMT), a music transformer that offers fill-in and melody generation capabilities for user supplied MIDIs Thickstun et al. (2023). However, the AMT currently has no way to condition its generation on natural language descriptions, unlike the current state of the art waveform music models.

To this aim, we have been experimenting with several techniques, including prepending BERT tokenized semantic information to the model's input to re-purposing placeholder tokens in the model to encode semantic information and more recently using k-means to assign text prompts to common categories of music descriptions. These choices help us better understand a fundamental dilemma in text-to-music models, best summarized by the following two research questions:

1. What is the proper balance of semantic to auditory information in a multimodal model?
2. What kinds of semantic information are helpful in training these models (e.g., instrument-based captions vs information about the artist)

3 Related Work

We begin with a discussion of common approaches in text-to-waveform music generation. We then introduce the AMT as a symbolic music model with impressive performance but missing human controls. Our project with synthesize both of these perspectives to produce a natural language supplement to the AMT.

3.1 Text-To-Waveform Music Generation: MusicLM

Recent work in high fidelity text-to-waveform music generation has been heavily inspired by the achievement that was Google's MusicLM by Agostinelli et al. (2023). This architecture relies on SoundStream (Zeghidour et al., 2022) to encode acoustic tokens that represent the final audio, w2v-BERT (Chung et al., 2021) with a k-means variant creates the descriptive captions for the text during training (semantic tokens), and MuLan (Huang et al., 2022) provides the joint text-audio embeddings that condition the semantic and acoustic modeling structures. Similar to how Borsos et al. (2023) in AudioLM generates a hierarchical set of tokens (semantic tokens for context, coarse acoustic tokens for general structure of the audio, and fine acoustic tokens to add depth to the music), MusicLM uses MuLan to generate RVQ audio embeddings which are then used to train a model that maps MuLan audio tokens onto a convolution of the k-best w2v-BERT generated captions. The MuLan audio tokens and the semantic tokens are then fed into a second block, which converts a combination of the MuLan audio and semantic tokens into encoded SoundStream audio tokens, which are ultimately decoded by SoundStream to become the final audio output. Using a Vision Transformer, an existing music piece's spectrogram can be converted into an input that can be paired with a text description to feed MusicLM's generation forward, thus allowing the model to handle text input as well as simultaneous audio and text input.

MusicLM, MusicFX are all publically available to interact with on Google's AI Test Kitchen. After experimenting with the model for some time, it is clear that the outputs often default to a similar "pop" style, they are not able to be easily edited on a track level, and they cannot be exported to other music editing software. It is for these reasons we began to experiment with discretized and controllable symbolic music generation.

3.2 Symbolic Music Generation: Anticipatory Music Transformer (AMT)

Symbolic music generation is the generation of codes that can be read by music production software and played using a digital soundfont. The most common symbolic music encoding is MIDI. Although popular throughout the pre-transformer era of music generation, and often applied to commonly transcribed Bach chorales, MIDI continues to be the language in which composers make music but is neglected by waveform-based models that rely on the waveform itself to condition music generation.

The Anticipatory Music Transformer (AMT) by Thickstun et al. (2023) is a GPT-2 based model that is trained to unconditionally generate MIDI-encoded music, fill in gaps between musical phrases, and provide accompaniment to an existing melody. After encoding notes into (ONSET, NOTE,

DURATION) triplets, these MIDI events are fed into a model that produces logits representing the next most likely note (a group of (ONSET, NOTE, DURATION) tokens representing a single auditory event). The disadvantage of this method relative to the waveform models, however, is there is no current support for text controls. While the model generates tonally "correct" music with reasonable harmonies and instrument selection considering prior and future events, there is no way to condition the model's generation based on the mood, instrumentation, or context of the song. Using the AMT tokenization scheme, we will both modify the tokenization to include semantic tokens and finetune existing AMT models to integrate semantic conditioning.

4 Approach

Our approaches can be summarized as follows:

1. Training a GPT-2 model from scratch using additional semantic tokens
2. Training a GPT-2 model from scratch using a k-means approach to encode semantic information
3. Finetuning an existing AMT checkpoint using a k-means approach to encode semantic information

All of the models we have trained adhere to the standard parameters of a GPT-2 "small" model: that is, inputs are 1024 token vectors, hidden layers are of size 768, there are 12 layers, and there are 12 attention heads. To perform the language modeling step, we have an additional linear layer with weights associated with the input embeddings. Our baselines will be an evaluation of the perplexity of the generation (elaborated upon in Section 5.2.2) in comparison to the available scores for small AMTs. Tokenization and training was achieved by making modifications to the Anticipation GitHub Repo. To finetune the existing AMT, we use the small AMT model trained for 800k steps, then add additional tokens as needed by initializing this linear layer with random additional weights associated with tokens in the expanded vocabulary.

Training a GPT-2 model directly using semantic tokens was largely unsuccessful, namely because finetuning existing AMTs after altering the token structure lead to malformed music tokens. Thus, this approach was quickly abandoned after promising results were achieved for early k-means based approaches.

Our k-means approach aimed to semantically cluster text data paired with our MIDI dataset entries into some set of k clusters. In order to do this, we aimed to implement a text vectorization scheme for the scikit-learn k-means algorithm which would allow us to capture as much semantic content as possible from our text sources.

This k-means approach allows us to insert one potent semantic token into the AMT architecture. In the sequence of 1024 auditory tokens, there is one token left unused by note events. We then approach fine-tuning by substituting this padding token with a k-means cluster for all training examples. We then modify the AMT music generation procedure to classify a user's text input into a cluster and prepend it to an existing input token sequence. The result is a version of the AMT that supports all of the music generation capabilities of the original model (from-scratch generation, infilling, accompaniment generation) with additional control provided by text input.

We believe this approach has several benefits, namely that it allows for semantic controls after the model undergoes a period of pre-training on a variety of music. The model first learns music patterns generally, and then it learns slight deviations to these patterns after being heavily conditioned in respect to our semantic information. Selfishly, the simplicity of this modification allowed us to focus our efforts on analyzing sources of high quality data and experimenting with hyperparameters to most appropriately balance musical quality and reponsiveness to semantic controls. Additionally, success with a small number of clusters on a small amount of training steps would allow us to make preliminary claims about the types of text that are appropriate for training a semantic music control model with significant changes to the tokenization scheme.

5 Experiments

5.1 Data

We believe that there are other types of rich textual information that are useful to conditioning a music model's performance outside of strict audio captions, which are short descriptions about various aspects of a song. So, rather than taking the MusicLM approach and using short, captioned clips like MusicCaps (Agostinelli et al., 2023), or the base AMT approach using the large number of short clips from recognizable songs in the Lakh MIDI dataset (Raffel, Colin, 2016), we focused on finding contextual data that mirrored how reviewers and musicians described these songs in detail. Using the MetaMIDI database by Ens and Pasquier (2021), we were able to link a MIDI recording to its MusicBrainz identifier, where we could access information about the song's name, artist, year of recording, and more. We used this information to scrape textual data about the relevant songs on Wikipedia. We also used the same approach with Pitchfork, a music review website which we hoped would provide richer semantic context about the music. We prioritized descriptions specific about the relevant songs, but we also included general information about the artist if the initial searches did not yield any data. This gave us ≈17,000 MIDI-Wikipedia pairs and ≈4,000 MIDI-Pitchfork pairs. Unfortunately, we were not able to find as much data from Pitchfork as we did with Wikipedia, which might have affected our ability to train on the data.

With these pairs of semantic and acoustic information, we were able to create training examples by tokenizing each note into a triplet of (TIME_ON, NOTE, DURATION) triplets, then prepend semantic tokens to get input vectors of length 1024 to feed into a GPT-2 model. The output of the model is a set of logits that can be used to predict the next note in the sequence by sampling a token from the most likely time, note, and duration intervals of the vocabulary using nucleus sampling (Holtzman et al., 2020).

5.2 Evaluation method

There were two phases to our evaluation. Because our model relies on k-means token clustering, we needed an independent evaluation scheme for these clusters. This allowed us to select an ideal k-means cluster set for our downstream transformer. We then evaluate this transformer model taking the clusters as given.

5.2.1 K-means Evaluation

For our k-means models, we fed our semantic information into a base GPT-2 model, extracted the final hidden layer activations, and used these as inputs to a k-means algorithm with a variety of clusters. We evaluated our k-means clusters with silhouette scores, which is given by

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

where $s(i)$ is the individual silhouette score for point i, $a(i)$ is the average distance from point i to other points in its assigned cluster, and $b(i)$ is minimized over all clusters as the smallest average distance from point i to points in a different cluster. Thus, S is the total silhouette score for a cluster assignment over N data points. The range of S is from -1 to 1, where a high score is a better score and indicates that a data point is well-placed. We performed a hyperparameter search over the number of clusters and chose the value of k that maximized the silhouette score while still featuring clusters of a reasonable size. The results of this search are listed below.

In general, we aimed to keep the number of clusters as low as possible in an attempt to enhance our tokenization and training process and avoid overfitting. At the same time, we aimed to select relatively high silhouette scores and relatively low standard deviation. This led us to we choose the $k = 128$ model for Wikipedia text data and the $k = 256$ model for Pitchfork text data, both of which approximately satisfy our requirements for more effective downtream transformer training.

Table 1: Silhouette Scores for Different K-values (Wikipedia)

K-value	Silhouette Score	Mean	SD
8	0.8047	2245	3353
16	0.8154	1123	2588
32	0.8169	561	1901
64	0.8318	281	1370
128	**0.8649**	**140**	**978**
256	0.8997	70	695
512	0.9392	35	493

Table 2: Silhouette Scores for Different K-values (Pitchfork)

K-value	Silhouette Score	Mean	SD
8	0.2813	498	259
16	0.2476	249	146
32	0.2597	124	73
64	0.3336	62	41
128	0.4691	31	24
256	**0.6149**	**16**	**18**
512	0.7778	8	13

5.2.2 Model Evaluation

For all of our models, we calculate the perplexity of our model when evaluated on the training set so they can be compared to the original AMT. This is done by individually calculating the average cross-entropy loss for each kind of token (L_t for the onset tokens, L_d for the durations, and L_n for the notes). This gives us an overall event loss $L_e = L_t + L_d + L_n$. To get each perplexity, simply calculate $\text{ppl}(x) = \exp(L_x)$ for $x \in \{t, d, n, e\}$. In addition to these perplexity scores, we include some sample generations to compare the quality of the generation.

For more qualitative evaulations and examples, see the Analysis section.

5.3 Experimental details

To form our k-means clusters, we tried an ensemble of vectorization approaches, including TF-IDF, Gensim Word2Vec, BERT last hidden state embeddings, and finally, GPT-2 last hidden state embeddings. TF-IDF, Word2Vec, and BERT approaches, though computationally quite fast, were approximately an order of magnitude worse in terms of silhouette scores than the GPT-2 approach.

After a few pilot models run on a base GPT-2 model, we finetuned several small AMTs for 2000 steps using a learning rate of 3E-5. Information on the size and number of training steps for each model is in the "Results" table (Table 3).

5.4 Results

The perplexity values for each model we trained are listed below. K-means data sources are indicated by “PF" for Pitchfork reviews and “Wiki" for Wikipedia articles.

Table 3: Evaluation Results

Model Name	Params	Steps	ppl(e)	ppl(t)	ppl(d)	ppl(n)
AMT Small (100k)	128M	100K	14.9	1.59	3.90	2.40
AMT Small (800k)	128M	800K	12.4	1.52	3.64	2.24
Wiki K-means	41M	2K	5094.37	4.881	14.197	73.512
PF K-Means	41M	2K	2437419.516	141.776	29.654	579.76
Wiki K-Means Finetuned (800k)	128M	800K+2K	**11.864**	**1.502**	**3.462**	**2.281**
PF K-Means Finetuned (800k)	128M	800K+2K	931.919	2.959	10.5	29.992
Wiki K-Means Finetuned (100k)	128M	100K+2K	12.827	1.526	3.598	2.336
PF K-Means Finetuned (100k)	128M	100K+2K	14.999	1.524	3.763	2.615

Preliminary results indicate that the Wikipedia semantic tokens were much more effective than the Pitchfork tokens, which surprised us. In fact, when finetuning the 800K AMT, performance degraded significantly when Pitchfork data was used, while further training with the Wikipedia dataset demonstrably improved the overall quality of the music. There are many possible explanations for this, including that the Wikipedia articles were often much longer and featured peer-edited phrasing whereas the Pitchfork reviews, while using more musically descriptive language, were shorter and the product of a single writer. This prompts additional investigations into the training of music models:

our project shows that the historical significance of the song and its artists could be influential in the production of high quality music from text.

6 Analysis

With respect to our k-means approach, we performed an inspection of how user song requests are clustered. We ran the NLTK top 10,000 English words through our GPT-2 tokenizer and extracted the final layer embedding to feed into our k-means model as designed. A cursory glance at the result of this experiment showed that there was a degree of randomness associated with the clustering. We attribute this in part to the fact that our k-means model was trained on large documents, and that individual words, taken out of context, do not possess clear semantic value in the same sense as the data on which we trained. We instead fed in a short list of example user song request prompts through our cluster assignment pipeline to test whether the clusters, in practice, seemed to aggregate around musical themes. We were encouraged to see that, for the most part, requests with markedly different themes were sent to different clusters; for example, “Epic orchestral soundtrack for a movie" was sent to cluster 44, while “Reggae-inspired beach vibes" was assigned to cluster 75. We then tested whether requests which were approximately synonymous were sent to the same cluster. We found that a set of 15 jazz-like requests were sent to 8 different clusters. Several of the clusters with multiple assigned prompts did appear to aggregate around other key descriptions, such as “chill", “mellow", and “relaxed", but strangely, the clusters which held prompts with these key words were themselves distinct. A natural attitude for sorting by music theme would likely place chill, mellow, and relaxed jazz in the same category, but our model did not accomplish this.

We think that a main reason for this inconsistency in cluster assignment is that the text data which we trained on was not systematically structured in terms of musical description. The Wikipedia data is, in large part, historical and not explicitly musical. Though the Pitchfork data was more refined in this sense, it was still written in terms of critique, which does not necessarily relate well to artist-side aspects of musicality. In both cases, the training data likely placed a limit on the ability of our k-means model to accurately capture *musical* semantic data, even if it did succeed in capturing other semantic data.

For the most, our clusters do seem to represent thematically similar ideas. Additionally, despite the average size of our clusters assigned during training being relatively small (average of around 10 examples per cluster, with one outlier cluster with several thousand examples), throughout testing we achieved assignment to a large variety of clusters. Any text longer than a few sentences gets classified into the large outlier cluster, suggesting that an additional, comprehensive pre-processing step which condenses the text inputs may significantly benefit the model’s ability to recognize a greater variety of genres. This theory is supported by feeding the names of artists in our corpus (e.g., John Williams, The Smiths) into our model, and the generations did not readily produce music that was tonally similar to these artists. While perhaps a failure of our system, we believe this is a positive effect of the act of using clustering to condition the generation. Because the artist name is only one token used to find the cluster, it is unlikely for an artist to achieve an entire cluster. Rather, clusters seem to represent thematically similar ideas. Additionally, despite the average size of our clusters assigned during training being relatively small (average of around 10 examples per cluster, with one outlier cluster with several thousand examples), throughout testing we achieved assignment to a large variety of clusters. Any text longer than a few sentences gets classified into the large outlier cluster, suggesting a more comprehensive pre-processing step that condenses the text inputs may significantly benefit the model’s ability to recognize a greater variety of genres.

Moving on to the music generated by the AMT itself, despite the finetuned 800K AMT using the Wikipedia dataset generating music with a lower perplexity, the music that it generates appears to be less responsive to the text inputs. For example, prompts that emphasize specific instruments (e.g., angry guitar, sad piano) are often ignored in favor of producing infilling sections that are most tonally and thematically similar to the default AMT. The 100K AMT, while producing slightly more dissonant music at times, experiences greater variation depending on the text prompt even with the same number of text clusters.

In experiments with long-length music generation (3 minutes or longer), we found mixed results. we found that after an interval of around one minute of tonally consistent music, the track would fade out and another seemingly different piece of music would start. When asked to generate "a

song from a British rock group comprised of four men with whimsical undertones", which you can listen to here, the first minute of the song was a reasonable attempt at this theme, the second minute featured a furious thirty second drum solo punctuated by several seconds of silence, and the third was a solemn choir with plodding bongos. Another generation prompted with a description of "The Cranberries", "an alternative rock group, but incorporated aspects of indie rock, jangle pop, dream pop, folk rock, post-punk and pop rock into their sound", can be listened to here and features a more tonally consistent output. Although not the intention of the model, preliminary experiments appear to support semantically conditioned generation of full length songs of three minutes.

7 Conclusion

We hope that our project emphasizes both the relative simplicity with which reasonable text controls for symbolic music generation can be achieved and the potential for improvement that exists in this space. If promising results were achieved by creatively retrofitting an existing architecture on relatively small datasets and a modest number of training steps, there is great potential for training symbolic music models with semantic information. In fact, our work has shown that historical and contextual information may help music models better understand complex concepts like genres and eras. Rather than training on coarse audio captions or music review data like that from Pitchfork, biographical information on the track and artist may prove crucial in helping text-to-music models learn about the tropes that human-generated music relies on.

Our work is limited in that our datasets were collected by greedily scraping publicly available semantic information from online sources. We did have some quality checks using metrics like article length, but would have liked to have greater assurance that each text-MIDI pair contained text that accurately represented the music it was paired with. Additionally, our finetuning steps were undertaken using a relatively small number of steps relative to the pre-training step (100K vs 2K). With a larger dataset and more training steps, stronger claims could be made to the effectiveness of the clusters to encode a representative range of semantic information across a multitude of genres.

Like other music models, our work does not claim to generate music that is representative of all cultures and traditions. Western music is heavily overrepresented in publicly available computer music datasets, and our requirement for there to be English language textual information about each track likely only further biased our dataset toward Western music. Future work would do well to train in multiple languages and consider methods of including music that is not often written about or recorded in a way that is meaningful.

In future work, we would consider the possibility of using more tokens in the architecture to encode semantic information. Additionally, we would consider directly using the activation layer (or a compressed version of it) to train a new GPT-2 model, which we currently lack the compute and training time for. It would be relevant to also explore alternative data sources which capture both the rich sociohistorical data offered by Wikipedia as well as precise musical terminology. Still, this is a very promising result and another step towards producing a model that augments the workflow of musicians in a helpful and cohesive capacity.

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. MusicLM: Generating Music From Text. ArXiv:2301.11325 [cs, eess].

Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour. 2023. AudioLM: A Language Modeling Approach to Audio Generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2523–2533. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.

David Brackett. 2023. *Interpreting Popular Music: With a new preface by the author*. University of California Press. Google-Books-ID: RSdcP4f24tgC.

Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. 2021. w2v-BERT: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 244–250.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and Controllable Music Generation.

Jeff Ens and Philippe Pasquier. 2021. MetaMIDI Dataset.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. ArXiv:1904.09751 [cs].

Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. 2022. MuLan: A Joint Embedding of Music Audio and Natural Language. ArXiv:2208.12415 [cs, eess, stat].

Raffel, Colin. 2016. Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.

John Thickstun, David Hall, Chris Donahue, and Percy Liang. 2023. Anticipatory Music Transformer. ArXiv:2306.08620 [cs, eess, stat].

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2022. SoundStream: An End-to-End Neural Audio Codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507. Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing.