Lyrics and Vocal Melody Generation conditioned on Accompaniment

Thomas Melistas
School of ECE
National Technical University of Athens
Greece
melistas.th@gmail.com

Theodoros Giannakopoulos
NCSR Demokritos
Greece
tyiannak@gmail.com

Georgios Paraskevopoulos
School of ECE
National Technical University of Athens
Greece
geopar@central.ntua.gr

Abstract

In this paper we present a previously unexplored task, the generation of lyrics and vocal melody for a given instrumental music piece in the symbolic domain. We model the above as a sequence-to-sequence task, using a memory efficient Transformer architecture, which we train on text event sequences that describe entire songs. Towards this end, we build a suitable dataset and apply musical analysis, compressing the instrumental part and making it key-independent. We further design a novel architecture to decouple lyrics and melody generation, making it possible to use pretrained language models and conditioning on lyrics. Finally, Mellotron is used to turn the generated sequences into singing audio.

1 Introduction

A significant part of research on singing has focused on information retrieval tasks, such as lyrics or melody transcription (Stoller et al., 2019; Nishikimi et al., 2019), as well as on singing voice synthesis (Nishimura et al., 2016). Generating the (symbolic) content of singing, namely lyrics and vocal melody, has only recently started gaining more attention. Relevant work has focused on generating lyrics for a specific music style or melody and lyrics-conditioned vocal melody generation.

Vocal music coexists with instrumental in most contemporary genres. However, despite the growing interest on studying the relation of lyrics and vocal melody, the connection of both to the accompaniment remains overlooked. In this work we aspire to fill this substantial gap.

We model the instrumental-conditioned generation of vocal melody and lyrics as a seq2seq task. First, we create pairs of text event sequences, which we use to train a baseline encoder-decoder Transformer architecture. We then propose a way to decouple lyrics from vocal melody generation, by inserting another decoder. Finally, we bring this symbolic output into the audio domain and perform a subjective evaluation study, using a singing voice synthesis model. In contrast to previous works that study vocal generation on the sentence level, we model full songs, which presents additional technical challenges.

Our main contributions to the field are the following: **(a)** We introduce the task of lyrics and vocal melody generation conditioned on the accompaniment. **(b)** We build a suitable dataset for this task by enforcing consistent tokenization. We apply musical analysis to compress the instrumental part up to 20% of the original, resulting to faster training. **(c)** We optimize the Transformer architecture in order to model full song sequences of up to 60k tokens in a single GPU. **(d)** We propose an architecture that decouples lyrics and vocal melody generation, providing the ability to use pretrained language models and predefined lyrics.

We release all code, datasets and some generated samples[1].

2 Related Work

Conditional Vocal Melody Generation In (Madhumani et al., 2020) a combination of word and syllable embeddings is used as input to an LSTM encoder (Hochreiter and Schmidhuber, 1997) that uses a vector to attend to three separate decoders, for note, duration and rest. Yu et al. (2021) use an LSTM that takes as input lyrics embeddings and noise vectors to sample MIDI sequences, which are provided to another LSTM alongside text embeddings and classified as real or fake. Another approach (Liu et al., 2020) studies singing voice generation without any melody or lyrics information, using GANs (Goodfellow et al., 2014) conditioned also on accompaniment.

[1] github.com/gulnazaki/lyrics-melody

Lyrics Generation In (Vechtomova et al., 2020) an LSTM-VAE creates latent representations of lyrics that condition lyrics generation on audio embeddings. Lu et al. (2019) use an encoder-decoder LSTM to generate lyrics, taking into account the only rhythmic quality of the melody. In (Watanabe et al., 2018) conditioning is done by concatenating each syllable to a local window of the corresponding melody note, before feeding it as input to an LSTM language model.

3 Dataset Description

Our dataset is built upon a subset of the Lakh MIDI Dataset (LMD) (Raffel, 2016) that consists of $45,129$ uniquely matched MIDI files.

3.1 Creating A More Consistent Dataset

LMD is not oriented towards the analysis of vocals. Therefore, many files do not include a vocal melody or synchronized lyrics. Moreover, the annotation of lyrics is not always consistent. Many tracks include different sentence and verse separators, mixing of MIDI lyrics and metadata, as well as inconsistent division of lyrics into sung syllables. The latter depends not only on the annotator but on the way the lyrics are sung, resulting, among others, to irregular tokenization of words.

In order to formalize our dataset, we construct a pre-processing pipeline. We keep only English lyrics, remove any metadata and use standard sentence and verse separators. To derive the vocal part, we assign each lyric to the closest note and choose the track with the most matches. To make the division of lyrics consistent and reversible at inference time we enforce a strict syllabified format, using Phonetisaurus (Novak et al., 2015) for grapheme-to-phoneme conversion. We split words into syllables, each one ending at a vowel. If a note corresponds to $n > 1$ syllables we divide it to n equal duration notes and if a syllable spans $n > 1$ notes we match it to the first one and assign the next $n - 1$ notes to a special symbol.

After completing the above process we are left with 8505 valid MIDI tracks.

3.2 Text Event Format

We create separate text event sequences for the instrumental and the vocal parts.

All sequences consist of the following types of tokens: (1) *Note on* (a note of this pitch starts), (2) *Note off* (a note ends), (3) *Wait time* (time passed in MIDI ticks[2]).

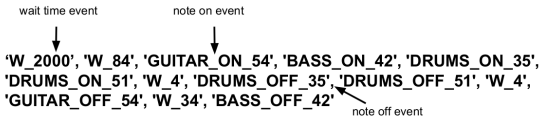

Figure 1: Instrumental Text Event Representation

We restrict notes in the piano range, shifting octaves if needed, resulting in 88 MIDI pitches. All instruments are grouped into 8 classes (*Guitar, Bass, etc.*) and each name is appended to the corresponding note token. An instrumental sequence can be seen in Figure 1.

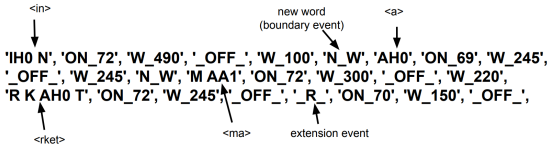

Figure 2: Vocal Text Event Representation with phonemes corresponding to the lyrics: *in a marke(-e)t*

Vocal sequences contain additional tokens: (1) *Syllable/phoneme* (syllable of the following note), (2) *Extension* (following note extends previous syllable), (3) *Boundary* (comma, word, line or verse separators) events.

For lyrics we use the extracted phonemes to reduce the vocabulary and account for rhyming and homophones. Figure 2 shows a vocal melody sequence with phonemes preceding each note.

3.3 Chord Reduction

The above representation results in very long sequences when applied to full instrumental tracks, imposing infeasible memory requirements and slow training. A more compact representation can benefit both model performance and robustness.

Vocal scores in jazz or orchestral music commonly use a reduced representation of the accompaniment that informs the singer about the harmonic and rhythmical structure of a song. Inspired by this, we create a chord reduction of the instrumental, using the *music21* library (Cuthbert and Ariza, 2010). Individual instruments are merged and every new note results in the formation of a

[2]Takes values in $[1, 2000]$ (more tokens needed for larger durations). Each tick corresponds to $\frac{60}{TR}$ seconds, T being tempo in *Beats Per Minute* and R being the file resolution in *Pulses Per Quarter Note*.

new chord. This method achieves a substantial reduction factor of $\frac{1}{5}$, as shown in Table 1.

	Vocal	Instrumental	Reduced
median	1645	13041	3220
max	6115	59120	11730

Table 1: Number of tokens for Vocal, Instrumental and Reduced Instrumental Sequences

We decide not to include the percussion part and instead use notions such as *beats* and *downbeats* events, which give more abstract but concrete rhythmical information. Besides the significant reduction in the instrumental sequences, we avoid inserting noise to the process, since drum parts are interpreted as pitches by *music21*.

3.4 Roman Numeral Analysis

To further capitalize on this chord representation, we employ a type of musical analysis, called Roman Numeral Analysis. Its core idea is that chords and notes can be represented by a degree of the musical scale they belong in[3]. Usage of roman numerals reduces the token vocabulary and enables us to model the relative position of chords, essentially performing data augmentation, since all songs are transposed to a common but abstract key.

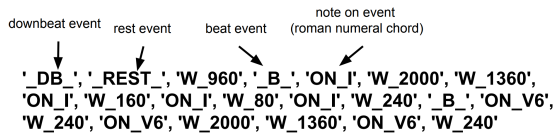

Figure 3: Instrumental Text Event Representation with Roman Numeral Chords, Rests, Downbeats and Beats

We get a key estimation using the Krumhansl-Schmuckler algorithm (Temperley, 1999) and based on that we represent each chord as a roman numeral. An example of this representation for instrumental sequences can be seen in Figure 3[4].

We apply a similar procedure for the vocal melody, by converting each note to a scale degree. Since pitch information is important to get more expressive vocal performances, we complement each token with its octave number.

[3]Uppercase roman numerals are used to represent major chords, while lowercase represent minor ones. Numbers denote inversions and extra chord notes.

[4]Since the formed chords mostly succeed each other directly, we do not need *note off* events and instead use *rest* events when required.

4 Proposed Method

4.1 Encoder-Decoder Architecture

We optimize a Transformer architecture to model long sequences and use it as our baseline to generate vocal sequences, given the instrumental.

A drawback of the Transformer architecture is that the memory footprint of the dot-product attention mechanism scales quadratically with the sequence length. To avoid this, we use the Performer (Choromanski et al., 2020), an architecture that achieves linear space and time complexity by using a mechanism called *FAVOR+*, which makes an unbiased linear estimation of full-rank softmax attention.

We further use reversible layers (Gomez et al., 2017), storing the activations of only the last layer, and feed-forward chunking as showcased in (Kitaev et al., 2020). We perform layer normalization before each sublayer (pre-norm) (Chen et al., 2018), reporting more stable training, with no need to do warm-up. Finally, we use learnable positional embeddings and tie the token embeddings of the decoder (Press and Wolf, 2017).

4.2 Decoupled Architecture

We augment the above architecture by adding a separate decoder for lyrics. We use the encodings of its last layer to further condition the vocal melody decoder, adding a second cross-attention layer to it (Libovický et al., 2018). The architecture is presented in detail in Figure 4.

It should be noted that in this architecture lyrics are generated without any instrumental conditioning. While we experimented with models that also use cross-attention in the lyrics decoder, we found them to perform poorly and be much harder to train in comparison to this simplified version.

We train our model to minimize the sum of the cross-entropy losses of the lyrics text and the vocal sequence (without phonemes). The lyrics decoder can use its own tokenization and the two sequences are merged at inference time, following the syllable tokenization process of Subsection 3.1. This decoupling allows us to predefine lyrics and use prior knowledge, which we achieve by using a language model pretrained on English lyrics.

To this end, we use a distilled[5] version of GPT-2 (Radford et al., 2019) that fits our low-resource requirements. We load the model's weights into an

[5]https://huggingface.co/distilgpt2

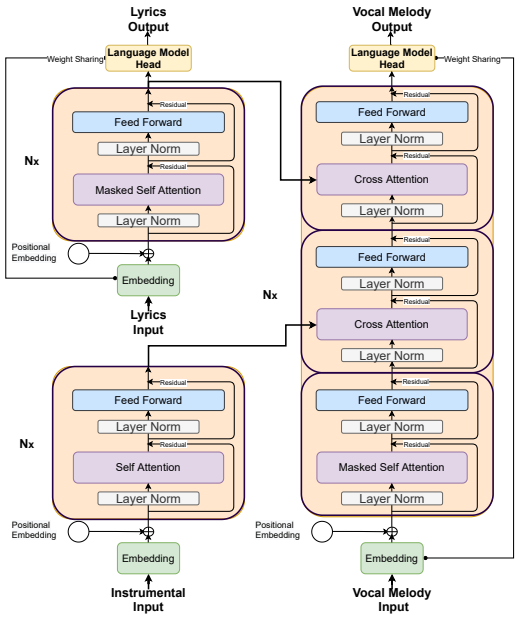

Figure 4: The Decoupled Architecture includes an instrumental encoder (bottom left), one decoder for lyrics (top left) and one decoder for vocal melody (right) with two cross-attention sublayers. The vocal melody decoder is conditioned by both instrumental and lyrics.

unconditional and causal decoder with *FAVOR+* attention, which we then fine-tune on a dataset of $263,666$ complete song lyrics. The purpose of this is twofold. As discussed in (Choromanski et al., 2020), fine-tuning is necessary for a Performer model to utilize the weights of a pretrained Transformer. Moreover, it helps our model adapt to the style of lyrics text (verse-chorus structure, repetition of words, onomatopoeia, etc.).

4.3 Experimental setup

We train **(a)** an encoder-decoder model on the full instrumental representation of Subsection 3.2 (*Vanilla Full - model A*), **(b)** an encoder-decoder on the reduced representation of Subsections 3.3, 3.4 (*Vanilla - model B*) and **(c)** a decoupled model on the latter representation (*Decoupled - model C*).

We use 6 layers with inner dimension of 512 and 8 attention heads for all models. The *AdamW* optimizer (Loshchilov and Hutter, 2019) is used with 0.001 learning rate and 0.1 weight decay.

During generation we use top-k sampling (Fan et al., 2018) keeping the top 0.1 tokens. We also take advantage of structural constrains in our representation (e.g. *note on* followed by *wait time* events only), by masking the valid tokens during inference.

We report that training for 6 epochs with batch size 8 takes a total of $\mathbf{74.6}$, $\mathbf{26.9}$ and $\mathbf{49.2}$ hours in an *NVIDIA Tesla T4 GPU* for models A, B and C respectively.

4.4 Subjective Evaluation

For 5 random instrumental tracks in the test set, we convert the outputs of our three models to audio, using a singing voice synthesis model called Mellotron (Valle et al., 2020). We then mix it with the synthesized instrumental. We ask 28 participant to choose between these samples on the basis of: **(a)** Rhythmic/Melodic Quality: how musical or interesting the vocal part is, **(b)** Relation to the Music: how well the vocal part fits with the instrumental, **(c)** Lyrical Content: quality of the generated lyrics. Table 2 shows the mean model preference values for all three objectives.

	Melody	**Relation**	**Lyrics**
Vanilla Full	0.369	0.246*	0.070
Vanilla	0.257*	0.374	0.052
Decoupled	**0.374**	**0.380**	**0.878***

Table 2: Mean value of each model preference for all 28 users. * denotes statistical significance ($p < 0.05$) using one-tail paired t-test (pairwise)

We observe that the reduced representation favors the instrumental-vocal relation, since it is more compact, but produces less interesting melodies. The decoupled model performs well on both metrics, which can be attributed to the separate modeling of the sequences, but does not reflect the independence between instrumental and lyrics. Finally, the lyrics generated by the decoupled model are significantly superior, which can be linked to the usage of a pretrained language model.

Using objective metrics to better understand the performance of these models remains to be done.

5 Conclusions

In this paper, we presented a pipeline to generate lyrics and vocal melody for any given instrumental MIDI file, using Transformer-based models. To this end, we have built and released a dataset oriented towards generation and study of vocals. We report that compressing the instrumental representation leads to substantially faster training and favors its connection to the vocal part. We further propose a decoupled architecture that allows us to use prior knowledge from language models and therefore generate more convincing lyrics.

References

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–86, Melbourne, Australia. Association for Computational Linguistics.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *CoRR*, abs/2009.14794.

Michael Scott Cuthbert and Christopher Ariza. 2010. Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*, pages 637–642. International Society for Music Information Retrieval.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Aidan N. Gomez, Mengye Ren, Raquel Urtasun, and Roger B. Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2214–2224.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer.

Jindřich Libovický, Jindřich Helcl, and David Mareček. 2018. Input combination strategies for multi-source transformer decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 253–260, Brussels, Belgium. Association for Computational Linguistics.

Jen-Yu Liu, Yu-Hua Chen, Yin-Cheng Yeh, and Yi-Hsuan Yang. 2020. Score and lyrics-free singing voice generation. In *Proceedings of the Eleventh International Conference on Computational Creativity, ICCC 2020, Coimbra, Portugal, September 7-11, 2020*, pages 196–203. Association for Computational Creativity (ACC).

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Xu Lu, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A syllable-structured, contextually-based conditionally generation of chinese lyrics. In *PRICAI 2019: Trends in Artificial Intelligence - 16th Pacific Rim International Conference on Artificial Intelligence, Cuvu, Yanuca Island, Fiji, August 26-30, 2019, Proceedings, Part III*, volume 11672 of *Lecture Notes in Computer Science*, pages 257–265. Springer.

Gurunath Reddy Madhumani, Yi Yu, Florian Harscoët, Simon Canales, and Suhua Tang. 2020. Automatic neural lyrics and melody composition. *CoRR*, abs/2011.06380.

Ryo Nishikimi, Eita Nakamura, Satoru Fukayama, Masataka Goto, and Kazuyoshi Yoshii. 2019. Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 161–165. IEEE.

Masanari Nishimura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. 2016. Singing voice synthesis based on deep neural networks. In *Interspeech*, pages 2478–2482.

Josef Novak, Nobuaki Minematsu, and Keikichi Hirose. 2015. Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework. *Natural Language Engineering*, -1:1–32.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Open AI Blog, 1(8)*.

Colin Raffel. 2016. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. PhD Thesis.

Daniel Stoller, Simon Durand, and Sebastian Ewert. 2019. End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model. In *IEEE International Conference on*

Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019, pages 181–185. IEEE.

David Temperley. 1999. What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered. *Music Perception: An Interdisciplinary Journal*, 17(1):65–100.

Rafael Valle, Jason Li, Ryan Prenger, and Bryan Catanzaro. 2020. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6189–6193. IEEE.

Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2020. Generation of lyrics lines conditioned on music audio clips. In *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, pages 33–37, Online. Association for Computational Linguistics.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 2018. A melody-conditioned lyrics language model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 163–172, New Orleans, Louisiana. Association for Computational Linguistics.

Yi Yu, Abhishek Srivastava, and Simon Canales. 2021. Conditional lstm-gan for melody generation from lyrics. *ACM Trans. Multimedia Comput. Commun. Appl.*, 17(1).