MODELING THE RHYTHM FROM LYRICS FOR MELODY GENERATION OF POP SONG

Daiyu Zhang **Ju-Chiang Wang** **Katerina Kosta** **Jordan B. L. Smith** **Shicen Zhou**
ByteDance
{daiyu.zhang, ju-chiang.wang, katerina.kosta, jordan.smith, zhoushicen}@bytedance.com

ABSTRACT

Creating a pop song melody according to pre-written lyrics is a typical practice for composers. A computational model of how lyrics are set as melodies is important for automatic composition systems, but an end-to-end lyric-to-melody model would require enormous amounts of paired training data. To mitigate the data constraints, we adopt a two-stage approach, dividing the task into lyric-to-rhythm and rhythm-to-melody modules. However, the lyric-to-rhythm task is still challenging due to its multimodality. In this paper, we propose a novel lyric-to-rhythm framework that includes part-of-speech tags to achieve better text-setting, and a Transformer architecture designed to model long-term syllable-to-note associations. For the rhythm-to-melody task, we adapt a proven chord-conditioned melody Transformer, which has achieved state-of-the-art results. Experiments for Chinese lyric-to-melody generation show that the proposed framework is able to model key characteristics of rhythm and pitch distributions in the dataset, and in a subjective evaluation, the melodies generated by our system were rated as similar to or better than those of a state-of-the-art alternative.

1. INTRODUCTION

Setting lyrics to a melody is a common but complex task for a composer. The form, articulation, meter, and symmetry of expression in lyrics can inspire, or set constraints on, the melodic arrangement. Given the importance of melody, it is unsurprising that the decades-long history of Music Metacreation systems includes countless melody-creation systems (see [1] for a review). However, less attention has been paid to the lyric-to-melody generation task (i.e., generating a melody for given input lyrics). The task is challenging for many reasons, including but not limited to: the need to handle the prosody of the text correctly (e.g., one should avoid setting an unstressed word like ‘the’ on a stressed note in the melody); the need to reflect the structure of the lyrics in the melody; and the need to create a good melody to begin with.

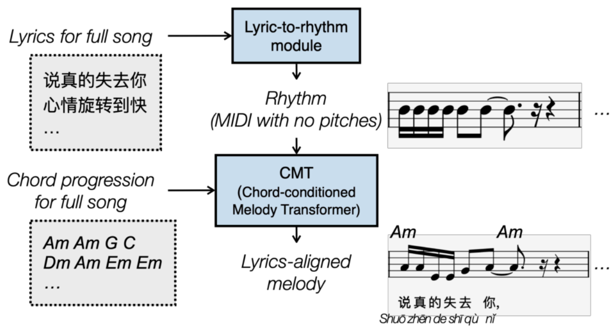


Figure 1: Diagram of the proposed system.

With the rapid growth of deep learning tools, this task has gained more attention, and there are many recent examples of lyric-to-melody creation systems, most using an end-to-end approach [2–5]. Modeling the relationship between lyric syllables and musical notes is a complex, cross-modal task, but it is hoped that we can succeed with a large amount of paired examples (i.e., lyrics aligned to their corresponding melodies). However, acquiring such data is expensive, and using unsupervised learning has shown limited performance gains [3]. All the systems mentioned here are trained on fewer than 200,000 examples of song lyrics; by contrast, the text-to-image system DALL-E has 12 billion parameters and involved hundreds of millions of paired text-image training data [6].

One alternative, suggested in [7], is to pick an intermediate representation and adopt a two-stage approach: one model to convert lyrics to the chosen representation, and a second to convert that to a melody. The motivation is that there is sufficient data to train each model separately, without the paired lyrics-melody data required by the end-to-end approach.

We choose ‘rhythm’ as the intermediate step because, if we disregard melismas and expressive singing techniques, we can assume there is a one-to-one correspondence between syllables and onsets, and between onsets and melody pitches. Also, there is plenty of data to model each step: first, from karaoke-style scrolling lyrics data, we can obtain an alignment between syllables in lyrics and note onsets in music, and thus note durations and metrical positions, too. Second, there are multiple public datasets from which to learn to assign pitches for each note given their duration. Our goal is then to solve two sub-tasks, namely lyric-to-rhythm and rhythm-to-melody, with an assumption that the rhythm generation process is independent of the pitch generation one [7].

There are many recent melody generation models [8–11], but lyric-to-rhythm modeling is rarely attempted. In this paper, we introduce a novel framework for converting lyrics to rhythms using an encoder-decoder Transformer architecture [12]. The proposed system is outlined in Fig. 1: given an input set of lyrics, a lyric-to-rhythm module assigns onset times and durations for each syllable. This rhythm, along with a user-provided chord progression, is fed into a Chord-conditioned Melody Transformer (CMT) [13], a state-of-the-art melody generation system, to predict the pitch for each note. The details of the lyric-to-rhythm module and the CMT are provided in Sections 3.3 and 3.2, respectively.

2. BACKGROUND

Lyrics and melody are not arbitrarily combined; common sense suggests and prior analysis [14] indicates that patterns in lyrics and melodies are related and can be modeled, in part, with features of the melody (e.g., note duration) and lyrics (e.g., syllable stress). One of the earliest lyric-to-melody systems was designed to handle Japanese prosody [15]: first, the input text was segmented into phrases; next, a set of pre-composed rhythms was searched for one that fit the syllable count and matched the accent pattern of the text; finally, pitches were assigned using dynamic programming to optimise the interval directions with the natural prosody of the words. An earlier lyric-to-rhythm system also leveraged a dataset of pre-composed rhythms that were scored based on their match to the input syllable-stress and word-rarity patterns [16]. Although our system has little in common with these works, we do share the use of rhythm as an intermediate representation.

Algorithms for automatic music generation are a subset of Music Metacreation systems [1], which have been present in Western music in many forms, including being used for the creation of standalone pieces and, either offline or in real-time, as part of the human composition process. With the help of machine learning and deep learning architectures, many such systems have shown to be capable of generating a plausible outcomes that match the musical characteristics of given datasets. Supervised generative models aim to learn a representation of the underlying characteristics of a training set distribution. Depending on the model, this representation can be either explicitly depicted or implicitly used to generate samples from the learned distribution [17].

Some systems aim to generate a part of a musical piece with the aid of another given part (including melody-to-lyrics creation [18], the inverse of the task we consider). Conditioning the choice of parameters in a generative model on data from other modalities, such as a bass line or a structure, can yield controllable generation systems [19][p.82-83]. For the case of using chords to condition melody generation, a recent system adjusting a general adversarial network architecture has been presented in [20] with the option of generating melody lines over a given accompaniment. The Chord-conditioned Melody Transformer (CMT) [13] is the most recent effort in this area; we adapt much of the design of this system, extending it to accept both lyrics and chords as input. Details of this system, and how we adapt it, follow in Section 3.

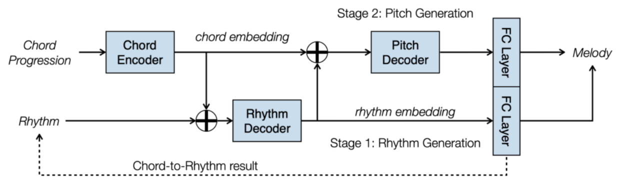


Figure 2: A two-stage structure of CMT, where $\oplus$ represents concatenation. Stage 1: chord-to-rhythm. Stage 2: rhythm+chord-to-pitch based on the result of Stage 1.

3. METHODOLOGY

3.1 System Overview

Our system design is motivated by the Chord-conditioned Melody Transformer (CMT) [13]. The authors of CMT proposed a two-stage system, assuming a hierarchy that the process of generating melodies is two-phase, as depicted in Fig. 2: *Stage 1*, generating the rhythm of notes from chord progressions; *Stage 2*, generating the pitch for each note depending on the chord progressions and generated rhythm. Our proposed system augments CMT by replacing chord-to-rhythm (i.e., Stage 1) with a novel *lyric-to-rhythm module*. As a result, users can input the lyrics and chord progression of a full song in our system (see Fig. 1). Then, the lyric-to-rhythm module generates the MIDI (with empty pitches). Second, CMT processes the MIDI and chord progression to generate the melody. As a result, the rhythm is generated with a global view of the lyrics, while the melody is generated with a causal view of the rhythm and chords.

In the following subsections, we will first review CMT and explain the difficulties of modifying it to handle the lyric-to-melody task in Section 3.2. Then, we will detail our solution in Section 3.3.

3.2 Chord-Conditioned Melody Transformer (CMT)

CMT adopts a pianoroll-like representation [13, 21] that includes chord, rhythm, and pitch (CRP) information. It splits the timeline into semiquaver-length frames (1/4 of a beat), each described by three vectors: a 12-dimensional binary *chord* vector (pitch classes in the chord get a 1); a 3-dimensional one-hot *rhythm* vector (onset, hold state, rest state); and a 22-dimensional one-hot melodic *pitch* vector (for this part we restrict MIDI pitches to between 48 and 67, plus a hold state and a rest state, giving a total dimension of 22). Please refer to [13][Fig. 1] for an illustration.

CMT contains three main modules: *Chord Encoder (CE)*, a bidirectional LSTM [22]; *Rhythm Decoder (RD)*, a stack of self-attention blocks; and *Pitch Decoder (PD)*, another stack of self-attention blocks. In Stage 1, given an input chord progression, the chord embedding encoded

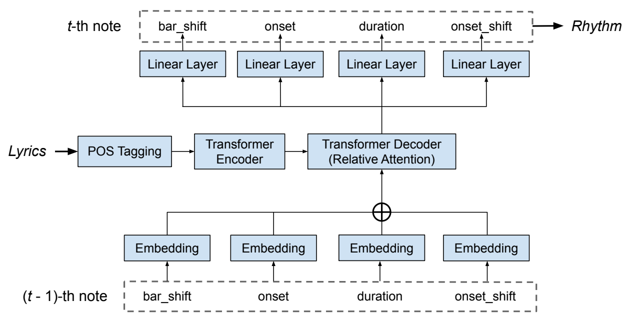


Figure 3: The proposed lyric-to-rhythm framework.

by CE is autoregressively sent into RD to output the rhythm embedding, followed by a fully-connected layer ("FC Layer" in Fig. 2) to predict the sequence of rhythm vectors for the entire song. In Stage 2, the concatenation of the chord and rhythm embeddings is autoregressively fed into PD, followed by a fully-connected layer to predict the sequence of pitch vectors. Finally, rhythm and pitch vectors are combined and converted to the melody.

However, to leverage CMT for the lyric-to-melody task, we face three problems. (1) *Multimodality*: CMT was designed to take the input of a chord progression to generate the melody. However, it is non-trivial to directly add a lyric encoder for lyrics input, as lyrics are more complicated sequential data than chords. (2) *Representation*: CMT uses a pianoroll-like (i.e. CRP) representation to encode melody, where the time axis is evenly scaled (e.g., 1/16 beat), so a note may require multiple tokens to carry the duration. This makes it difficult to create a one-to-one mapping that ties a syllable (or character) to a single note token. (3) *Constraint on length*: in CMT, the CE generates the melody on a segment-to-segment basis (e.g., 8 bars at a time) without exploiting the global context of a full-song chord progression. However, we believe the structural information carried in the input lyrics is crucial to determine the repetitive pattern for the output melody. The next subsection details how we address these problems: (1) is addressed with a POS tagger that compactly encodes useful lyrics information; and (2) and (3) are addressed by adapting a Compound Word representation.

3.3 Lyric-to-Rhythm Framework

Fig. 3 shows our lyric-to-rhythm framework, which is analogous to a language translation task: i.e., an input sequence of lyrics is translated into an output sequence of notes. In this work, we assume that each syllable (or character) is mapped onto one note as a simplification; handling melismas remains a future challenge. To this end, we adapt an encoder-decoder Transformer architecture [12]. To enhance the repetitive coherence modeling in note sequences, we incorporate relative self-attention [23, 24].

To extract the features of lyrics, we employ *part-of-speech (POS) tagging* with a Transformer encoder. Following prior works [25, 26], we characterize the rhythmic features of a note with a tuple of (`bar_shift`, `position`, `duration`, `onset_shift`), and model the sequence of tuples using the *Compound Word (CP)* Transformer decoder [27]. The lyric-to-rhythm module generates the t-th note based on the full context of lyrics and the previously generated notes (from the first to (t-1)-th) in an auto-regressive manner, with the future notes being masked. We describe the POS tag representation and CP Transformer in the next subsections.

Token Name	Vocab.	Description
`bar_shift`	0, 1, 2	Time shift in bar to current bar
`onset`	0 – 15	Onset in 1/4 beat in current bar
`duration`	0 – 31	Duration in 1/4 beat
`onset_shift`	0 – 15	Time shift in 1/4 beat to previous note's onset

Table 1: Rhythmic features for a note.

3.3.1 Part-of-Speech (POS) Tagging

In natural language processing, POS tagging refers to the process of labeling every word in a text with its part of speech. The taxonomy of POS tags varies by language, but commonly includes 'noun,' 'verb,' 'adjective,' 'adverb,' and others. POS tags can augment the text information by indicating the structure of sentences [28], and thus plays an important role in tokenizing the input words in conventional text-to-speech (TTS) systems [29, 30].

POS are word-level descriptors, but we want syllable-level descriptors in order to align the lyrics with the rhythm. (When dealing with Chinese lyrics, we can also say 'character-level' since each Chinese character is one syllable.) Thus, we combine each POS tag with the syllable index to create a POS 'token': e.g., the input English sentence "Why not tell someone," would result in: ['adverb-0', 'adverb-0', 'verb-0', 'noun-0', 'noun-1'], where the two syllables in "someone" are represented by ['noun-0', 'noun-1'].

3.3.2 Compound Word Transformer

In contrast to the CP proposed in [27], we do not distinguish between *note*- and *metre*-related events. Instead, we include four tokens in every compound word (see Table 1), so that we can have one set of tokens per syllable. From karaoke scrolling lyrics data, we can obtain the onset and duration of each syllable, and by tracking the downbeats, we can obtain the metric position in the bar. Following [27], each of the four tokens is converted into an embedding, and then the embeddings are concatenated before being sent to the Transformer decoder. Each of the four output embeddings is linearly projected to predict the value for the associated token of the t-th note.

Once all the notes are ready, we convert them to MIDI (with unspecified pitch) with the following steps:

1. Place an empty note at bar 0 and position 0.
2. Determine the onset by shifting (`bar_shift`$\times 16+$ `onset_shift`) units from the previous onset.
3. Set the duration by min(`duration`, next note's `onset_shift`).
4. Repeat 2 and 3 until all the notes are processed.

We note that `onset` is not used for generating MIDIs. Instead, we use the position shifted to determine the onset so that notes are placed in an incremental order. Nevertheless, we suspect that `onset` can help regularization in training. Using the CP representation addresses the "constraint on length" issue mentioned in Section 3.2, as it permits a more compact sequence of tokens that can model a longer duration, such as a full song.

4. SYSTEM CONFIGURATION

This section describes how we trained the lyric-to-rhythm and rhythm-to-melody models. For each model we explain what data were used and how they were collected. We focus on Chinese pop songs to validate our system, but the framework could be adaptable to other languages since parts of speech and syllables are broadly useful concepts.

4.1 Lyric-to-Rhythm Model

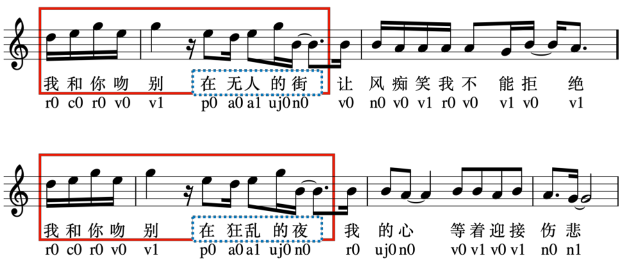
We collected data for 45K Chinese pop songs using a similar pipeline as [31] and [7][Appendix A]. That is, we crawled online to obtain paired lyrics and audio, with timestamps indicating the onset of each line of the lyrics. Then, for each song, we performed the following steps: (1) isolate the vocal audio using source separation; (2) convert lyrics to phoneme sequences; (3) estimate the phoneme onset timestamps using forced phoneme alignment; and (4) estimate the time signature and beat and downbeat times. From the phoneme and beat data, we can derive the syllable onsets and thus the bar-shift, onset, duration and onset-shift attributes required by the model. The steps were performed using in-house tools comparable to those used in [7]: Spleeter [32], Phonemizer [33], Montreal Forced Aligner [34], and Madmom [35], respectively.

We kept songs with a detected time signature of 4/4 (around 90%), and quantized the timestamp of each syllable in quarter beats. Errors in automatic lyrics alignment, in beat tracking, and in the detected time signature can all degrade the model quality, so we selected 330 songs to manually adjust the timestamps. This subset was used to fine-tune the model.

For POS tagging, we adopted Jieba [1], an open-source tool that supports 56 tags commonly used in Chinese. Without POS tags, the vocabulary size for our dataset was 5,368 unique characters. Reducing to POS and then adding the syllable index resulted in a vocabulary of 123 unique POS tokens. In Sec. 5.2, we will compare a model using this 123-dimensional POS vector to an ablated version of the system that encodes the raw characters index in a 5368-dimensional vector.

In Chinese pop songs, symmetric expression of text structure is commonly reflected in melody repetition. Fig. 4 shows one example: the chorus melody of "Goodbye Kiss" [2] by Jacky Cheung. The two phrases outlined in solid boxes are identical in melody, and nearly identical in text; but even where the lyrics are different (in the dotted

[1] `https://github.com/fxsjy/jieba`
[2] `https://www.youtube.com/watch?v=bJRkEmrkIO4`



我 和 你 吻 别 在 无人 的 街 让 风 痴 笑 我 不 能 拒 绝
r0 c0 r0 v0 v1 p0 a0 a1 uj0 n0 v0 n0 v0 v1 r0 v0 v1 v0 v1
我 和 你 吻 别 在 狂乱 的 夜 我 的 心 等 着 迎 接 伤 悲
r0 c0 r0 v0 v1 p0 a0 a1 uj0 n0 r0 uj0 n0 v0 v1 v0 v1 n0 n1

Figure 4: The melody-lyrics-POS example in the chorus section of "Goodbye Kiss". POS abbreviation key: {'r': pronoun, 'c': conjunction, 'v': verb, 'p': preposition, 'a': adjective, 'n': noun, 'uj': auxiliary}.

boxes), they have the same POS tags. With the POS tagging representation, we believe the lyric-to-rhythm model can learn to generate similar rhythms for two text phrases if they have a common structure.

We use the following parameters for the encoder-decoder Transformer: the input length is 1000; the numbers of heads, encoder-layers, and decoder-layers are 8, 6, and 6, respectively; the embedding sizes of lyrics, bar_-shift, onset, duration, and onset_shift are 512, 32, 128, 128, and 128, respectively; dropout is 0.1; batch size is 16; and learning rate is 1e-5 with Adam optimizer. Using a single Tesla-V100-SXM2-32GB GPU to train a satisfactory model takes $\sim$10 hours on the automatically aligned dataset plus 1.5 hours on the manually annotated subset. In both cases we use 10 percent of the dataset for validation.

4.2 Rhythm-to-Melody Model

To train the rhythm-to-melody model, we used POP909 [36] and Lead-Sheet-Dataset [37]. POP909 contains data on 909 Chinese pop songs including chords, melody in MIDI format, and other information not used here. Lead-Sheet-Dataset (LSD), a collection of symbolic content sourced from HookTheory, [3] contains lead-sheets (i.e., melodies and chords) of 16K song segments in MIDI format. We used the songs in 4/4 time (dropping roughly 10% of the data) and transposed all pieces to the key of C major or A minor. This resulted in about 30K bars of music from POP909 and about 40K from LSD.

We followed [13] to train the model, setting the input length to be 8 bars but reducing the pitch range from 48 to 20; any pitches outside the range were octave-shifted to lie within the range. After obtaining the rhythm MIDI of a full song from the lyric-to-rhythm module, pitches were generated for 8 bars autoregressively, with a 4-bar sliding window, i.e., the model composes the next 4 bars given the previous 4 bars already composed.

5. EVALUATION

We would like to answer two questions: first, does our system succeed in emulating basic musical qualities of the training data? And second, does it produce pleasing, viable

[3] `https://www.hooktheory.com/`

settings of lyrics? To answer the first, we compare the output melodies of our model (denoted 'pop-melody') to the held-out training data and discuss their similarity. For the second, we conducted a listening test in which participants rated the quality of the lyric settings of our model as well as those of a state-of-the-art alternative, TeleMelody [7].

5.1 Objective Results

We analyze the melodies created by our system in two objective evaluation strands. The first one is to demonstrate how similar the rhythms generated by our model are to the original data (see Fig. 5); the second is to look for and characterize the differences between the melodies produced by the two models (see Fig. 6). We compare statistics over several musical quantities computed on the dataset and compositions generated by both systems. For this comparison, we have generated 400 scores from each system and used the same amount of scores from the dataset.

Most of the musical quantities we compute are adapted from [38] and [39]. These symbolic descriptors have been shown to enhance melodic expectation when embodied in a cognitively plausible system for music prediction. Expectation and memorability have been shown to be important characteristics for identifying a plausible melody, and surprise and repetition are measurable elements that relate to these charactertistics. (For more background on such descriptors and on the concepts of predictability and uncertainty in the pleasure of music, see [40, 41].)

We showcase two sets of descriptors, one for each evaluation strand. The first contains: the *duration* of the melody notes; their *inter-onset intervals* (IOIs; the distance between the start of a note to the start of the preceding one); and their metrical *position in bar*. Fig. 5 shows the distributions of these descriptors for the dataset and for the outputs of our system (tagged as "pop-melody") before and after fine-tuning (see Section 4.1). Note that we exclude TeleMelody from this comparison since it was trained on a different dataset (of around 110K samples), so it is not meaningful to compare it to our training data.

Judging from the distributions, the outputs of both models are broadly similar to the melodies in the dataset. However, there is clearly a surfeit of short notes (0.25 crochets, or sixteenth notes) in the generated melodies, which skews the distribution of IOIs as a result. Also, regarding note position in the bar, there is a subtle variation of the likely onset positions in the dataset that is not reflected in the generated data, which, prior to fine-tuning, has an almost uniform distribution.

The other set of descriptors contains: the *pitch contour*, which gives the likelihood that the next note in the melody will be lower (descending), higher (ascending) or the same; *note sparsity*, which gives the fraction of the timeline which has no note in the melody (a value of 0 indicates no rests in the melody); and the *pitch-in-chord-triads ratio*, a kind of 'consonance' metric, calculated as the fraction of notes in the melody that belong to the accompanying chord triad.

These descriptors are illustrated in Fig. 6, comparing

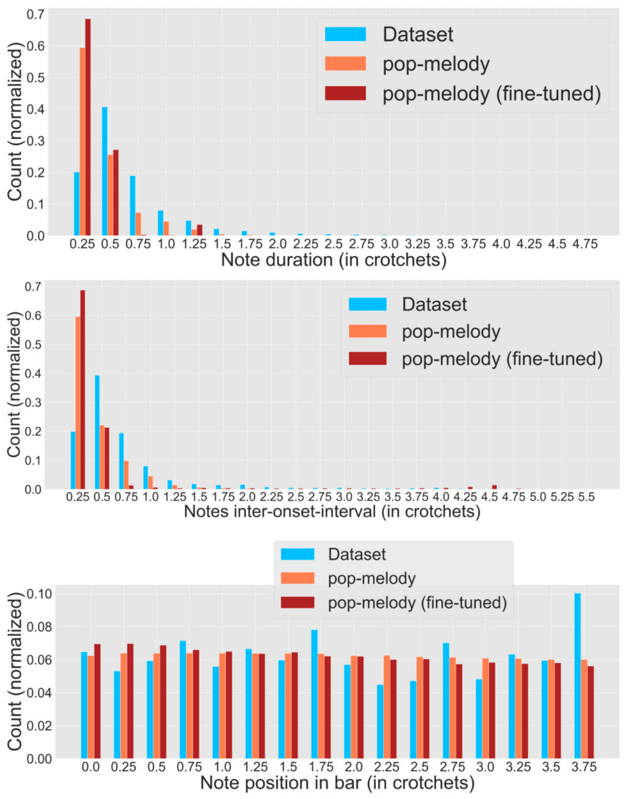


Figure 5: Distributions of descriptors values derived from melodies from the dataset and melodies generated by the proposed pop-melody system.

the melodies from our fine-tuned system ("pop-melody") with TeleMelody. Here, the purpose is not to compare the systems to the data—they were trained on different data, and may each reflect their training set well—but to assess how the melodies of the systems differ. From the contour descriptors, it is clear that TeleMelody is more likely to generate many consecutive notes with the same pitch, whereas melodies from the proposed system have more variation. The melodies from our system also tend to have fewer rests, and tend to include more notes that appear in the underlying chord. The latter can be interpreted as a tendency to stay in consonance and limiting the space of "dissonant" or "unexpected" moments.

5.2 Subjective Results

We conducted a subjective listening test using a similar design as [3, 7, 42]: we selected lyrics from ten random songs from the test portion of the dataset of 45K songs and used these as input to three systems to generate melodies: "TeleMelody"; "Pop-melody", the proposed system; and "Baseline", an ablated version of our system that does not use POS tokens (see Sec 4.1). This resulted in 30 full songs: ten triples with the same lyrics, chords, and tempo. We rendered the lyrics and melodies to audio with an in-house singing voice synthesis comparable to Xiaoice [42] and rendered a simple accompaniment with the chords.

We had 20 participants, all of whom had some musical background and could read musical scores and play an in-

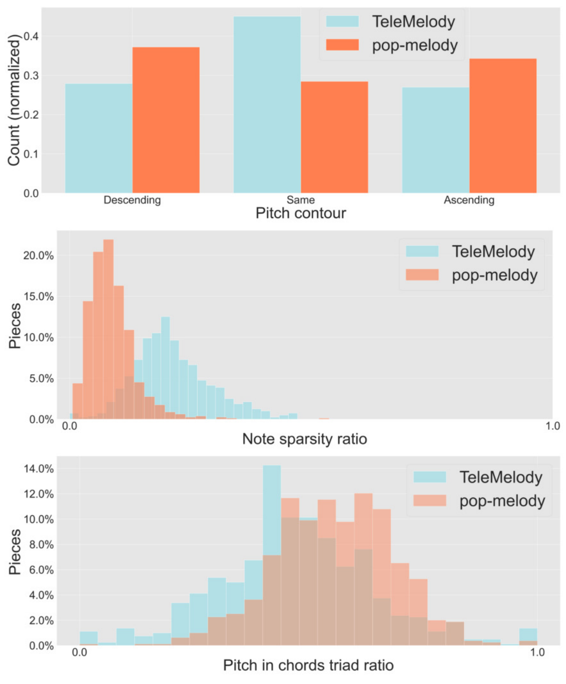


Figure 6: Distributions of descriptor values derived from melodies generated by TeleMelody [7] and by the proposed pop-melody system.

strument. Participants listened to a triple at a time, where the system identities were masked and the order is at random. Then, they rated each song on four criteria on a Likert scale from *Bad* (1) to *Excellent* (5):

1. **Rhythm**: is the timing of notes suitable for the lyrics?
2. **Harmony**: do the pitches fit the chords and key?
3. **Melody**: does the melody line sound natural with the lyrics?
4. **Overall**: what is the overall quality of the melody?

After rating each triple, listeners also rated their familiarity with the original song of the input lyrics on a 5-point Likert scale. The average rating here was 1.5: somewhere between "1. Never heard the title or melody" and "2. Heard the song title, but not the melody".

The results of the study are shown in Table 2. Overall, listeners gave the three systems similar average ratings: all lie within 3.6 ± 0.25. However, Wilcoxon signed-rank tests reveal small but consistent differences between the systems; see Table 3 for the p-values of all comparisons. First, we see that the proposed system is consistently better than Baseline, suggesting that POS-based tokenization is effective. Second, we find that the proposed system also matches or outperforms TeleMelody; the difference is greatest for rhythmic quality. Despite the broadly positive ratings, mostly between *Fair* (3) and *Good* (4), comments from the participants mostly cited shortcomings of the output. TeleMelody and Pop-melody both earned comments that the "melody is a little weird" and sometimes "too repetitive", but only the TeleMelody outputs earned comments that the "rhythm is a little weird" and "fragmented".

System	Rhythm	Harmony	Melody	Overall
Baseline	3.42(.93)	3.67(.86)	3.46(.93)	3.42(.82)
TeleMelody	3.58(.96)	3.69(.90)	3.38(.85)	3.57(.73)
Pop-melody	3.84(.87)	3.87(.81)	3.64(.89)	3.68(.72)

Table 2: Subjective result and comparison.

Comparison	Rhythm	Harmony	Melody	Overall
Pop vs Tele	0.0006	0.007	0.001	0.1
Pop vs Baseline	1.1e-08	0.002	0.006	1.8e-05
Tele vs Baseline	0.01	0.89	0.22	0.02

Table 3: P-values of the subjective result comparison.

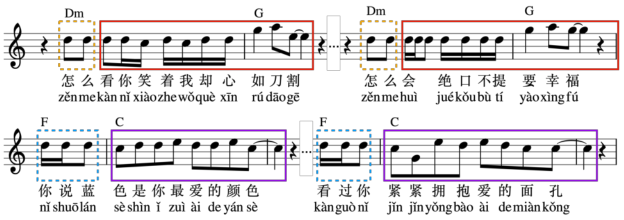

Dm
G
Dm
G
怎 么 看 你 笑 着 我 却 心 如 刀 割
zěn me kàn nǐ xiào zhe wǒ què xīn rú dāo gē
怎 么 会 绝 口 不提 要 幸 福
zěn me huì jué kǒu bù tí yào xìng fú
F
C
F
C
你 说 蓝 色 是 你 最 爱 的 颜 色
nǐ shuō lán sè shìn ǐ zuì ài de yán sè
看 过 你 紧 紧 拥 抱 爱 的 面 孔
kàn guò nǐ jǐn jǐn yǒng bào ài de miàn kǒng

Figure 7: Output examples (a) above and (b) below.

Two output examples from our system are shown in Figs. 7(a) and 7(b). In both cases the melodies follow the input chord progressions and we find parallelism and variation in the melody when the lyric structure recurs. E.g., in Fig. 7(a), the similar lyrics begin with the same two melody notes (dashed boxes), and the remainders have similar rhythm and contour (solid boxes). Similarly, in Fig. 7(b), the similar lyrics are given identical openings (dashed boxes) with rhythmically identical continuations (solid boxes).

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a new approach to generate melody for a given lyric by combining lyric-to-rhythm and rhythm-to-melody modules. We found that listeners rated the long-term text-settings provided by our system as acceptable, and at least as good as a competing system.

In order to achieve a cross-modal mapping from syllables to onsets to melody notes, we made the simplifying assumption that each syllable is sung on one note. There is a clear way to improve this in the lyric-to-rhythm module by adding a syllable-state token to the Compound Word, indicating whether we are at the onset of a syllable, or the continuation of one. However, allowing a one-to-many syllable-to-note mapping would also complicate the automatic syllable alignment step, making the hand-corrected data even more precious.

We also found that POS tags were valuable text tokens; using them led to a boost in text-setting quality. Given this success, we ought to leverage more linguistic information, such as syllable stress and word frequency, as in [15, 16]. Music structure labels (e.g., verse and chorus) could also prove valuable. This is an under-explored area, but may become feasible with the introduction of more datasets, or with automatic labelling systems [43] to further augment existing data. if more datasets become available.

7. REFERENCES

[1] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov, "An introduction to musical metacreation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 1–14, 2017.

[2] Y. Yu, A. Srivastava, and S. Canales, "Conditional LSTM-GAN for melody generation from lyrics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1, pp. 1–20, 2021.

[3] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "Songmass: Automatic song writing with pre-training and alignment constraint," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 798–13 805.

[4] H. Bao, S. Huang, F. Wei, L. Cui, Y. Wu, C. Tan, S. Piao, and M. Zhou, "Neural melody composition from lyrics," in *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 2019, pp. 499–511.

[5] H.-P. Lee, J.-S. Fang, and W.-Y. Ma, "iComposer: An automatic songwriting system for Chinese popular music," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 84–88.

[6] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proc. 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8821–8831.

[7] Z. Ju, P. Lu, X. Tan, R. Wang, C. Zhang, S. Wu, K. Zhang, X. Li, T. Qin, and T.-Y. Liu, "TeleMelody: Lyric-to-melody generation with a template-based two-stage method," *arXiv preprint arXiv:2109.09617*, 2021.

[8] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. ISMIR*, 2020, pp. 77–84.

[9] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. ISMIR*, 2021, pp. 143–150.

[10] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. IEEE, 2020, pp. 128–135.

[11] H. H. Tan, "ChordAL: A chord-based approach for music generation using Bi-LSTMs." in *Proc. International Conference on Computational Creativity (ICCC)*, 2019, pp. 364–365.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] K. Choi, J. Park, W. Heo, S. Jeon, and J. Park, "Chord conditioned melody generation with transformer based decoders," *IEEE Access*, vol. 9, pp. 42 071–42 080, 2021.

[14] E. Nichols, D. Morris, S. Basu, and C. Raphael, "Relationships between lyrics and melody in popular music," in *Proc. ISMIR*, 2009, pp. 471–476.

[15] S. Fukayama, K. Nakatsuma, S. Sako, T. Nishimoto, and S. Sagayama, "Automatic song composition from the lyrics exploiting prosody of the Japanese language," in *Proc. 7th Sound and Music Computing Conference (SMC)*, 2010, pp. 299–302.

[16] E. Nichols, "Lyric-based rhythm suggestion," in *Proc. International Computer Music Conference*, 2009.

[17] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[18] X. Ma, Y. Wang, M.-Y. Kan, and W. S. Lee, "AI-Lyricist: Generating music and vocabulary constrained lyrics," in *Proc. 29th ACM International Conference on Multimedia*, 2021, pp. 1002–1011.

[19] J.-P. Briot, G. Hadjeres, and F. D. Pachet, *Deep learning techniques for music generation*. Springer Cham, 2020.

[20] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. ISMIR*, 2017, pp. 324–331.

[21] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. ISMIR*, 2019, pp. 596–603.

[22] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 799–804.

[23] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 464–468.

[24] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[25] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," *arXiv preprint arXiv:2106.05630*, 2021.

[26] Y.-H. Chou, I. Chen, C.-J. Chang, J. Ching, Y.-H. Yang *et al.*, "MidiBERT-Piano: Large-scale pre-training for symbolic music understanding," *arXiv preprint arXiv:2107.05223*, 2021.

[27] 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," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 178–186.

[28] M. Sun and J. R. Bellegarda, "Improved POS tagging for text-to-speech synthesis," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5384–5387.

[29] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Prentice Hall, 2008.

[30] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0." in *SSW*, 2007, pp. 294–299.

[31] L. Xue, K. Song, D. Wu, X. Tan, N. L. Zhang, T. Qin, W.-Q. Zhang, and T.-Y. Liu, "DeepRapper: Neural rap generation with rhyme and rhythm modeling," in *Proc. Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 2021, pp. 69–81.

[32] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[33] M. Bernard and H. Titeux, "Phonemizer: Text to phones transcription for multiple languages in Python," *Journal of Open Source Software*, vol. 6, no. 68, p. 3958, 2021.

[34] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable text-speech alignment using Kaldi." in *Proc. Interspeech*, 2017, pp. 498–502.

[35] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. 24th ACM International Conference on Multimedia*, 10 2016, pp. 1174–1178.

[36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. ISMIR*, 2020, pp. 38–45.

[37] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research (JNMR)*, vol. 50, no. 1, pp. 37–51, 2021.

[38] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[39] M. T. Pearce, "The construction and evaluation of statistical models of melodic structure in music perception and composition," Ph.D. dissertation, City University London, 2005.

[40] B. P. Gold, M. T. Pearce, E. Mas-Herrero, A. Dagher, and R. J. Zatorre, "Predictability and uncertainty in the pleasure of music: A reward for learning?" *Journal of Neuroscience*, vol. 39, no. 47, pp. 9397–9409, 2019.

[41] D. Müllensiefen, "Fantastic: Feature analysis technology accessing statistics (in a corpus): Technical report v1," *London, England: Goldsmiths, University of London. Retrieved from http://www.doc.gold.ac.uk/isms/m4s*, 2009.

[42] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.

[43] J.-C. Wang, Y.-N. Hung, and J. B. L. Smith, "To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 416–420.