

KeYric: Unsupervised Keywords Extraction and Expansion from Music for Coherent Lyrics Generation

XICHU MA, VARUN SHARMA, MIN-YEN KAN, WEE SUN LEE, and YE WANG, School of Computing, National University of Singapore, Singapore

We address the challenge of enhancing coherence in generated lyrics from symbolic music, particularly for creating singing-based language learning materials. Coherence, defined as the quality of being logical and consistent, forming a unified whole, is crucial for lyrics at multiple levels—word, sentence, and full-text. Additionally, it involves lyrics’ musicality—matching of style and sentiment of the music. To tackle this, we introduce KeYric, a novel system that leverages keyword skeletons to strengthen both coherence and musicality in lyrics generation. KeYric employs an innovative approach with an unsupervised keyword skeleton extractor and a graph-based skeleton expander, designed to produce a style-appropriate keyword skeleton from input music. This framework integrates the skeleton with the input music via a three-layer coherence mechanism, significantly enhancing lyric coherence by 5% in objective evaluations. Subjective assessments confirm that KeYric-generated lyrics are perceived as 19% more coherent and suitable for language learning through singing compared to existing models. Our analyses indicate that integrating genre-relevant elements, such as pitch, into music encoding is crucial, as musical genres significantly affect lyric coherence.

CCS Concepts: • **Applied computing** → **Sound and music computing**; *Education*; • **Computing methodologies** → **Natural language processing**.

Additional Key Words and Phrases: Lyrics Generation; Keyword Extraction; Textual Coherence; Language Learning; Graph Learning

ACM Reference Format:

Xichu Ma, Varun Sharma, Min-Yen Kan, Wee Sun Lee, and Ye Wang. 2023. KeYric: Unsupervised Keywords Extraction and Expansion from Music for Coherent Lyrics Generation. *J. ACM* 1, 1 (September 2023), 27 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

1 INTRODUCTION

Coherence is often lacking in existing lyrics generation systems. This reduces the comprehensibility, engagement, and artistic expression of the content. This paper focuses on enhancing the coherence of generated lyrics, defined as the quality of being logical and consistent, forming a unified whole. In lyrics writing, coherence means using a broader context and establishing semantic relationships between words and sentences to enhance understanding and interpretation [59].

Coherence in lyrics is manifested at four levels [59]: Word level coherence: Words form logical concepts or actions with their surrounding words. Sentence level coherence: Each line supplements, contrasts, or expands on the previous line. Full-text level coherence: The entire text revolve around a single theme. Musicality level coherence: The lyrics match the rhythm, sentiment, and style

Authors’ address: Xichu Ma, ma_xichu@nus.edu.sg; Varun Sharma, sharmavarun.s@u.nus.edu; Min-Yen Kan, kanmy@comp.nus.edu.sg; Wee Sun Lee, leews@comp.nus.edu.sg; Ye Wang, wangye@comp.nus.edu.sg, School of Computing, National University of Singapore, Singapore, Singapore.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2023 ACM.

ACM 0004-5411/2023/9-ART

<https://doi.org/XXXXXXXX.XXXXXXX>

♩ = 72

(a) Human written (Good): Early one day we shall see The sun rising brightly

(b) Generated (Bad): Imagine there's a treehouse Life is easy if you hop

5

(a) Singing - Birds at peace After the dark night - ly

(b) No spiders fixing toast, yeah Above us only barber shops

Fig. 1. Examples of (a) coherent human-composed lyrics (at top) and (b) incoherent automatically generated lyrics (at bottom) using the music from John Lennon’s song Imagine. The human composition (a) exhibit good coherence due to appropriate collocation. Subsequent lines extend the preceding ones, and all sentences revolve around the theme of “natural scenery during a day.” And the contents are harmonious with the music style. In contrast, the generated sample (b) suffer from poor coherence due to inappropriate word combinations, off-topic sentences, and a style incongruous with the music.

of the accompanying music. For example, in Figure 1-a, the example shows high-quality human-composed lyrics with appropriate collocation. Each subsequent sentence extends the previous one, and the entirety of the sentences describes the theme “natural scenery during a day,” creating a peaceful atmosphere that aligns with the music’s sentiment. In contrast, the example (b) of machine-generated lyrics exhibits poor word choices leading to unclear meaning (e.g., “spider fixes toast”), weak sentence connections, a lack of central theme, and a bizarre style that conflicts with the musical style. Additional generated examples can be found in Appendix B.

Current models for lyrics generation typically achieve the proficiency demonstrated by the lower lines in the previous example, where grammar and rhythm are acceptable, but coherence across all four levels remains problematic [45, 65]. Although ChatGPT¹ generates coherent and rhyming lyrics, it has notable limitations: it cannot process music input, meet syllable requirements, or avoid the risk of plagiarism. Customizing lyrics for specific applications, such as language learning, introduces even greater challenges. While research has shown that singing songs with appropriate lyrics can help language learners acquire vocabulary [18, 19, 22, 24, 48, 50, 62, 75], human-composed songs that appeal to learners’ musical tastes [51] often lack the necessary target vocabulary and may be subject to copyright. Existing lyrics generation systems are also inadequate in this context, as they struggle to incorporate user-specified keywords while maintaining four-level coherence.

A recent study propose using a keyword skeleton as prompts for lyrics generators to enhance coherence [70]. A “keyword skeleton” is defined as a structured list of terms, each corresponding sequentially to a sentence or line in the lyrics. Nevertheless, this approach has limitations. It does not thoroughly investigate and clearly define lyric coherence, fails to analyze the root causes of incoherence in generated lyrics, and does not fully leverage the keyword skeleton framework. We identified the main issues as follows: (1) The system employs the YAKE keyword extraction method [70], which generates skeletons based on word frequency. This approach does not effectively capture the narrative structure of lyrics. (2) The system may overlook essential user-input keywords during the skeleton generation process. (3) The system does not account for musical features, relying heavily on the rhythm and syllable templates of original lyrics, thus limiting its generalizability.

¹<https://openai.com/chatgpt/>

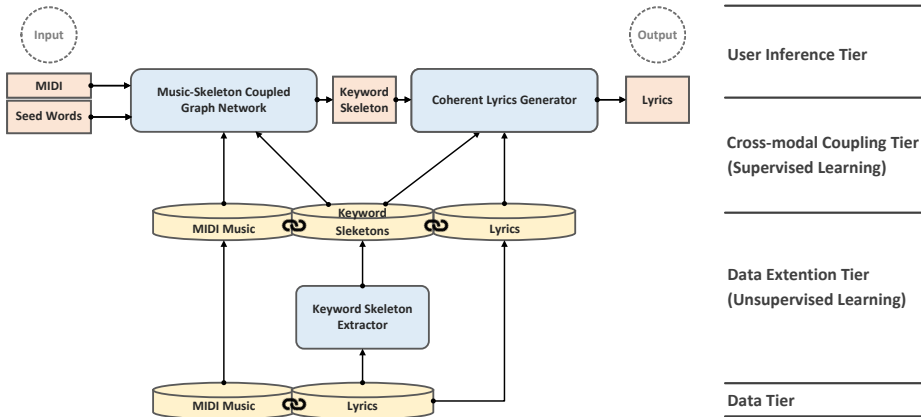


Fig. 2. The KeYric system extracts semantically significant keywords from a song's lyrics, forms a keyword skeleton that align with the music, and uses this skeleton to fit human-composed lyrics in training. During inference, the system takes a MIDI file and seed words as inputs to produce personalized coherent lyrics.

(4) The system utilizes keywords as prompts without incorporating targeted innovations in its language model appropriately to enhance coherence. As a result, the coherence of the generated lyrics remains less satisfying compared against human-composition. It is crucial to integrate a deeper understanding of musical elements and user inputs into the generation process to improve musicality and personalization.

In response to the above challenges, we introduce the KeYric system, which enhances coherence, musical association, and personalization in generated lyrics. As illustrated in Figure 2, the KeYric system generates coherent lyrics by first taking a MIDI file [63] and user-specified seed words as inputs. The Keyword Skeleton Extractor module uses unsupervised learning to extract semantically significant keywords from a lyrics database, forming keyword skeletons. This extends the original {MIDI, lyrics} tuple into a {MIDI, keyword skeleton, lyrics} triplet dataset. The cross-modal tier, which includes a graph network and a lyrics generator, then couples MIDI with skeletons and skeletons with lyrics. The Music-skeleton Coupled Graph Network selects a keyword for each musical phrase, linking them to form a skeleton, and the Coherent Lyrics Generator creates lyrics based on this skeleton. During inference, the system predicts a skeleton from the given MIDI file and seed words, and then generates personalized lyrics suitable for language learning through singing [45].

Our approach differs from traditional keyword extraction methods that rely on word frequency or manual annotation. Instead, we conceptualize our keyword skeletons as an interpretable latent space, capable of compressing the lyric space and sampling it with comprehensible keywords. For instance, the example lyrics (a) in Figure 1 can be summarized into the keyword skeleton ["see", "sun", "birds", "nightly"]. We propose a novel unsupervised method for extracting keyword skeletons from human-composed lyrics, identifying the most representative words through a process of text compression and reconstruction. Furthermore, we leverage deep graph networks to establish pairwise connections between musical phrases and keywords within a skeleton. This graph model predicts a skeleton that includes the user-specified keywords, aligning with the sentiment and style of the music, thereby enhancing musicality coherence. Additionally, we present a coherent lyrics generation model that uses a skeleton and melody as prompts. This model incorporates three levels

of coherence mechanisms, which enhance coherence at the word, sentence, and full-text levels throughout the generation process.

Objective and subjective evaluations demonstrate that the KeYric system achieves a 5% and 19% improvement in lyric quality over the compared models, respectively. Experts, including linguists, songwriters, and language teachers, have verified that KeYric generates lyrics that facilitate language learning through singing. Furthermore, we investigate the impact of genres and musical elements on lyric coherence. The findings reveal that pop and country songs produce the most coherent skeletons and lyrics. Additionally, our analysis indicates that, in comparison to bar boundary, genre-relevant information such as pitch and note duration of music play a more significant role in maintaining lyric coherence.

In summary, this study makes three major contributions:

- We address coherent lyrics generation and propose a solution that improves coherence at word (smoothness between adjacent words), sentence (continuity in describing the same subject in adjacent sentences), full-text levels (the entire lyrics center the same theme), and musicality (lyrics match the style and sentiment of the music).
- We propose KeYric to generate lyrics from a keyword skeleton. This novel unsupervised method extracts keyword skeletons from lyrics, expands input music and seed words for language learning into a skeleton, and incorporates the skeleton in the lyrics generator.
- Our analysis of the experiment results reveals that incorporating genre-relevant musical components (i.e., pitch and note duration) in data encoding substantially enhances the coherence of the generated lyrics.

2 RELATED WORK

2.1 Automatic Lyrics Generation

With the rise of neural network technology, Recurrent Neural Networks (RNNs) [21, 45, 46, 58, 66, 79] and Transformers [7, 36, 41, 54, 56, 65, 81] dominate automatic lyrics generation. These autoregressive models select the next lyric token or line until the desired length is reached. Considering the musical nature and unique requirements of writing lyrics, many studies have focused on using prompts or conditional embeddings to improve the generated lyrics' topical words [76, 85], content [54], rhyme [36, 81], matching of syllables [43, 53, 77], audio features [7, 72–74], and realism [47]. To produce well-balanced lyrics, SongNet [36] and ChipSong [41] both constrain several of the above characteristics. Other studies use adversarial learning to reward results with desirable lyric characteristics from a consequentialist perspective [8, 12, 14, 45]. Researchers find that, to make lyrics singable, syllable patterns must match the melody [42]. Accordingly, some studies attempt to produce lyrics from melody input [34, 65, 77].

Unfortunately, few studies have examined coherence in lyrics generation thoroughly. Although state-of-the-art (SOTA) work attempts to use keyword skeletons as prompts to improve the coherence of lyrics generation [70], it does not define or explain what coherence in lyrics means in detail. Additionally, this system lacks an in-depth discussion on how to extract keyword skeletons and how to use the keyword framework to enhance coherence in the lyrics generator. It also overlooks the coherence between lyrics and music, failing to generate lyrics that match the prosody and style of the input music. These issues result in a significant gap between automatic lyrics generation and human songwriting.

2.2 Coherence in Text Generation

Existing methods for coherent text generation fall into three categories. The first generates text from prerequisite keywords, topics, or sentences. Early approaches used the hidden state of the

previous sentence as context for the next [78], while later work suggested condensing sentences to condition future generations [26]. Other studies expanded phrase-based storyline plans into coherent stories. However, these methods often rely on (1) high-frequency words, leading to homogenized storylines [83], (2) predicate-argument structures, which ignore sentiment and style [20], or (3) human annotations, which lack uniform standards [80]. Improved methods for extracting keyword skeletons from lyrics are needed.

The second branch of methods enhances coherence by selecting the most coherent candidate. One study includes an independent model's coherence judgment score in the generation loss [68]. SeqGAN with coherence and cohesion discriminators evaluates candidates' probabilities of co-occurrence and adjacency with previous text [10]. Phrase-level reward replaces full-sentence reward in the roll-out process to improve efficiency [12]. The GEDI model uses an independent language model to compute priors for candidate words under a given topic code and previous words, approximating the posteriors given by the discriminators [32]. Lin et al. suggest adding a topic transition planner to GEDI for gradual topic transitions [39]. A recent poem generation study used prompt templates requiring candidates to predict the title, previous sentence, or topic, keeping only the winners in the beam search [89]. These "inverse prompts" predict back from the current generation (e.g., "current generation is from a <STYLE> style poem titled <TITLE>"). Since lyrics, as artistic texts, cannot be generalized into a few limited themes like technology, society, or economy, as in the case of GEDI, and song titles often fail to comprehensively summarize the lyrics' content, we believe that using a single coherence mechanism alone is unlikely to improve lyric coherence. Therefore, it is necessary to design a new architecture that includes multiple coherence mechanisms.

The third branch suggests using a keyword skeleton as prompts to improve coherence in long texts [26]. However, applying this to lyrics generation is complex. It requires subjective human annotation of keyword skeletons, with one word per lyric line capturing salient semantic, sentimental, and narrative information. Current automatic plot planning and keyword skeleton extraction techniques often overlook words expressing sentiment and music style [20, 52, 80, 82, 83], making them less suitable for coherent lyrics generation. The lyrics generation in this study uses the keyword skeleton as prompts. Through an integrated model architecture, it combines GEDI, a lyrics generator, and inverse prompt techniques to enhance the coherence of the generated lyrics before, during, and after word selection.

2.3 Unsupervised Keyword Extraction

To create keyword skeletons that enhance lyric coherence and maintain the style and sentiment without human annotation bias, we reviewed existing unsupervised keyword extraction studies.

Unsupervised keyword extraction has evolved significantly. Initially, researchers used statistical, linguistic, machine learning, and graph theory methods to extract keywords from text [55]. With advancements in deep learning and language models, text-embedding models have gained prominence [2, 67]. Techniques like PageRank [13] and TextRank [49] construct document graphs to evaluate vertex importance. Improvements include clustering similar phrases into topics and weighting them by semantic relations [6], PositionRank which considers word position and frequency [23], and multipartite graphs that ensure topical diversity [5]. To address the limitations of graph-based methods, deep learning-based embedding methods like EmbedRank [4] and SIFRank [69] use high-dimensional vectors. UkeRank [38] and AttentionRank [16] improve accuracy with global and local contexts and a hybrid attention model with BERT [15], respectively. Recent research ranks all phrases from a corpus by relevance to new documents [64] and uses autoencoding variational Bayes to build a latent topic tree [86].

However, these methods are inadequate for lyric keyword skeleton extraction: (1) They struggle with poetic and metaphorical language, such as “time is a thief” (without simile indicators “like” or “as”), which requires understanding abstract concepts. (2) They overlook plot and topic transitions, complicating the generation of coherent texts from keywords. (3) Lyrics feature more repetitions and fewer explicit topic markers, making accurate keyword identification challenging. Thus, lyric keyword skeleton extraction necessitates specialized methods.

Imagine – Original	Imagine – Composition 1	Imagine – Composition 2	Imagine – Composition 3
Imagine there's no heaven It's easy if you try No hell below us Above us, only sky Imagine all the people Livin' for today Ah	Early one day we shall see → (that) The sun rising brightly Singing birds at peace → After the dark, nightly Nature is all around us Listen if you can Ah~	Nothing around but what you see No ghosts or spirits high Just that which can be perceived With an unaided eye Live not for some world beyond Just live for today Ah	I found a chest in my basement It opened with a creak To a candy jungle → As far as I could see Imagine all that sugar Ain't great for my teeth
Imagine there's no countries It isn't hard to do Nothing to kill or die for And no religion , too Imagine all the people Livin' life in peace You	Perhaps one day we shall know → (that) Beauty is all around Trees, flowers , and animals → Living above the ground Nature is all around us Breathe , enjoy the air You~	If no borders existed As the world appears from space No rivals, conflicts, chaos, war No fight s for what's called 'race' Live not to crush some enemy But to uplift all You	I wandered there for miles I passed a chocolate grove Sour patches and pop rocks There were all-the treats I know Imagine all that sugar Ain't great for my teeth
You may say I'm a dreamer But I'm not the only one I hope someday you'll join us And the world will be as one	You may not see , or know this But life does surround us all Maybe we'll join together And we'll live without a fall	We know that which is real Yet fight for make-believe Let's stop the endless battle And bring about world peace	You may say I'm a glutton But I'm not just thinkin' treats I hope someday our real world Will be-e just as sweet
[heaven, easy, hell, sky, people, living, country, hard, kill, religion, people, peace, dreamer, only, hope, world] Relevance 0.247	[see, rising, peace, night, nature, listen, perhaps, beauty, flower, living, nature, air, see, life, join, live] Relevance 0.189	[nothing, spirit, perceive, unaided, world, today, border, space, war, fight, enemy, uplift, real, fight, battle, peace] Relevance 0.236	[basement, open, candy, see, sugar, tooth, wander, chocolate, rock, treat, sugar, tooth, glutton, treat, hope, sweet] Relevance 0.185
Average relevance to the song name = 0.203 Random keyword skeleton's relevance to the song name = 0.129			
Text : seed words for learning	Text : Skeleton keyword	→ : Clause	Text : Conjunctions/ Antecedent Reference

Fig. 3. Human composed lyrics given the mandatory seed words **know** and **see** based on John Lennon's song Imagine. Seed words are denoted by green text, skeleton keywords by orange text, clauses by an arrow →, and conjunctions by underscore.

3 KEYWORD SKELETON EXTRACTION AND EXPANSION

3.1 Motivation

We commissioned lyricists to write lyrics suitable for language learning based on given seed words and music. By monitoring their creation process, we observed common procedures among human lyricists writing for linguistic pedagogy. As illustrated in Figure 3, these keywords evolve into lyrical cues and are extended into sentences, considering pivotal terms, rhyme, melodic alignment, and seed words to be learned. An initial assessment of the semantic relevance [61] between these keyword skeletons and song titles showed a 57.4% improvement compared to randomly selected keywords (as 0.203 vs. 0.129 shown in Figure 3). There are typically four methods to achieve textual coherence: repeating key nouns, using pronouns, employing transition signals, and maintaining logical order [57]. Compared to previous automated models, human lyricists use these components more effectively, enhancing coherence between consecutive lyric lines.

However, this process is labor-intensive, requiring over 20 minutes per song lyric to align with the provided music and keywords. In lyrics-based language learning, this effort increases as lyricists tailor compositions to learners' backgrounds (e.g., linguistic proficiency, vocabulary). To improve efficiency, we propose KeYric, which emulates human lyricists' writing processes through deep learning.

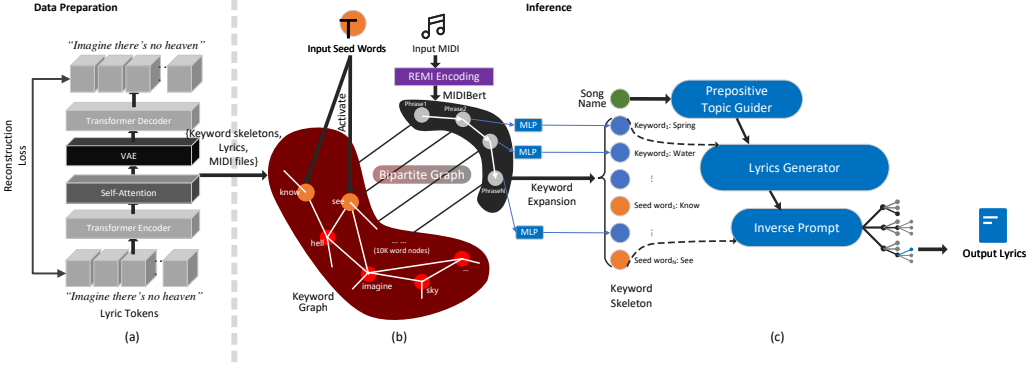


Fig. 4. KeYric architecture. (a) Keyword skeleton extractor creates keyword skeletons from lyrics. The skeletons are then packaged along with the lyrics text and their corresponding MIDI files for model training. Specifically, paired {keyword skeletons, MIDI files} datasets will be utilized to train a cross-modal graph model, i.e. the keyword skeleton expander, that communicates between music and keywords. Paired {keyword skeleton, lyrics} datasets will be employed to train a coherent lyrics generator, with a keyword skeleton serving as heuristic prompts. (b) Keyword skeleton expander trains on pairs to build a keyword skeleton from input MIDI file and seed words. The bipartite graph consists of two sub-graphs, the word sub-graph and music sub-graph. (c) Coherent lyrics generator takes a keyword skeleton, melody and a song name to generate coherent lyrics supporting language learning.

The core idea is to extract salient vocabulary from each lyric sentence using unsupervised learning. We build a keyword skeleton with the words gaining highest attention weights during compression and reconstruction, refining coherent connections between lyric sentences. Then a keyword skeleton expander, trained on extracted keyword skeletons and corresponding MIDI files, predicts a suitable keyword skeleton for unseen input songs. Simultaneously, a lyrics generator, trained on full lyrics using an expanded keyword skeleton as prompts, employs multi-layer coherence mechanisms to select prevalent conjunctions, pronouns, clauses, and cohesive terms. This approach generates coherent lyrics by stringing together the keywords, ultimately achieving overall coherence.

3.2 Keyword Skeleton Extraction

Given a vocabulary set V , the keyword skeleton is defined as a sequence of word tokens $K = \{k_1, k_2, \dots, k_n\}, K \neq \emptyset$, where each element $k_t \in V$ corresponds sequentially to s_t , the t^{th} line in specific lyrics. As a condensed version of the entire text, a keyword skeleton should present coherence akin to a storyline, showcasing the narrative development and central theme. The selected keywords should meet the following criteria: (1) Each keyword represents a lyric sentence and conveys its stylistic information concisely. (2) Keywords should link coherently to nearby keywords. (3) Repeated keywords are allowed to present lyric structure. Thus, the skeleton can serve as a synopsis and developmental framework for the lyrics.

We propose an unsupervised learning model to interpretably select the best keyword from each lyric line. As shown in Figure 4-a, the keyword extractor compresses lyrics into latent space and reconstructs them using a hierarchical Transformer-VAE. The word-to-sentence encoder's attention scores determine the most semantic and stylistic words contributing to the latent variables for each line. These chosen words form the keyword skeleton. The use of VAE improves generalizability and robustness by capturing the underlying semantic structure of lyrics. This accommodates variations and different versions of the same song while maintaining keyword extraction consistency. This probabilistic approach ensures effective handling of diverse lyric representations. A hierarchical

architecture separates sentence and word attention computation, making word token attention values more representative of their contributions to a sentence.

3.2.1 Lyrics Compression. In Figure 5, the green blocks illustrate the compression process. The VAE has a hierarchical Transformer encoder $q_\theta(z|x)$, decoder $p_\phi(x|z)$, and latent variable $z \in \mathbb{R}^{d_z}$ [31]. q and p are parameterized by θ and ϕ respectively.

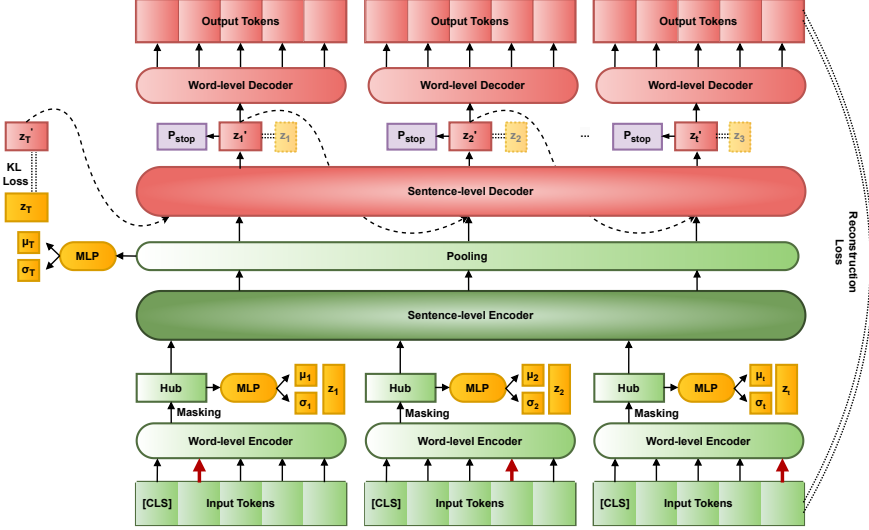


Fig. 5. Keyword skeleton extractor network made of symmetric hierarchical Transformer-VAE encoder and decoder.

The hierarchical Transformer encoder [71] aggregates word tokens $X = \{x_1, x_2, \dots, x_n\}$ into sentence-level representations $Z = \{z_1, z_2, \dots, z_u\}$ stored in “hub” vectors, i.e., the foremost vector. Each sentence is prefixed with a prepositive virtual [CLS] token. A sentence’s aggregated latent vector is a Gaussian sample of its Transformer encoding at the hub vector’s location. We choose an isotropic Gaussian distribution with unit variance, $p(z) = \mathcal{N}(0, I)$, as our prior. This simplifies the latent space structure, ensuring each latent variable contributes equally and independently. It aids efficient learning of diverse lyric representations while maintaining consistency and reducing complexity in the hierarchical Transformer encoder. We incorporate positional, part-of-speech (POS), and dependency embeddings [17] to include word tokens’ syntactic features.

The lyric-level encoder computes cross-sentence information and averages the outputs through a pooling layer to obtain the compressed latent vector of the entire lyrics, $Z_T \in \mathbb{R}^{d_z}$.

$$\begin{aligned} z_t &= M_{Hub}(f(\text{Emb}([\text{CLS}]||s_t) + \text{Emb}^+([\text{CLS}]||s_t))) \\ z_T &= \text{Pool}(F(Z)) \\ \mu_t, \sigma_t &= \text{MLP}(z_t); \quad \mu_T, \sigma_T = \text{MLP}(z_T) \end{aligned} \quad (1)$$

where $\text{Emb}(\cdot)$ is the word embedding layer and $\text{Emb}^+(\cdot)$ is the sum of positional and semantic embeddings. s_t ($t \in [1, v]$) is the t^{th} sentence. $||$ denotes concatenation, $f(\cdot)$ and $F(\cdot)$ are the word-to-sentence and sentence-to-lyrics Transformer encoders, respectively. M_{Hub} is the masking operation that retains only the hub vector.

3.2.2 Lyrics Reconstruction. Unlike previous research [33, 40, 87], our lyric reconstruction uses a symmetric Transformer-based encoder and decoder, as the model compresses and reconstructs lyrics rather than generating them from random latent variables.

We sample the lyric’s latent vector, z'_T , from the approximate posterior $q\theta(z|x) = \mathcal{N}(\mu_z, \sigma_z)$. To match the decoder’s input shape, we expand z'_T into the set $C' = \{c'_1, c'_2, \dots, c'_u\}$, where $c'_i \in \mathbb{R}^{d_c}$. We decode z'_T into sentence-level latent variables $Z' = \{z'_1, z'_2, \dots, z'_u\}$ and project each $z'_i \in \mathbb{R}^{d_z}$ into the word-level decoder’s input shape to regenerate lyric tokens.

A Multi-layer Perceptron (MLP) predicts whether the current sentence is the end of the lyric reconstruction. The MLP assigns each z'_i a probability P_{stop} , indicating if the current sentence should be the last.

$$C' = W_z \cdot z'_T; \quad Z' = G(C'); \quad X' = g(C'); \quad P_{stop} = MLP(Z') \quad (2)$$

where W_z is a linear projection to sequentially expand z'_T ; $G(\cdot)$ and $g(\cdot)$ are the lyric-to-sentence and sentence-to-word decoders.

3.2.3 Loss Design. The loss function of the keyword extractor is formulated as follows:

$$\mathcal{L}_{VAE} = \alpha \mathbb{E}_{q\theta(z|x)} [\log p_\phi(x|z)] + \beta \mathcal{L}_{Stop}(P_{stop}) - \gamma D_{KL} [q\theta(z|x) || p(z)] \quad (3)$$

The loss function has three weighted terms. The first term, Reconstruction Loss, compares generated lyrics to the ground truth. The second term, Sentence Loss on the stopping distribution P_{stop} , encourages the model to select an appropriate length for the generated lyrics [33]. The third term, the Kullback-Leibler Divergence, penalizes deviations of the latent variable distribution from a Gaussian prior with unit variance. We employ the “reparameterization trick” [31] to sample latent variables in a differentiable manner by predicting the mean and variance parameters of the Gaussian distribution.

3.2.4 Keyword Selection. After compressing and reconstructing the lyrics, we examine the accumulative self-attention matrices of all word-level Transformer encoder blocks to identify each line’s keyword. For each lyric line, each token’s attention scores across all layers, $W_i^{Att} \in \mathbb{R}^{n \times n}$, are multiplied along the propagation path to the hub vector’s attention score, $W_{Hub_t}^{Att} \in \mathbb{R}^{n \times 1}$. As illustrated by the red arrows in Figure 5, we select k_t , the token with the highest product, as the keyword for its lyric line since this product indicates the token’s contribution to the sentence encoding. We then concatenate all selected keywords to form the skeleton of the lyrics.

$$k_t = \arg \max \prod_{i=1}^{\psi-1} W_i^{Att} \cdot W_{Hub_t}^{Att} \quad (4)$$

where k_t is the extracted keyword for sentence t and ψ is the layer number of the word-to-sentence encoder.

3.3 Keyword Skeleton Expansion

The keyword skeleton extractor creates matched triplets of {keyword skeletons, lyrics, MIDI files}. It also generates two static matrices showing keyword co-occurrence and adjacency statistics, forming a keyword relationship graph. By adding input music phrases as nodes and connecting them to the keyword graph, the expander uses a graph Transformer [27] to learn the cross-modal relevance of keywords and music (Figure 4-b).

During inference, the expander generates a keyword skeleton as lyric storylines from user-input seed words K_{seed} and input MIDI music m . the expander augments seed words by predicting additional keywords from the music and rearranging them to form a keyword skeleton. Each music phrase node predicts a keyword matching its musical features after graph propagation

and neighbor feature aggregation. The skeleton is the concatenation of the input seed words and predicted keywords from all music nodes.

3.3.1 Graph Building. As shown in Figure 4-b, a bipartite graph connects the keyword textual modality with the symbolic music modality, consisting of two subgraphs: the keyword graph and the music graph.

The keyword graph contains word nodes of the entire vocabulary, identified by their token IDs. These nodes are connected by bidirectional edges representing co-occurrence and adjacency frequencies based on extractor statistics. For instance, the probability of the keywords “seasons” and “spring” appearing together in a skeleton is 0.6, and the probability of “spring” following “seasons” in skeletons is 0.4. Thus, the edge from “seasons” to “spring” in the keyword graph has features [0.6, 0.4].

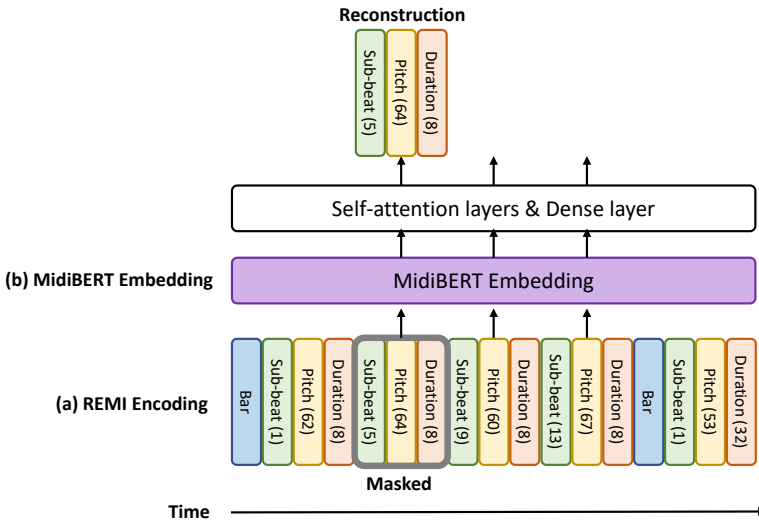


Fig. 6. Illustration of REMI encoding and MidiBERT embedding. (a) The REMI encoding of a MIDI file is a music event representation that converts MIDI scores into discrete tokens with metrical context, aligning with lyric phrase divisions. (b) The MidiBERT embedding, a large-scale pre-trained model for symbolic music understanding, is trained by masked language modeling task which masks and reconstructs the tokens in REMI encodings.

The music graph includes music nodes represented by MidiBERT embeddings [11] of all music phrases’ REMI encodings [28]. First, a MIDI file is split into phrases following [45]. Then, as demonstrated in Figure 6-a, REMI, a music event representation, converts each music phrase’ MIDI score into discrete tokens, providing metrical context for rhythmic patterns and segmenting music encoding into distinct nodes aligned with lyric phrase divisions. Next, MidiBERT, a large-scale pre-trained model for symbolic music understanding, uses masked language modeling (MLM) to learn high-level features by masking and reconstructing input REMI tokens (Figure 6-b), capturing intricate musical patterns, harmonies, and structures.

In the music graph, music nodes are connected by directed edges indicating performance order. Keyword nodes are bidirectionally connected to all music nodes, forming a bipartite graph to model cross-modal relationships (Figure 4-b). That is, in reference initialization, every music phrase is connected to every keyword. Integrating text and music modalities in a graph network, this model

predicts keywords from paired music phrases, generating a keyword skeleton aligned with the musical context to provide a coherent storyline for lyrics generation.

3.3.2 Keyword Skeleton Expansion. The graph Transformer computes hidden states for all nodes, propagating keyword information throughout the graph. Unlike sequential or grid models, a graph network (1) de-emphasizes autoregressive generation, enabling parallel keyword expansion; (2) captures topological long-term keyword dependencies; and (3) unifies music and text as graph nodes for cross-modal relevance. Hidden states for nodes and edges are represented uniformly as \mathbb{R}^{d_g} . After information propagation, an MLP predicts and samples a keyword for each music node, representing the music phrase. This procedure is formulated as:

$$h_i^x = Emb_{word}(x); \quad h_i^m = Emb_{MIDI}(R(m)) \quad (5)$$

$$h_i = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_h h_j\right); \quad e_{i,j} = \sigma\left(\sum_{k \in \mathcal{N}(i)} \alpha_{i,k} W_e e_{i,k}\right) \quad (6)$$

$$P(k_i) = Softmax(MLP(h_i^m)); \quad Loss = CE(P(k_i), \hat{k}_i) \quad (7)$$

where h and e are graph node and edge hidden states, whose superscripts distinguish music and word nodes. $R(\cdot)$ denotes REMI encoding while Emb_{word} and Emb_{MIDI} represent word and MidiBERT Embedding respectively. $\mathcal{N}(i)$ are node i 's incoming neighbors, and W_h and W_e are graph Transformer model's trainable parameters.

The expander is trained using the Cross-Entropy loss between the predicted keyword from the music node $P(k_i)$ and the extracted keyword \hat{k}_i from the lyrics. Thereby, the expander produces a coherent keyword skeleton that fits the music during inference.

3.3.3 Specified Seed Words Insertion. During lyrics generation inference, users typically input several seed words to indicate the words they wish to learn through singing. The seed words provided by users are usually insufficient to form a complete keyword skeleton. Therefore, we employ the keyword expander to predict additional keywords from the input MIDI file and organize both seed words and expanded keywords into a keyword skeleton in a specified order. We utilize melody identification from [44] and musical snippet segmentation techniques from AI-Lyricist [45] to estimate an appropriate sentence number L (equal to the number of phrases in the input music). After predicting keywords for the first number of $l_{exp} = L - l_{seed}$ music nodes, we insert the remaining l_{seed} specified seed words into the keyword skeleton, ensuring the total number of keywords in the skeleton equals the number of musical phrases. We use average co-occurrence and adjacency probabilities to determine seed words' positions within the skeleton. Each seed word is inserted sequentially to maximize these probabilities for the entire keyword skeleton thus improving the storyline's coherence. The expanded keyword skeleton is finalized after all seed words are inserted.

Compared to Plan2Lyrics [70] and AI-lyricist, inserting seed words during skeleton expansion avoids conflicts with surrounding words. Maximizing co-occurrence and adjacency probabilities ensures sentence-level coherence. The graph model also establishes cross-modal coherence between music and lyrics.

4 COHERENT LYRICS GENERATION

After training, the keyword skeleton expander can produce a keyword skeleton from unseen input MIDI music. Thus, the lyrics generation module takes any MIDI music and a keyword skeleton produced from the MIDI as input to generate coherent lyrics. We propose a three-layer mechanism to ensure coherence in its generation, utilizing three stacked GPT-2-based submodules: prepositive topic guider, main-body lyrics generator, and inverse prompts. These submodules enforce coherence

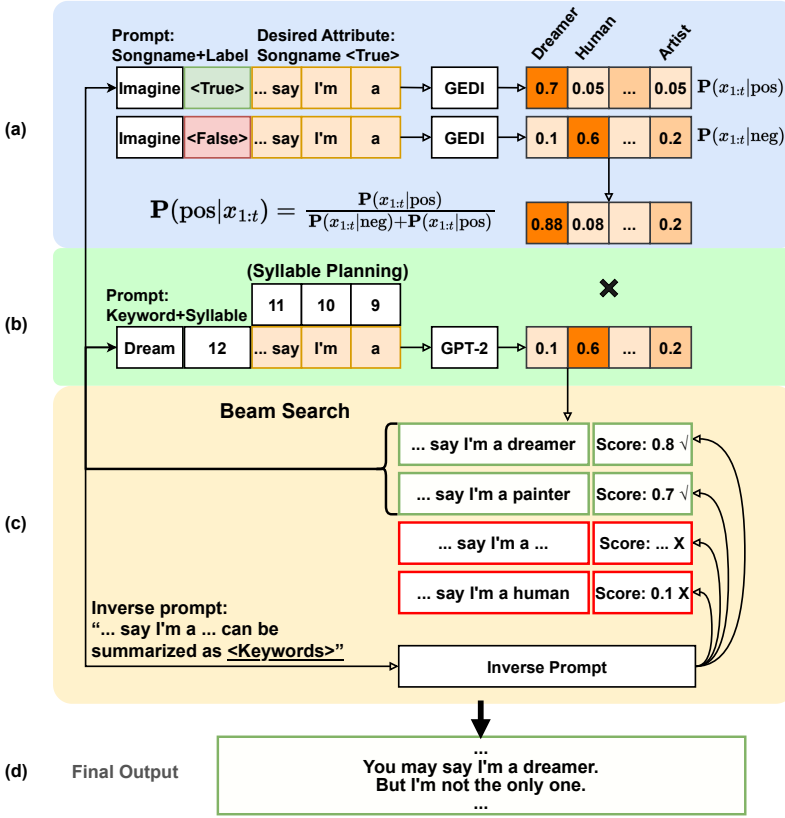


Fig. 7. Coherent lyrics generator’s network architecture. (a) The prepositive topic guider. (b) The main-body lyrics generator. (c) Post-beam search driven by inverse prompts. (d) Final output of generated lyrics.

before, during, and after lyric probability computation. (1) The expanded keyword skeleton prompts the main-body lyrics generator. (2) The prepositive topic guider uses the song name and previously generated words to constrain the next word selection. (3) Beam search with inverse prompts evaluates lyric candidates based on their alignment with the keyword skeleton.

GPT-2 is chosen as the foundational model for all three sub-modules due to its power, reproducibility, interpretability, and computational affordability. While more advanced models might perform better, GPT-2 enables us to explore coherence-enhancing factors and techniques within generally acceptable resource constraints. A textual lyric dataset pre-trains these models for poetic lyric adaptation, followed by fine-tuning for specific tasks. The three-layer mechanisms work together to generate fluent, coherent, and musically relevant lyrics.

4.1 Main-body Lyrics Generator

The main-body lyrics generator produces subsequent tokens autoregressively. It is fine-tuned to generate lyrics based on a specified number of syllable and a keyword in the skeleton as prompts after pre-training on a lyric dataset. As illustrated in Figure 7-b, the fix-length keyword and syllable prompts precede each lyric sentence. A syllable planning $SL = \{sl_i^1, sl_i^2, \dots, sl_i^T\}$, which is a list of predicted remaining syllable counts, is added to each token’s word embedding to indicate the remaining syllables in the current sentence. Training with keyword and lyric association

improves word-level coherence by familiarizing the generator with specific keywords. Using the keyword k_i and remaining syllable number sl_i^t as prompts, the generator maximizes $P_w(x_t = k_i | x_{<t}, sl_i^t) \times P_w(x_{>t} | x_{<t}, sl_i^{t+1}, x_t = k_i)$ to generate probabilities for subsequent token. For ease of computation, this process is approximated mathematically as

$$P_w(x_t | x_{<t}, sl_i^t, k_i) \quad (8)$$

4.2 Prepositive Topic Guider

Usually, to generate lyrics that match desired attributes, discriminators $P_d(\cdot)$ typically measure how well the generated lyrics align with a given attribute. The entire generation and discrimination process is formulated as follows:

$$P_y(x_t | x_{<t}, y) \propto P_w(x_t | x < t) P_d(y | x_t, x < t) \quad (9)$$

However, the subjective and multifaceted nature of song lyrics makes them difficult to describe. Instead of attribute classes used in [32], we propose using the song name to guide generation, as song names often summarize themes, sentiments, and content.

We enhance the main-body generator with a prepositive topic guider based on GEDI [32]. GEDI is called a “prepositive” topic guider because it influences the token selection process with the song name and previous generated lyrics before the main-body generator makes its final prediction, ensuring alignment with the desired song name from the outset. This guider computes the probability that each candidate token x_t matches the desired features in the song name prompt y (i.e., $P_d(y | x_t, x_{<t})$), replacing the ineffective roll-out and reward processes of conventional discriminators. As shown in Figure 7-a, the topic guider computes the probability of each candidate token given the song name prompt, the corresponding anti-prompt (“<SONGNAME> <FALSE>”), and previous tokens at each step. This probability is multiplied to constrain token selection alongside the main-body generator’s prediction.

$$P_d(y_{pos} | x_{1:t}) = \frac{P_y(x_{1:t} | y_{pos})}{P_y(x_{1:t} | y_{neg}) + P_y(x_{1:t} | y_{pos})} \quad (10)$$

GEDI can enhance sentence-level coherence in lyrics generation. Its optimization objective ensures the current sentence is judged as a continuation of the same topic as the previous sentence, thus increasing the probability of selecting more relevant candidate words.

4.3 Inverse Prompts

Generating long texts often deviates from the prompt and includes irrelevant content. To address this, we use the inverse prompt mechanism [89], a beam scoring function that evaluates the log likelihood in reverse. Traditional beam search calculates beam scores using the log likelihood of generating lyrics from prompts: $BeamScore(X|K) = \log P_w(X|K)$. In contrast, the inverse prompt assumes that if the prompts can be generated back from the lyrics, they must be closely related, formulated as $BeamScore_{IP}(X|K) = \log P_w(K|X)$. Traditional prompting strategy is “K results in X”, whereas inverse prompt is “X inferred from K”.

However, reversing the order of prompts and lyrics can produce unnatural texts [89]. A more natural inverse prompt predicts the original prompts from the generated text. Here, the inverse prompt summarizes generated lines X' back into a keyword skeleton K' , and beams are rated by: $BeamScore_{IP}(X|K) = \log P_w(K'|X')$.

An example is shown in Figure 7-c. Given the keyword “Dream” and previously generated text “... *say I’m a*” in the search beams, the inverse prompt is constructed as “... *say I’m a* {*dreamer / painter / human*} *can be summarized as* <KEYWORD>”. A GPT-2 model optimized for inverse

prompt predicts the <KEYWORD> for each beam and scores them based on how closely it matches “Dream”. In the example, beams ending in “dreamer” and “painter” receive higher scores and remain in the search while other results are eliminated. To ensure the inclusion of seed words, the proposed scorer will return 0 if a candidate beam does not contain the specified seed word for learning.

The skeleton extracted by the KeYric system is a compressed representation of the lyrics in a latent space, serving as the song’s theme. The essence of Inverse Prompt is to have the model generate lines during beam search that can be summarized as the core idea, ensuring the lyrics maintain a consistent theme and enhancing full-text coherence.

5 OBJECTIVE EXPERIMENT

5.1 Dataset

We used the Netease API to extract English lyrics with the 100 most frequent tags from a lyric dataset [84], creating the *Netease-lyrics* dataset of 160,171 {*songname, lyric*} pairs for training the keyword skeleton extraction model. For training the keyword skeleton expansion model, we built the *LMD-lyrics* dataset from the Lakh MIDI dataset [60], which contains 7,211 {*songname, lyric, MIDI file*} triplets. All lyrics are segmented by lines. Both datasets are split into 8:1:1 train, validation, and test subsets.

5.2 Configurations

The keyword skeleton extractor employs a standard encoder-decoder block [71] with hidden states size $z \in Z$ set to 256 ($d_z = d_c = 256$). Preliminary experiments determined $\alpha = 1.0$, $\beta = 4.0$, and $\gamma = 0.2$ [33]. The keyword skeleton expander’s graph network has an embedding size of 256 ($d_g=256$), 7 propagation layers to accommodate an average of two verses and two choruses, and includes the 10,000 most frequent words. Graph propagation uses sub-graphs of size 1024 in batches. We pretrained three GPT-2 models (prepositive topic guider, main-body generator, and inverse prompt scorer) on the Netease-lyrics dataset with a masked language model (MLM) task [15] and used the LMD-lyrics dataset with respective prompt templates.

5.3 Compared Methods

We compared our keyword skeleton extraction (Proposed-K) and keyword expansion (Proposed-G, with G representing “graph”) models against various unsupervised keyword extraction techniques, including graph-based algorithms (TextRank, TopicRank, MultipartiteRank, PositionRank), embedding-based algorithms (EmbedRank, SIFRank), and attention-based algorithms (AttentionRank, UkeRank) [38].

We also compared our KeYric system (Proposed) with the SOTA lyrics generation model [70], referred to as Plan2Lyrics in this paper, and with AI-Lyricist [45], based on SeqGAN, and SongMASS [65], which uses a Transformer to generate lyrics from a melody line.

An ablation study assessed the impact of each coherence mechanism in our lyrics generator. We evaluated three versions: (1) a vanilla GPT-2 generator with the keyword skeleton as prompts (Proposed-Lite, a simplified generator without coherence mechanisms), (2) a generator with a prepositive topic guider (Proposed-Pre), and (3) a generator with only inverse prompts (Proposed-IP). This allowed us to determine each mechanism’s contribution to enhancing lyrics’ coherence and musicality.

5.4 Objective Measures

We objectively evaluated the keyword skeletons and the lyrics generator’s applicability to language learning.

The keyword extractor and expander were evaluated on five metrics. The first metric, “**representativeness**” [70], assesses how well the keyword skeleton represents the semantic content and linguistic characteristics of the original lyrics [20]. This is measured by the average cosine similarity between each lyric sentence and its keyword embedding, indicating their interchangeability. The second metric, “**coherence**”, evaluates the skeleton’s topic transitions [29]. It is the average log probability of keyword graph edges, reflecting the frequency of consecutive keyword pairs in adjacent lines and thus measuring the coherence of the storyline. It is formulated as:

$$C_K = \frac{1}{|E|} \sum_{e_{i,j} \in E} \log P(e_{i,j}) \quad (11)$$

where each $e \in |E|$ denotes the edge in the keyword graph defined in subsection 3.3.1 that connects two subsequent words in the keyword skeleton K and $P(e_{i,j})$ represents the probability associated with edge $e_{i,j}$ in the keyword graph, indicating the frequency of subsequent occurrence of the keywords k_i and k_j .

We propose the third metric, “**uniformity**”, which evaluates the distribution of keywords, aiming for one keyword per lyric sentence. It is the ratio of lyric sentences without keywords, multiplied by the number of keywords in the skeleton as a balancing coefficient. This ensures each sentence contributes a keyword, supporting the storyline cohesively without missing key points. High uniformity suggests concentrated keywords, potentially losing content from other lines. The fourth metric, “**cross-modal relevance**”, measures the correlation between text and musical features, computed as the normalized dot product of their feature vectors [45]. The fifth metric, “**diversity**”, assesses the word choice diversity in the lyrics dataset [20, 83]. It is the average pairwise difference between two keyword skeletons, formulated as:

$$D = \frac{1}{|S|(|S| - 1)} \sum_{i=1}^{|S|} \sum_{j=i+1}^{|S|} |(S_i \cup S_j) - (S_i \cap S_j)| \quad (12)$$

where S is the keyword skeleton of the lyrics in the test set and $||$ denotes the size of a set. Diversity is beneficial, but excessive diversity can result in random keywords.

Following previous studies on coherent text generation, we evaluate lyrics generators using two metrics: local [25, 35] and global coherence scores [26]. Local coherence is measured by topic switching detection, calculating the probability that two consecutive sentences share the same topic [3]. Global coherence is evaluated by a model predicting the document’s overall coherence through supervised regression [1]. Additionally, we performed part-of-speech (POS) tagging on the generated lyrics and computed the proportion of elements that significantly contribute to coherence, including conjunctions, subordinate clause indicators, and pronouns.

5.5 Objective Experiment Results

The keyword skeleton evaluation results are shown in Table 1-a. Our proposed model outperforms others in all five metrics. Notably, our one keyword per line strategy avoids distribution bias and enhances episodic coherence. It improves uniformity by 26% compared to the second-best method. The VAE captures essential keywords, increasing diversity by 15% and cross-modal relevance by 14%. Our keyword expander improves coherence and cross-modal relevance by 20% and 14%, respectively, demonstrating the effectiveness of deep graph networks in correlating music and lyrics.

In contrast, AttentionRank employs an attention mechanism but lacks a lyric reconstruction process, leading to the selection of articles and auxiliary words that misrepresent lyrics, lowering its representativeness score. TopicRank and TextRank build text graphs and select keywords without

Table 1. Results of the objective experiments. (a) Objective experiment results of keyword skeleton level evaluation. (b) Objective experiment results of lyrics evaluation. The bold red values represent the best performing values, while the underlined values represent second best performing. We omit the values of the keyword expander (Proposed-G) on representativeness, uniformity and faithfulness as the expander is independent of the original lyrics. **REP**: representativeness, **CO**: coherence, **UNI**: uniformity, **CMR**: cross-modal relevance, **DIV**:diversity, **PCT**: proportion of coherent terms.

(a) Keyword Skeleton Generation						(b) Lyrics Generation			
Model	REP↑	CO↓	UNI↓	CMR↑	DIV↑	Model	Local Coherence↑	Global Coherence↑	PCT↑
SIFRank	0.5	6.69	5.30	0.52	6.2	Original	<u>0.88</u>	<u>1.66</u>	0.28
MultiPartiteRank	0.56	6.73	4.11	0.55	13.59	AI-Lyricist	0.83	1.52	0.16
TopicRank	0.55	6.7	4.06	<u>0.56</u>	12.48	SongMASS	0.85	1.54	0.19
AttentionRank	0.42	6.66	5.91	0.39	5.27	Plan2Lyrics	0.85	1.59	0.25
TextRank	0.54	6.7	4.2	0.48	11.19	Proposed	0.881	1.69	<u>0.27</u>
PositionRank	0.45	6.5	6.7	0.45	3.86	Proposed-Lite	0.854	1.57	0.22
EmbedRank	<u>0.57</u>	6.48	<u>2.75</u>	0.51	13.46	Proposed-Pre	0.86	1.65	0.24
UKERank	0.44	6.43	3.98	0.42	4.62	Proposed-IP	<u>0.876</u>	1.61	0.24
Plan2Lyrics	0.29	<u>6.42</u>	2.91	0.33	4.33				
Proposed-K	0.6	6.33	2.04	0.64	15.68				
Proposed-G	-	5.13	-	0.64	<u>13.82</u>				

considering sentence order, resulting in weaker coherence. PositionRank, relying on keyword frequency and previous occurrences, produces undiversified keyword skeletons. EmbedRank, which uses embeddings for keyword extraction, ranks second in competitiveness by selecting heavily modified words, like nouns surrounded by many adjectives, creating information-dense keywords. However, ignoring sentiment and style modifiers weakens EmbedRank’s cross-modal relevance with music.

The evaluation results for the coherent lyrics generated by our proposed model are presented in Table 1-b. Our model shows a 5% improvement over the SOTA Plan2Lyrics, demonstrating that our compression-reconstruction skeleton extraction method produces a more effective latent space. It also surpasses AI-Lyricist by 9% in overall coherence, validating the effectiveness of our three-layer mechanisms. Additionally, our model outperforms SongMASS by 7%, indicating that incorporating human knowledge, such as syllable templates and keyword skeleton input, is more effective than relying solely on automatic cross-modal relevance capture.

Lyrics’ coherence improvement is calculated by averaging the percentage improvements for each metric in Table 1-b. For example, the improvement in local coherence compared to AI-Lyricist is $(0.88 - 0.83) / 0.825 = 0.06$, and the improvement in global coherence is $(1.69 - 1.52) / 1.52 = 0.112$. The overall improvement is then $(0.06 + 0.112) / 2 = 0.09$ (9%). The +5% and +7% improvement over Plan2Lyrics and SongMASS is calculated similarly.

Although Plan2Lyrics increases the use of conjunctions and referential words in generated lyrics, the skeleton quality largely determines coherence improvement. Plan2Lyrics employs the YAKE method, which relies on text word frequency for keyword extraction. While these skeleton keywords show high coherence, they lack diversity and fail to adequately represent the original lyrics, resulting in an ineffective compression space. Additionally, the absence of musical input leads to significant deviations in musicality.

Our objective experiments revealed a strong positive correlation between the coherence of lyrics and the proportion of coherent elements they contain (Table 1-b). This suggests that using these elements more extensively in lyrics generation enhances coherence.

6 SUBJECTIVE EVALUATION

6.1 Experiment Participant and Procedures

We recruited 32 participants from the university via email, requiring English as their first language. After completing the experiment and passing a manipulation check, participants received S\$30. The participants included 3 professional lyricists and 1 language teacher.

Participants underwent training before the main experiments. They were shown a sample keyword skeleton and lyric paired with music and then asked to rate keyword skeletons and lyrics. They also reviewed rating standards with examples for marks 1-5. The main experiment had two sections: in section 1, participants rated 100 keyword skeletons paired with music in random order; in section 2, they rated 40 lyrics paired with music in random order.

To maximize validity, we used several approaches: (1) a within-subjects experiment with randomized display order; (2) online training for participants with detailed example ratings and reasons to ensure consistency. For singability, we provided a clear rating question with examples for scores 1-5 to avoid ambiguity: *“Please listen to the synthesized singing of the lyrics and rate the following aspects of the lyrics on a scale of 1 to 5, where 1 represents the lowest rating and 5 represents the highest rating. Singability: How well do the syllables of the generated lyrics align with the melody notes of the input music? It is a 5 score if all syllables and music notes are perfectly matched so that you can sing the lyrics naturally, without syllables needing elongations or compressions into more/less music notes, and without a word’s syllables separated by a downbeat. You should subtract 1 point for every mismatch that you feel.”* (3) We balanced participant recruitment to include a variety of other spoken languages to mitigate linguistic biases, ensuring all participants’ first language was English. (4) We used manipulation checks to verify genuine engagement.

6.2 Subjective Metrics

We conducted user rating surveys to evaluate the quality of extracted keyword skeletons, expanded keyword skeletons, and generated lyrics, focusing on key aspects. We assessed keyword skeletons based on coherence, faithfulness, musicality, and sentiment. Coherence examines storyline progression [9], faithfulness assesses accurate summarization of the original lyrics, musicality checks the match with the music style [45], and sentiment evaluates emotional expression. We refined the previously defined four levels of coherence in lyrics into six evaluation criteria for subjective experiments: fluency, local coherence, global coherence, learnability, singability, and musicality. Fluency [70] ensures natural English [88], local coherence ensures smooth sentence transitions, global coherence maintains a consistent theme [88], and learnability integrates seed words seamlessly. Participants judged the presence and contextual relevance of user-specified words in the lyrics. Singability ensures syllables align with melody notes [30, 37], and musicality evaluates cross-modal relevance with the genre, sentiment, and style of the paired music [45].

6.3 Subjective Experiment Results

As shown in Table 2-a, our model’s keyword skeletons assist users in envisioning storylines aligned with the music’s style and sentiment. Participants noted that our keyword extractor summarizes lyrics more accurately than compared models. Overall, our keyword extractor and expander outperform compared methods by 15% and 8%, respectively, based on the average improvement over the second-best model in each metric.

As shown in Table 2-b, Our lyrics generator with a three-layer coherence mechanism surpasses competitors in text quality, local and global coherence, and cross-modal relevance with the expanded keyword skeleton. The KeYric system improves coherent lyrics generation by 19%, based on the

Table 2. Results of the subjective experiments. (a) Subjective experiment results of keyword skeleton level evaluation. (b) Subjective experiment results of lyrics evaluation. The bold red values represent the best performing values, while the underlined values represent second best performing. **CO**: coherence, **FA**: faithfulness, **MUS**: musicality, **SEN**: sentiment, **FLU**: fluency, **LRN**: learnability, **SIN**:singability

(a) Keyword Skeleton Generation					(b) Lyrics Generation						
Model	CO \uparrow	FA \uparrow	MUS \uparrow	SEN \uparrow	Model	FLU \uparrow	Local CO \uparrow	Global CO \uparrow	LRN \uparrow	SIN \uparrow	MUS \uparrow
SIFRank	3.24	2.91	2.88	2.72	Original	3.99	3.90	3.84	1.40	3.45	3.24
MultiPartiteRank	3.16	2.93	2.91	3.22	AI-Lyricist	1.79	1.66	1.51	2.33	2.27	1.83
TopicRank	3.22	2.93	2.90	1.60	SongMASS	2.42	2.16	2.02	1.03	2.45	2.07
AttentionRank	1.75	1.60	1.73	2.90	Plan2Lyrics	2.92	3.00	2.79	0.38	2.63	2.04
TextRank	3.25	<u>3.09</u>	2.90	2.51	Proposed	3.69	3.23	3.07	Δ 2.98	3.08	2.61
PositionRank	2.83	2.68	2.62	2.48							
EmbedRank	3.38	3.18	3.04	2.94							
UKERank	2.48	2.33	2.35	1.55							
Plan2Lyrics	3.38	2.79	2.25	2.29							
Proposed-K	<u>3.59</u>	3.69	3.59	3.69							
Proposed-G	3.60	-	3.55	3.23							

average improvement over the second-best model in each metric. It excels particularly in coherence at the word, sentence, and whole-piece levels, validating our design motivation.

Subjective experiments show that lyrics generated with a skeleton are perceived as more coherent and fluent by singers. Compared to the Plan2Lyrics model, our method improves coherence by 7.6% in local coherence and 10% in global coherence, highlighting the importance of coherence constraints in lyrics generation. Additionally, our model outperforms Plan2Lyrics in musicality by 17%, demonstrating the need for cross-modal associations between music and text.

One important function of the KeYric system is to help generate personalized lyrics for language learning through singing [45, 51]. In this method, users enhance their understanding and memory of words by singing songs that include the keywords they wish to learn. Our lyrics generator enhances personalized language learning by integrating input seed words naturally, showing a 28% improvement in learnability compared to AI-lyricist. Specifically, the seed words that users want to learn are successfully incorporated into the generated lyrics. And the surrounding words and sentences help users understand the meanings of these seed words. This integration aids vocabulary comprehension and language acquisition effectively. In summary, our generated lyrics are engaging, coherent, pleasant, and artistic, making them ideal for language learning.

7 MULTIFACETED ANALYSIS: THE IMPACTS OF MUSICAL FACTORS ON LYRIC COHERENCE

To analyze the effects of musical factors on lyric coherence, we divided the LMD-lyrics dataset into 17 genres and independently trained and evaluated their expanded keyword skeletons. This approach helped us understand how music genres influence lyric coherence. We extracted songs from the database with identified genre attributes, covering 17 genres: bluegrass (0.66%), blues (4.76%), Christian-gospel (1.38%), classical (1.16%), country (8.46%), dance-electric (5.48%), disco (0.32%), folk (2.08%), hip-hop (3.28%), jazz (3.54%), metal (19.92%), new age (1.98%), pop (11.96%), punk (6.84%), reggae (0.6%), R&B (0.96%), and rock (26.62%). Table 3-a shows the coherence rankings, revealing that pop and country music produce the most coherent results. The coherence and narrativity of human songwriting may be the primary reason. In contrast, classical music lacks lyrics, and gospel songs use chanting and exclamations, leading to less coherent keyword expansion and lyrics generation.

Our case studies show that classical songs' lyrics are often incomplete, and punk and folk songs lack a fixed arrangement. Large language models (LLMs) like ChatGPT struggle to detect patterns

Table 3. Multifaceted analysis of keyword and lyric coherence on datasets divided by (a) music genres. (b) encoded elements. The bold red values represent the best performing values, while the bold blue values indicate the least ideal performing values. The underlined values mean the second best performing value.

(a) Results of Subdatasets of Different Music Genres.				
Genres	Keyword Coherence \uparrow	Local Lyrics Coherence \uparrow	Global Lyrics Coherence (Global) \uparrow	Normalized Avarage \uparrow
Pop (11.96%)	5.48	0.91	1.67	0.83
Bluegrass (0.66%)	5.01	0.92	1.64	0.75
Newage (1.98%)	5.08	0.91	1.61	0.70
Jazz (3.54%)	4.95	0.91	1.63	0.69
Reggae (0.6%)	4.66	0.91	1.66	0.69
Country (8.46%)	5.14	0.93	1.62	0.68
Hip-hop (3.28%)	5.05	0.91	1.62	0.67
Disco (0.32%)	4.68	0.92	1.58	0.59
Rock (26.62%)	4.55	0.90	1.63	0.58
R&B (0.96%)	5.17	0.89	1.58	0.54
Metal (19.92%)	4.6	0.89	1.58	0.53
Blues (4.76%)	5.08	0.90	1.55	0.52
Dance-electric (5.48%)	4.06	0.91	1.61	0.50
Christ-Gospel (1.38%)	3.36	0.85	1.50	0.45
Punk (6.84%)	4.79	0.90	1.50	0.37
Folk (2.08%)	4.47	0.90	1.50	0.31
Classical (1.16%)	4.33	0.88	1.49	0.15

(b) Results of Subdatasets of Different REMI Encoded Elements.			
REMI Element	Keyword Coherence \uparrow	Local Lyrics Coherence \uparrow	Global Lyrics Coherence (Global) \uparrow
Full	5.13	0.881	1.69
w/o Bar Line	5.23	0.80	1.61
w/o Position	5.10	<u>0.83</u>	1.66
w/o Pitch	4.44	0.86	1.57
w/o Duration	<u>4.94</u>	0.89	<u>1.61</u>

in these genres. Thus, we trained our generation model on genres with average coherence scores of 0.53 and above (e.g., metal). The trained model also performs well on unseen genres.

We further investigated which musical elements influence cross-modal lyric coherence. By excluding elements in REMI melody representation, we observed changes in coherence. As shown in Table 3-b, removing pitches significantly decreases keyword and whole-piece coherence by 13% and 7%. Surprisingly, removing bar line information increases keyword coherence by 2% but reduces line-to-line coherence by 9%, as lyric lines do not always match bar lines. These findings help us select musical elements to establish cross-modal relevance between text and music in future work.

7.1 Ablation Study and Case Study

The analysis of lyrics from our model and its variants (Proposed-Lite, Proposed-Pre, Proposed-IP) shows that Proposed-Pre produces more sentimentally and thematically coherent lyrics than Proposed-Lite (+2.9%). Most sentences generated by Proposed-Pre maintain a consistent tone and focus on a shared topic, influenced by the prepositive topic guider.

Lyrics from Proposed-IP exhibit tone and subject shifts but use more consecutive and supplementary words to link these shifts. This variant also shows an increase in longer compound and complex sentences with attributive and adverbial clauses. The Proposed model integrates features

of both Proposed-Pre and Proposed-IP, achieving smooth topic transitions between paragraphs and coherence within paragraphs.

8 DISCUSSION AND FUTURE WORK

Despite the success of LLMs like ChatGPT in natural language processing, our findings suggest that their proficiency in generating lyrics for language learning is inferior to our proposed system, KeYric. As shown in Appendix A, ChatGPT struggles with generating original lyrics without mimicking existing structures and often fails to meet specific syllabic requirements. In contrast, KeYric adapts to new musical inputs and consistently produces accurate, flexible results. To fully exploit the potential of LLMs in lyrics generation, future research could enhance LLMs' abilities to understand lyric prosody, incorporate explicit mandatory control instructions, and intentionally avoid plagiarism. Developing models that can process various musical inputs and integrating more comprehensive paired datasets could improve the applicability of LLMs in music-related tasks.

Our research focuses on the English language, but extending our model to other languages is a promising future direction. While this extension is beyond the current paper's scope, it could significantly enhance our system's applicability and impact. Future research should explore adapting the model for multilingual lyrics generation by addressing language-specific nuances and integrating lyric datasets in diverse languages.

9 CONCLUSION

This paper examines lyric coherence at word, sentence, full-text, and cross-modal (musicality) levels. We address issues in existing keyword extraction methods for improving lyric coherence. We propose unsupervised keyword extraction from lyrics and keyword expansion from music. Additionally, we suggest using multiple coherence mechanisms with a keyword skeleton to enhance coherence before, during, and after lyric token prediction. Users believe that the KeYric system outperforms SOTA model by 19%. Finally, we offer insights into how genres and musical elements influence coherent lyrics generation.

ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for their valuable suggestions. This project was funded by research grant A0008150-00-00 from the Ministry of Education, Singapore.

REFERENCES

- [1] Tushar Abhishek, Daksh Rawat, Manish Gupta, and Vasudeva Varma. 2021. Transformer models for text coherence assessment. *arXiv preprint arXiv:2109.02176* (2021).
- [2] Lahbib Ajallouda, Fatima Zahra Fagroud, Ahmed Zellou, et al. 2023. Automatic keyphrases extraction: an overview of deep learning approaches. *Bulletin of Electrical Engineering and Informatics* 12, 1 (2023), 303–313.
- [3] Dennis Aumiller, Satya Almasian, Sebastian Lackner, and Michael Gertz. 2021. Structural text segmentation of legal documents. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*. 2–11.
- [4] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470* (2018).
- [5] Florian Boudin. 2018. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721* (2018).
- [6] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*. 543–551.
- [7] Jia-Wei Chang, Jason C Hung, and Kuan-Cheng Lin. 2021. Singability-enhanced lyric generator with music style transfer. *Computer Communications* 168 (2021), 33–53.
- [8] Yihao Chen and Alexander Lerch. 2020. Melody-conditioned lyrics generation with seqgans. In *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 189–196.

- [9] Yun-Nung Chen, Yu Huang, Hung-Yi Lee, and Lin-Shan Lee. 2012. Unsupervised two-stage keyword extraction from spoken documents by topic coherence and support vector machine. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5041–5044.
- [10] Woon Sang Cho, Pengchuan Zhang, Yizhe Zhang, Xiujun Li, Michel Galley, Chris Brockett, Mengdi Wang, and Jianfeng Gao. 2018. Towards coherent and cohesive long-form text generation. *arXiv preprint arXiv:1811.00511* (2018).
- [11] Yi-Hui Chou, I Chen, Chin-Jui Chang, Joann Ching, Yi-Hsuan Yang, et al. 2021. MidiBERT-piano: Large-scale pre-training for symbolic music understanding. *arXiv preprint arXiv:2107.05223* (2021).
- [12] Yun-Yen Chuang, Hung-Min Hsu, Ray-I Chang, and Hung-Yi Lee. 2022. Adversarial Rap Lyric Generation. In *2022 4th International Conference on Natural Language Processing (ICNLP)*. IEEE, 414–419.
- [13] Fan Chung. 2014. A Brief Survey of PageRank Algorithms. *IEEE Trans. Netw. Sci. Eng.* 1, 1 (2014), 38–42.
- [14] Aidan Cookson, Auguste Hirth, and Krish Kabra. 2020. SloGAN: Character Level Adversarial Lyric Generation. (2020).
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
- [16] Haoran Ding and Xiao Luo. 2021. AttentionRank: unsupervised keyphrase extraction using self and cross attentions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 1919–1928.
- [17] Sufeng Duan, Hai Zhao, Junru Zhou, and Rui Wang. 2019. Syntax-aware transformer encoder for neural machine translation. In *2019 International Conference on Asian Language Processing (IALP)*. IEEE, 396–401.
- [18] Dwayne Engh. 2013. Why use music in English language learning? A survey of the literature. *English Language Teaching* 6, 2 (2013), 113–127.
- [19] Judith Weaver Failoni. 1993. Music as Means To Enhance Cultural Awareness and Literacy in the Foreign Language Classroom. *Mid-Atlantic Journal of Foreign Language Pedagogy* 1 (1993), 97–108.
- [20] Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. *arXiv preprint arXiv:1902.01109* (2019).
- [21] Haoshen Fan, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A hierarchical attention based Seq2Seq model for Chinese lyrics generation. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 279–288.
- [22] Douglas Fisher. 2001. Early language learning with and without music. *Reading Horizons: A Journal of Literacy and Language Arts* 42, 1 (2001), 8.
- [23] Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1105–1115.
- [24] Arla J Good, Frank A Russo, and Jennifer Sullivan. 2015. The efficacy of singing in foreign-language learning. *Psychology of Music* 43, 5 (2015), 627–640.
- [25] Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. 2021. Long text generation by modeling sentence-level and discourse-level coherence. *arXiv preprint arXiv:2105.08963* (2021).
- [26] Zhe Hu, Hou Pong Chan, Jiachen Liu, Xinyan Xiao, Hua Wu, and Lifu Huang. 2022. Planet: Dynamic content planning in autoregressive transformers for long-form text generation. *arXiv preprint arXiv:2203.09100* (2022).
- [27] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*. 2704–2710.
- [28] Yu-Siang Huang and Yi-Hsuan Yang. 2020. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*. 1180–1188.
- [29] Haixin Jiang, Rui Zhou, Limeng Zhang, Hua Wang, and Yanchun Zhang. 2019. Sentence level topic models for associated topics extraction. *World Wide Web* 22 (2019), 2545–2560.
- [30] Haven Kim, Kento Watanabe, Masataka Goto, and Juhan Nam. 2023. A Computational Evaluation Framework for Singable Lyric Translation. *arXiv preprint arXiv:2308.13715* (2023).
- [31] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
- [32] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367* (2020).
- [33] Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. 2017. A hierarchical approach for generating descriptive image paragraphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 317–325.
- [34] Sankar Kuppan, Sobha Lalitha Devi, et al. 2009. Automatic generation of Tamil lyrics for melodies. In *Proceedings of the workshop on computational approaches to linguistic creativity*. 40–46.
- [35] Junyi Li, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Knowledge-based review generation by coherence enhanced text planning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 183–192.
- [36] Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Songnet: Rigid formats controlled text generation. *arXiv preprint arXiv:2004.08022* (2020).

- [37] Qihao Liang, Xichu Ma, Finale Doshi-Velez, Brian Lim, and Ye Wang. [n. d.]. XAI-Lyricist: Improving the Singability of AI-Generated Lyrics with Prosody Explanations. ([n. d.]).
- [38] Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Unsupervised keyphrase extraction by jointly modeling local and global context. *arXiv preprint arXiv:2109.07293* (2021).
- [39] Zhiyu Lin and Mark O Riedl. 2021. Plug-and-blend: a framework for plug-and-play controllable story generation with sketches. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 17. 58–65.
- [40] Danyang Liu and Gongshen Liu. 2019. A transformer-based variational autoencoder for sentence generation. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.
- [41] Nayu Liu, Wenjing Han, Guangcan Liu, Da Peng, Ran Zhang, Xiaorui Wang, and Huabin Ruan. 2022. ChipSong: A Controllable Lyric Generation System for Chinese Popular Song. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*. 85–95.
- [42] Peter Low. 2003. Singable translations of songs. *Perspectives: Studies in Translatology* 11, 2 (2003), 87–103.
- [43] Xu Lu, Jie Wang, Bojin Zhuang, Shaojun Wang, and Jing Xiao. 2019. A syllable-structured, contextually-based conditionally generation of chinese lyrics. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 257–265.
- [44] Xichu Ma, Xiao Liu, Bowen Zhang, and Ye Wang. 2022. Robust Melody Track Identification in Symbolic Music. In *Ismir 2022 Hybrid Conference*.
- [45] Xichu Ma, Ye Wang, Min-Yen Kan, and Wee Sun Lee. 2021. AI-Lyricist: Generating Music and Vocabulary Constrained Lyrics. In *Proceedings of the 29th ACM International Conference on Multimedia*. 1002–1011.
- [46] Eric Malmi, Pyy Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 195–204.
- [47] Enrique Manjavacas, Mike Kestemont, and Folgert Karsdorp. 2019. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. In *Proceedings of the 12th International Conference on Natural Language Generation*. 301–310.
- [48] Suzanne L Medina. 1990. The Effects of Music upon Second Language Vocabulary Acquisition. (1990).
- [49] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.
- [50] Susan Bergman Miyake. 2004. Pronunciation and music. *Sophia Junior College Faculty Bulletin* 20, 3 (2004), 80.
- [51] Dania Murad, Riwu Wang, Douglas Turnbull, and Ye Wang. 2018. SLIONS: A karaoke application to enhance foreign language learning. In *Proceedings of the 26th ACM international conference on Multimedia*. 1679–1687.
- [52] Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan McDonald. 2021. Planning with learned entity prompts for abstractive summarization. *Transactions of the Association for Computational Linguistics* 9 (2021), 1475–1492.
- [53] Hieu Nguyen and Brian Sa. 2009. Rap lyric generator. *New York, USA* (2009), 1–3.
- [54] Nikola I Nikolov, Eric Malmi, Curtis G Northcutt, and Loreto Parisi. 2020. Conditional rap lyrics generation with denoising autoencoders. *CoRR, vol. abs* (2020), 1–13.
- [55] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. 2016. Ensemble of keyword extraction methods and classifiers in text classification. *Expert Systems with Applications* 57 (2016), 232–247.
- [56] Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. PoeLM: A Meter-and Rhyme-Controllable Language Model for Unsupervised Poetry Generation. *arXiv preprint arXiv:2205.12206* (2022).
- [57] Alice Oshima and Ann Hogue. 2006. *Writing academic english*. Pearson.
- [58] Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. GhostWriter: Using an LSTM for Automatic Rap Lyric Generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 1919–1924. <https://doi.org/10.18653/v1/D15-1221>
- [59] Dian Nauwala Putri. 2021. Cohesiveness in informal written text, song lyrics. *ASELS_2021* (2021).
- [60] C Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. 331 Ph. D. Ph. D. Dissertation. thesis, Columbia University.
- [61] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
- [62] Andrés Roberto Rengifo. 2009. Improving pronunciation through the use of karaoke in an adult English class. *Profile Issues in TeachersProfessional Development* 11 (2009), 91–106.
- [63] Joseph Rothstein. 1995. *MIDI: A comprehensive introduction*. Vol. 7. AR Editions, Inc.
- [64] Xianjie Shen, Yinghan Wang, Rui Meng, and Jingbo Shang. 2022. Unsupervised deep keyphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 11303–11311.

- [65] Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. 2021. Songmass: Automatic song writing with pre-training and alignment constraint. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 13798–13805.
- [66] Sung-Hwan Son, Hyun-Young Lee, Gyu-Hyeon Nam, and Seung-Shik Kang. 2019. Korean song-lyrics generation by deep learning. In *Proceedings of the 2019 4th International Conference on Intelligent Information Technology*. 96–100.
- [67] Mingyang Song, Yi Feng, and Liping Jing. 2023. A Survey on Recent Advances in Keyphrase Extraction from Pre-trained Language Models. *Findings of the Association for Computational Linguistics: EACL 2023* (2023), 2108–2119.
- [68] Ruixiao Sun, Jie Yang, and Mehrdad Yousefzadeh. 2020. Improving language generation with sentence coherence objective. *arXiv preprint arXiv:2009.06358* (2020).
- [69] Yi Sun, Hangping Qiu, Yu Zheng, Zhongwei Wang, and Chaoran Zhang. 2020. SIFRank: a new baseline for unsupervised keyphrase extraction based on pre-trained language model. *IEEE Access* 8 (2020), 10896–10906.
- [70] Yufei Tian, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Gunnar Sigurdsson, Chenyang Tao, Wenbo Zhao, Yiwen Chen, Tagyoung Chung, Jing Huang, et al. 2023. Unsupervised Melody-to-Lyrics Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 9235–9254.
- [71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [72] Olga Vechtomova and Gaurav Sahu. 2023. LyricJam Sonic: A Generative System for Real-Time Composition and Musical Improvisation. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer, 292–307.
- [73] Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2020. Generation of lyrics lines conditioned on music audio clips. *arXiv preprint arXiv:2009.14375* (2020).
- [74] Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2021. LyricJam: a system for generating lyrics for live instrumental music. *arXiv preprint arXiv:2106.01960* (2021).
- [75] Wanda T Wallace. 1994. Memory for music: Effect of melody on recall of text. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 20, 6 (1994), 1471.
- [76] Jie Wang and Xinyan Zhao. 2019. Theme-aware generation model for chinese lyrics. *arXiv preprint arXiv:1906.02134* (2019).
- [77] 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)*. 163–172.
- [78] Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, and Masataka Goto. 2014. Modeling structural topic transitions for automatic lyrics generation. In *Proceedings of the 28th Pacific Asia conference on language, information and computing*. 422–431.
- [79] Xing Wu, Zhikang Du, Yike Guo, and Hamido Fujita. 2019. Hierarchical attention based long short-term memory for Chinese lyric generation. *Applied Intelligence* 49, 1 (2019), 44–52.
- [80] Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. *arXiv preprint arXiv:1808.06945* (2018).
- [81] Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural rap generation with rhyme and rhythm modeling. *arXiv preprint arXiv:2107.01875* (2021).
- [82] Kevin Yang, Nanyun Peng, Yuandong Tian, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. *arXiv preprint arXiv:2210.06774* (2022).
- [83] Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7378–7385.
- [84] Yi Yu, Abhishek Srivastava, and Simon Canales. 2021. Conditional lstm-gan for melody generation from lyrics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 17, 1 (2021), 1–20.
- [85] Rongsheng Zhang, Xiaoxi Mao, Le Li, Lin Jiang, Lin Chen, Zhiwei Hu, Yadong Xi, Changjie Fan, and Minlie Huang. 2022. Youling: an AI-assisted lyrics creation system. *arXiv preprint arXiv:2201.06724* (2022).
- [86] Yuxiang Zhang, Tao Jiang, Tianyu Yang, Xiaoli Li, and Suge Wang. 2022. Htkg: Deep keyphrase generation with neural hierarchical topic guidance. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1044–1054.
- [87] Kun Zhao, Hongwei Ding, Kai Ye, and Xiaohui Cui. 2021. A transformer-based hierarchical variational autoencoder combined hidden Markov model for long text generation. *Entropy* 23, 10 (2021), 1277.
- [88] Wei Zhao, Michael Strube, and Steffen Eger. 2022. Discoscore: Evaluating text generation with bert and discourse coherence. *arXiv preprint arXiv:2201.11176* (2022).
- [89] Xu Zou, Da Yin, Qingyang Zhong, Hongxia Yang, Zhilin Yang, and Jie Tang. 2021. Controllable generation from pre-trained language models via inverse prompting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2450–2460.

A LYRICS GENERATION BY CHATGPT

A.1 Generated from existing lyrics

The conversation in Figure 8 is the process of ChatGPT generating lyrics from existing lyrics in ChatGPT’s training dataset. Using query template like “Generate lyrics from the song called <SONG NAME>.”, ChatGPT’s high-quality lyrics generation often contain traces of adaptation from the original lyrics. This could cause copyright issues and is not comparable to our task of generating lyrics without reference lyrics. ChatGPT also copies the song’s structure while our KeYric model can be flexibly applied to new MIDI file input.

A.2 Generated from syllable templates

The conversation in Figure 10 is the process of ChatGPT generating lyrics from a syllable template that contains a sequence of syllable number of each line. “Generate lyrics of sentences having syllable numbers of 8,7,....,6” is the query sentence in ChatGPT. Despite providing lyric examples with syllable numbers, ChatGPT does not understand this specific requirement and still produces lyrics with incorrect syllable numbers. ChatGPT also always uses the same rigid versus-chorus structure, possibly because it favours common lyric structures.

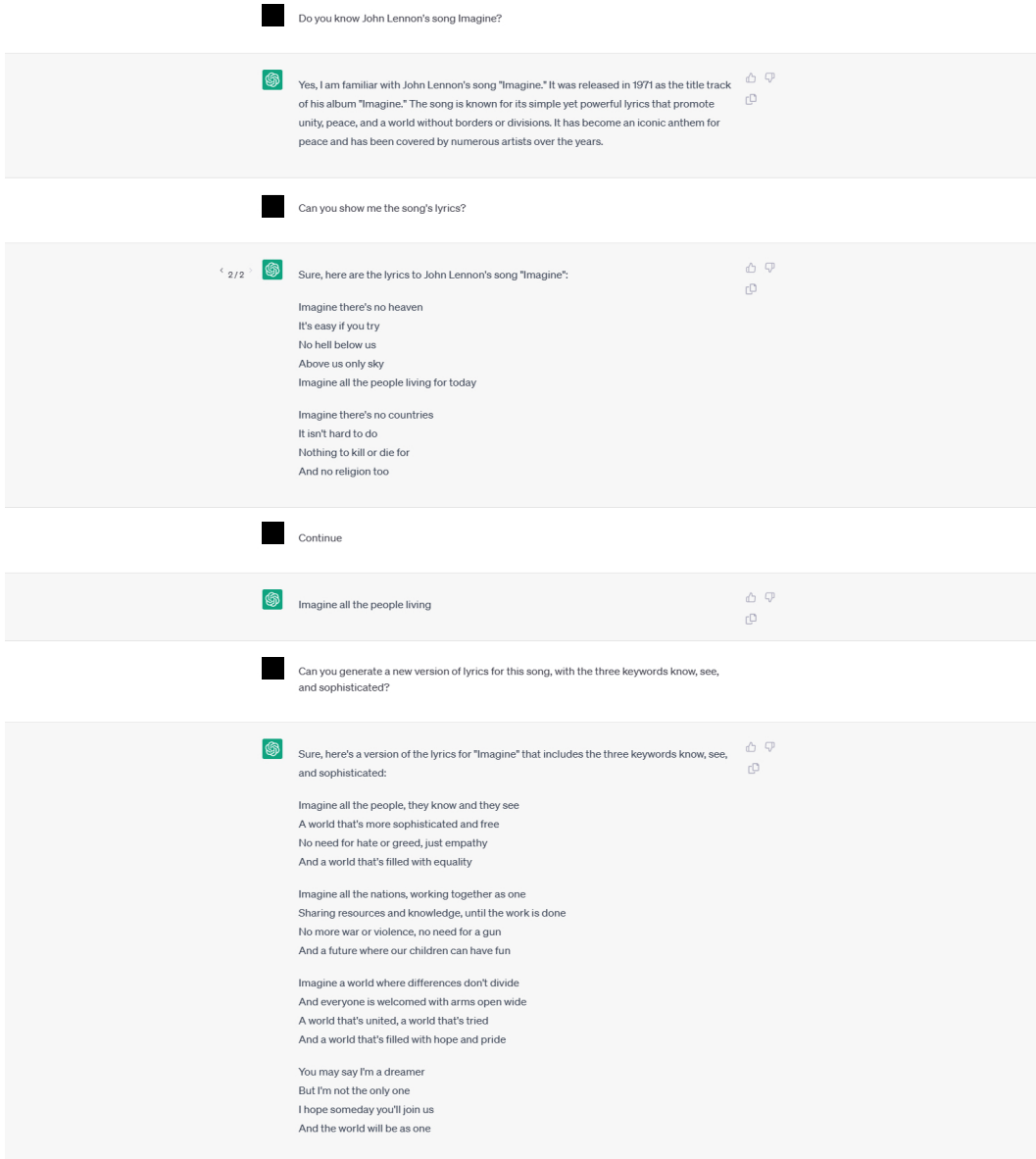


Fig. 8. The conversation with ChatGPT generating lyrics from existing songs.

Let me tell you something. A syllable template of a song's lyrics is a sequence of numbers, each number per lyric line, describing how many syllables this line have.

Thank you for sharing that information with me. Syllable templates can be helpful for songwriters when writing lyrics and setting them to music. By using a syllable template, they can ensure that the lyrics fit within the rhythm and melody of the song. It can also be used to create a particular structure or pattern in the lyrics.

For example, the following lyrics have a syllable template of (7, 5, 8, 5, 6, 5, 6, 9, 5, 5, 8).

As I look at the letters
that you wrote to me
It's you that I am thinking of
As I read the lines,
that to me were so dear
I remember our faded love
I miss you, darling,
more and more everyday
As Heaven would miss the stars above
With every heartbeat,
I still think of you
And remember our faded love

Yes, that is correct! The syllable template for those lyrics is (7, 5, 8, 5, 6, 5, 6, 9, 5, 5, 8), meaning the first line has 7 syllables, the second line has 5 syllables, the third line has 8 syllables, and so on. This syllable pattern helps to give the lyrics a consistent rhythm and flow when set to music.

Can you generate new lyrics that have a syllable number of (8, 6, 8, 5, 8, 11, 10, 7)?

Sure, here's an example of lyrics with a syllable template of (8, 6, 8, 5, 8, 11, 10, 7):

When I wake up in the morning light
I see your face and it feels so right
Your smile shines like the rising sun
It's you that I need, you're the only one

In my heart, you're always there
With every breath, I feel you near
Our love is strong, it's meant to be
Together forever, you and me

No, the syllable number of each line does not match the given template. Please try again.

I apologize for the mistake. Here's another attempt at generating lyrics with a syllable template of (8, 6, 8, 5, 8, 11, 10, 7):

As I walk along the shore at night
I see the stars shining so bright
The waves crash against the shore
My heart races, I want you more

Your love is like the ocean deep
A treasure that I want to keep
You're the light in my darkest days
In your embrace, my heart stays

I know that we're meant to be
Together for eternity

Can you generate new lyrics that have a syllable number of (8, 6, 8, 5, 8, 11, 10, 7), with the three key words know see and sophisticated?

Sure, here's an example of lyrics with a syllable template of (8, 6, 8, 5, 8, 11, 10, 7) that include the key words "know", "see", and "sophisticated":

I know you're out there somewhere
Waiting for me with a love so rare
I can feel your presence in the air
Even though I can't see you there

Your love is sophisticated
Like a fine wine that's aged and waited
For the perfect moment to be savored
And the memories to be created

When we're together, time stands still
Our love is deeper than any thrill
With you, I can conquer any hill
Our bond, unbreakable and real

Fig. 9. The conversation with ChatGPT generating lyrics from syllable templates.

B LYRICS GENERATION SAMPLES FROM COMPARED MODELS

Original

Imagine there's no *heaven*
It's easy if you try
 No *hell* below us
 Above us, only sky
 Imagine all the *people*
 → *Livin'* for today
 Ah - - -

You may say I'm a dreamer
But I'm not the only one
 I hope someday you'll join us
And the world will be as one

AI-lyricist - Auto

I could get me my back unmoved
 We got my head out
 You need to **know** me now
 How bright **see** all light
 So it's all **what** you did before
 I've never been so **sophisticated**
 That's why you go
 Broken Syllable Template

I am casting at the door to go
 Let it be for the world now
 when you go I'll be my conscience oh
 I don't want to alter at all

AI-lyricist - Interact

I lift you up to **see** the world
 → like when you are a girl
 Now we found the love
 And I got a smile girl
 Now I have to be a man
 Doing everything
 → that I **know** about
 Broken Syllable Template

I never got this chance of life
 In a world of only you
 I just need you to be in **sophisticated** mind
 When it finally goes to dust and fire

SongMASS

Well until what please tied you
 Like to talk like
 Nothing choose where you did
 You think you were gave me cry
 We let me stay
 Let me feel you pain
 This time to keep me flat

I think time
 Land just leave you more plane
 And if I know what I just wanna
 → make you until remember

ChatGPT

Imagine there's a world so **sophisticated**
 → where skies are always blue.
 And love and peace is all we ever **knew**.
 Where every soul
 → can be as they desire.
 And we can all **see**
 → what life could be like.
 Broken Syllable Template

Let's imagine a world
 → with no need to fight.
 Where love and hope
 → is what we all know and light.

Proposed

I can't *wait for* a second
 → as a teen *not patient*
Know that I can't find
 → a **sophisticated** heart
 His fingertips trace my skin
 → digging what I hide
 → that I'm in love with you
 Plot Reversal

My *heart is closed* after duels
 Next day we *don't talk* anymore
 Scared the moment will pass, I **see** this
Then the pain will fill in all our heads

Text: Seed words to learn → : **Clauses** Text: **Conjunctions** *Text:* **Cohesive terms** Comments

Fig. 10. Lyrics generated by different lyrics generation models from John Lennon’s song “Imagine”, with seed words **sophisticated**, **know** and **see**. Seed words are denoted by green text, clauses by an arrow →, conjunctions by underscore and comments by a dotted box. Words that tie together related concepts, behaviors and attributes are denoted by italic bold formatting.