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

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

We explore the generation of coherent lyrics from symbolic music for the purpose of creating singing-based language learning materials. This task requires semantic coherence at multiple resolutions — at the word, sentence, and full-text levels, while conforming to the musical style and appropriate syllabic patterns. To solve this task, we introduce KeYric, which links music with lyrics using a keyword skeleton to enhance coherence and musicality. KeYric employs an unsupervised keyword skeleton extractor and a graph-based skeleton expander to create style-appropriate keyword skeletons from input MIDI files. Subsequently, a three-layer coherence mechanism integrates the input MIDI file and its expanded skeleton into lyric generation. The objective evaluation demonstrates a 9% enhancement in lyric coherence overall. Through the participants' subjective evaluation, our generated lyrics are considered to be more coherent and appropriate than compared models for language learning through singing (+31%). Our detailed analyses suggest that incorporating genre-determining components (e.g., pitch) in music encoding is critical for task success as we find musical genres greatly influence lyric's coherence.

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

Additional Key Words and Phrases: Lyric Generation; Keyword Extraction; Textual Coherence; 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 Lyric Generation. *J. ACM* 1, 1 (March 2023), 26 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Psychology and neuroscience research shows that singing songs with appropriate lyrics helps language learners acquire vocabulary [18, 19, 22, 25, 48, 51, 63, 74]. Prior work developed language learning that harnesses learners' natural interests in singing by choosing appropriate songs [52]. However, human-written lyrics may not cover the target words for learning and may be copyrighted, rendering them unusable for language learning. SongMASS [65] and AI-Lyricist [45] have attempted to generate lyrics from input instrumental music. Nonetheless, these efforts still encounter problems such as insufficient semantic coherence, contextual conflicts with incorporated seed words (i.e., target words for learning), inconsistent sentence structures, and deviations from the main theme. AI-Lyricist, as illustrated in Figure 1, requires human intervention to maintain coherence in interactive mode (where the user selects sentences line by line from candidate generation), and its automatic generation frequently produces illogical sentence transitions, and incorporates seed words divergent

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.


0004-5411/2023/3-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

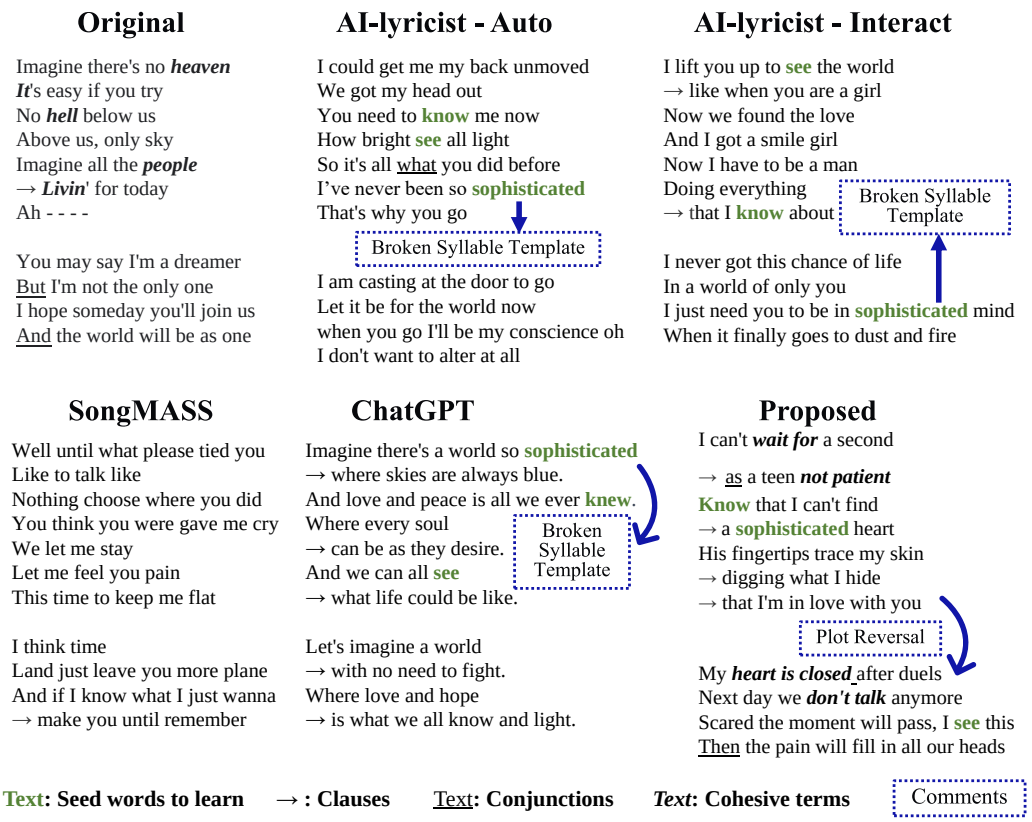


Fig. 1. Lyrics generated by different lyrics generation models from John Lennon's song "Imagine", with seed words **sophisticated**, **know** and **see**. Seed words are denoted by green text, clauses by an arrow →, conjunctions by underscore and comments by a dotted box. Words that tie together related concepts, behaviors and attributes are denoted by italic bold formatting.

from contexts or lacking a common theme. The system can also compromise the number of syllables to incorporate seed words. (e.g. the AI-lyricist Auto & sample in Figure 1). The lyrics generated by SongMASS often appear disjointed and illogical due to the lack of coherence between sentences, failing to extend the same topic or describe a single entity. These are because lyrics generated by AI-lyricist and SongMASS consist of relatively independent sentences, with fewer cohesive devices such as conjunctions, subordinate clauses, pronouns, and cohesive terms, which are crucial for textual coherence [24, 49, 61]. Although ChatGPT[1] can generate coherent and rhyming lyrics, it cannot process music input or fulfill syllable requirements for singing. Moreover, it is prone to plagiarism risks and exhibits black-box characteristics with the latter making it more difficult to incorporate specific requirements (such as seed words) into the lyric generation process. These limitations make automatic lyric generation challenging, especially for language learning purposes.

A "keyword skeleton" is defined as a structured list comprised of dictionary terms, with each element sequentially corresponding to each sentence/line of lyrics. This entire skeleton serves as a synopsis and developmental framework for the lyrics. Using a keyword skeleton as a prompt

[1]https://openai.com/blog/chatgpt

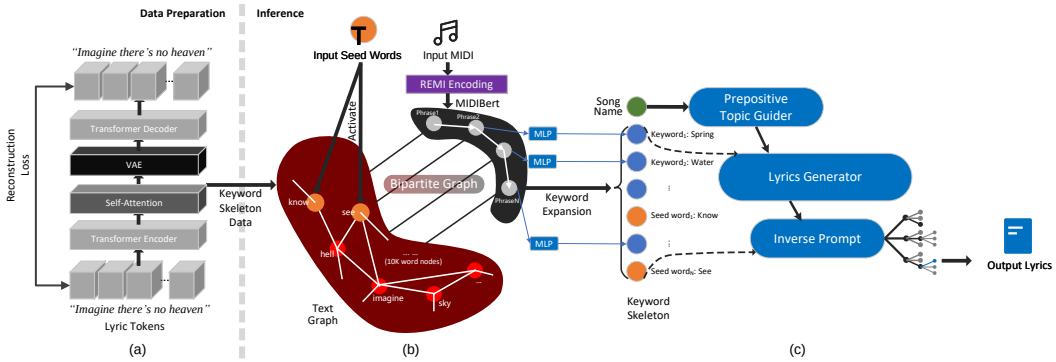


Fig. 2. KeYric architecture. (a) Keyword skeleton extractor creates keyword skeletons from lyrics. The skeletons are then packaged along with the lyrics text and their corresponding MIDI files for model training. Specifically, paired {keyword skeletons, MIDI files} datasets will be utilized to train 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 keyword skeletons serving as heuristic prompts. (b) Keyword skeleton expander trains on pairs to build keyword skeletons 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 lyric generator takes a keyword skeleton and a song name to generate coherent lyrics supporting language learning.

before generating long texts can improve coherence, according to recent research [27]. For example, the keyword skeleton ["spring", "vitality", "breeze", "willow", "burgeon", "breeze", "squirrel", "jump", "breeze", "youth", "love"] can delineate the objects, plot, emotion, and central theme of the entire song titled "Chant of Spring". However, applying this method to lyrics generation introduces additional complications. For one thing, it requires subjective, unstandardized human annotation of keyword skeletons that consist of a sequence of word tokens, one word per lyric line, conveying the most salient semantic, sentimental, and narrative information in that line. For another, current automatic plot planning and keyword skeleton extraction techniques often neglect words that express sentiment and music style [20, 53, 79, 81, 82]. Thus, their created keyword skeletons are less suitable for coherent lyric generation.

We propose unsupervised keyword skeleton extraction from human-composed lyrics, which does not require subjective and time-consuming human annotation. The resulting {lyric-skeleton dataset} then supports expanding input seed words into a keyword skeleton that accords with some input MIDI music paired with the seed words. This expanded skeleton aims to improve local and global coherence in the generated lyrics, helping students understand the seed words' meanings and usages. To achieve this skeleton extraction, we addressed the challenges of: (1) balancing the narrative contents of the lyrics with the sentiment expression, which requires an interpretable and unsupervised method to replace human annotation; (2) developing an effective method for deducing the cross-modal correlation between music and keywords to create skeletons for unseen music; and (3) ensuring that the generated lyrics incorporate the skeletons properly.

Based on the innovative keyword extraction method, we've developed KeYric, a system for unsupervised keyword extraction and expansion from music for coherent lyric generation. The system architecture is shown in Figure 2. The hierarchical Transformer-based Variational Auto Encoder (VAE) [32] extracts keyword skeletons from lyrics unsupervised and interpretably by compressing lyrics into latent space and reconstructing them from Gaussian-resampled latent variables (Figure 2-a). The skeleton is made from the words gaining the most attention (i.e., of the

highest attention scores). A graph-learning-based model (Figure 2-b) converts MIDI files and their paired keyword skeletons into graphs and jointly learns their cross-modal relevance by predicting keyword skeletons from music. Finally, we propose (Figure 2-c), a lyric generator with three layered coherence mechanisms based on GPT-2 [59], using the expanded skeleton as coherence clues. This improves the local and global coherence of the generated lyrics.

Linguists, song writers and language teachers all confirm that KeYric produces language-learning-friendly lyrics. An example generation is shown in Figure 1-Proposed. It can be observed that the lyrics generally follow the storylines by skeletons, matching the style and syllable structure of the input music. Moreover, the user-input seed words "Know," "See," and "Sophisticated" fit seamlessly into their contexts, with consecutive sentences linked by compound sentences, complex sentences and pronouns. As the plot unfolds, the sentences present a consistent theme of love and separation.

Objective and subjective experiments show that the KeYric system gains 9% and 31% improvement in lyric quality over compared models respectively. We also examine how genres and musical elements affect lyric coherence. The results show that the examined pop and country songs yield the most coherent skeletons and lyrics, and that compared to bar boundary information, pitch and note duration are more important to lyric coherence.

In summary, this study makes three major contributions:

- We are the first to address coherent lyric generation and propose a solution that improves coherence at word, sentence, and full-text levels and musicality. This streamlines lyric creation for language learning.
- We propose KeYric to generate lyrics from keyword skeletons. This novel system extracts keyword skeletons from lyrics, expands them using input music and seed words for learning, and incorporates the skeletons in the lyric 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 Lyric Generation

With the rise of neural network technology, Recurrent Neural Networks (RNNs) [21, 45, 46, 58, 66, 78] and Transformers [7, 37, 41, 55, 57, 65, 80] dominate automatic lyric 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 [75, 84], content [55], rhyme [37, 80], matching of syllables [43, 54, 76], audio features [7, 71–73], and realism [47] . To produce well-balanced lyrics, SongNet [37] 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 [35, 65, 76].

Unfortunately, few studies have examined coherence in lyric generation. Even the state-of-the-art lyrics generation frameworks can produce incoherent lyrics due to contextual word conflicts, sentence inconsistency, digressions from the main topic, and musical style mismatches. These problems result in a tangible gap between automatic lyric generation and human song writing.

### 2.2 Coherence in Text Generation

Existing methods on coherent text generation can be categorised into two branches. The first branch generates texts from prerequisite keywords, topics, or sentences. Early work used the hidden state of the previously generated sentence as the context for the following sentence [77], while another work suggested condensing sentences to condition future generations [27]. Other studies have expanded phrase-based storyline plans to coherent stories. Unfortunately, their storylines are selected from (1) high-frequency words [82], which often homogenizes the storylines, (2) predicate-argument structure [20], which ignores sentiment and style-related lyrics, or (3) human annotations [79], which lack uniform and explainable standards. Better methods are needed to extract keyword skeletons from lyrics.

The second branch of methods, enhances coherence through the selection of the most coherent candidate. One study includes an independent model's coherence judgement score in the generation loss [68]. Full-fledged SeqGAN with coherence and cohesion discriminators evaluates the candidates' probabilities of co-occurrence and adjacency with the previous texts [10]. Phrase-level reward then replaces full-sentence reward in the roll-out process to improve efficiency [12]. To be more efficient, a model called GEDI was developed which uses an independent language model to compute the priors of candidate words under a given topic code and previous words, approximating the posteriors given by the discriminators [33]. Lin et al. suggest adding a topic transition planner to GEDI for gradual topic transitions [39]. A recent poem generation study created prompt templates that require candidates to predict the title, previous sentence, or topic and only keep the winners in the beam search [88]. Such templates to predict back (e.g., "... current generation is from a <STYLE> style poem titled <TITLE>") are called "inverse prompts". While the second-branch methods show potential in aiding textual coherence, it is still a crucial and challenging task to incorporate them effectively into lyric generation with appropriate model architectures.

### 2.3 Unsupervised Keyword Extraction

To create keyword skeletons that improve lyric coherence and preserve lyrics' style and sentiment without human annotation bias, we reviewed existing unsupervised keyword extraction studies.

Unsupervised keyword extraction research has spanned decades. In the early stages, researchers primarily employed methodologies from statistics, linguistics, machine learning, and graph theory to extract keywords from text [56]. Subsequently, deep learning and language models have made text-embedding models more popular [2, 67]. PageRank [13] and its derivative system TextRank [50] both construct a document graph and recursively determine each vertex's importance using global information. Clustering similar phrases into topics and weighting them by semantic relation improves this method [6]. PositionRank weights words by position and frequency but treats all topic candidates equally [23]. A multipartite graph structure addresses the issue and enforces topical diversity [5]. Deep learning-based embedding methods have been developed to overcome the slowness and over-generation of graph-based methods. EmbedRank [4] and SIFRank [69] use vectors to represent documents and candidate phrases in a high-dimensional space. UkeRank [38] combines global and local contexts to improve accuracy, while AttentionRank [16] uses a hybrid attention model with BERT [15] to identify keywords. Recent research creates a "phrase bank" by ranking all phrases extracted from a corpus by relevance to new documents [64]. HTKG builds a latent topic tree from corpus documents using autoencoding variational Bayes [85].

However, these methods may not work for lyric keyword skeleton extraction: (1) These methods cannot capture poetic and metaphorical words (often without simile indicators "like" or "as").For instance, "time is a thief" compares time to a thief, implying that time steals moments. Extracting metaphorical keywords from requires understanding the underlying concept being conveyed, which

can be challenging for existing models. (2) They also ignore plot and topic transition, making it difficult to generate coherent texts from extracted keywords. (3) Lyrics have more repetitions and fewer explicit topic markers than typical text documents, making it difficult to identify keywords that accurately reflect their main themes and ideas. Hence, lyric keyword skeleton extraction requires special methods.

## 3 KEYWORD SKELETON EXTRACTION AND EXPANSION

### 3.1 Motivation

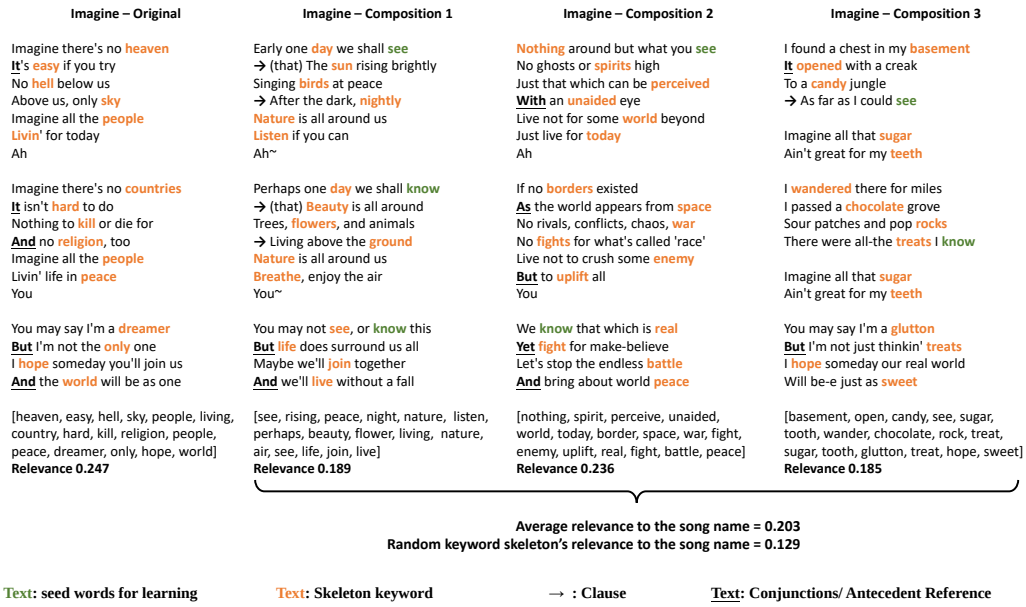


Fig. 3. Human composed lyrics given the mandatory seed words **know** and **see** based on John Lennon's song Imagine. Seed words are denoted by green text, skeleton keywords by orange text, clauses by an arrow →, and conjunctions by underscore.

As illustrated in Figure 3, we've discovered common creative procedures in our testing of human lyricists writing lyrics for linguistic pedagogy. Typically, during writing, lyricists first associate the melody with a given theme or song title, yielding an array of keywords throughout the composition. Subsequently, these words evolve into lyrical cues, and are systematically extended into sentences. During this extension, lyricists consider the integration of pivotal terms, rhyme, melodic alignment, and seed words to be learned. We conducted an initial assessment of the semantic relevance [62] between these keyword skeletons and song titles, detecting a 57.4% enhancement in relevance compared to randomly selected keywords (0.203 v.s 0.129).

Furthermore, we have observed that in comparison to previous automated lyric generation models, human lyricists tend to employ connective elements, referential devices, and subordinate clauses more extensively, enhancing the coherence between consecutive lines of lyrics. Numerous studies indicate that the utilization of conjunctions, pronouns, subordinate clauses, and cohesive terms significantly influences the coherence of text. Specifically, conjunctions join and delineate relationships among ideas within contexts; pronouns track the same referent across multiple sentences; subordinate clauses enable the collective formation of comprehensive expressions

and concepts across multiple sentences, or extend and supplement topics from preceding text to subsequent text; while cohesive terms (as exemplified in Figure 1) interconnect related concepts across sentences [24, 49, 61].

This laborious compositional process necessitates extensive contemplation and iterative revisions by lyricists: Our tests show that lyricists devote over 20 minutes to drafting each song lyric to be aligned with the provided music and designated keywords. This cost can drastically increase in the context of lyrics-based language learning, where lyricists have to additionally tailor their composition based on the background of learners (e.g., linguistic proficiency, vocabulary). Therefore, to enhance the efficiency of lyrics composition for language learning, we propose KeYric, which emulates the writing process of human lyricists through deep-learning.

The core idea is to extract the most salient vocabulary from each lyric sentence through unsupervised learning. We construct a keyword skeleton of words with the highest attention weights during compression and reconstruction, which refines the coherent connections between lyric sentences. A keyword skeleton expander is trained on extracted keyword skeletons and their corresponding MIDI files to predict a suitable keyword skeleton from an unseen input song. Simultaneously, we train a lyrics generator on full lyrics, using the extracted keyword skeleton as prompts. Armed with our proposed multi-layer coherence mechanisms, the generator can select more prevalent conjunctions, pronouns, clauses, and cohesive terms commonly found in human songwriting and stringing together the keywords in the list into coherent lyrics, ultimately achieving the overall coherence.

### 3.2 Keyword Skeleton Extraction

As a condensed version of the entire lyrics, a keyword skeleton extracted from lyrics should present a coherence akin to a storyline, showcasing the narrative development of the lyrics and the central theme around which it revolves. Therefore, the selected keywords should meet the following criteria. (1) Each keyword represents a lyric sentence and conveys its stylistic information in a concise and dense manner. (2) These keywords should also coherently link to nearby keywords. (3) In addition, repeated keywords are allowed in order to present lyric structure. Given a vocabulary set $V$, we define the keyword skeleton extracted from the lyrics as a sequence of word tokens $K = \{ k_1, k_2,..., k_n \}$, where each element $k_t \in V$ in the list corresponds sequentially to a line $t^{th}$ in a specific lyrics $s_t$. Thus, the entire skeleton $K$ serves as a synopsis and developmental framework for the lyrics.

To produce training data for the expander, an unsupervised Transformer-VAE model extracts keyword skeletons from human-written lyrics. This extractor yields keyword skeletons and pair them with the original MIDI files. It also generates two static matrices showing the statistics of keywords' co-occurrence and adjacency. The matrices, along with MidiBERT embeddings [11] of music, are used to construct a graph.

We propose an unsupervised learning model to interpretably select the best keyword from each lyric line (sentence). As shown in Figure 2-a, the keyword extractor compresses lyrics into latent space and reconstructs them using a hierarchical Transformer-VAE. The attention scores of the word-to-sentence encoder decide the lyrics' most semantic and stylistic words that contribute most to the latent variables for each line. These chosen words form the keyword skeleton of the lyrics. The use of VAE improves generalizability and robustness because one song can have many versions of lyrics. A hierarchical architecture separates sentence and word attention computation, making word token attention values more dedicated and representative of their contributions to a sentence.

*3.2.1 Compression.* In Figure 4, 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 $z \in \mathbb{R}^{d_z}$ [32]. $q$ and $p$ are parameterized by $\theta$ and $\phi$ respectively.

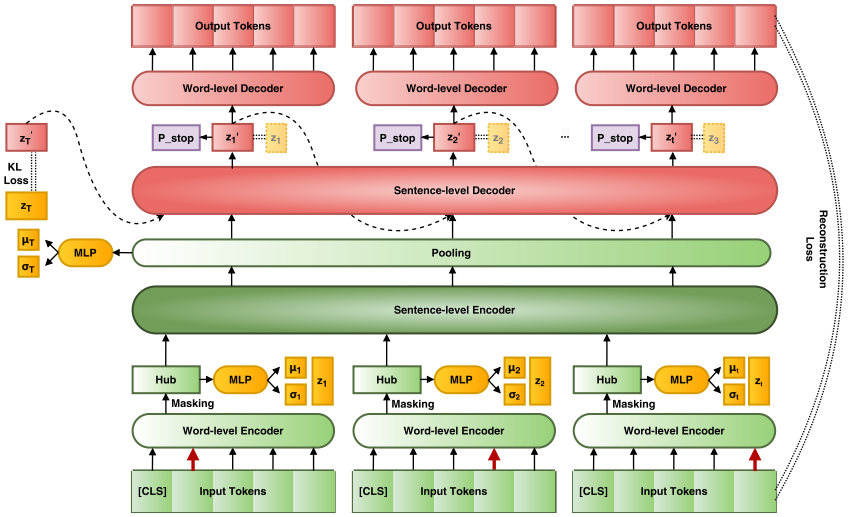


Fig. 4. Keyword skeleton extractor network made of symmetric hierarchical Transformer-VAE encoder and decoder.

The hierarchical Transformer encoder [70] aggregates the word tokens $X = \{x_1, x_2, ..., x_n\}$ into the sentence-level representations $Z = \{z_1, z_2, ..., z_u\}$ stored in hub vectors. We prefix each sentence with a prepositive virtual [CLS] token. A sentence's aggregational latent vector is a Gaussian sample of its Transformer encoding at the foremost hub location. Following standard practice, our prior is an isotropic Gaussian distribution with unit variance, $p(z) = \mathcal{N}(0, I)$. We add positional, part-of-speech (POS), and dependency embeddings [17] in the encoding to incorporate word token's syntactic features.

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

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

where $Emb(\cdot)$ is the word embedding layer and $Emb^+(\cdot)$ is the sum of positional and semantic embeddings. $s_t$ ($t \in [1, v]$) is the $i^th$ 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 only keeps the hub's vector.

*3.2.2 Reconstruction.* Unlike previous research [34, 40, 86], our lyric reconstruction uses a symmetric Transformer-based encoder and decoder because the model seeks to compress and reconstruct 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 lyrics-level decoder's input shape, we expand $z'_T$ into the set $C' = \{c'_1, c'_2, ..., c'_u\}$, where

$c'_i \in \mathbb{R}^{d_c}$. We then decode $z'_T$ into sentence-level latent variables $Z' = \{z'_1, z'_2, ..., z'_u\}$. Finally, we project each $z'_i \in \mathbb{R}^{d_z}$ into the word-level decoder's input shape to re-generate lyric tokens back.

We add a Multi-layer Perceptron (MLP) to predict whether the current sentence is the end of the lyric reconstruction. The MLP predicts each $z'_i$'s probability $P_{stop}$, indicating whether current sentence should be the last.

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

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:

$$\begin{aligned} \mathcal{L}_{VAE} = \alpha \mathbb{E}_{q_\theta(z|x)} \left[ log p_\phi(x|z) \right] + \beta \mathcal{L}_{Stop}(P_{stop}) \\ -\gamma D_{KL} \left[ q_\theta(z|x) || p(z) \right] \end{aligned} \tag{3}$$

The loss has three weighted terms. Reconstruction loss, which compares generated lyrics to ground truth lyrics, is the first term. The second term, a sentence loss on the stopping distribution $P_{stop}$, encourages the model to choose an appropriate length for the re-constructed lyrics, as suggested in previous works [34]. Finally, the Kullback-Leibler divergence penalizes latent variable distribution divergence from a prior in the third term. This term ensures that latent variables have a Gaussian distribution with unit variance. We utilize the "reparameterization trick" [32] to sample latent variables in a differentiable manner by predicting the mean and variance parameters of a Gaussian distribution to calculate the KL divergence.

*3.2.4 Keyword Selection.* After compressing and reconstructing the lyrics, we examine all word-level Transformer encoder blocks' accumulative self-attention matrices to determine each line's keyword. For each lyric line, each token multiplies the attention scores of all layers $W_i^{Att} \in \mathbb{R}^{n \times n}$ along the forwarding propagation path to the hub vector's attention score $W_{Hub_t}^{Att} \in \mathbb{R}^{n \times 1}$. As the example presented by the red arrows in Figure 4, we select $k_t$, the token with the greatest product, as the keyword of its lyric line because this product reflects the tokens' contributions to the sentence encoding. We concatenate all selected keywords to form the skeleton for the whole lyrics.

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

where $k_t$ is the extracted keyword for sentence t, $\psi$ is the layer number of the word-to-sentence encoder.

### 3.3 Keyword Skeleton Expansion

Using two statistical matrices extracted from the lyrics dataset and matched {keyword skeletons, MIDI files} data pairs, the expander then employs a graph-Transformer [28] to jointly learn the cross-modal relevance of keywords and music (Figure 2-b) by modeling keywords and music as graph nodes. The expander then yields keyword skeletons as lyric storylines from user-input seed words $K_{seed}$ and input MIDI music $m$.

Specifically, the expander augments the provided seed words by predicting additional keywords from music and subsequently rearranging them to collectively formulate a keyword skeleton. Each music phrase node predicts a keyword that matches 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.

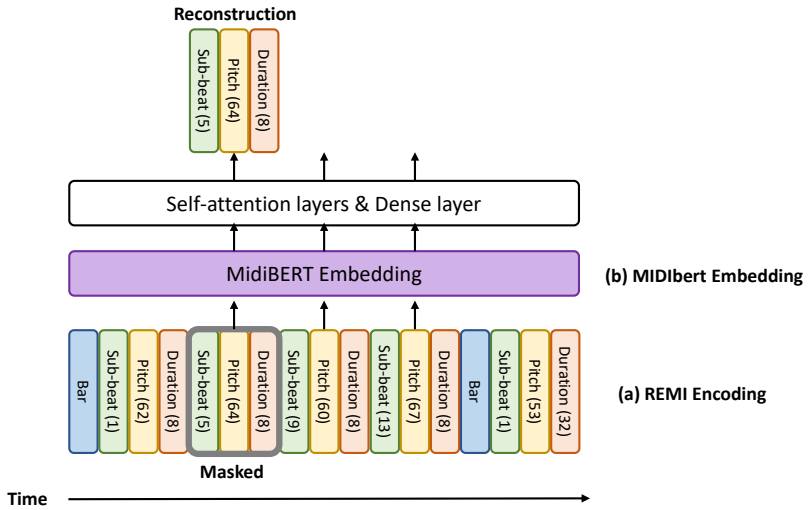


Fig. 5. Illustration of REMI encoding and MidiBERT embedding. (a) The REMI encoding of a MIDI file. (b) The MidiBERT embedding is trained by masked language modeling task which masks and reconstructs the tokens in REMI encodings.

the bipartite graph consists of two sub-graphs, the word sub-graph and music sub-graph...

*3.3.1 Graph Building.* As shown in Figure 2-b, a bipartite graph is built to connect the keyword textual modality with the symbolic music modality. The bipartite graph has two subgraphs, i.e. text graph and music graph. The text graph consists of word nodes of the entire vocabulary, featured by their token IDs. Word nodes are connected by bidirectional edges that show the co-occurrence and adjacency frequency in the statistics yield by the extractor. Music nodes are represented by the MidiBERT embedding [11] of a music phrase's REMI encoding [29]. As demonstrated in Figure 5-(a), REMI is an music event representation proposed for converting MIDI scores into text-like discrete tokens. REMI provides sequence models with a metrical context for modeling the rhythmic patterns. It also encompasses explicit bar delineations, facilitating the segmentation of music encoding into distinct nodes according to lyric phrase divisions within the graph model. On this basis, a large-scale pre-training model for symbolic music understanding, MidiBERT, employs the mask language modeling (MLM) approach learns high-level features by masking and reconstructing input REMI tokens (Figure 5-(b)). It enables the embedding to capture intricate patterns, harmonies, and developmental structures of music.

The music graph connects music nodes by directed edges that show their performance order. Then all word nodes are connected bidirectionally to all music nodes to form a bipartite graph to model cross-modal relationships. By integrating information from both the text and music modalities within a graph network and training it to predict the keywords from paired music phrases, such a graph model generates keyword skeletons that effectively correspond to the musical context, providing a coherent storyline to guide lyric generation.

*3.3.2 Keyword Expansion.* The graph-Transformer computes hidden states for all nodes and propagates keyword information throughout the graph. Compared to sequential or grid models, a graph network (1) deemphasizes autoregressive generation and allows parallel keyword expansion; (2)

captures the topological long-term dependencies among keywords; and (3) unifies music and texts as graph nodes to build cross-modal relevance. The hidden states for nodes and edges are unified as $\mathbb{R}^{d_g}$ to represent all graph elements consistently. After sufficient information propagation, an MLP is applied to each music node to predict and sample a keyword that best represents this music phrase. This procedure is formulated as:

$$\begin{gathered} h_i^x = Emb_{word}(x);\ \ h_i^m = Emb_{MIDI}(R(m)) \\ h_i = \sigma(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W_h h_j);\ \ e_{i,j} = \sigma(\sum_{k \in \mathcal{N}(i)} \alpha_{i,k} W_e e_{i,k}) \\ P(k_i) = Softmax(MLP(h_i^m));\ Loss = CE(P(k_i), \hat{k_i}) \end{gathered} \tag{5}$$

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 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.* We leverage melody identification from [44] and musical snippet segmentation techniques from AI-Lyricist to estimate an appropriate sentence number $L$. After predicting keywords for the first $l_{exp} = L - l_{seed}$ music nodes, we insert the $l_{seed}$ specified seed words into the keyword skeleton. We use average co-occurrence and adjacency probabilities to determine the positions of seed words. Inserting each seed word sequentially maximizes these probabilities of the whole keyword skeleton. The expanded keyword skeleton is output after all seed words are inserted.

Compared with AI-Lyricist, inserting seed words in the skeleton expansion phase avoids conflicts with their surroundings. Maximizing co-occurrence and adjacency probabilities ensures sentence-level coherence. The graph model also creates cross-modal coherence between music and lyrics.

## 4 COHERENT LYRIC GENERATION

We propose to apply three-layer mechanisms to ensure coherence in lyric generation. Its three stacked GPT-2-based submodules, { prepositive topic guider, main-body lyric generator, inverse prompts }, respectively impose coherence constraints before, during, and after lyric probability computation. (1) The expanded keyword skeleton prompts the main-body lyric generator, while (2) the prepositive topic guider uses the song name with previously generated words to constrain next word selection. (3) Finally, beam search with inverse prompt evaluates lyric candidates based on their ability to embody the keyword skeleton.

We employ GPT-2 as the foundation model for all the three stacked layers due to its power, reproducibility, interpretability, and computational affordability. It allows us to explore the factors and techniques beneficial to lyric coherence within the available computing resources, although GPT-3.5 or GPT-4 may perform better. A textual lyric dataset pre-trains the three•models•to adapt to poetic lyrics. Each•model is then fine-tuned for their•tasks. These mechanisms collaborate to generate fluent, coherent, musically relevant lyrics.

### 4.1 Main-body Lyric Generator

Based on GPT-2, the main-body lyric generator generates subsequent tokens autoregressively. The main-body generator is fine-tuned to generate lyrics prompted by a given number of syllables and

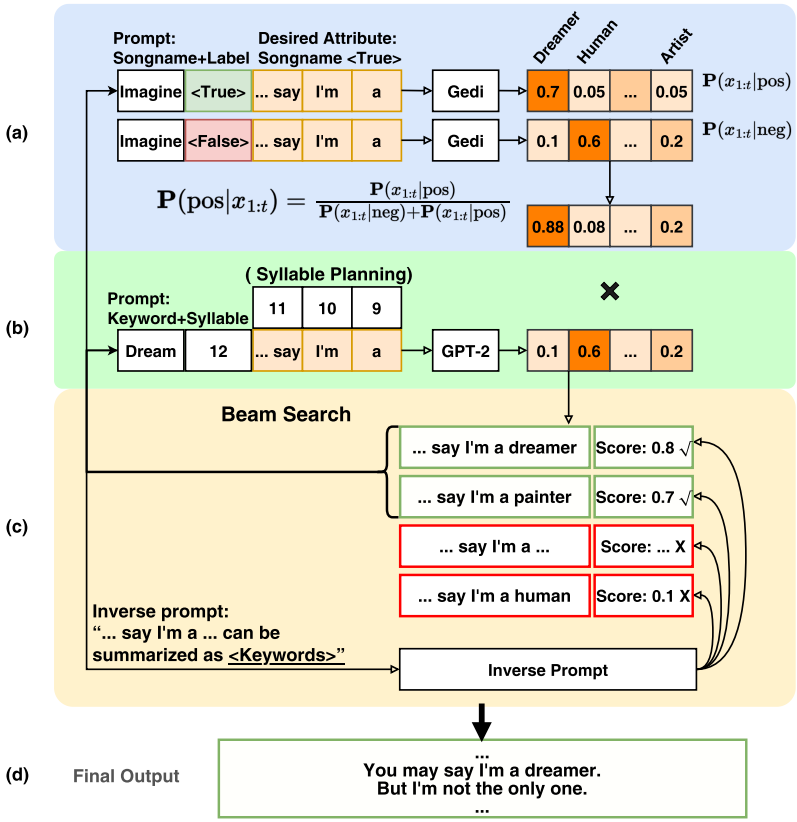


Fig. 6. Coherent lyric generator's network architecture. (a) The prepositive topic guider. (b) The main-body lyric generator. (c) Post-beam search driven by inverse prompts.

a keyword in the skeleton after pre-training on a lyric dataset. As shown in Figure 6-b, keywords and syllable counts precede each lyric sentence due to their fixed length. We also add a syllable planning embedding $sl$ to each token's word embedding, indicating how many syllables remain in the current sentence. The lyric generator then generates probabilities for subsequent token selection, denoted by $P_w(x_t|x_{<t}, sl_{<t}, k_{<i})$.

### 4.2 Prepositive Topic Guider

To guide the generation of lyrics that match the desired attributes, typically discriminators are used to compute the degree to which the generated lyrics align with a given attribute, formulated as:

$$P_y(x_t|x_{<t}, y) \propto P_w(x_t|x< t)P_d(y|x_t, x< t) \tag{6}$$

However, the subjective and multifaceted characteristics of song lyrics make them difficult to describe. Rather than attribute classes utilized in [33], we propose to employ song name to guide the generation because song names often summarize themes, sentiments, and content.

Concretely, we augment the main-body generator with a prepositive topic guider based on GEDI [33]. The prepositive topic guider computes the probability that each candidate next token $x_t$ matches the desirable features contained in the given song name prompt $y$ (i.e., $P_d(y|x_t, x_{<t})$), thereby

replacing the ineffective and inefficient roll-out and reward processes of conventional discriminators. As illustrated in Figure 6-a, the prepositive 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.

$$P_d(y_{pos}|x_{1:t}) = \frac{P_y(x_{1:t}|y_{pos})}{P_y(x_{1:t}|y_{neg}) + P_y(x_{1:t}|y_{pos})} \tag{7}$$

This probability is then multiplied to constrain the token selection alongside the main-body generator's prediction.

### 4.3 Inverse Prompts

Generating long texts often deviate from the prompt and include irrelevant content. To address this issue, we employ the inverse prompt mechanism [88], which is a beam scoring function that evaluates the log likelihood in reverse. Traditional beam search calculates beam scores using log likelihood of generating lyrics from prompts: $BeamScore(X|K) = logP_w(X|K)$, but inverse prompt does the opposite way, assuming that if the prompts can be generated back from lyrics, they must be closely related. The inverse beam score can thus be formulated as $BeamScore_{IP}(X|K) = logP_w(K|X)$. For instance, traditional prompting strategy is intuitively formatted 'K results in X', but inverse prompt is "X inferred from K"

However, simply reversing the order of prompts and lyrics can produce unnatural texts [88]. A more natural inverse prompt is to predict the original prompts intended from the generated text. In our case the inverse prompt is tasked to summarize generated lyrics $X'$ back into a keyword skeleton $K'$. After that, beams are rated by: $BeamScore_{IP}(X|K) = logP_w(K'|X')$.

An example is shown in Figure 6-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.

## 5 OBJECTIVE EXPERIMENT

### 5.1 Dataset

We used the Netease API to crawl English lyrics with the 100 most frequent tags from a lyric dataset [83]. We call this dataset ***Netease-lyrics*** consisting of 160,171 {song name, lyric} pairs for training the keyword skeleton extraction model. As for training the keyword skeleton expansion model, we built the ***LMD-lyrics*** dataset from the Lakh MIDI dataset [60]. The LMD-lyrics dataset contains 7,211 {song name, lyric, MIDI file} triplets. All lyrics in the datasets are segmented by lines. We split both datasets into 8:1:1 train, validation, and test subsets.

### 5.2 Configurations

The keyword skeleton extractor uses a standard encoder-decoder block implementation [70]. The size of the hidden states $z \in Z$ is 256 ($d_z = d_c = 256$). Based on our preliminary experiments, we set $\alpha$ to 1.0, $\beta$ to 4.0, and $\gamma$ to 0.2 [34]. The graph network in the keyword skeleton expander has a unified embedding size of 256 ($d_g$=256), a propagation layer number of 7 to match the average length of 2 verse and 2 chorus sections, and covers the 10,000 most frequent words. The graph propagation is conducted with sub-graphs of size 1024 in batches. We also reused the Netease-lyrics

dataset to pretrain the three GPT-2 models (prepositive topic guider, main-body generator, and inverse prompt scorer) with masked language model (MLM) task [15] and reused the LMD-lyrics dataset to them with respective prompt templates.

### 5.3 Compared Methods

Our keyword skeleton extraction (Proposed-K) and keyword expansion (Proposed-G) models were compared to a number of unsupervised keyword extraction techniques. These included graph-based algorithms such as TextRank, TopicRank, MultipartiteRank, and PositionRank, embedding-based algorithms such as EmbedRank and SIFRank, and attention-based algorithms such as AttentionRank and UkeRank [38].

We compared our proposed coherent lyric generation model (Proposed) to two studies with similar task definitions. One is AI-Lyricist, which is based on SeqGAN [45]. The other is SongMASS [65] using a Transformer to write lyrics from a melody line.

An ablation study evaluated the ability of each coherence mechanism to improve lyrics compared other lyric generators. Three versions of our proposed lyric generator were evaluated: (1) a vanilla GPT-2 generator with the keyword skeleton as prompts (Proposed-Lite), (2) a generator with prepositive topic guider (Proposed-Pre), and (3) a generator with only inverse prompts (Proposed-IP). This allowed us to isolate and determine the importance of each coherence mechanism in enhancing the lyrics' coherence and musicality.

### 5.4 Objective Measures

We conducted an objective experiment to evaluate the extracted and expanded keyword skeletons and the applicability of the proposed lyric generator to language learning.

The keyword extractor and the expander were evaluated on five metrics. The first metric, **"representativeness"**, measures the extent to which the extracted keyword skeleton represents the semantic content and linguistic characteristics of the original lyrics [20]. This metric is the average cosine similarity between each lyric sentence and its corresponding keyword embedding since similar embeddings imply their interchangeability. The second metric, **"coherence"** measures the skeleton's topic transitions [30]. It is the average log probability of the text graph edges that indicate keyword adjacency. This metric reflects the frequency of occurrence of consecutive pairs of words from the keyword skeleton appearing in closely adjacent lines. Thus, it measures the extent to which the words in this skeleton can represent a coherent storyline. It is formulated as:

$$C_K = \frac{1}{|E|} \sum_{e_{i,j} \in E} \log P(e_{i,j}) \tag{8}$$

where each $e \in |E|$ denotes the edge in the text graph defined in subsubsection 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 text graph, indicating the frequency of subsequent occurrence of the keywords $k_i$ and $k_j$. We propose the third metric, **"uniformity"**, which evaluates the skeleton's keyword distribution, with one keyword per lyric sentence being optimal. It is the ratio of lyric sentences without extracted keywords multiplied by the number of keywords in the skeleton as a balancing coefficient. This ensures that each sentence in the skeleton contributes a keyword, effectively supporting the storyline cohesively without missing any key points. High uniformity suggests that all key words come from a concentrated few lines, resulting in the loss of content from other lines. The fourth metric, **"cross-modal relevance"**, measures the correlation between text and musical features and is computed as the normalized dot product of the text feature vector and the musical feature vector [45]. The fifth metric, **"diversity"**, measures the lyrics dataset's word choice diversity [20, 82]. It is the average pairwise difference between two keyword skeletons and

is formulated as:

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

where S is keyword skeleton of the lyrics in the test set and | | denotes the size of a set. Diversity is beneficial, but too much can cause random keywords.

Following previous studies on coherent text generation, we evaluate lyric generators in terms of two related metrics: local [26, 36] and global coherence scores [27]. The local coherence is measure by topic switching detection, calculating the probability that two consecutive sentences share the same topic [3]. Global coherence is evaluated by a documentary model that predicts the coherence value of the entire document through supervised regression [1]. Additionally, we conducted part-of-speech (POS) tagging on the generated lyrics and computed the proportion of elements contributing significantly to coherence in the generated lyrics, including conjunctions, subordinate clauses indicators, and pronouns.

Table 1. Results of the experiments. (a) & (b) are objective and subjective measures of keyword skeleton level evaluation while (c) & (d) are objective and subjective measures of lyric evaluation. The bold red values represent the best performing values, while the underlined values respresent second best performing.

**Evaluation on Keyword Skeleton**

(We omit the values of the keyword expander on representativeness, uniformity and faithfulness as the expander is independent of the original lyrics.)

| | (a) Objective Evaluation (Keyword Skeleton) | | | | | (b) Subjective Evaluation (Keyword Skeleton) | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Representativeness↑** | **Coherence↓** | **Uniformity↓** | **Cross-modal Relevance↑** | **Diversity↑** | **Coherence↑** | **Faithfulness↑** | **Musicality↑** | **Sentiment↑** |
| SIFRank | 0.5 | 6.69 | 5.30 | 0.52 | 6.2 | 3.24 | 2.91 | 2.88 | 2.72 |
| MultiPartiteRank | 0.56 | 6.73 | 4.11 | 0.55 | 13.59 | 3.16 | 2.93 | 2.91 | 3.22 |
| TopicRank | 0.55 | 6.7 | 4.06 | 0.56 | 12.48 | 3.22 | 2.93 | 2.90 | 1.60 |
| AttentionRank | 0.42 | 6.66 | 5.91 | 0.39 | 5.27 | 1.75 | 1.60 | 1.73 | 2.90 |
| TextRank | 0.54 | 6.7 | 4.2 | 0.48 | 11.19 | 3.25 | 3.09 | 2.90 | 2.51 |
| PositionRank | 0.45 | 6.5 | 6.7 | 0.45 | 3.86 | 2.83 | 2.68 | 2.62 | 2.48 |
| EmbedRank | 0.57 | 6.48 | 2.75 | 0.51 | 13.46 | 3.38 | 3.18 | 3.04 | 2.94 |
| UKERank | 0.442 | 6.43 | 3.98 | 0.42 | 4.62 | 2.48 | 2.33 | 2.35 | 1.55 |
| **Proposed-K** | **0.6** | 6.33 | **2.04** | **0.64** | **15.68** | 3.59 | **3.69** | **3.59** | **3.69** |
| **Proposed-G** | - | **5.13** | - | **0.64** | 13.82 | **3.60** | - | 3.55 | 3.23 |

**Evaluation on Generated Lyrics**

| | (c) Objective Evaluation (Lyrics) | | | (d) Subjective Evaluation (Lyrics) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **Local Coherence↑** | **Global Coherence↑** | **Proportion of Coherent Terms↑** | **Fluency↑** | **Local Coherence↑** | **Global Coherence↑** | **Learnability↑** | **Singability↑** | **Musicality↑** |
| Original | 0.875 | 1.66 | **0.28** | 3.99 | 3.90 | 3.84 | 1.40 | 3.45 | 3.24 |
| AI-Lyricist | 0.825 | 1.52 | 0.16 | 1.79 | 1.66 | 1.51 | 2.33 | 2.27 | 1.83 |
| SongMASS | 0.845 | 1.54 | 0.19 | 2.42 | 2.16 | 2.02 | 1.03 | 2.45 | 2.07 |
| **Proposed** | **0.881** | **1.69** | 0.27 | **3.69** | **3.23** | **3.07** | Δ **2.98** | **3.08** | **2.61** |
| Proposed-Lite | 0.854 | 1.57 | 0.22 | - | - | - | - | - | - |
| Proposed-Pre | 0.86 | 1.65 | 0.24 | - | - | - | - | - | - |
| Proposed-IP | 0.876 | 1.61 | 0.24 | - | - | - | - | - | - |

## 5.5 Objective Experiment Results

The keyword skeleton evaluation results are shown in Table 1-a. Our proposed model for extracting keyword skeletons outperforms other methods in all five metrics. The most notable enhancement is in uniformity (the lower the better), showing a 26% reduction compared to the second-best approach. The one keyword per line strategy prevents distribution bias and enhances the episodic coherence of the lyric skeleton. The VAE captures important keywords from lyrics while avoiding sole reliance on high-frequency words, resulting in a 15% increase in diversity. This approach may also explain the 14% increase in cross-modal relevance. Our proposed keyword expander exhibits superior coherence and cross-modal relevance compared to other models, with improvements of 20% and 14%, respectively. This indicates the effectiveness of deep graph networks in capturing the correlation between music and lyrics.

In contrast, AttentionRank uses an attention mechanism, but without a lyric reconstruction process it may choose articles and auxiliary words that misrepresent the lyrics. This lowers its representativeness score. TopicRank and TextRank algorithms construct text graphs and select keywords

without considering sentence order, resulting in weaker coherence. Due to its reliance on keyword frequency and previous occurrence, PositionRank procuces undiversified keyword skeletons. EmbedRank, which extracts keywords using embeddings, ranks second in competitiveness. Its strategy is to select heavily modified words, such as a noun surrounded by many adjectives, creates information-dense words that effectively represent the lyrics. However, disregarding 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-c. Our model surpasses the AI-Lyricist by 9% overall in coherence. It validates that the three-layer mechanisms effectively enhance the coherence of the generated lyrics. Our model also outperforms SongMASS (+7%), suggesting that incorporating additional human knowledge, such as syllable template and keyword skeleton input, is more effective than relying solely on the model to capture cross-modal relevance automatically. The calculation of coherence improvement involves obtaining the average improvement percentage for each metric listed in Table 1-(c). For example, when comparing our model to AI-Lyricist, the improvement in local coherence is calculated as (0.881 - 0.825) / 0.825 = 0.068, and the improvement in global coherence is calculated as (1.69 - 1.52) / 1.52 = 0.112. Thus, the overall improvement is calculated as (0.068 + 0.112) / 2 = 0.09 (9%). The calculation method for the +7% improvement over SongMASS is the same as this approach.

Simultaneously, we observed a significant positive correlation between the local and global coherence of lyrics in objective experiments and the proportion of coherent elements they contain (as shown in Table 1-(c)). This indicates that employing these elements more extensively during lyric generation tasks is beneficial for enhancing lyrical coherence. Furthermore, our proposed approach demonstrates a greater utilization of these words compared to other models during lyric generation (+68% over AI-Lyrcist and +42% over SONGMASS). It suggests that our proposed three-layer coherence mechanism successfully guides the model to logically interconnect the keywords of skeleton.

In addition, it is also surprising to find KeYric-generated lyrics even more coherent than the original ones. It verifies the competence of our proposed keyword expander to create coherent keyword skeleton from only input music and seed words. By incorporating the Prepositive Topic Guider and Inverse Prompts, the preceding and subsequent sentences are subject to a unified constraint, resulting in higher local (and global) coherence compared to human composition. The lyrics generation proposed in this paper serves the purpose of language learning, where coherence is a crucial aspect. This requirement, however, is not as explicitly emphasized in general human composition compared to our proposed method. Therefore, both at the local and global levels of coherence, our model surpasses the original lyrics.

## 6 SUBJECTIVE EVALUATION

### 6.1 Experiment Participant and Procedures

We recruited participants via email by sending invitations within the university. The invitation required that English should be the participants' first language. Upon completing the experiment properly and passing the manipulation check, participants were reimbursed S$30. We recruited 24 participants, with 3 professional lyricists and 1 language teacher.

The participants were trained before they started the main experiments. The participants were shown a sample keyword skeleton, a sample lyric with their paired music. Then they were asked to rate a series of keyword skeletons and lyrics. During this process, they were also shown the descriptions of rating standards of keyword skeletons and lyrics, each followed by an example corresponding to marks 1-5. After the training, the participants moved on to the main experiment, which consisted of two sections. In section 1, the participants read 100 keyword skeletons, listened

to their paired music in random order and rated them. In section 2, the participants read 40 lyrics, listened to their paired music in random order and rated them.

We employed the following approaches in our experiment to maximize the validity of the subjective experiment. (1) We conducted a within-subjects experiment with randomised display order. (2) Since the experiment was conducted online, we trained the participants by fully demonstrated the example ratings and the rating reasons to avoid inconsistent responses among participants. For instance, when designing the following question regarding singablity, we have designed the following question followed by examples that should be rated as 1-5 to avoid common flaws of survey question such as ambiguity, inductivity and, uncertainty: *"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) During the participant recruitment process, we aim to balance the number of participants such that while their first language is all English, their other spoken language(s) may vary. This stratified/quota sampling serves to mitigate linguistic biases. (4) We implement manipulation check to verify that users genuinely engaged in the experiment rather than randomly assigning scores.

### 6.2 Subjective Metrics

User rating surveys were conducted to evaluate the quality of the extracted keyword skeletons from lyrics, expanded keyword skeletons from music and the generated lyrics, with a particular focus on the most important aspects. We evaluated the keyword skeletons' coherence, faithfulness, musicality, and sentiment. (1) Coherence evaluates the coherent progression of storylines [9], (2) faithfulness to the accurately summarizing the original lyrics, (3) musicality if the lyrics match the music style [45], and (4) sentiment to the expression of emotions in the original songs or input music. Six criteria were used to evaluate the generated lyrics: fluency, local coherence, global coherence, learnability, singability, and musicality. (1) Fluency ensures meaningful and natural English [87]. (2) Local coherence ensures smooth transitions between sentences, whereas (3) global coherence ensures that the entire lyrics focus on the same theme [87]. (4) Learnability guarantees that seed words are seamlessly integrated into the lyrics. In the experiment, participants are required to judge the presence of user-specified words (such as "know" and "see") in the generated lyrics. They were also asked to evaluate whether these words embody the correct semantics and whether they are appropriately contextualized and coherent. (5) Singability ensures that the syllables align to the melody notes [31], and (6) 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 1-b, our models' keyword skeletons help users envision storylines that match the music style and sentiment in their minds. Moreover, participants said that the keyword extractor could summarize the lyrics more accurately than compared models. Our keyword extractor and expander improve over compared methods by 15% and 8% overall, calculated by the average improvement over the second-best model in each metric.

Our lyric generator with three-layer coherence mechanism outperforms its competitors in text quality, local and global coherence, as well as the cross-modal relevance after incorporating the expanded keyword skeleton. Our KeYric system improves coherent lyric generation comprehensively by 31%, calculated by the average improvement over the second-best performing model in

each metric. It is especially strong in the three metrics of coherence at the word, sentence, and whole-piece levels. These results support our system design motivation.

Despite the limited availability of scalable methods for searching human-written lyrics that contain seed words for learning and matching personal language proficiency, our lyric generator supports personalized language learning through singing by incorporating input seed words naturally than the original lyrics of input songs (+113%). This is because neither the original lyrics of a user-preferred song nor lyrics generated by other generic lyrics generation models explicitly include the words that the users need to learn.

Furthermore, our proposed model exhibits remarkble enhancements (+28%) in learnability compared to its most immediate competitor, AI-lyricist. Accurate manifestation of keyword semantics in lyrics holds paramount significance for comprehending new vocabulary and language acquisition. This improvement underscores the superior effectiveness of our model in language learning because it seamlessly and appropriately integrates pivotal keywords to learn into the generated lyrics.

In summary, our subjective experiments indicated that our lyrics are engaging, coherent, pleasant and artistic, making them suitable for language learning.

Table 2. Meso 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 valus.

**(a) Results of Subdatasets of Different Music Genres.**

| Genres | Keyword Coherence↑ | Local Lyrics Coherence↑ | Global Lyrics Coherence (Global)↑ | Normalized Avarage↑ |
|---|---|---|---|---|
| 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↑ | Local Lyrics Coherence↑ | Global Lyrics Coherence (Global)↑ |
|---|---|---|---|
| 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> |

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

To further study the effects of diifferent musical factors on lyrics coherence, we divided the LMD-lyrics dataset into 17 genres, trained and evaluated their expanded keyword skeletons and generated lyrics independently to understand how music genres affect lyric coherence. Specifically, we have extracted songs from the database with identified genre attributes, including a total of 17 distinct musical genres. These cover 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%), newage (1.98%), pop (11.96%), punk (6.84%), reggae (0.6%), R&B (0.96%), and rock (26.62%). Table 2-a shows the rankings of their results of coherence at different levels, where it is clearly that pop and country music produce the most coherent keyword skeleton/lyrics. Their original human-written lyrics' coherence and narrativity may be the main reason. In contrast, classical music lacks lyrics, while gospel songs employ chanting and exclamatory words, resulting in less coherent keyword skeleton expansion and lyric generation.

According to our case studies, classical songs' lyrics may be incomplete or missing; punk and folk songs' lyrics may not always follow a predetermined arrangement, etc. Large language models (LLMs) like ChatGPT cannot grasp onto any patterns in such genres. As a result, we decide to train our full-fledged lyrics generation model on genres with average coherence scores of 0.53 and higher (metal and above). The trained model has also been tested to perform well on unseen genres.

We further investigated which musical elements are more decisive to the cross-modal coherence of lyrics. Specifically, we studied the change in lyric coherence after excluding different elements in REMI melody representation. As presented in Table 2-b, the absence of pitches leads to significant decreases in keyword and whole-piece coherence (-13% and -7%). Surprisingly, removing bar line information increases keyword coherence by 2%, but reduces the coherence between consecutive lines by 9%. This is because the 'line of lyrics' does not necessarily match the bar lines. These findings aid us in selecting musical elements to establish cross-modal relevance between texts and music in our future work.

### 7.1 Ablation Study and Case Study

The analysis of lyrics generated by our proposed model and ablated models (Proposed-Lite, Proposed-Pre, and Proposed-IP) reveals that Proposed-Pre generates more sentimentally and thematically coherent lyrics than Proposed-Lite (+2.9%). We observed that the majority of the lyrics' sentences generated by Proposed-Pre have a consistent tone and focus on a shared topic. We believe that the condition created by the prepositive topic guider influences each step of the generation process consistently.

The lyrics generated by Proposed-IP display tone and subject shifts. These lyrics, however, contain more consecutive and supplementary words to link these shifts. We also observe a rise in the prevalence of longer compound and complex sentences, especially those with attributive and adverbial clauses. The Proposed model incorporates both Proposed-Pre and Proposed-IP, thereby showing their features simultaneously. Its generation has smooth topic transitions between paragraphs and coherence within paragraphs.

## 8 DISCUSSION AND FUTURE WORK

Despite the recent success and attention garnered by large language models such as ChatGPT in natural language processing tasks, we contend that their proficiency in generating lyrics for language learning falls short of our proposed system. We have found two query templates (prompts) that ChatGPT can use to generate lyrics, though each has its own challenges.

The first query template is "Generate lyrics from the song called <SONG NAME>." ChatGPT's high-quality lyrics 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. On the other hand, our proposed model takes keywords as input, and during the generation phase, it remains blind to the original lyrics. Upon comparison, we have found that, except for very short sequences, the probability of our model copying lyric lines from the original lyrics for a given keyword is extremely low.

"Generate lyrics of sentences having syllable numbers of 8,7,...,6" is the second query template. 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. Finally, ChatGPT cannot accommodate music inputs due to the scarcity of paired data. Some samples of ChatGPT generating lyrics are presented in Appendix A. In contrast, KeYric is able to produce lyrics with correct syllable numbers and more flexible song structures.

However, while KeYric can generate coherent lyrics matching the music's emotion and style, it can be improved. The literary structure of the lyrics can conflict with the music structure, confusing listeners. For instance, the last music snippet is assigned a sentence that is expected to continue, causing ambiguity about whether the phrase is ending or continuing. Effective music structure analysis, a field-wide question, may solve this problem in future work.

## 9 CONCLUSION

This paper examines lyric coherence at word, sentence, full-text, and cross-modal levels. We study the problem of existing keyword extraction methods in improving lyric coherence. We then propose unsupervised keyword extraction from lyrics and keyword expansion from music to address this challenge. We also suggest using multiple coherence mechanisms with keyword skeletons to improve coherence before, during, and after the lyric token prediction of a lyric generator. Our KeYric system outperforms the state-of-the-art models by 31%. Finally, we analyze the influence of genres and musical elements on coherent lyric generation.

## ACKNOWLEDGMENTS

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] Talmy Givón. 1993. Coherence in text, coherence in mind. *Pragmatics & Cognition* 1, 2 (1993), 171–227.

[25] 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.

[26] 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).

[27] 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).

[28] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*. 2704–2710.

[29] 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.

[30] 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.

[31] Haven Kim, Kento Watanabe, Masataka Goto, and Juhan Nam. 2023. A Computational Evaluation Framework for Singable Lyric Translation. *arXiv preprint arXiv:2308.13715* (2023).

[32] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[33] 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).

[34] 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.

[35] 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.

[36] 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.
[37] Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Songnet: Rigid formats controlled text generation. *arXiv preprint arXiv:2004.08022* (2020).
[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, Pyry 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] Vesna Bagarić Medve and Višnja Pavičić Takač. 2013. The influence of cohesion and coherence on text quality: A cross-linguistic study of foreign language learners' written production. *Language in cognition and affect* (2013), 111–131.
[50] 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.
[51] Susan Bergman Miyake. 2004. Pronunciation and music. *Sophia Junior College Faculty Bulletin* 20, 3 (2004), 80.
[52] 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.
[53] 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.
[54] Hieu Nguyen and Brian Sa. 2009. Rap lyric generator. *New York, USA* (2009), 1–3.
[55] 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.
[56] 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.
[57] 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).
[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] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[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] Gisela Redeker. 2000. Coherence and structure in text and discourse. *Abduction, belief and context in dialogue* 233, 263 (2000).
[62] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[63] 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.

[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] 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).

[71] 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.

[72] Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2020. Generation of lyrics lines conditioned on music audio clips. *arXiv preprint arXiv:2009.14375* (2020).

[73] Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. 2021. LyricJam: a system for generating lyrics for live instrumental music. *arXiv preprint arXiv:2106.01960* (2021).

[74] 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.

[75] Jie Wang and Xinyan Zhao. 2019. Theme-aware generation model for chinese lyrics. *arXiv preprint arXiv:1906.02134* (2019).

[76] 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.

[77] 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.

[78] 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.

[79] 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).

[80] 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).

[81] 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).

[82] 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.

[83] 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.

[84] 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).

[85] 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.

[86] 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.

[87] Wei Zhao, Michael Strube, and Steffen Eger. 2022. Discoscore: Evaluating text generation with bert and discourse coherence. *arXiv preprint arXiv:2201.11176* (2022).

[88] 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 7 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 8 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.

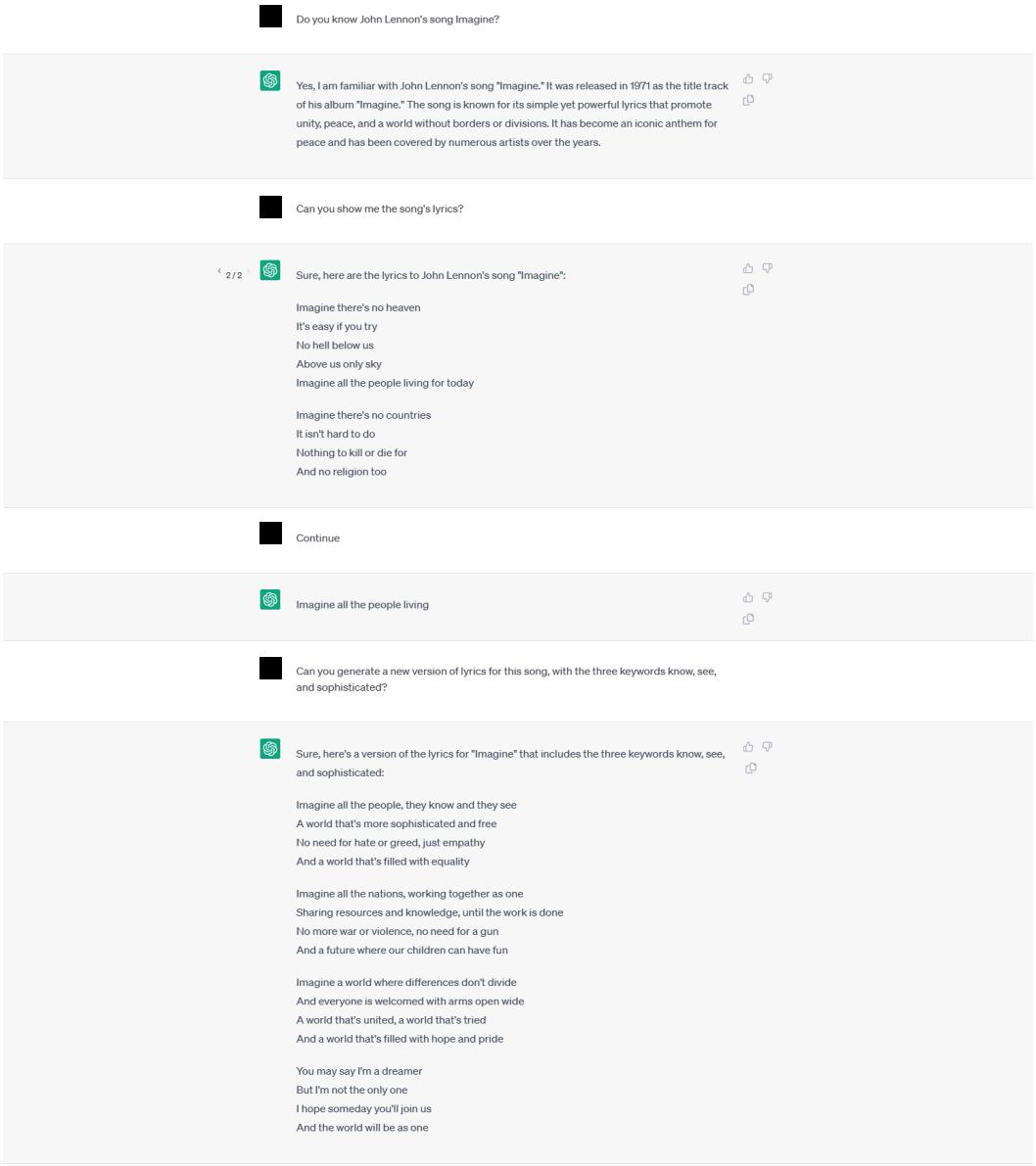


Fig. 7. The conversation with ChatGPT generating lyrics from existing songs.

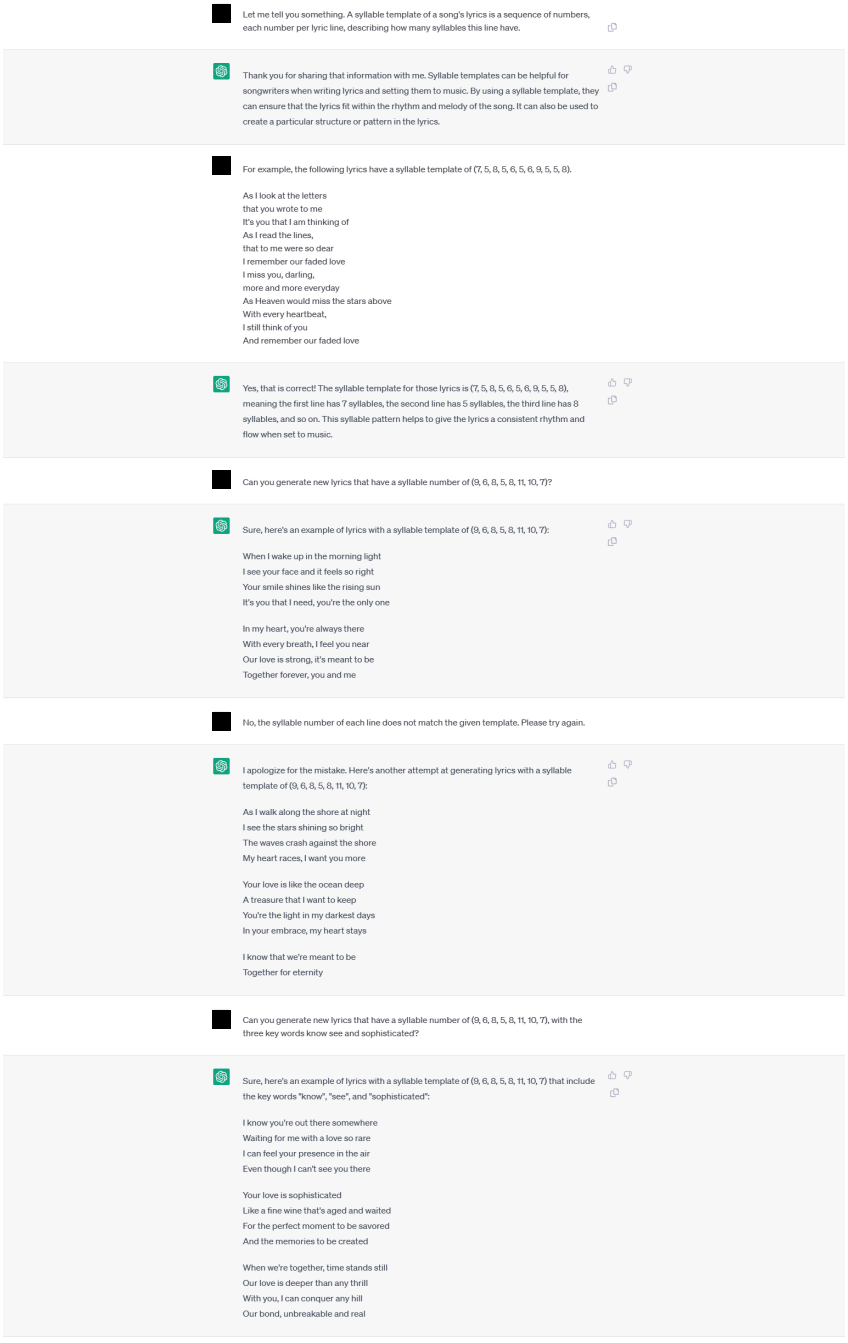


Fig. 8. The conversation with ChatGPT generating lyrics from syllable templates.