SMILE: Section-Aware Music-Driven Lyric Generation

Anonymous authors
Paper under double-blind review

Abstract

Music-driven lyric generation seeks to create lyrics that are semantically coherent across a song and aligned with its musical phrasing and section structure. A song's sections, such as verses, choruses, and bridges, organize where lyrical ideas develop, recur, and change, making them an effective interface between musical structure and lyric generation. However, existing approaches often rely on externally supplied lyric conditions or extract musical cues without inferring sections to organize the complete lyrics. To connect these levels of organization, we introduce SMILE (**S**ection-aware **M**usic-dr**I**ven **L**yric g**E**neration), a framework that generates full-song lyrics from a complete recording and aligned notes without user-provided themes, lyric templates, or drafts. First, a Music Section Analyzer identifies section boundaries and functions. Within these sections, a Lyric Line Planner assigns notes to lines and predicts their target syllable counts, providing a common plan for lyrical form and local musical conditioning. A Section Lyric Generator then follows this plan to produce all passage drafts, using the corresponding notes and shared semantic context from the recording. Finally, a Cross-Section Lyric Refiner uses the complete draft and optional musical relationships to coordinate passage development and repetition while preserving the line plans. In the evaluated section-level comparisons, SMILE outperforms state-of-the-art lyric-generation baselines (Sheng et al., 2021; Ding et al., 2025; Chae et al., 2025; You et al., 2025; Zhao et al., 2025) in music-lyric association, syllable-count accuracy, and section-format adherence.

1 Introduction

Music-driven lyric generation aims to compose lyrics that reflect a song's musical expression and fit its structure (You et al., 2025; Ou et al., 2023). Words should convey compatible ideas and emotions, while lines follow musical phrasing. For a complete song, however, matching individual lines is only part of the task: the generator must also decide how passages develop, return, and relate to one another.

Musical *sections*, such as verses, choruses, and bridges, provide a useful basis for this organization (Kim & Nam, 2023). Their boundaries define passages, their internal phrasing guides lyric lines, and their functions and recurring material suggest how ideas should develop across the song. For example, a new verse can advance a narrative while a returning chorus preserves a recognizable refrain. Therefore, section-aware generation must connect two decisions: how to organize lines within a passage and how to coordinate passages across the song (Figure 1c).

Existing methods obtain generation conditions from two main sources. Some methods rely on externally supplied content or structure, whereas others derive generation conditions from music. These sources may be combined within one method.

The first type uses externally prepared content or structure: titles, keywords, or drafts indicate what to write (Zhang et al., 2020; Tian et al., 2023; Zhao et al., 2025; Yoshida et al., 2026), while templates, rhyme, song forms, and syllable counts specify its form (Li et al., 2020; Liu et al., 2022; Chae et al., 2025). The musical input can itself carry external supervision through presegmented phrases or lyric-line intervals (Sheng et al., 2021; Ou et al., 2023; You et al., 2025). Likewise, UniLG accepts supplied rhythmic templates (Qian et al., 2023), and SongComposer uses functional section labels prepared during dataset construction (Ding et al., 2025). Such conditions help control lyrics, but

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

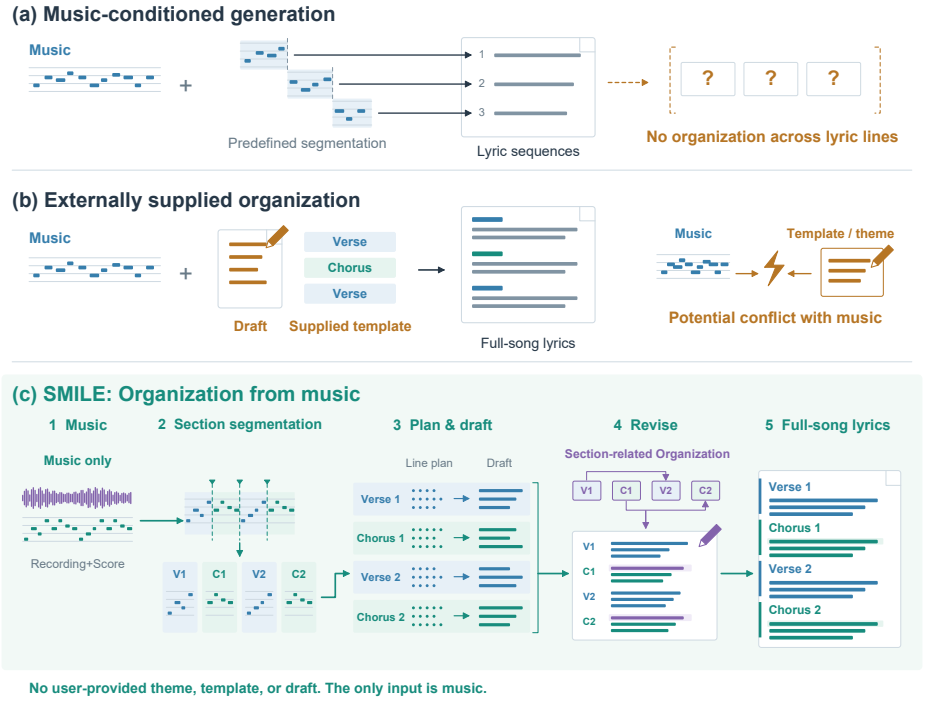

Figure 1: **Motivation for music-derived lyric organization.** (a) Predefined segments guide generation; the gap concerns explicit section planning, not whole-song semantic context (Sheng et al., 2021; You et al., 2025). (b) Supplied themes, templates, or drafts may conflict with music (Zhao et al., 2025). (c) SMILE infers sections, plans and drafts lyrics, then revises the song. Its inputs are a recording and score (aligned notes), without user-provided themes, templates, or drafts. Schematic illustration, not model outputs.

leave part of their organization to a user or dataset instead of deriving it from the input music during generation. However, supplied semantics can conflict with musical expression (You et al., 2025), and independently chosen templates may conflict with musical structure (Figure 1b).

Another type derives generation conditions from music. Audio-based systems associate short clips with lyric lines (Vechtomova et al., 2020; 2021); melody-based methods extract syllable, stress, alignment, or rhythm constraints (Tian et al., 2023; Liang et al., 2024; Ou et al., 2023; Qian et al., 2023). AI-Lyricist also detects musical repetition to reuse lyrics (Ma et al., 2021). Going beyond isolated clips, You et al. (2025) combine whole-song semantics, local multi-track notes, and predicted syllable counts, but generate lyrics sentence by sentence without explicitly inferring section boundaries and functions to organize the complete song (Figure 1a). Thus, the remaining organizational challenge is to make inferred sections jointly determine passage form, each line's musical context, and development across passages.

To address this challenge, we introduce **SMILE**, short for **S**ection-aware **M**usic-dr**I**ven **L**yric g**E**neration. It derives lyric organization from a complete recording and aligned notes, without user-provided themes, templates, or drafts. Specifically, the Music Section Analyzer (MSA) combines note detail, bar context, and temporal changes to identify boundaries and functions. The function of each section is inferred from its own music and the surrounding sections. Within each section, the Lyric Line Planner (LLP) assigns notes to lyric lines and predicts their target syllable counts. This plan serves two purposes: it specifies the passage's form and identifies the local music that should guide each line.

To realize this plan, the Section Lyric Generator (SLG) drafts each passage using shared audio-derived semantic context. Line-dependent attention connects each line to its assigned notes, linking planned form to musical detail. Finally, to coordinate passages across the song, the Cross-Section Lyric Refiner (CSLR) uses all completed drafts, section structure, and optional musical relationships

to make one song-level revision plan. It then edits passages in sequence using finalized earlier lyrics and unchanged later drafts, while preserving their line plans. In this way, the same inferred section organization connects music analysis, line-level realization, and whole-song revision.

Our contributions are:

- A section-aware framework that organizes full-song lyric generation from a recording and aligned notes, inferring boundaries and functions without user-provided themes, lyric templates, or drafts.
- A music section analyzer that combines note and bar information to infer passage boundaries and functions, providing a musical basis for lyric organization.
- A line-planning and generation design that connects lyrical form to local notes, followed by full-draft planning and section-wise revision to coordinate passages.
- An evaluation of generation and its organizing components. In the evaluated section-level comparisons, SMILE outperforms the compared methods in music-lyric association, syllable-count accuracy, and section-format adherence.

2 Related Work

2.1 Lyric Generation

Following the distinction above, controllable generation uses externally supplied content and form requirements. SongNet (Li et al., 2020), Youling (Zhang et al., 2020), and ChipSong (Liu et al., 2022) control format, content, length, or rhyme; Chae et al. (2025) extend these controls to full-song generation and infilling. Melody-conditioned methods add musical constraints: SongMASS (Sheng et al., 2021) learns melody-lyric alignment, Lyra (Tian et al., 2023) and Ou et al. (2023) use phrases and notes to guide syllable counts, and XAI-Lyricist (Liang et al., 2024) derives prosodic conditions from beat positions and durations. These sources can coexist: using music does not imply that input phrase or lyric-line boundaries are inferred by the generator.

At a broader scale, UniLG (Qian et al., 2023) supports externally supplied rhythmic templates as well as modes deriving them from beats, while SongComposer (Ding et al., 2025) uses functional labels prepared during dataset construction, rather than inferred by its lyric generator. REFFLY (Zhao et al., 2025) fits drafts to melodies under song-form instructions, and CoLyricist (Yoshida et al., 2026) incorporates melody fitting into interactive writing. Audio-based systems generate lines or suggestions from recorded or live music (Vechtomova et al., 2020; 2021); You et al. (2025) combine whole-song semantics, local notes, and predicted syllable counts. In contrast to externally organized form or global and line-level conditioning alone, SMILE infers section boundaries and functions from notes, uses them to plan and draft passages, and then revises the complete lyrics while preserving those line plans.

2.2 Music Structure Analysis

To derive this organization from music, music structure analysis supplies section boundaries and functions. All-In-One (Kim & Nam, 2023) jointly tracks beats and downbeats and predicts sections; SongFormer (Hao et al., 2025) combines temporal scales for boundary and function prediction. Complementing segmentation, Tralie & McFee (2019) recover repetition by fusing musical similarities, and AI-Lyricist (Ma et al., 2021) reuses lyrics in detected recurring passages. While these studies identify musical structure or reuse lyrics in recurring passages, SMILE uses inferred sections to organize both local writing and full-song revision: boundaries delimit planning, functions condition passages, and optional musical relations guide revision references. The revision plan specifies which recurring wording to preserve and where content should develop, while keeping each passage's line plan fixed.

3 METHOD

3.1 PROBLEM FORMULATION

Given a complete recording $\mathbf{x}$ and its time-aligned note sequence $\mathcal{N}$, music-driven lyric generation aims to produce lyrics that reflect the music and fit its structure. Section boundaries and functions are inferred from notes, without user-provided themes or lyric templates. The task is therefore section-aware at two levels: lines must fit their own musical passage, and passages must form an organized whole song.

Let $\mathcal{S} = \{(a_s, b_s, c_s)\}_{s=1}^{S}$ denote the inferred sections indexed in time order. Section s spans $[a_s, b_s)$ seconds and has function c_s, such as verse or chorus. The set $\mathcal{V} \subseteq \{1, \ldots, S\}$ contains sections requiring lyrics; instrumental and silent sections remain in the song structure but receive no lyrics. We assume lyric-bearing status is available through a separate interface: function labels alone do not determine it, since an intro or outro may contain lyrics.

As shown in Figure 2, SMILE first infers musical sections, then plans and generates a draft $Y_s^{(0)}$ for each $s \in \mathcal{V}$, and finally uses the full-song context and optional musical relations to revise these drafts into final lyrics $Y_s^{(1)}$. Both versions use the same line plan P_s^{line}, which specifies the line count, the notes assigned to each line, and its target syllable count. Whole-song audio supplies shared semantic context, while the note sequence provides local musical detail for individual lines.

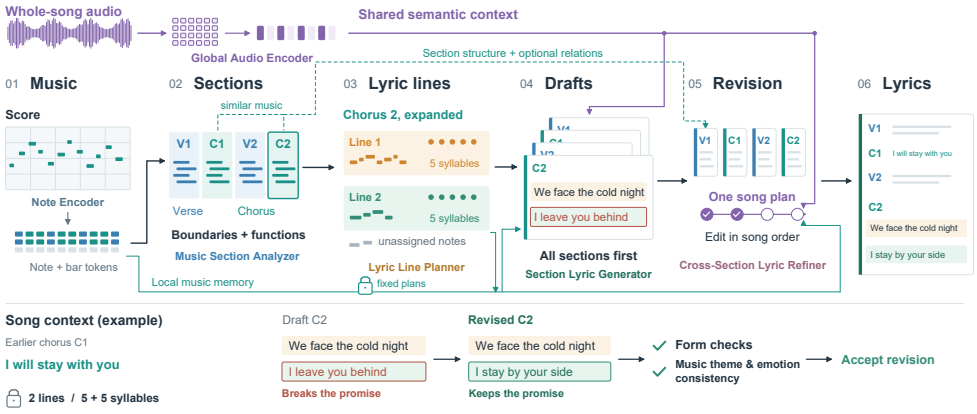

Figure 2: **Data flow of SMILE.** Whole-song audio supplies shared semantic context; encoded notes provide section structure and local musical detail. All section drafts precede one song-level revision plan. Editing uses finalized earlier text and unchanged later drafts while retaining line plans; musical relations are optional. The example preserves the earlier chorus's promise of companionship by replacing *I leave you behind* with *I stay by your side*, retaining two five-syllable lines. Form checks and content review determine whether a revision is accepted; otherwise its draft is retained. Notes, durations, counts, and lyrics are schematic, not model outputs.

3.2 MUSIC SECTION ANALYSIS

To establish the passages that organize lyric writing, MSA divides the song and identifies each section's function. It predicts boundaries and functions within one model, following the joint formulation used in music structure analysis (Wang et al., 2022; Hao et al., 2025). To make the notes available to this model, the Note Encoder converts each note's pitch, start time, duration, and other attributes into a learned vector, called a note token. Local attention lets each token incorporate information from nearby notes. The encoder also aggregates notes within each notated bar into a bar token, giving the analyzer both individual-note detail and a summary of the surrounding music.

With these complementary representations, the analyzer processes the song in short, fixed time steps. At each step, it combines nearby notes with bar context and bar-to-bar correspondences computed from the note sequence. A shared temporal network follows how this musical information changes

over time. From its state $\mathbf{h}_r$ at step r, a boundary head predicts

$$p_r^{\text{bd}} = \text{sigmoid}(\mathbf{w}_{\text{bd}}^\top \mathbf{h}_r + b_{\text{bd}}), \qquad \{(a_s, b_s)\}_{s=1}^{S} = \text{Split}(\text{Peaks}(\boldsymbol{p}^{\text{bd}})). \tag{1}$$

The sigmoid function converts the output to a probability. The learned parameters $\mathbf{w}_{\text{bd}}$ and b_{bd} map the state to a boundary probability. Peak selection retains likely boundaries, and splitting at their times, together with the song's endpoints, gives the section spans.

However, a boundary alone does not identify a passage's role. The same temporal states therefore provide initial function scores through classifier f_0. To assign a single function to a complete section, the model combines evidence within that passage with its position among the other passages. Attention summarizes the time steps I_s inside section s as $\mathbf{v}_s$; a bidirectional recurrent network then reads the section summaries in song order to obtain contextual state $\mathbf{u}_s$. These two summaries refine the initial scores:

$$\mathbf{o}_s^0 = \frac{1}{|I_s|} \sum_{r \in I_s} f_0(\mathbf{h}_r), \qquad c_s = \arg\max_{c \in \mathcal{C}} \left[\mathbf{o}_s^0 + f([\mathbf{v}_s; \mathbf{u}_s])\right]_c. \tag{2}$$

The set $\mathcal{C}$ contains the section functions, f learns the score correction, and brackets denote vector concatenation. Thus, a passage is classified using both its own music and the surrounding sections. Appendix A.1 gives the training objectives and the detailed readout. When relation analysis is enabled, a postprocessing step compares the music within the fixed section spans to obtain relations $\mathcal{R}$ between musically similar passages. Only groups whose member sections are mutually similar are retained, rather than treating all sections with the same function as repeats. These relations provide references, such as returning choruses, for later lyric refinement.

3.3 Lyric Line Planning and Generation

Once the sections are established, LLP converts their musical phrasing into a form the generator can follow: note-to-line assignments and target syllable counts. This connects lyrical form to musical phrasing, whose role in lyric generation has been studied by Watanabe et al. (2018). SLG then uses the resulting plan to write each line from its associated music and the shared semantic context.

Within section s, a Transformer processes the note tokens together with their start times, durations, positions within bars, corresponding bar tokens, and function c_s. To avoid splitting one musical onset across lyric lines, notes starting within 20 milliseconds of a group's first onset share a line-assignment decision. Anchoring this window to its first onset prevents a chain of nearby notes from merging into an arbitrarily long group. Each note is still encoded individually; only the scores for the assignment decision are averaged within each group. The model scores three choices: beginning a new line (B), continuing the current line (I), or leaving the notes unassigned (O). These are the begin-inside-outside (BIO) labels. With K_s groups and score $e_{s,k,b}$ for label b in group k, the best assignment and its line count are

$$\widehat{\boldsymbol{z}}_s = \arg\max_{\boldsymbol{z} \in \mathcal{A}_s} \sum_{k=1}^{K_s} e_{s,k,z_k}, \qquad \widehat{L}_s = \sum_{k=1}^{K_s} \mathbf{1}[\widehat{z}_{s,k} = B]. \tag{3}$$

The legal set $\mathcal{A}_s$ allows I only after B or I and requires at least one line, up to a preset maximum. Every B or I group belongs to the line indexed by the number of B labels up to and including that group; O groups remain unassigned. The notes inherit their group's assignment, producing $\widehat{\boldsymbol{g}}_s$. The line count therefore follows from the decoded BIO labels. For example, B-I-O-B-I gives two lines; the intervening O group supplies no lyric-line assignment.

For line l, the planner summarizes the assigned note tokens and their start times and durations as $\mathbf{r}_{s,l}$. A count predictor f_{syll} uses this summary to estimate the probability of each possible syllable count:

$$q_{s,l}^{\text{syll}}(n) = \text{softmax}(f_{\text{syll}}(\mathbf{r}_{s,l}))_n, \qquad \widehat{n}_{s,l} = \min\left\{ n : \sum_{v=1}^{n} q_{s,l}^{\text{syll}}(v) \geq \tfrac{1}{2} \right\}. \tag{4}$$

The selected count is the smallest integer whose cumulative probability reaches one half. The line plan $P_s^{\text{line}} = (c_s, \widehat{L}_s, \widehat{\boldsymbol{g}}_s, \widehat{\boldsymbol{n}}_s)$ now states how many lines to write, which notes inform them, and how many syllables each should contain. The training design learns note-to-line assignment before syllable prediction, using annotated lyric-line intervals and syllable counts as supervision. Appendix A.2 specifies the grouping rules, count summary, and training variants.

Given the completed line plan, SLG turns it into a draft. The Global Audio Encoder adopts the audio Q-Former of You et al. (2025), derived from BLIP-2's query-based connector (Li et al., 2023), to summarize the recording as $G = \mathcal{E}_A(\mathbf{x})$. This semantic context and selected note tokens introduce the music to the language model. During generation, the section function, line count, and target syllable count are also supplied at selected decoder layers, so that the format remains available as the text grows. Gated cross-attention, adapted from Flamingo (Alayrac et al., 2022), lets those layers read the note and bar tokens.

To help each line attend to its assigned notes, the attention score includes a line-dependent bias. Let $l(t)$ be the line being written at text position t, determined by the lines already completed, and let $\widetilde{g}_j$ record its musical source. Retained note tokens inherit line assignments from $\widehat{\boldsymbol{g}}_s$ through their source-note identifiers; unassigned notes and bar tokens receive separate markers. Then

$$A_{t,j} = \frac{\mathbf{q}_t^\top \mathbf{k}_j}{\sqrt{d_{\text{head}}}} + b_{\text{line}}(l(t), \widetilde{g}_j) + b_{\text{time}}(t, j) + \mu_j, \tag{5}$$

where $\mathbf{q}_t$ and $\mathbf{k}_j$ are the text query and music key, and d_{head} is their width. The line bias favors notes assigned to the current line while keeping other notes accessible; bar tokens carry no line-distance penalty. The mask μ_j excludes invalid positions. A time bias can additionally use annotated line intervals during training; automatic inference sets $b_{\text{time}} = 0$ and uses predicted note assignments. Alongside next-token learning, a ranking objective favors the target lyrics under matched notes over another passage's notes, keeping audio and target format controls fixed. The complete loss and bias definitions appear in Appendix A.3.

3.4 Cross-Section Lyric Refinement

Although the line plan organizes each passage, relationships between passages require whole-song context. CSLR therefore coordinates lyrical development and repetition after all section drafts are complete. Building on draft-based melody-constrained lyric editing (Zhao et al., 2025), it first plans changes across the song and then edits individual sections. Seeing the complete draft lets it consider both earlier and later passages before deciding what to change.

First, the Song Revision Planner receives the initial drafts $\mathbf{Y}^{(0)} = (Y_s^{(0)})_{s\in\mathcal{V}}$, section structure, optional musical relations, and line plans as X_P. It produces one plan Π that specifies how each section should develop, which passages to refer to, what wording to preserve, and which lines may change. The Section Lyric Editor then works in song order. Following the distinction between temporal and structural context in melody infilling (Tan et al., 2022), its input $X_{E,s}$ includes the section-specific instructions from Π, the unchanged line plan P_s^{line}, finalized earlier sections, unchanged copies of later drafts, and the references designated by the plan, along with the current draft and its musical conditions. The two steps are

$$\Pi = \mathcal{P}_\omega(X_P), \qquad \Delta_s = \mathcal{E}_\eta(X_{E,s}), \tag{6}$$

where ω and η are the planner and editor parameters. Each edit in Δ_s specifies an original line number and its replacement text. Adapting the separation of editable regions and preserved context in music proofreading (Ramoneda et al., 2024), the controller applies only these replacements and keeps the other lines intact. Every replacement retains its line's assigned notes and target syllable count.

To preserve musical form during revision, the Constraint Controller checks the complete revised section for line count, syllable counts, permitted edits, and preserved wording. An external content-review interface compares the candidate with the draft under the same plan and records judgments on thematic consistency, progression, and recurrence. This interface requires explicit judgments; the constraint controller does not itself score lyric quality. The section is accepted only if the checks pass and the content review judges that the requested content improves without regression on the other assessed requirements; otherwise its draft is retained. A final song-level review checks references to passages edited later and records unresolved issues. Appendix A.4 details this procedure. The planner and editor use separate LoRA adapters (Hu et al., 2022) and token-prediction objectives, keeping the Section Lyric Generator fixed.

Table 1: Generation under the declared conditions on 64 songs. Exact and Format are percentages. CLaMP3 uses Payload eligibility; its valid song count is in parentheses. SongComposer MAE covers 55 songs; other MAEs cover 64.

Method	Complete	Syl MAE↓	Exact↑	Format↑	CLaMP3↑
SongMASS	64/64	2.4848	13.84	7.81	0.05677 (63)
SongComposer	22/64	8.3654	5.58	1.56	0.03627 (30)
Back-S	64/64	0.4769	58.13	75.00	0.04173 (63)
You et al. (2025)	0/64	5.9387	6.36	0.00	0.04422 (63)
MASS→REFFLY	0/64	3.1667	16.03	0.00	0.05014 (59)
SMILE	63/64	**0.1612**	**88.92**	**96.88**	**0.07111 (63)**

4 Experiments

We evaluate lyrical form and music–lyric association, then examine section organization, line planning, and editing. Component studies and inference interventions are distinguished from complete-system ablations.

4.1 Experimental setup

TriDataset and Popular Hooks provide 43,582 aligned sections, including 23,784 from TriDataset; 30,795 train the generator and 1,707 support validation. The generation panel contains one section from each of 64 TriDataset English-test songs, totaling 302 reference lines. Selection precedes generation using input/scoring eligibility and fixed hash ordering (Appendix C.1).

SMILE is selected by validation NLL after domain adaptation, control instruction tuning, and Oracle training. It receives score-rendered section audio, notes, and reference-derived functions, line counts, syllable counts, and line intervals. Greedy generation uses no target words, hooks, or history; a controller blocks early EOS and excess lines. Reference intervals supply note assignments and temporal attention bias. Thus, this comparison evaluates section generation under supplied plans, before refinement.

Baselines are SongMASS (Sheng et al., 2021), SongComposer (Ding et al., 2025), Back-S (Chae et al., 2025), our local adaptation of You et al. (2025), and SongMASS followed by REFFLY (Zhao et al., 2025). Music-conditioned methods share selected passages and receive external line windows where required. Back-S uses reference counts/functions and neutral text without music; the You adaptation receives reference counts, and REFFLY additionally receives the frozen SongMASS draft (Appendix C).

Syllable MAE measures count deviation; Exact measures exact syllable-count agreement, counting missing, extra, or unscorable lines as failures. Section Format ± 1 requires verified completion, correct line count, and each admissible nonempty line within one target syllable. Completion requires successful, nonempty, untruncated output with known termination. Payload CLaMP3-SAAS (Wu et al., 2025) permits admissible partial lyrics; Strict requires completion. Songs receive equal weight, failures remain where defined, and coverage is reported. Paired intervals resample common songs. CLAP, PPL, and historical INT provide supplementary diagnostics (Appendix B).

4.2 Lyric generation performance

Table 1 shows that SMILE achieves 88.92% Exact and 96.88% Format, versus 58.13% and 75.00% for Back-S, supporting more accurate execution of line plans. SMILE completes 63/64 sections; SongMASS and Back-S complete all 64.

SMILE improves paired Payload CLaMP3 over Back-S by 0.029382 (95% CI: [0.018725, 0.040060]); the Strict difference is 0.027689 [0.017136, 0.037590] on 62 common songs. This supports stronger evaluator-measured association under the stated inputs. Because CLaMP3 deduplicates lines, it does not assess repetition or order.

Table 2: Component decoding comparisons. Left: variant C, two-seed means; count metrics use 1,308 records and boundary F1 uses 1,238. Right: eight sections and 41 target lines; failures remain in the denominators.

Line planning

Decoder	MAE↓	Exact%	F1%
MAP	2.4759	18.04	14.44
Count-aware	1.7821	19.99	22.24

Section editing

Decoder	Complete	Exact	Allowed
Ordinary	8/8	28/41	36/41
Hard	5/8	25/41	32/41
Search	7/8	29/41	37/41

4.3 Music section analysis

To test the structure supporting lyric organization, we examine function readout and boundary aggregation separately. On the original same-source development cohort, work-weighted readout reaches $84.31 \pm 1.04\%$ accuracy and $26.30 \pm 0.13\%$ macro-F1 across three seeds, versus 63.54% and 20.33% for the frozen base head.

With Context frame labels fixed, replacing note-model boundaries with SongFormer boundaries changes work-weighted accuracy by +3.416 points [0.898, 6.319] on 17 Harmonix works, but only +0.003 [-0.191, 0.196] on 410 TriDataset works. Boundary choice therefore helps on one cohort without consistently correcting function predictions. These partial-support cohorts have historical exposure; the intervention changes final majority-vote intervals without recomputing representations (Appendix D).

The section study separates the source of boundaries from the source of function labels. Fixed frame predictions ensure that the boundary comparison measures aggregation over musical passages, rather than changes to the underlying classifier. Work-level averaging prevents longer recordings from dominating the comparison, and paired resampling retains the dependence between two boundary choices applied to the same work. This makes the observed Harmonix gain interpretable as a benefit of the alternative passage boundaries under the fixed representation.

4.4 Lyric line planning

To connect predicted counts with note assignments, we compare legal BIO MAP with count-aware decoding: select the count-posterior mode, then find the best legal path with exactly that many line starts. Both arms use identical logits from six frozen checkpoints (variants A/B/C, two seeds). Line-count metrics cover 1,308 Popular Hooks inner-development records; internal-boundary F1 uses 1,238 eligible records and exact onset-group matching, excluding each sequence's first boundary.

For C, count-aware decoding reduces MAE from 2.4759 to 1.7821 and raises internal F1 from 14.44% to 22.24% (Table 2). The effect depends on the training variant (full A/B/C results in the appendix). This comparison evaluates count and boundary decisions: the study evaluates segmentation without syllable prediction or downstream generation (Appendix D.2).

Count-aware decoding enforces the same line count in the count decision and the decoded BIO path. This consistency matters because the resulting assignments determine which notes inform each lyric line. The comparison retains records whose reference count is unreachable under the grouping rules, rather than restricting evaluation to easy cases. Separate populations for count error and internal-boundary F1 distinguish passage size from the placement of line starts.

4.5 Cross-section lyric refinement

To test editing under line constraints, we reconstruct masked sections from neighboring reference lyrics and supplied counts on eight adapter-held-out development songs (41 target lines), using the same frozen editor and 512-token budget. Inputs contain neither music, generated drafts, nor a predicted revision plan.

Bounded search recovers two hard-decoding failures and attains 29/41 exact lines, versus 28/41 for ordinary decoding and 25/41 for hard constraints. Failures remain in the fixed denominator. Its song-macro Exact advantage over ordinary decoding is +10.42 points [-14.58, 39.58], so recovery does

Table 3: Single-bias inference interventions on 64 sections. Exact and Format are percentages; CLaMP3 uses 63 eligible songs. Audio, notes, plans, and routing remain available in every condition.

Biases	MAE↓	Exact↑	Format↑	Distinct-2	CLaMP3↑
Time + line	0.1612	88.92	96.88	0.3870	0.07111
Time only	0.1425	90.01	98.44	0.3860	0.07058
Line only	0.1677	88.27	96.88	0.3812	0.07273

not establish a reliable accuracy gain. The allowed interval $[\max(1, n-1), n]$ differs from generation Format ± 1 (Appendix D.3).

4.6 Musical grounding

Finally, we isolate the two attention biases while retaining audio, notes, Oracle plans, routing, and decoding. All settings complete 63/64 sections (Table 3); each intervention changes 7/64 outputs. Exact-score difference intervals all include zero. The small set of changed outputs shows stability to these additive biases; the single-bias comparisons expose the corresponding form, diversity, and association tradeoffs. These interventions assess a fixed generator, not separately trained architectures.

5 Conclusion

SMILE organizes music-driven lyric generation around inferred sections, connecting passage form to local notes and coordinating drafts through whole-song revision. The evaluated generator improves syllable accuracy, section-format adherence, and music–lyric association over the compared configurations. Controlled component studies further examine how section context, line decoding, and constrained editing support this organization.

6 Limitations

The generation comparison uses English sections, supplied structural controls, and score-rendered audio. Component studies use separate development sets; editor reconstruction conditions on neighboring reference lyrics. These settings delimit the evidence for the complete framework. Baseline inputs and decoding differ, and automatic music–text association does not establish listener preference or singability. CLaMP3 deduplicates lines, so repetition is evaluated separately.

Reproducibility Statement

The appendices specify evaluated configurations, metric denominators, and uncertainty estimates. Companion records retain protocols, outputs, per-example scores, and source hashes, distinguishing executed models from architectural designs. The component comparisons retain fixed evaluation populations and failure denominators; paired confidence intervals accompany the saved predictions. The records also document checkpoints, decoding settings, and scoring populations for each reported comparison.

AI Use Statement

AI assisted literature and code inspection, methodological feedback, writing, translation, and consistency checks. The authors are responsible for verifying the manuscript and its final disclosure.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning, 2022. URL https://arxiv.org/abs/2204.14198.

Yunkee Chae, Eunsik Shin, Suntae Hwang, Seungryeol Paik, and Kyogu Lee. Song Form-aware Full-Song Text-to-Lyrics Generation with Multi-Level Granularity Syllable Count Control. In *Interspeech 2025*, pp. 1273–1277, 2025. doi: 10.21437/Interspeech.2025-1247. URL https://www.isca-archive.org/interspeech_2025/chae25_interspeech.html.

Shuangrui Ding, Zihan Liu, Xiaoyi Dong, Pan Zhang, Rui Qian, Junhao Huang, Conghui He, Dahua Lin, and Jiaqi Wang. SongComposer: A large language model for lyric and melody generation in song composition. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7108–7127, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.352. URL https://aclanthology.org/2025.acl-long.352/.

Yixing Guan, Jinyu Zhao, Yiqin Qiu, Zheng Zhang, and Gus Xia. Melodic phrase segmentation by deep neural networks. arXiv:1811.05688, 2018. URL https://arxiv.org/abs/1811.05688. Preprint.

Chunbo Hao, Ruibin Yuan, Jixun Yao, Qixin Deng, Xinyi Bai, Yanbo Wang, Wei Xue, and Lei Xie. SongFormer: Scaling music structure analysis with heterogeneous supervision, 2025. URL https://arxiv.org/abs/2510.02797.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Shaojie Jiang, Ruqing Zhang, Svitlana Vakulenko, and Maarten de Rijke. A simple contrastive learning objective for alleviating neural text degeneration, 2022. URL https://arxiv.org/abs/2205.02517.

Taejun Kim and Juhan Nam. All-in-one metrical and functional structure analysis with neighborhood attentions on demixed audio, 2023. URL https://arxiv.org/abs/2307.16425.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023. URL https://proceedings.mlr.press/v202/li23q.html.

Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. Rigid formats controlled text generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 742–751, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.68. URL https://aclanthology.org/2020.acl-main.68/.

Qihao Liang, Xichu Ma, Finale Doshi-Velez, Brian Lim, and Ye Wang. XAI-Lyricist: Improving the singability of AI-generated lyrics with prosody explanations. In Kate Larson (ed.), *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pp. 7877–7885. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/872. URL https://doi.org/10.24963/ijcai.2024/872. Human-Centred AI.

Nayu Liu, Wenjing Han, Guangcan Liu, Da Peng, Ran Zhang, Xiaorui Wang, and Huabin Ruan. ChipSong: A controllable lyric generation system for Chinese popular song. In Ting-Hao 'Kenneth' Huang, Vipul Raheja, Dongyeop Kang, John Joon Young Chung, Daniel Gissin, Mina Lee, and Katy Ilonka Gero (eds.), *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pp. 85–95, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.in2writing-1.13. URL https://aclanthology.org/2022.in2writing-1.13/.

Xichu Ma, Ye Wang, Min-Yen Kan, and Wee Sun Lee. AI-Lyricist: Generating music and vocabulary constrained lyrics. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, pp. 1002–1011. ACM, October 2021. doi: 10.1145/3474085.3475502. URL http://dx.doi.org/10.1145/3474085.3475502.

Longshen Ou, Xichu Ma, and Ye Wang. Joint learning of wording and formatting for singable melody-to-lyric generation, 2023. URL https://arxiv.org/abs/2307.02146v3. Version 3, revised 12 December 2025.

Tao Qian, Fan Lou, Jiatong Shi, Yuning Wu, Shuai Guo, Xiang Yin, and Qin Jin. UniLG: A unified structure-aware framework for lyrics generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 983–1001, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.56. URL https://aclanthology.org/2023.acl-long.56/.

Pedro Ramoneda, Martín Rocamora, and Taketo Akama. Music proofreading with RefinPaint: Where and how to modify compositions given context. In *Proceedings of the 25th International Society for Music Information Retrieval Conference*, 2024. doi: 10.5281/zenodo.14877319. URL https://doi.org/10.5281/zenodo.14877319.

Zhonghao Sheng, Kaitao Song, Xu Tan, Yi Ren, Wei Ye, Shikun Zhang, and Tao Qin. SongMASS: Automatic song writing with pre-training and alignment constraint. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13798–13805, 2021. doi: 10.1609/aaai.v35i15.17626. URL https://arxiv.org/abs/2012.05168.

Chih-Pin Tan, Alvin W. Y. Su, and Yi-Hsuan Yang. Melody infilling with user-provided structural context. In *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, 2022. URL https://archives.ismir.net/ismir2022/paper/000100.pdf.

Yufei Tian, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Gunnar Sigurdsson, Chenyang Tao, Wenbo Zhao, Yiwen Chen, Tagyoung Chung, Jing Huang, and Nanyun Peng. Unsupervised melody-to-lyrics generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9235–9254, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.513. URL https://aclanthology.org/2023.acl-long.513/.

Christopher J. Tralie and Brian McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 201–205. IEEE, May 2019. doi: 10.1109/ICASSP.2019.8683492. URL https://doi.org/10.1109/ICASSP.2019.8683492.

Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. Generation of lyrics lines conditioned on music audio clips, 2020. URL https://arxiv.org/abs/2009.14375.

Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. LyricJam: A system for generating lyrics for live instrumental music, 2021. URL https://arxiv.org/abs/2106.01960.

Ju-Chiang Wang, Yun-Ning Hung, and Jordan B. L. Smith. To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions. arXiv:2205.14700, 2022. URL https://arxiv.org/abs/2205.14700. Preprint.

Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. 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)*, pp. 163–172. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1015.

Shangda Wu, Zhancheng Guo, Ruibin Yuan, Junyan Jiang, SeungHeon Doh, Gus Xia, Juhan Nam, Xiaobing Li, Feng Yu, and Maosong Sun. CLaMP 3: Universal music information retrieval across unaligned modalities and unseen languages. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova,

and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2605–2625, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.133. URL https://aclanthology.org/2025.findings-acl.133/.

Masahiro Yoshida, Bingxuan Li, Songyan Zhao, Qinyi Zhou, Shiwei Hu, Xiang Anthony Chen, and Nanyun Peng. CoLyricist: Enhancing lyric writing with AI through workflow-aligned support, 2026. URL https://arxiv.org/abs/2602.22606.

Mu You, Fang Zhang, Shuai Zhang, and Linli Xu. S^{2}MILE: Semantic-and-structure-aware music-driven lyric generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(21): 22208–22217, 2025. doi: 10.1609/aaai.v39i21.34375. URL https://doi.org/10.1609/aaai.v39i21.34375.

Rongsheng Zhang, Xiaoxi Mao, Le Li, Lin Jiang, Lin Chen, Zhiwei Hu, Yadong Xi, Changjie Fan, and Minlie Huang. Youling: an AI-assisted lyrics creation system. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 85–91, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.12. URL https://aclanthology.org/2020.emnlp-demos.12/.

Songyan Zhao, Bingxuan Li, Yufei Tian, and Nanyun Peng. REFFLY: Melody-constrained lyrics editing model. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 11295–11315, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.564. URL https://aclanthology.org/2025.naacl-long.564/.

A Implementation Details

A.1 Note encoding and music section analysis

The Note Encoder converts each input note into a vector and provides summaries at several musical scales:

$$(H^{\text{note}}, H^{\text{beat}}, H^{\text{bar}}, H^{\text{role-bar}}, \mathcal{T}) = \mathcal{E}_N(\mathcal{N}). \tag{7}$$

Each row of H^{note} is a note token. The other tensors contain tokens for beats, bars, and musical roles within bars; $\mathcal{T}$ records their times, source-note identifiers, and valid positions. Beats are the pulse units specified by the score, while bars follow its notated boundaries. The encoder groups notes directly by bar. In parallel, it groups notes by beat and by their estimated musical roles within each bar. Musical roles include melody, bass, harmony, and drums, and are distinct from section functions such as verse and chorus. The bar branch combines the note and role summaries with metrical information, then models relationships between bars. It does not require notes to pass through beat tokens before forming bar tokens.

Note tokens encode pitch, onset, duration, velocity, track, instrument, and metrical attributes. Local attention uses relative musical positions. The inspected checkpoint has internal width 256, four local note-attention layers with eight heads and a 256-note window, and six bar-attention layers with eight heads. Its output width is 512. Long inputs use windows of up to 224 bars and 10,240 notes, with 16 overlapping bars; shared source coordinates reconcile the overlapping outputs. Two context layers with kernel size five add longer-range bar information while retaining the individual note tokens. These are saved checkpoint settings, not source defaults.

Pretraining uses masked reconstruction of pitch, duration, and velocity, musical-role supervision, and additional context-corruption and time-warp objectives where valid targets are available. The encoder is frozen for the downstream modules. The task takes a time-aligned note sequence as input and does not perform transcription. Whole-song analysis and section-level generation encode their respective input ranges while retaining the correspondence to source notes.

The Music Section Analyzer evaluates the music at 8.333 time steps per second. The center of step r is $(r + 0.5)/f_{\text{step}}$ seconds for zero-based r, with $f_{\text{step}} = 8.333$. It attends to notes beginning within

0.25 seconds of that center, using the relevant bar context as the query. Bar correspondences describe recurring musical material before section boundaries are predicted. The resulting note, bar, time, and recurrence information enters a shared temporal network that produces the 128-dimensional states $\mathbf{h}_r$. Its boundary and initial function heads use these same states.

The first training stage combines class-weighted cross-entropy on valid function labels with positive-weighted binary cross-entropy on valid boundary targets. The boundary targets are Gaussian-smoothed with a scale of two-thirds of a time step, and the positive-class weight is 50. Decoding retains peaks of height at least 0.5 separated by at least 33 steps, approximately 3.96 seconds. These are minimum-spacing constraints; section boundaries need not be evenly spaced.

After this stage, the section-function readout is learned with predicted boundaries, temporal states, and the base function classifier fixed. A separately fitted linear frame classifier supplies f_0. Attention selects relevant time steps within each predicted section, and a bidirectional gated recurrent unit (BiGRU) supplies context from preceding and following sections:

$$\begin{aligned} \alpha_{s,r} &= \mathrm{softmax}_{r \in I_s}(g_{\text{att}}(\mathbf{h}_r)), \quad \mathbf{v}_s = \sum_{r \in I_s} \alpha_{s,r} \mathbf{h}_r, \\ (\mathbf{u}_1, \ldots, \mathbf{u}_S) &= \mathrm{BiGRU}(\mathbf{v}_1, \ldots, \mathbf{v}_S), \qquad \mathbf{o}_s = \mathbf{o}_s^0 + f([\mathbf{v}_s; \mathbf{u}_s]). \end{aligned} \tag{8}$$

The attention scorer g_{att} has widths $128 \to 32 \to 1$. The BiGRU has one layer and 32 units per direction; the correction network has widths $192 \to 64 \to 7$ and a zero-initialized final layer. Thus $\mathbf{v}_s$ has 128 entries and $\mathbf{u}_s$ has 64. Equation (2) selects the largest corrected score. Fixed-input representation probes and later label-sequence postprocessing are separate variants; this formulation describes the contextual readout on the original temporal states.

A predicted section can overlap more than one annotated function. Its supervision retains the amount of labeled time belonging to each class, rather than assigning the entire span the majority label. Let m_r^{ann} indicate whether step r has a valid annotation and y_r^{func} be that label. For class weight w_c, the labeled duration and per-record loss are

$$\begin{aligned} d_{s,c} &= \frac{1}{f_{\text{step}}} \sum_{r \in I_s} \mathbf{1}[m_r^{\text{ann}} = 1,\ y_r^{\text{func}} = c], \\ \mathcal{L}_{\text{function}} &= -\frac{\sum_{s,c} w_c d_{s,c} \log \mathrm{softmax}(\mathbf{o}_s)_c}{\sum_{s,c} w_c d_{s,c}}. \end{aligned} \tag{9}$$

Only labeled time contributes to the loss; pooling still reads all available input steps in the predicted section. The class order is intro, verse, chorus, bridge, instrumental, outro, and silence, with inherited weights approximately $(0.999597, 0.5, 0.5, 1.019356, 1.612331, 1.513579, 1.0)$. The weights come from source-weighted training frequencies, using inverse square roots and clipping to $[0.5, 4]$ after normalization; a class with zero frequency receives weight one. Losses are also balanced by musical work when aggregating records. These duration weights apply to the section readout stage, not the initial boundary loss.

Musical relations $\mathcal{R}$ are computed separately on fixed section spans. The available relation interface compares musical views, obtains a pairwise section-similarity matrix, and groups sections only when every pair within a group passes the similarity threshold. Its current provider uses the existing V2 section scorer and complete-link grouping; it does not change the predicted spans. This optional post-processing is distinct from both the section-function classifier and the bar correspondences supplied before segmentation. Its activation and threshold configuration must be specified for an execution; the latest function-classification experiments do not by themselves establish an integrated relation-enabled run. Shared function labels alone do not establish musical repetition.

A.2 Lyric line planning

The planner processes every note token before grouping its label scores. Notes beginning within 20 milliseconds of the first onset in a group share a decision. Measuring from the first onset prevents a chain of nearby starts from forming an arbitrarily long group. If $\mathcal{G}_{s,k}$ contains the notes in group k, and $a_{s,i,b}$ is note i's score for label $b \in \{B, I, O\}$, then

$$e_{s,k,b} = \frac{1}{|\mathcal{G}_{s,k}|} \sum_{i \in \mathcal{G}_{s,k}} a_{s,i,b}, \qquad S_s(\boldsymbol{z}) = \sum_{k=1}^{K_s} e_{s,k,z_k}. \tag{10}$$

The score S_s sums the decisions over the section. The constraints in $\mathcal{A}_s$ forbid an initial I, an O-to-I transition, and an all-O sequence; they permit consecutive B labels and at most $L_{\max} = 16$ line starts. Transition constraints are fixed, with no learned transition scores. Equation (3) selects the highest-scoring legal sequence. Each assigned note receives the cumulative B count including its own group, while an O note receives zero in the planner output. Decoding ties use the lexicographically smallest complete path under $O < B < I$.

Annotated lyric-line intervals provide grouping supervision: a note's start time determines its line under left-closed, right-open intervals. A group that spans conflicting labels cannot supply an unambiguous target. Invalid paths and counts beyond the supported range are excluded, not clipped. This is supervision from lyric timing, rather than an annotation of which syllable is sung on each note. During supervised planner training, section functions are annotated; automatic inference uses predicted functions.

The current training design separates grouping from syllable prediction. The segmentation study compares two sequence losses, each with three boundary-weighting choices. For target sequence $\boldsymbol{z}_s^*$, the sequence term is

$$\mathcal{L}_{\text{sequence}} = \frac{\log \sum_{\boldsymbol{z} \in \mathcal{A}_s} \exp(S_s(\boldsymbol{z}) + D(\boldsymbol{z}, \boldsymbol{z}_s^*)) - S_s(\boldsymbol{z}_s^*)}{K_s}. \tag{11}$$

The ordinary conditional-random-field loss uses $D = 0$. The softmax-margin variant uses a non-negative label cost D: missing an internal target B costs two and adding B at a non-B position costs one; the first target B is excluded from this cost. Both losses use the same constrained decoding at inference.

The three boundary choices are no auxiliary term, balanced binary cross-entropy for line starts, and the same term with greater weight near a true internal boundary. The auxiliary loss excludes the first target B; later B labels are positives and the remaining I/O labels are negatives. The weighted sum is divided by the complete group count K_s. This comparison is motivated by work on sparse phrase-boundary supervision (Guan et al., 2018). The nearby-boundary variant multiplies the hard-label loss by $1 + 2\exp(-d^2/8)$, where d is distance in onset groups to the nearest internal boundary. It does not replace the labels with decay curves. The auxiliary coefficient is zero when absent and one when enabled. These six combinations are experimental variants; no one variant is implicitly selected as the final model.

Once notes have been assigned, the count summary $\mathbf{r}_{s,l}$ concatenates their token means and standard deviations, mean start times and durations, and transformed note and group counts. The count transformation is $\log(1+x)$. The distribution in Equation (4) covers $n \in \{1, \ldots, N_{\max}\}$, with $N_{\max} = 32$. The existing count interface projects 512-dimensional inputs to width 128, uses two Transformer layers with four heads, feed-forward width 512 and dropout 0.1, and has a count head with hidden width 256. The hard-assignment count readout and cumulative-probability rule are implemented in the preceding complete planner. The later segmentation study trains only grouping; matching predicted lines to annotated lines and the final second-stage training recipe have not been fixed. An earlier joint variant used soft assignments and an expected-line-count auxiliary loss, which are not objectives of this staged segmentation study.

A.3 Generation conditions and matched-pair training

Assigned note tokens use positive line identifiers, unassigned events use -1, and bar tokens use -2. With learned scale $\gamma \geq 0$, the line-distance bias is

$$b_{\text{line}}(l, g) = \begin{cases} -\gamma |l - g|, & l > 0,\ g > 0, \\ -\gamma \max(L_s^{\text{cond}} - 1, 1), & l > 0,\ g = -1, \\ 0, & g = -2 \text{ or } l \leq 0, \end{cases} \tag{12}$$

For valid supplied controls, L_s^{cond} is the supervised count during training and the decoded count $\widehat{L}_s$ during automatic inference. Training can also use $b_{\text{time}}(t, j) = -((\tau_j - \bar{\tau}_{l(t)})/\sigma_{\text{time}})^2$, where τ_j is normalized event/bar time, $\bar{\tau}_{l(t)}$ is the supervision-derived line center, and $\sigma_{\text{time}} > 0$ is a learned per-head scale. Invalid indices or line boundaries give zero time bias; nonpositive line indices also

give zero line bias. The validity mask μ_j is zero at valid positions and excludes invalid ones from the attention softmax. Softmax-weighted value states are normalized, projected, scaled, and gated before addition to the decoder. Structural conditioning is applied to embeddings as well as the selected decoder layers.

Generator training uses supervised format controls and assigns events by their midpoints within annotated lyric-line intervals, subject to condition masking. This differs from the onset-based grouping supervision of the Lyric Line Planner. With matched conditions X_s^+, note-mismatched conditions X_s^-, target tokens $x_{s,t}^*$, valid target positions $\mathcal{U}_s$, and parameters θ, the per-record likelihoods used in the ranking term are

$$\ell_s^{\pm} = -\frac{1}{|\mathcal{U}_s|} \sum_{t \in \mathcal{U}_s} \log p_\theta(x_{s,t}^* \mid x_{s,<t}^*, X_s^{\pm}). \tag{13}$$

With $\mathcal{L}_{\text{LM}} = \ell_s^+$, ranking margin m, and positive-only auxiliary count and syllable cross-entropies $\mathcal{L}_{\text{note-aux}}$, the objective is

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{LM}} + \lambda_{\text{rank}} \max(0, m + \ell_s^+ - \ell_s^-) + \lambda_{\text{aux}} \mathcal{L}_{\text{note-aux}}. \tag{14}$$

The coefficients λ_{rank} and λ_{aux} weight the additional losses.

The auxiliary count and syllable heads are separate from the Lyric Line Planner. They receive matched-note supervision only, and their losses are not repeated in ranking forwards. Missing negative pairs give zero ranking contribution.

The inspected 36-layer generator uses structural and music-conditioning layers at zero-based indices 7, 14, 21, 28, and 35. Music cross-attention has width 1024 and 16 heads, while the structural state has width 512. The total retained event budget is 384; bar tokens are additional context and do not count toward this event budget. Event selection preserves source identities and line assignments. The line-distance scale starts at 0.5 and is constrained to be nonnegative. The per-head time scale starts at 0.5, uses a softplus parameterization, and is lower-bounded by 10^{-4} during computation.

Time positions use event midpoints and bar starts. Each is converted to a position relative to its own musical section, divided by the section duration with a 10^{-6}-second lower bound, and clipped to $[0, 1]$. Supervision-derived normalized line boundaries define the query's line center. Nonpositive or out-of-range line indices and invalid endpoints give zero temporal bias. Automatic inference uses predicted event membership and disables this supervision-derived temporal term. Bar states receive valid temporal bias but no line-distance penalty. The count L_s^{cond} in Equation (12) is read from the supplied line-count distribution; it is not necessarily a prediction from the Lyric Line Planner during training.

The generator's categorical format interface has capacity for 64 lines and 64 syllables, while the Lyric Line Planner supports 16 lines and 32 syllables. Encoding or padding a predicted plan preserves this smaller support. A further structural feature categorizes the number of distinct melodic onsets assigned to a line. This count merges onsets within 10 milliseconds of the first onset in a group; it is distinct from the 20-millisecond grouping used to learn lyric lines. Function labels are semantic categories, and their integer encodings are specific to each module.

Equation (14) uses ranking margin $m = 0.10$, ranking weight $\lambda_{\text{rank}} = 0.30$, and auxiliary weight $\lambda_{\text{aux}} = 0.10$. Training negatives replace note tokens together with masks, timing, event assignments, roles, and melody-count categories. Audio, target lyrics, token-to-line indices, and target function and lyric-format controls remain fixed. A donor's event assignments are retained, rather than transferred to the target events. Cross-song donors require the same function and different known artists; within-song selection uses another record and prefers matching functions. Candidate selection then prioritizes bar-count and duration similarity. It does not require identical line counts or a fixed 25-percent duration window. Unavailable negatives give zero ranking contribution, while positive language and auxiliary supervision remain available. The joint audio-note intervention in the experiments is a separate evaluation protocol.

A.4 Language adaptation and refinement training

Write the fixed initial drafts as $\mathbf{Y}^{(0)} = (Y_s^{(0)})_{s\in\mathcal{V}}$. The Song Revision Planner receives $X_P = (\mathbf{Y}^{(0)}, \mathcal{S}, \mathcal{R}, \{P_s^{\text{line}}\}_{s\in\mathcal{V}})$ as text and structured summaries and predicts $\Pi = \mathcal{P}_\omega(X_P)$. For target plan tokens $\Pi^* = (\pi_t^*)_t$ of length T_P, its objective is

$$\mathcal{L}_{\text{plan}} = -\sum_{t=1}^{T_P} \log p_\omega(\pi_t^* \mid \pi_{<t}^*, X_P). \tag{15}$$

For section s, the editor input $X_{E,s}$ contains its draft and plan, finalized earlier sections, unchanged copies of later drafts, designated references, and the original musical conditions and line plan. Ordinary references follow these version choices; protected wording is pinned to fixed spans. The Section Lyric Editor predicts $\Delta_s = \mathcal{E}_\eta(X_{E,s})$, consisting of original line identifiers and replacements. For target edit tokens $\Delta_s^* = (\xi_{s,t}^*)_t$ of length $T_{E,s}$,

$$\mathcal{L}_{\text{edit}} = -\sum_{t=1}^{T_{E,s}} \log p_\eta(\xi_{s,t}^* \mid \xi_{s,<t}^*, X_{E,s}). \tag{16}$$

The two modules have separate parameters ω and η. Both objectives sum over target output tokens, including structural tokens and explicit empty edits, while excluding prompts and padding.

The Constraint Controller reconstructs the complete candidate $\widetilde{Y}_s$ by applying the line replacements. With $\kappa_s = 1$ only when both formal checks and content review accept the candidate, the final section is

$$\widetilde{Y}_s = \text{Apply}(Y_s^{(0)}, \Delta_s), \qquad Y_s^{(1)} = \begin{cases} \widetilde{Y}_s, & \kappa_s = 1, \\ Y_s^{(0)}, & \kappa_s = 0. \end{cases} \tag{17}$$

For each replacement $y_{s,l}$, the syllable condition is

$$\max(1, \widehat{n}_{s,l} - \epsilon_{\text{syll}}) \le \text{Syll}(y_{s,l}) \le \widehat{n}_{s,l}, \tag{18}$$

Here Syll counts linguistic syllables, and the preset integer $\epsilon_{\text{syll}} \ge 0$ allows underfilling only, never exceeding the target count. Zero tolerance requires the exact count. When enabled for verified vocal-melody units, an order-preserving one-to-many assignment also checks alignment. All applicable constraints and content acceptance must pass for the complete candidate section to be committed.

The inherited domain-adaptation recipe uses contrastive token learning (Jiang et al., 2022):

$$\mathcal{L}_{\text{DAPT}} = \mathcal{L}_{\text{CE}} + \frac{\alpha_{\text{CT}}}{N} \sum_{t\in\mathcal{U}^{\text{text}}} \log\left(1 + \sum_{v\in C_t} \exp[z_t(v) - z_t(y_t)]\right). \tag{19}$$

Here z_t denotes next-token logits, y_t is the target token, $\mathcal{U}^{\text{text}}$ is the set of supervised positions, $N = |\mathcal{U}^{\text{text}}|$, and C_t contains distinct tokens from a bounded preceding-context window. The inspected recipe uses a 128-token window, excludes the current target and EOS, does not cross EOS boundaries, and sets $\alpha_{\text{CT}} = 0.75$. This deduplicated candidate set is an implementation choice; the original paper uses a multiset. This token-learning objective is separate from musical recurrence and cross-section revision.

The Song Revision Planner and Section Lyric Editor have separate adapters, supervision sequences, and objectives, as defined above. The refinement training design specifies adapter rank $r_{\text{LoRA}} = 8$ and scaling parameter $\alpha_{\text{LoRA}} = 16$ on query, key, value, and output projections, with zero adapter dropout. The generator remains fixed. The Section Lyric Editor first learns from supplied plans, and the Song Revision Planner subsequently learns to predict the plans. These design settings are recorded separately from the evaluated section-generation checkpoints; final optimizer, target-mask, batch loss normalization, context, and decoding settings are tied to the respective refinement runs.

The Constraint Controller treats a section's edits as one candidate. It verifies line count, edit scope, protected spans, and target syllable counts, and compares the candidate and draft under the same revision plan. Exact syllable checking uses $\epsilon_{\text{syll}} = 0$; any relaxed tolerance and the optional melody-alignment mode are fixed before decoding. A rejected candidate leaves the complete draft unchanged, and any remaining violations stay visible in the final audit. This controller is distinct from the format controller used by the frozen Oracle section-generation evaluations.

B Evaluation Definitions and Text Diagnostics

B.1 Line slots, completion, and syllable counts

For section s, let L_s and $\widehat{L}_s$ be the reference and generated line counts, and let $J_s = \max(L_s, \widehat{L}_s)$. Lines are compared by their existing order. A missing or blank generated line contributes zero syllables and retains its slot; an extra generated line is compared against a zero reference count. Phonology failures and unsupported scripts can make an absolute error undefined. Let $\mathcal{C}_s$ be the set of slots with computable errors $e_{s,l} = |n_{s,l} - \widehat{n}_{s,l}|$. Then

$$\text{MAE}_s = \frac{\sum_{l \in \mathcal{C}_s} e_{s,l}}{|\mathcal{C}_s|}, \qquad \text{Acc}_{s,d} = \frac{1}{J_s} \sum_{l=1}^{J_s} \mathbf{1}[\text{valid nonempty paired lines and } e_{s,l} \le d]. \tag{20}$$

An unpaired or unscorable slot cannot count as an accuracy success. We retain it in the accuracy denominator. A section with no computable errors has undefined MAE, and the macro-average reports its reduced coverage. SongComposer has 338 computable slots out of 420 reference/generated slots and a computable section MAE on 55 songs. Its mean per-song slot coverage is 0.8510; this differs from the pooled fraction 338/420. All main-table accuracies still average all 64 songs.

Line EM is $\mathbf{1}[L_s = \widehat{L}_s]$. Line-count MAE is $|L_s - \widehat{L}_s|$. Verified completion is a separate indicator requiring successful status, nonempty text, known termination, and no truncation. Section Format at tolerance one additionally requires $L_s = \widehat{L}_s$ and an admissible, nonempty aligned prediction with $e_{s,l} \le 1$ for every target line (equivalently, $\text{Acc}_{s,1} = 1$). Complete output can therefore fail the format criterion, and an accurately counted partial output is not complete. The main panel contains one section per song. In the extension, base syllable errors and accuracy counts are pooled over each song's line slots before averaging songs; line-count metrics, completion, and section-format rates first average the sections within each song.

The frozen English phonology pipeline uses CMU pronunciations, selecting the modal syllable count across pronunciations and the smaller count in a tie; out-of-vocabulary words use a vowel-group heuristic with retained warnings. Syllable agreement is distinct from stress alignment, rhythm, or singability. The source records retain per-line counts, failure categories, missing/extra slots, and micro totals alongside the reported song-macro metrics.

B.2 Supplementary length and text measures

SDD reuses the absolute syllable-count error above. SDD-S normalizes each eligible slot's absolute error by its positive reference syllable count. SDD-N divides the same reference-versus-generated error by the number of melody-note onsets in the shared half-open Oracle line window; it is not duration normalization or error against the note count. Coverage is reported separately when a reference count or note-count denominator is unavailable. SCD is defined only for positive reference and generated syllable counts:

$$\text{SCD}(n, \widehat{n}) = \tfrac{1}{2} \left(\frac{|n - \widehat{n}|}{n} + \frac{|n - \widehat{n}|}{\widehat{n}} \right). \tag{21}$$

SCErr is the exact-count mismatch rate on the fixed line-slot denominator. SDD, SDD-S, SDD-N, SCD, and SCErr first average eligible slots within sections, then sections within songs, and finally songs equally. SDD equals base syllable MAE on this one-section-per-song panel, but not necessarily on a multi-section panel. These diagnostics complement completion and conservative accuracy.

Within-section Distinct-k is the number of unique word k-grams divided by the total number of such k-grams. N-grams are constructed within lines, without crossing a line boundary. We report the

song-macro average of section ratios. Rep-4 equals one minus Distinct-4. If no k-gram exists, that metric is undefined rather than zero. Duplicate-line rate measures the proportion of nonempty token-normalized lines beyond their unique forms. ROUGE-2 is a multiset F1 of normalized Latin-word bigrams against the reference lyric, with section newlines converted to spaces and no stemming. Unlike diversity, the fixed-panel ROUGE score retains empty outputs as zero-overlap cases.

B.3 Eligibility for external evaluators

Payload analysis requires admissible nonempty text, but permits generation truncation. Strict analysis additionally requires the declared English lyric-bearing target, successful generation, known normal termination, and no truncation; it also checks the entire text for script and tokenizer admissibility. This full-text eligibility rule is not identical to the format scorer's completion flag, and neither requires perfect syllable accuracy. Each evaluator then adds its own audio, context-length, or token-availability rules. Consequently, a single global valid-sample count cannot be reused for all metrics.

CLaMP3 uses frozen official SAAS weights, native audio processing, line deduplication, and complete 128-token text windows. Its text preprocessing may also reorder deduplicated lines; the frozen hash seed is retained. The metric therefore does not evaluate line order or repetition. The 0.29-second audio exclusion is independent of the one truncated SMILE section: 63 verified complete text outputs intersect 63 eligible audios to yield 62 Strict CLaMP3 pairs. A common-set paired comparison uses exactly the same songs for both models, not the difference between means over different eligible sets.

B.4 Language-model and audio-text diagnostics

GPT-2 Large perplexity is $\exp(\overline{\mathrm{NLL}})$, with song-macro token NLL, not the average of exponentiated section scores. The historical line-ending measure is $\mathrm{INT} = 2^{-\overline{\log p_{\mathrm{end}}}}$: it intentionally retains the original mixed-logarithm implementation, where stored log probabilities use natural logs. One trailing punctuation/EOS token is removed from each eligible line; p_{end} is the maximum next-token probability over punctuation and EOS. Empty or overlong lines are unscored; valid extra lines remain eligible. Lower PPL or INT can favor repetitive text and does not establish overall lyric quality.

Public CLAP uses a separate audio-text model and a 77-token text limit, with truncation recorded; its values are not directly comparable with CLaMP3. The current SMILE run has PPL 6.4341, INT 7.5421, and CLAP 0.02104 on 64 songs. Two texts are truncated for CLAP. Reference-text BERTScore and CoSENT, where reported, measure similarity to reference lyrics rather than music.

B.5 Uncertainty

Intervals are nominal 95% song-paired bootstrap intervals, without correction for multiple comparisons. The new bias and planner analyses use 2,000 resamples. They describe the frozen checkpoints and evaluated populations, not training-seed or evaluator-model uncertainty. Metrics with different eligibility use their own common paired sets; missing audio scores are not replaced by zero.

C Evaluated Configurations and Supplementary Results

C.1 Corpus construction and panel selection

The retained section corpus combines 23,784 sections and 115,961 lyric lines from TriDataset with 19,798 sections and 80,330 lines from Popular Hooks. It contains 30,795 training sections, 1,707 validation sections, 1,603 test sections, and 9,477 artist-disjoint test sections. The construction report counts 4,230 source packages, not unique songs, and records 52 excluded packages with nonmonotonic time mappings. Each section record retains aligned note events, original lyric text, line intervals, and available structural controls; unavailable conditions are recorded explicitly.

The generation panel is selected from 964 TriDataset English test records. Input and reference-count scoring eligibility retain 822 records; a fixed seeded hash ordering selects one section per song before generation, giving 64 songs and 302 target lines. The recorded overlap with training song groups is zero; this check concerns the assembled training corpus, not external model pretraining.

Table 4: Inputs supplied to the evaluated configurations.

Configuration	Music / structure	Text
SongMASS	Melody and supplied line windows	No target words
SongComposer	Melody and supplied line windows	No target words
Back-S (Chae et al., 2025)	No music; reference counts and function	Neutral sentence
You et al. (2025)	Section audio, notes, supplied windows and refiner counts	Native draft, no target words
SMILE	Section audio, notes; Oracle function, lines, counts and timing	No themes, hooks or reference passages
MASS→REFFLY	Music-derived constraints and supplied windows	Neutral title and frozen MASS draft

The companion provenance inventory retains the construction report and input preflight, including their source hashes.

C.2 Compared inputs and decoding

The compared configurations receive different inputs and search budgets, summarized below. Oracle means reference-derived structure supplied before generation; it is not an automatic prediction.

SongMASS uses greedy decoding with a 500-token output limit. SongComposer uses its public SFT release, one sample, temperature 1, top-p 0.8, top-k 50 and up to 1,000 joint music-text tokens. Its conservative parser retains raw output and does not repair lyric tokens. The adaptation of Back-S (Chae et al., 2025) uses the released GPT-2 generator, temperature 1, top-k 20, top-p 0.9 and 1,024 context positions; intro/outro map to verse in seven cases. A line-end token must actually be sampled to count toward verified completion.

The local adaptation of You et al. (2025) uses the native Mistral-7B-v0.1 generator and v0.2 refiner with reference counts. Native generation uses a 128-token limit and 256 notes per line; 27/302 inputs reach the note limit. Query tokens are initialized with a fixed seed rather than restored from trained query parameters. Native/refiner calls reach their length budgets in 299/302 and 273/302 cases. These initialization and budget differences prevent treating this row as a verified reproduction of the published final system.

MASS→REFFLY uses the released 13B structural rewriting adapter and deterministic group-beam search, 16 candidates, up to five retries, and 50/150-token revision/rephrasing limits. Its conversion supplies music-derived slot and stress constraints without target lyrics; two unsupported-meter input failures remain in the denominator. Draft access and search differ from direct generation. Unknown release-weight identities are not presented as verified final-model reproductions.

SMILE is selected by minimum validation NLL, independently of the 64-song output scores. Its section training uses 30,795 records. Greedy decoding permits 256 new tokens and 48 per line. The controller masks early EOS and excess line breaks; no forced-newline or forced-EOS events occur. Section audio, note routing and reference line timing are supplied, with history and references disabled. All sections are generated independently, before cross-section refinement.

D Component Evaluation Protocols

D.1 Section functions and boundary aggregation

The fixed-readout study uses 572 development works and 2,288 training works, three seeds and frozen predicted spans and representations. Accuracy weights works equally; five-class macro-F1 retains the declared zero-denominator rule. An always-verse diagnostic reaches 74.39% accuracy. These are same-source weak-label results. The new diagnostic fixes C0/Context frame predictions and changes only final majority-vote intervals, without recomputing the contextual representation or classifier. SongFormer-minus-original accuracy differences in percentage points are $+0.003$ $[-0.191, 0.196]$ on 410 Tri works; $+3.416$ [0.898,6.319] on 17 Harmonix works; $+2.142$ $[-0.171, 5.140]$ on 13

Table 5: Function classification on 572 weak-labeled development works with the same frozen predicted segmentation. Values are percentages; ± denotes the sample standard deviation across three seeds. Epochs identify selected checkpoints, not separate data folds.

Function readout	ACC	Macro5-F1	Epochs
Frozen base head	63.54±0.00	20.33±0.00	-
Local readout	63.54±0.00	20.33±0.00	0/0/0
Context readout	79.20±13.56	24.29±3.44	3/3/0
Work-weighted loss	84.31±1.04	26.30±0.13	3/2/3
Work resampling	83.02±2.02	26.12±0.42	3/3/3

Table 6: A/B/C decoding results: descriptive means of two seeds.

Branch	Decoder	L MAE	L Exact (%)	Internal F1 (%)
A	MAP	4.5302	10.82	21.25
A	Count-aware	7.1292	3.67	20.12
B	MAP	2.2951	19.38	18.20
B	Count-aware	2.9021	11.01	22.25
C	MAP	2.4759	18.04	14.44
C	Count-aware	1.7821	19.99	22.24

with old supervised overlap; $+7.556$ [2.112,13.000] on four without known overlap; and -0.025 $[-0.237, 0.185]$ on 335 Tri works after title-alias and exact-MSA exclusions. No known overlap does not establish pretraining independence. Ground-truth cuts are diagnostic, not a full-model upper bound.

D.2 Line-planning decoding

Six fixed A/B/C checkpoints (two seeds each) share 1,308 repeatedly used Popular Hooks train-inner-development records. All branches share a BIO-CRF architecture: A uses path NLL plus balanced internal-boundary BCE, each normalized by onset-group count; B adds 0.1 times count NLL; C additionally adds $0.1\,\mathbb{E}[\max(K - K^*, 0)]$. Both count terms use the same CRF count posterior; K^* is the training reference count, unavailable to inference. Both decoders use the same newly computed logits. MAP chooses the best legal BIO path; count-aware decoding chooses the smallest posterior mode, then the best legal path with exactly that many B labels. The predicted count equals the path's B count and all note assignments follow the path.

Count metrics retain all 1,308 records, including unreachable reference counts. Internal-boundary F1 uses 1,238 eligible records and exact onset-group matches, excluding the first gold and predicted B independently. The other 70 records remain in count metrics. This evaluates assignment and count decoding, without a syllable head or downstream full-plan generation. Count-aware decoding improves C but increases A/B count MAE. C still assigns 80.82% of gold background groups to lines. Its two seed-specific paired MAE intervals are $[-1.0864, -0.7784]$ and $[-0.6607, -0.3093]$; internal-F1 intervals are [8.03,10.75] and [4.92,7.52] percentage points. The 2,000-resample intervals do not include training-seed uncertainty.

D.3 Section Lyric Editor

The reconstruction adapter uses a frozen CT-adapted Qwen3-8B parent, rank-8 LoRA with scale 16, and 128 updates on 2,048 records from 1,312 songs. The new eight-song/41-line panel excludes adapter-training songs and the older development panel; parent-model exposure remains unresolved. Inputs are reference neighboring passages and supplied syllable counts, without music, a generated draft, or a predicted song-level revision plan. Ordinary, hard-constrained and bounded-search decoding complete 8/8, 5/8 and 7/8, with 28/41, 25/41 and 29/41 exact lines. The allowed range $[\max(1, n-1), n]$ differs from symmetric ± 1. Failures retain their fixed denominators. Search recovers two hard-decoding failures; its song-macro exact difference from ordinary decoding is 10.42 points $[-14.58, 39.58]$. Reconstruction does not establish full-draft revision quality.

D.4 Attention-bias diagnostic

The 2×2 diagnostic changes only time and line-position score biases, retaining checkpoint, music, Oracle plan, routing, controller and 256/48-token limits. Every arm retains music and line-organized memory; this is inference sensitivity, not matched-training architectural ablation. Full/time-only/line-only/neither yield CLAP 0.02104/0.01658/0.01670/0.01952, PPL 6.4341/6.4119/6.3864/6.4377 and historical INT 7.5421/7.0210/7.7068/7.5489. Format/PPL/CLAP cover 64 songs, CLaMP3 Payload 63 and Strict 62. Each intervention changes 7/64 texts; paired intervals for Exact, CLAP, PPL and INT differences include zero.

D.5 Music-input intervention

The completed intervention fixes a legal relative Oracle timing-and-count plan P_{eval}. Audio and notes are replaced jointly using a donor from the same source, section function, and line count, from a different known artist, and within 25% of the target duration. Each condition rebuilds its own routing; the target's source-note assignments are not transplanted to the donor. This evaluates a different manipulation from the note-only negative pairing in the generator's training objective.

$$\Delta_{\text{music}} = \text{NLL}(Y \mid X^-, P_{\text{eval}}) - \text{NLL}(Y \mid X^+, P_{\text{eval}}). \tag{22}$$

Here X^+ and X^- are matched and donor music, and target lyric Y is fixed. Song-macro NLL is 2.580468 with matched music and 2.594540 with donor music on 52 songs; their paired difference is 0.014072 $[-0.003423, 0.031989]$. In 12 preselected regenerated pairs, both conditions complete, and 10 songs change their text. We test whether the changes track the corresponding music using

$$D = \big[s(Y^+, A^+) - s(Y^+, A^-)\big] - \big[s(Y^-, A^+) - s(Y^-, A^-)\big], \tag{23}$$

where s is CLaMP3 cosine, Y^+ and Y^- are the generated lyrics, and A^+ and A^- the two audio inputs. The contrast is 0.003379 $[-0.005935, 0.014126]$. Output changes demonstrate sensitivity; neither matching contrast excludes zero.