Libretto: Giving LLM Agents a Sense of Musical Structure

Yichen Xu[1]

[1]University of California, Berkeley

Abstract

Generative music systems can now produce impressive audio from text prompts, but audio outputs are difficult to inspect, edit, and diagnose as musical structure. We introduce Libretto, an agent-facing framework for symbolic music generation and revision. Libretto uses an LLM-native grammar with explicit onset slots, voices, and bar-level organization, then evaluates each piece in a corpus-calibrated statistical space over rhythm, harmony, melody, texture, form, and variation. The same structural axes support retrieval, diagnosis, copy-risk control, and iterative self-revision. Across gap filling, reference-guided full-piece generation, gradual morphing, and educational music generation, Libretto turns symbolic music from a raw token sequence into a measurable and editable object for language-model agents.

1 Introduction

Generative AI has made music creation increasingly accessible. Commercial systems such as Suno, Udio, Stable Audio, Eleven Music, and MusicFX can produce complete audio from short prompts, and research systems such as MusicLM and MusicGen show strong text-conditioned audio generation capabilities [Suno, 2026, Udio, 2026, Stability AI, 2026, ElevenLabs, 2026, Google Labs, 2026, Agostinelli et al., 2023, Copet et al., 2023]. These systems are powerful, but their audio outputs are difficult to inspect as musical objects. A waveform can be heard and rated, but it does not directly expose note timing, voice assignment, phrase structure, harmonic motion, repetition, or local edit boundaries.

Symbolic music offers a complementary path because it keeps music in an editable representation. Prior work has generated symbolic music with chorale models, multi-track GANs, latent musical spaces, long-range Transformers, beat-aware event formats, and multi-track orchestral representations [Hadjeres et al., 2017, Dong et al., 2018, Roberts et al., 2018, Huang et al., 2019, Huang and Yang, 2020, Yu et al., 2022, Liu et al., 2022]. More recent work connects symbolic music with large language models through ABC-based music-language models, symbolic pretraining, text-to-MIDI adaptation, and multi-agent composition [Yuan et al., 2024, Qu et al., 2025, Wu et al., 2025, Deng et al., 2024, Xing et al., 2025]. However, most trained sequence models require specialized deployment, and current agentic systems leave open a basic interface question: what symbolic form should an agent read, write, and revise? Compact formats such as ABC are useful, but their timing can be implicit, making onset reasoning and local editing harder for a language-model agent.

Evaluation is another limitation. Existing systems often rely on human preference, likelihood, prompt adherence, contrastive audio-text alignment, or learned aesthetic predictors [Elizalde et al., 2023, Tjandra et al., 2025]. These measurements are useful, but they do not always explain what structural property of a generated piece failed or how an

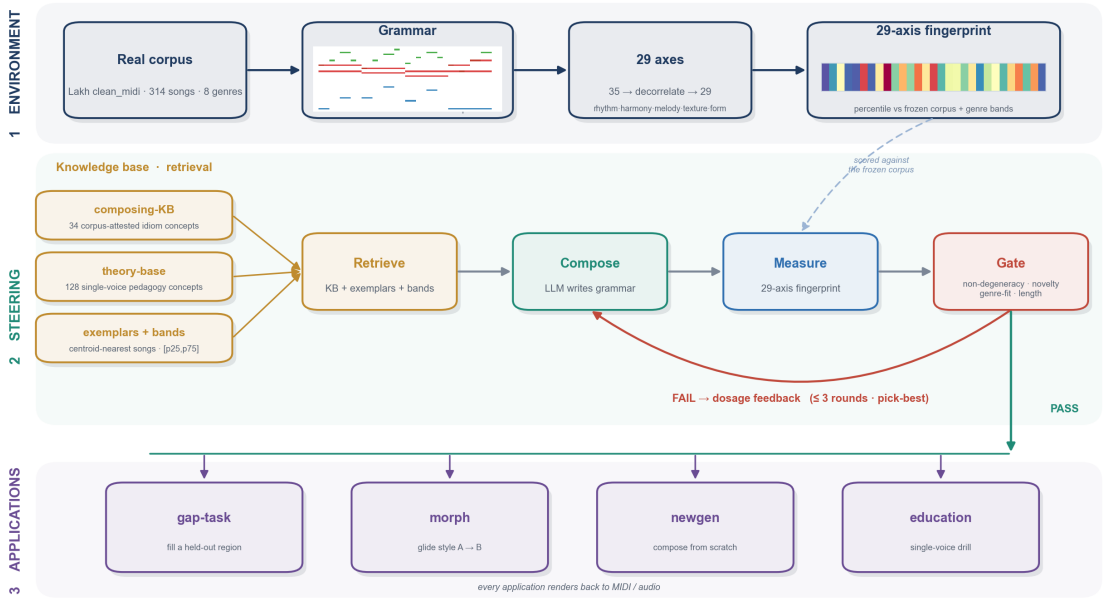

Figure 1: Overview of the LIBRETTO workflow.

agent should revise it. Music theory has long treated music as organized structure across meter, grouping, harmony, and repetition [Lerdahl and Jackendoff, 1983]. This motivates an evaluation interface that describes music through interpretable structural properties rather than only through a global quality score.

We introduce LIBRETTO, an agent-facing framework for symbolic music generation and revision. Libretto represents each piece in an LLM-native grammar with explicit onset slots, voices, and bar-level organization, so that timing and structure are directly readable and locally editable. It then places each piece in a corpus-calibrated statistical cloud: a set of interpretable axes over rhythm, harmony, melody, texture, form, and within-song variation. These axes make music structure measurable, allowing generation to be diagnosed by where it falls relative to existing music. Figure 1 gives an overview of the full workflow, from symbolic representation and retrieval to measurement, feedback, and revision.

Our contributions are threefold:

- We characterize symbolic music through a corpus-calibrated statistical cloud rather than a subjective quality score. Each piece is located along interpretable axes of rhythm, harmony, melody, texture, form, and variation, enabling generation to be diagnosed by where it falls relative to existing music. An LLM-native grammar serves as the interface through which these axes become readable and editable by a language-model agent, while also supporting straightforward downstream MIDI rendering.

- We build an agentic composition system that combines knowledge bases, retrieval, measurement, and iterative self-improvement. The agent retrieves relevant musical concepts and examples, generates a candidate, evaluates it using the structural axes, and revises it through musician-readable feedback.

- We show that the framework supports multiple symbolic-music applications, including gap filling, reference-guided new-piece generation, gradual morphing between musical

styles or pieces, and educational music generation for targeted theory concepts. The same framework also supports longer-form generation than the short fixed-length settings common in prior work, with new multi-voice pieces typically spanning around 100 bars.

2 Related Work

Symbolic music generation and representations. Deep learning for symbolic music generation has been studied through many representations, model families, and evaluation protocols; we refer readers to the survey by Ji et al. [2023] for a broader overview. Early Transformer-based work such as Music Transformer showed that relative self-attention can generate symbolic music with long-term structure, motif continuation, and coherent accompaniment [Huang et al., 2019]. Subsequent work emphasized that representation design is as important as model architecture. Pop Music Transformer introduced REMI, a beat-based event representation with explicit bar, position, tempo, and chord tokens, making metrical and harmonic structure easier for a sequence model to learn [Huang and Yang, 2020]. Museformer addressed long symbolic sequences by combining fine-grained attention over structure-related bars with coarse-grained summaries of other bars [Yu et al., 2022]. SymphonyNet focused on complex orchestral scores, proposing a multi-track, multi-instrument representation and permutation-aware modeling for symphonic generation [Liu et al., 2022]. Anticipatory Music Transformer treated symbolic music as a temporal point process and used arrival-time tokenization for controllable infilling, making onset time explicit in the event sequence [Thickstun et al., 2024]. MuPT scaled ABC-based symbolic pretraining and introduced synchronized multi-track ABC to improve bar alignment across tracks [Qu et al., 2025]. Libretto follows this line of work in treating representation as central, but it targets a different interface: the grammar is designed for language-model agents to read, edit, and diagnose musical structure directly.

LLMs and agentic symbolic composition. Recent work has explored whether large language models can understand and generate symbolic music as text. ChatMusician continues pretraining and fine-tuning LLaMA2 on ABC notation and music-language data, treating music as a second language and evaluating both generation and music-theory understanding [Yuan et al., 2024]. NotaGen adopts large-language-model training paradigms for symbolic generation, including pretraining and finetuning to improve musicality [Wang et al., 2025]. MIDI-LLM instead expands a pretrained text LLM with MIDI tokens and trains it for text-to-MIDI generation, using explicit onset, duration, and instrument-pitch tokens [Wu et al., 2025]. MetaScore builds a large symbolic-score dataset with rich metadata and LLM-generated captions, then trains text- and tag-conditioned symbolic music generators [Xu et al., 2025]. In parallel, ComposerX and CoComposer explore training-free multi-agent composition: ComposerX decomposes symbolic composition into leader, melody, harmony, instrument, review, and arrangement agents [Deng et al., 2024], while CoComposer uses leader, melody, accompaniment, revision, and review agents and evaluates generated outputs with AudioBox-Aesthetics [Xing et al., 2025]. Libretto is closest to these LLM-based symbolic systems, but shifts the contribution from model training or agent role design to the representation-evaluation loop: generated music is written in a directly inspectable grammar, measured by corpus-calibrated structural axes, and revised through explicit feedback.

Text-to-audio generation, benchmarks, and evaluation. Text-conditioned audio generators provide another important comparison point. MusicGen models compressed audio tokens with a single-stage Transformer and supports text and melody conditioning, producing strong audio outputs but not directly exposing symbolic structure for editing or diagnosis [Copet et al., 2023]. Automatic audio evaluation has also developed in this direction: AudioBox-Aesthetics predicts production quality, production complexity, content enjoyment, and content usefulness for speech, music, and sound, offering a scalable alternative to human listening tests [Tjandra et al., 2025]. For symbolic-music understanding, ABC-Eval benchmarks LLMs on ABC notation across syntax, segment-level, and sequence-level tasks, showing that text-based symbolic music reasoning remains difficult for current models [Zhao et al., 2025]. These works motivate Libretto's focus on symbolic structure rather than only audio quality or prompt adherence. Libretto does not replace audio-domain systems or trained symbolic generators; instead, it provides a text interface where timing, voices, and structural measurements remain available throughout retrieval, generation, diagnosis, and self-revision.

3 Methods

Libretto is a text interface for symbolic music. It converts MIDI-like note structure into an LLM-readable grammar, places each piece inside a corpus-calibrated statistical music cloud, and uses that position to guide an agent through retrieval, generation, diagnosis, and revision. The method consists of five parts: the grammar, the structural axis system, the agent loop, the knowledge bases, and the application-specific task setups. We use Claude Code with Opus 4.8 as the LLM agent throughout all experiments, and use 314 real MIDI files spanning eight genres as the raw music corpus, curated from the Lakh MIDI Dataset [Raffel, 2016].

Grammar. Libretto represents a piece as plain text with a global header, a voice declaration, and one block per bar. The header specifies key, meter, tempo, grid, and bar count; the voice line declares the ordered parts; and each bar contains a required chord label followed by voice-specific note tokens. Each token specifies pitch, onset slot, and duration slot, while simultaneous pitches are joined with a plus sign. The grammar uses integer slots rather than floating-point time. In a 16th-note 4/4 grid, for example, the beat positions are slots 1, 5, 9, and 13. This makes rhythm compact and explicit, while preserving separate voices for bass lines, melody, accompaniment, and other parts. The encoder also supports adaptive grids, including triplet grids, so that timing can be preserved without making the text unnecessarily dense. The representation is faithful to pitch, quantized onset, quantized duration, and voice separation, and deliberately abstracts away velocity, micro-timing, original timbre, and unpitched percussion. Its goal is to expose score-like structure in a form that an LLM agent can read, edit, and regenerate.

Structural axes. Libretto evaluates music by locating it inside a statistical cloud of existing pieces rather than assigning a subjective quality score. Each piece is mapped to a 29-axis fingerprint covering rhythm, harmony, melody, texture, form, and within-song variation. These axes are computed directly from grammar tokens, so they describe observable musical structure rather than human preference. Candidate axes are selected by two principles: they must vary meaningfully across the corpus, and they should not be redundant with one another. Near-constant metrics are removed, and highly correlated metrics are pruned, leaving a compact set of relatively independent structural descriptors.

Each raw axis value is then converted to a percentile against a frozen 314-song corpus, giving all axes a common distribution-free scale. A percentile is descriptive: high or low does not mean good or bad. Extreme percentiles are used only to diagnose structural degeneracy, such as outputs that become unusually repetitive, dense, sparse, harmonically unstable, or rhythmically atypical relative to real music.

Agent loop. Generation is organized as a bounded generate–measure–revise loop. The agent first produces a candidate grammar; the system parses it, computes the structural fingerprint, checks copy risk, and evaluates task-specific gates; then the next prompt receives musician-readable feedback. The feedback does not expose raw metric identifiers or ask the model to hit exact numerical targets. Instead, it describes musical tendencies to adjust, such as making the texture less sparse, reducing harmonic instability, increasing genre fit, or moving an extreme axis back toward the idiomatic middle. The loop keeps the best candidate found so far, so refinement is safe at selection time: a later attempt is retained only if it improves the measured structural score.

Knowledge bases and retrieval. Retrieval gives the agent concrete musical grounding. Libretto uses two knowledge bases. The composing knowledge base is corpus-grounded and contains idiomatic concepts for harmony, groove, melody, voicing, form, and jazz-specific techniques. Each entry includes corpus attestation, real examples, and an actionable composition instruction; it is used for genre-conditioned generation and morphing. The theory knowledge base is pedagogical and contains single-voice examples for scales, chords, progressions, rhythm patterns, cadences, texture, and form. Each entry includes a challenge that the generated drill must satisfy; it is used for education tasks. For new-piece generation, Libretto also retrieves short real excerpts from songs nearest to the target genre centroid in fingerprint space. These excerpts provide style references, while copy-risk gates prevent direct reuse.

Copy and novelty. Libretto separates idiomatic similarity from direct copying. Copy risk is measured at the note level by comparing bar-aligned onset–pitch pairs between generated material and real references. The system checks overlap against retrieved examples, likely corpus matches, and, in gap filling, the hidden answer region. This is stricter than comparing chord labels or bar-level summaries: a piece may share genre idioms, harmonic language, or rhythmic feel, but it should not reproduce the same note-level material. In education, the same principle is applied against the shown theory example, so the drill must instantiate the requested concept without simply copying the demonstration.

Applications. The same grammar, statistical axes, retrieval mechanism, and refinement loop support four tasks. In gap filling, the model receives surrounding musical context and fills a held-out region. The output must match the missing length, fit the local context, avoid structural degeneracy, and avoid copying either the hidden answer or corpus material. In new generation, the model composes a full piece in a target genre from scratch, using retrieved concepts and prototypical excerpts as references; the loop pushes the result away from extremes, weak genre fit, and copy risk. In our experiments, new-generation outputs typically span 92–128 bars, with a mean length of about 102 bars. In morphing, the model gradually moves from one source style or component toward another, and evaluation checks source-like beginning, target-like ending, smooth progress, and non-abrupt transition. In education, the model generates a short drill for a requested theory concept, requiring valid

Table 1: Representation and metric validation. The grammar preserves downstream musical structure with quantified losses, makes timing directly readable, and the 29-axis fingerprint is low-redundancy but genre-informative.

Check	Metric	Result
Pitch and voice	18 files; 9 genres; 57,885 non-drum notes	Pitch set preserved exactly; 11 notes lost, or 0.019%; source parts map 1:1 to grammar voices.
Timing fidelity	Round-trip note starts and durations	Start-time error after round trip: median 0 ms, mean 3.1 ms, worst note within one grid slot; durations rounded to the bar grid.
Timing readability	ABC vs. Libretto on 5 tunes, 98 notes, 28 bars	Recovering note starts takes 70 running additions in ABC, or 131 additions from scratch; Libretto takes 0 because onset slots are explicit.
Edit locality	Single duration edits under relative timing	ABC causes 1005 downstream onset rederivations over the corpus; Libretto causes 0. Cost: ABC 335 body characters, Libretto 1176.
Intentional abstractions	Dimensions not represented	Velocity flattened; micro-timing quantized; drums dropped, mean 21% of source notes; instrument identity reassigned.
Axis construction	Spread and redundancy filters	35 candidate measurements reduced to 29 axes over rhythm, harmony, melody, texture, form, and within-song variation.
Axis independence	Pairwise correlations on 314 songs	Mean $\|r\| = 0.16$; 12/406 retained axis pairs exceed 0.5; largest remaining $\|r\| = 0.77$.
Genre signal	Five-fold classification on 255 songs	Logistic regression on the 29 axes reaches 0.384 top-1 accuracy over 8 genres, about 3.1× chance.

grammar, key adherence, satisfaction of user constraints, detection of the target concept, and novelty relative to the shown example.

4 Experiments

4.1 Representation and evaluation

Before evaluating generation, we first validate the text grammar. The audit measures whether the representation preserves the musical information used by the downstream language-model pipeline, including pitch, timing, and voice assignment, while explicitly quantifying the dimensions that are intentionally abstracted away. We also test timing readability: in ABC, note starts are recovered by accumulating earlier durations in the bar, while in Libretto each note carries its absolute onset slot. We then define a 29-dimensional structural fingerprint, where each axis is evaluated as a corpus percentile against a fixed 314-song reference set. A value near 50 is corpus-typical, while values at or below the 5th percentile or at or above the 95th percentile are treated as structural extremes.

Table 1 establishes the measurement substrate used in the rest of the experiments. The grammar is effectively exact for pitch and voice, grid-faithful for timing, and explicit about the musical information it abstracts away. It also makes timing local: Libretto uses more characters than ABC, but avoids the duration accumulation needed to recover note starts

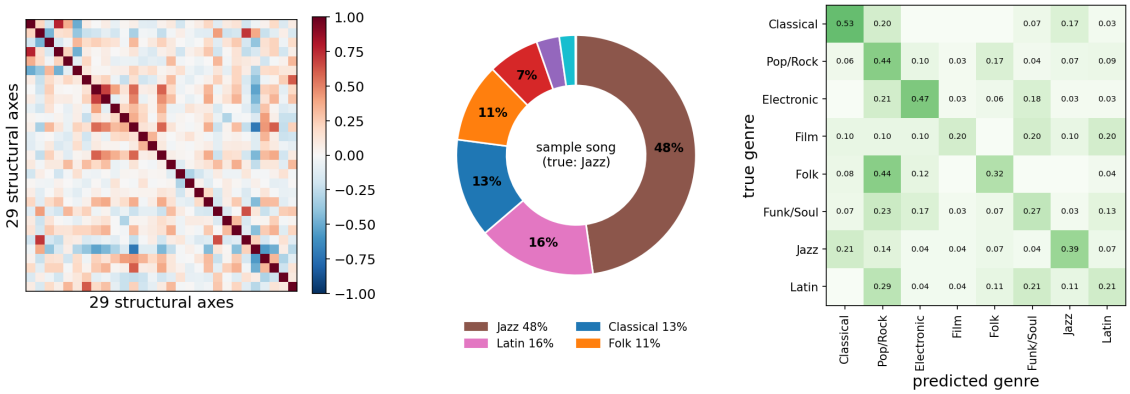

Figure 2: Axis structure and soft genre signal.

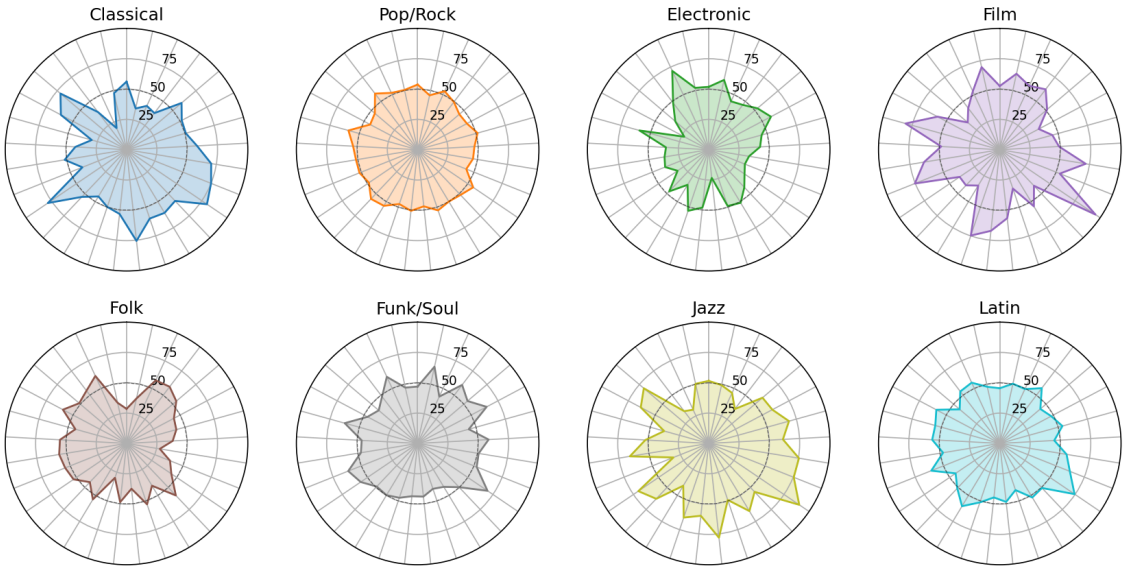

Figure 3: Faceted genre fingerprints over the 29 axes.

and keeps duration edits from shifting downstream onsets. This is an encoding-cost result rather than an LLM benchmark. The structural axes are checked separately: they are filtered for spread and redundancy, then evaluated both for independence and for musical signal.

Figure 2 visualizes the 29-axis fingerprint as a measurement system. The correlation panel shows that most axis pairs are weakly related. The genre-composition panel shows that a song is represented as a soft stylistic mixture rather than as a hard category. The confusion matrix makes the same point at corpus scale: the diagonal is visible, but neighboring genres blur into one another. This is the desired behavior for a structural metric: it captures genre tendencies without forcing musical style into clean clusters.

Figure 3 makes the axes musically interpretable. Each spoke is one structural measurement, and each value is the genre mean corpus percentile. Jazz expands on harmonic-complexity axes, with chromaticism near the 75th percentile and diminished/augmented color near the 76th percentile. Folk is lower on the same axes, around the 28th and 39th percentiles. Electronic music is highly self-similar, around the 72nd percentile, while classical is much lower, around the 20th percentile. These gaps of roughly 30–50 percentile

Table 2: Gate calibration and discrimination. Each gate is calibrated on real or acceptable music and targets a distinct generated-output failure mode.

Gate	Calibration	Generated-output behavior
Degeneracy	Real songs: mean 3.4 extreme axes out of 29, median 3, 90th percentile 7; final genre budgets allow 4–6 extremes.	Gap-task failures average 4.36 extremes versus 2.25 for passes; ungrounded full-piece generations average 5.9.
Genre fit	A fixed six-of-eight fit rule admits only 10–29% of real songs in their own genres; final floors are genre-calibrated.	Calibrated floors avoid rejecting human music while still requiring generated pieces to occupy genre-typical structural bands.
Copy risk	Real-song copy statistics have 90th percentile at most 0.29 across genres, grounding a threshold near 0.30.	Passing fills: mean 0.233, max 0.294; flagged fills: mean 0.383, min 0.312.
Novelty	Accepted education drills have copy-vs-shown mean 0.065 and max 0.127, below the 0.50 novelty threshold.	Rejects drills that reuse the shown material instead of producing a new example of the target concept.

points show that the fingerprint carries readable stylistic structure.

The pass gates use the same percentile language. The main structural gate counts how many axes land in the extreme tails of the real-song distribution. A distinctive piece of music may naturally contain some extreme measurements, so the budgets are calibrated against real songs rather than fixed uniformly. A separate copy-risk gate rejects outputs that are too close to the source, context, or hidden answer.

Table 2 shows that the gates are calibrated against real or acceptable music before being applied to generated outputs. Real music naturally contains some extreme axes, so the degeneracy gate allows a small calibrated budget rather than requiring every axis to be central. Genre fit is likewise calibrated per genre because a fixed rule would reject too much human music. Copy risk is anchored to real-song overlap statistics, and novelty is checked against the examples shown in the education setting. Together, the gates target the intended generated-output failures: structural collapse, weak genre fit, replication of corpus or held-out material, and copying from demonstrations.

4.2 Application-level results

We evaluate the grammar and gates across four applications: filling missing regions, generating complete pieces, morphing between styles, and creating education drills. A generation passes only if it satisfies the relevant structural, fit, copy, and task-specific checks. The same metric family is used across tasks, but each task stresses a different failure mode: local coherence for gap filling, non-degeneracy for full-piece generation, continuity for morphing, and requirement satisfaction for drills.

Table 3 gives the main outcomes. The loop is most useful when a first draft is close but structurally flawed: it raises gap-filling pass rate from 12% to 39% and full-piece pass rate from 62% to 94%. Retrieval addresses a different weakness. In full-piece generation, it triples the pass rate from 25% to 75% by grounding the model in concrete musical examples. In education, where the desired scale, rhythm, and challenge constraints are already explicit, retrieval adds little.

Figure A.2 shows the breadth of the system in a single visual language. The examples include a passing jazz gap-fill with one extreme axis, a 96-bar jazz full-piece generation with low copy risk across rounds, an electronic-to-folk morph, and an E harmonic minor education

Table 3: Main application results. Retrieval and the self-evolving loop improve the settings where generation otherwise collapses or underspecifies structure.

Application	Comparison	Baseline	Result
Gap filling	Single-shot vs. loop, up to 3 rounds	6/51 pass, 12%	20/51 pass, 39%; 33/51 improve.
Full-piece generation	Single-shot vs. loop, up to 3 rounds	10/16 pass, 62%	15/16 pass, 94%; loop runs only on failures.
Full-piece generation	Retrieval off vs. retrieval on	2/8 pass, 25%	6/8 pass, 75%; copy risk below gate in every genre.
Education drills	Retrieval off vs. retrieval on	7/8 pass	6/8 pass; retrieval is nearly null because concepts are already explicit.
Education drills	Retrieval-on model comparison	All models output valid grammar	Opus 6/8, Sonnet 3/8, Haiku 3/8 pass.
Gradual morphing	Source-to-target component morph	–	11/21 pass all morph checks.

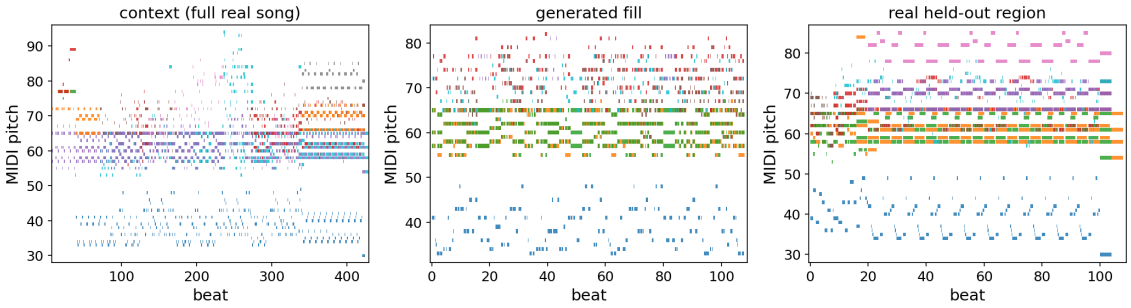

Figure 4: Gap-task triptych: context, generated fill, and held-out answer.

drill with 2.2% out-of-scale notes and copy-vs-shown score 0.065. This demonstrates that the same text representation supports coherent structure across different musical objectives.

Figure 4 focuses on the gap-filling setting. The left panel is the surrounding real context, the middle panel is the generated fill, and the right panel is the held-out answer. The task is to land in the same musical neighborhood as the context, without copying the hidden answer. This jazz continuation passes at round 3 with one extreme axis, copy risk 0.251, answer overlap 0.145, and beat alignment 98%, showing that it fits the local texture while remaining distinct from the ground truth.

Figure 5 illustrates the morph task, where the metric checks whether the output starts like source style A, ends like target style B, and moves gradually rather than jumping. In this electronic-to-folk example, the seven-segment progress curve rises from 0.22 to 0.42, 0.49, 0.61, and eventually 1.0. The largest single step is 0.38, below the abrupt-jump ceiling, and the copy-to-A and copy-to-B curves cross in opposite directions. The figure shows a morph that is both anchored and progressive.

Figure A.6 shows example scores generated for education drills. A student can request new practice material in different keys, modes, rhythmic patterns, or melodic settings, allowing the same theory concept to be practiced through fresh short scores rather than repeated from a fixed exercise.

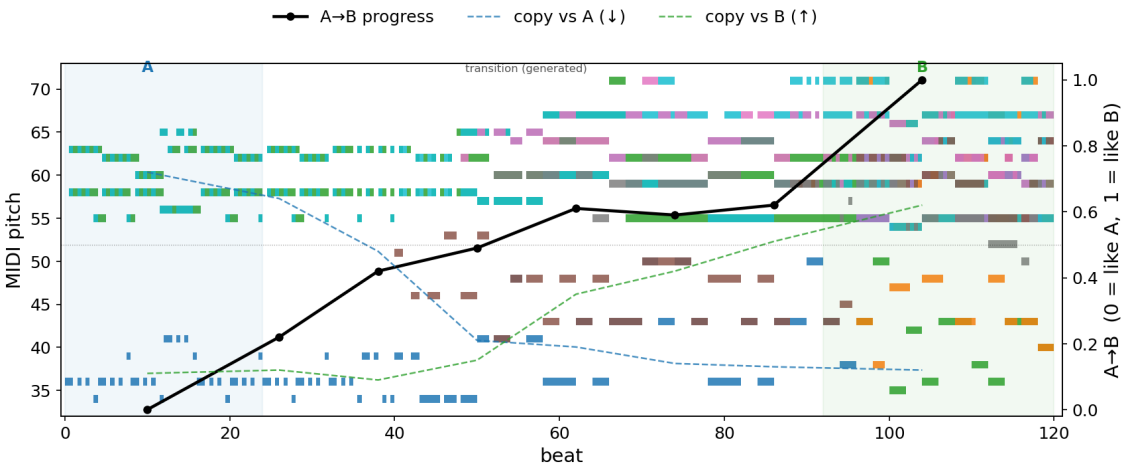

Figure 5: Gradual morph with measured progress curve.

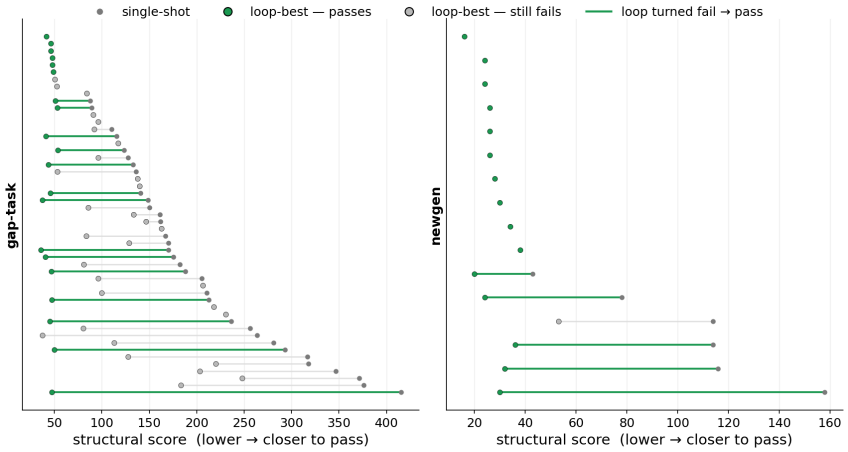

Figure 6: Per-song effect of the self-evolving loop.

4.3 Mechanisms and diagnostics

The aggregate gains in Table 3 come from two different mechanisms. Retrieval helps before generation by grounding the prompt in real musical targets; the loop helps after generation by iteratively repairing outputs that fail the gates. We visualize both mechanisms at the level of the structural axes.

Figure A.3 shows that retrieval improves full-piece generation by pulling extreme axes back toward the corpus band. Without retrieval, Pop/Rock collapses to the 5th percentile for mean note duration and the 95th percentile for diminished/augmented color; Film reaches the 98th percentile for ascending-step ratio, the 97th percentile for root-motion variety, and the 100th percentile for novelty. With retrieval, Pop/Rock moves to 52, 60, and 46 on the corresponding axes, while Film moves root motion from 97 to 46, ascending-step ratio from 98 to 12, maximum chord width from 1 to 59, and novelty from 100 to 74. Retrieval therefore improves pass rate by de-degenerating the fingerprint, not by increasing similarity to the corpus through copying.

Figure 6 shows the loop at the level of individual songs. Each row compares the single-shot score with the best loop candidate, and lower is better. In the gap task, 33 of 51 songs improve and the pass count rises from 6 to 20. In full-piece generation, only six

rows move because the other ten already passed single-shot; among those looped pieces, five cross the pass gate. This explains why the loop is useful but not indiscriminate: it acts only where the gates expose a concrete failure.

Figure A.4 shows the same supervision signal directly. Rows are generated pieces, columns are axes, colors are corpus percentiles, and dots mark tail extremes. In the eight retrieval-on full-piece generations, the average piece has 4.6 extreme axes, but the distribution is diagnostic: electronic has one extreme, Latin and Funk/Soul have three, Pop/Rock and Film have four, and Folk accumulates 10 extremes and fails. The heatmap turns a pass/fail decision into a readable map of which musical dimensions still need repair.

Figure A.5 shows why the diagnostic is musically meaningful rather than just numerical. This Latin gap-fill fails with six extreme axes against a budget of three: harmonic rhythm at the 100th percentile, chord variety at the 99th, average note length at the 96th, stepwise motion at the 95th, bar distinctness at the 100th, and density variation at the 3rd percentile. Those numbers correspond to what is visible in the roll: a chord-conveyor-belt texture with roughly two chords per bar and a scalar, highly stepwise melody. The case is not a copy, with copy risk 0.134 and answer overlap 0.098, and it is not simply off-grid, with beat alignment 86%. The failure is structural degeneracy, and the named axes make that failure inspectable.

5 Conclusion

We presented Libretto, an agent-facing framework that makes symbolic music readable, measurable, and revisable by an LLM agent. Instead of treating generation quality as a single subjective score, Libretto represents each piece in a corpus-calibrated structural space over rhythm, harmony, melody, texture, form, and variation. This lets the agent diagnose where a candidate departs from real music, retrieve relevant examples, and revise through musician-readable feedback. Across gap filling, full-piece generation, gradual morphing, and educational music generation, the same representation-evaluation loop supports multiple symbolic-music tasks without training a new music model.

Several directions remain open. The structural axes can be refined, expanded, and pruned using the same principles used here: each added axis should capture meaningful musical variation while remaining sufficiently decorrelated from existing measurements. The feedback loop also suggests a natural path toward agentic reinforcement learning for music generation, where actions such as retrieval, editing, rewriting, and accepting can be optimized against structural rewards, copy-risk constraints, and human preference. More broadly, Libretto points toward symbolic-music agents that do not merely emit notes, but learn to reason over measurable musical structure.

References

Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text, 2023. URL `https://arxiv.org/abs/2301.11325`.

Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. Simple and controllable music generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Qixin Deng, Qikai Yang, Ruibin Yuan, Yipeng Huang, Yi Wang, Xubo Liu, Zeyue Tian, Jiahao Pan, Ge Zhang, Hanfeng Lin, Yizhi Li, Yinghao Ma, Jie Fu, Chenghua Lin, Emmanouil Benetos, Wenwu Wang, Guangyu Xia, Wei Xue, and Yike Guo. Composerx: Multi-agent symbolic music composition with llms, 2024. URL https://arxiv.org/abs/2404.18081.

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11312. URL https://doi.org/10.1609/aaai.v32i1.11312.

ElevenLabs. Elevenlabs music. https://elevenlabs.io/music, 2026. Accessed 2026-06-19.

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095889.

Google Labs. Musicfx. https://labs.google/fx/tools/music-fx, 2026. Accessed 2026-06-19.

Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: a steerable model for Bach chorales generation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1362–1371. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/hadjeres17a.html.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJe4ShAcF7.

Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1180–1188, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379885. doi: 10.1145/3394171.3413671. URL https://doi.org/10.1145/3394171.3413671.

Shulei Ji, Xinyu Yang, and Jing Luo. A survey on deep learning for symbolic music generation: Representations, algorithms, evaluations, and challenges. 56(1), August 2023. ISSN 0360-0300. doi: 10.1145/3597493. URL https://doi.org/10.1145/3597493.

Fred Lerdahl and Ray Jackendoff. *A generative theory of tonal music*. The MIT Press, 1983.

Jiafeng Liu, Yuanliang Dong, Zehua Cheng, Xinran Zhang, Xiaobing Li, Feng Yu, and Maosong Sun. Symphony generation with permutation invariant language model, 2022. URL https://arxiv.org/abs/2205.05448.

Xingwei Qu, yuelin bai, Yinghao Ma, Ziya Zhou, Ka Man Lo, Jiaheng Liu, Ruibin Yuan, Lejun Min, Xueling Liu, Tianyu Zhang, Xeron Du, Shuyue Guo, Yiming Liang, Yizhi LI, Shangda Wu, Junting Zhou, Tianyu Zheng, Ziyang Ma, Fengze Han, Wei Xue,

Gus Xia, Emmanouil Benetos, Xiang Yue, Chenghua Lin, Xu Tan, Wenhao Huang, Jie Fu, and Ge Zhang. MuPT: A generative symbolic music pretrained transformer. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iAK9oHp4Zz.

Colin Raffel. Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching. 2016. URL https://api.semanticscholar.org/CorpusID:63439223.

Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4364–4373. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/roberts18a.html.

Stability AI. Stable audio 3.0. https://stability.ai/stable-audio, 2026. Accessed 2026-06-19.

Suno. Suno: Ai music generator. https://suno.com/, 2026. Accessed 2026-06-19.

John Thickstun, David Leo Wright Hall, Chris Donahue, and Percy Liang. Anticipatory music transformer. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=EBNJ33Fcrl.

Andros Tjandra, Yi-Chiao Wu, Baishan Guo, John Hoffman, Brian Ellis, Apoorv Vyas, Bowen Shi, Sanyuan Chen, Matt Le, Nick Zacharov, Carleigh Wood, Ann Lee, and Wei-Ning Hsu. Meta audiobox aesthetics: Unified automatic quality assessment for speech, music, and sound. 2025. URL https://arxiv.org/abs/2502.05139.

Udio. Udio: Ai music generator. https://www.udio.com/, 2026. Accessed 2026-06-19.

Yashan Wang, Shangda Wu, Jianhuai Hu, Xingjian Du, Yueqi Peng, Yongxin Huang, Shuai Fan, Xiaobing Li, Feng Yu, and Maosong Sun. Notagen: Advancing musicality in symbolic music generation with large language model training paradigms. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 10207–10215. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/1134. URL https://doi.org/10.24963/ijcai.2025/1134.

Shih-Lun Wu, Yoon Kim, and Cheng-Zhi Anna Huang. MIDI-LLM: Adapting large language models for text-to-MIDI music generation. In *AI for Music Workshop*, 2025. URL https://openreview.net/forum?id=GVW9YixIAI.

Peiwen Xing, Aske Plaat, and Niki van Stein. Cocomposer: Llm multi-agent collaborative music composition, 2025. URL https://arxiv.org/abs/2509.00132.

Weihan Xu, Julian McAuley, Taylor Berg-Kirkpatrick, Shlomo Dubnov, and Hao-Wen Dong. Generating symbolic music from natural language prompts using an llm-enhanced dataset, 2025. URL https://arxiv.org/abs/2410.02084.

Botao Yu, Peiling Lu, Rui Wang, Wei Hu, Xu Tan, Wei Ye, Shikun Zhang, Tao Qin, and Tie-Yan Liu. Museformer: Transformer with fine- and coarse-grained attention for music generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=GFiqdZOm-Ei.

Ruibin Yuan, Hanfeng Lin, Yi Wang, Zeyue Tian, Shangda Wu, Tianhao Shen, Ge Zhang, Yuhang Wu, Cong Liu, Ziya Zhou, Liumeng Xue, Ziyang Ma, Qin Liu, Tianyu Zheng, Yizhi Li, Yinghao Ma, Yiming Liang, Xiaowei Chi, Ruibo Liu, Zili Wang, Chenghua Lin, Qifeng Liu, Tao Jiang, Wenhao Huang, Wenhu Chen, Jie Fu, Emmanouil Benetos, Gus Xia, Roger Dannenberg, Wei Xue, Shiyin Kang, and Yike Guo. ChatMusician: Understanding and generating music intrinsically with LLM. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6252–6271, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.373. URL `https://aclanthology.org/2024.findings-acl.373/`.

Jiahao Zhao, Yunjia Li, Wei Li, and Kazuyoshi Yoshii. Abc-eval: Benchmarking large language models on symbolic music understanding and instruction following, 2025. URL `https://arxiv.org/abs/2509.23350`.

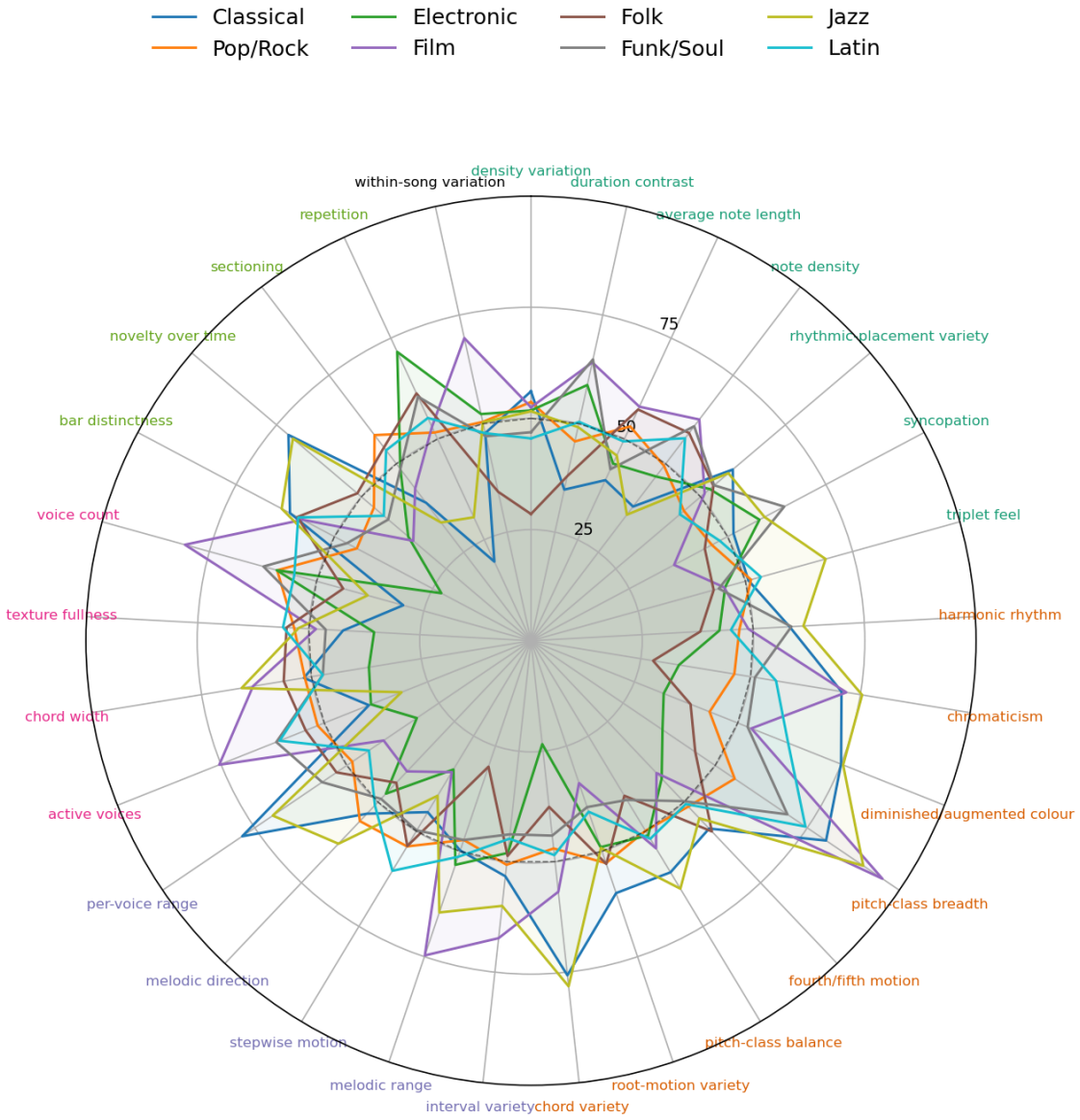

Figure A.1: Overlaid genre fingerprints over the 29 axes.

Appendix

A Auxiliary results

We illustrate the Libretto grammar on three real generated outputs. The examples span single-voice pedagogy, multi-voice from-scratch generation, and anchored multi-voice gap filling. All examples are actual model outputs, trimmed for readability and edited only by elisions marked with ....

A. Education drill: single-voice pedagogy

E harmonic minor, one `Piano` voice. The excerpt shows the raised seventh `D##`, the dominant `B7`, and closing block chords written with `+`.

```
KEY: E minor | METER: 4/4 | TEMPO: 138 | GRID: 16th | BARS: 20
```

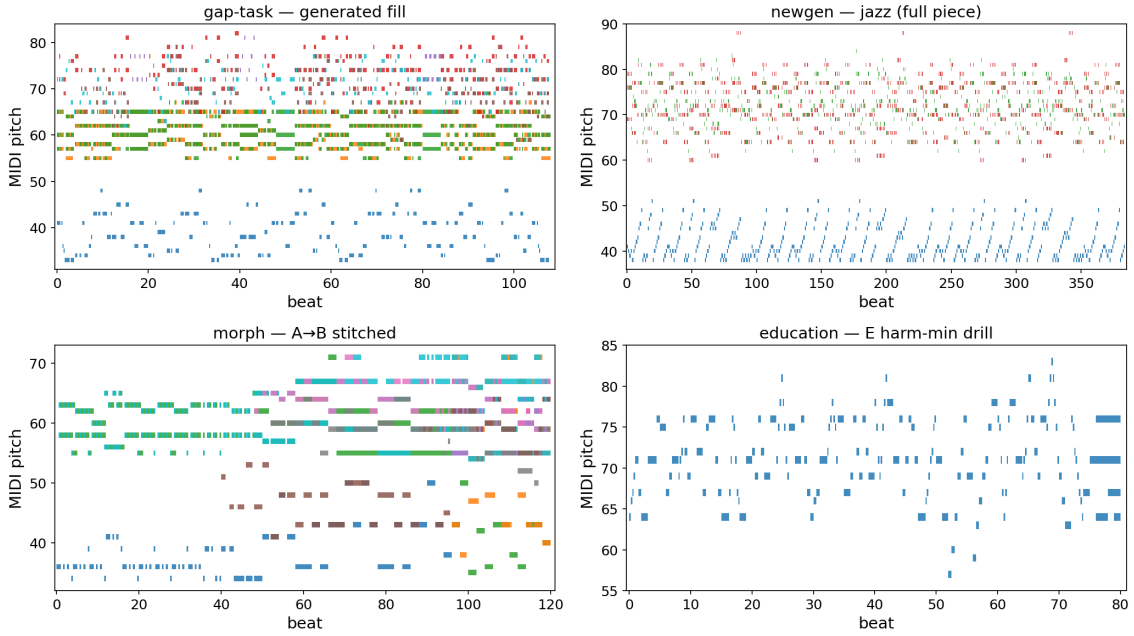

Figure A.2: Representative generated outputs across four applications.

```
VOICES: Piano
@1 [Em]
  Piano: E4@1>1 F#4@2>1 G4@3>1 A4@4>1 B4@5>2 G4@7>2 E4@9>4 B4@13>4
@2 [Em]
  Piano: B4@1>2 E5@3>2 D#5@5>4 G4@9>2 A4@11>2 B4@13>4
@3 [Am]
  Piano: A4@1>1 B4@2>1 C5@3>1 E5@4>1 C5@5>2 A4@7>2 E5@9>4 C5@13>4
@4 [Em]
  Piano: G4@1>2 D#5@3>2 E5@5>4 B4@9>2 G4@11>2 E4@13>4
@5 [Em]
  Piano: E4@1>1 G4@2>1 B4@3>1 E5@4>1 D#5@5>1 B4@6>1 G4@7>1 F#4@8>1 E4@9>4 B4@13>4
@6 [Am]
  Piano: C5@1>2 A4@3>2 E5@5>2 C5@7>2 A4@9>4 E5@13>4
@7 [B7]
  Piano: B4@1>1 D#5@2>1 F#5@3>1 A5@4>1 F#5@5>1 D#5@6>1 B4@7>1 A4@8>1 D#5@9>4 B4@13>4
@8 [Em]
  Piano: E5@1>2 B4@3>2 G4@5>2 E4@7>2 F#4@9>2 G4@11>2 B4@13>4
...
@20 [Em]
  Piano: E4+B4+E5@1>8 G4+B4+E5@9>4 E4+G4+B4+E5@13>4
```

Read: In a 16th-note 4/4 grid, slot 1 is beat 1 and the beats fall on slots 1, 5, 9, and 13. A duration of `>4` is a quarter note, while `>1` is a sixteenth note.

B. New generation: funk/soul multi-voice band

A 96-bar funk/soul generation in G minor with five declared voices: `BASS`, `GTR`, `KEYS`, `HORNS`, and `LEAD`. The excerpt shows bars 9–12, where four active voices interlock over a `Gm7-Cm7-D7` vamp: off-beat guitar stabs, seventh-chord keyboard comping, a syncopated bass with pickups, and a lead line above.

```
KEY: G minor | METER: 4/4 | TEMPO: 102 | GRID: 16th | BARS: 96
VOICES: BASS, GTR, KEYS, HORNS, LEAD
...
@9 [Gm7]
  BASS : G1@1>2 G1@4>1 Bb1@7>1 D2@9>2 D2@12>1 F1@15>2
  GTR  : Bb3+D4+F4@3>1 Bb3+D4+F4@7>1 Bb3+D4+F4@11>1 Bb3+D4+F4@15>1
```

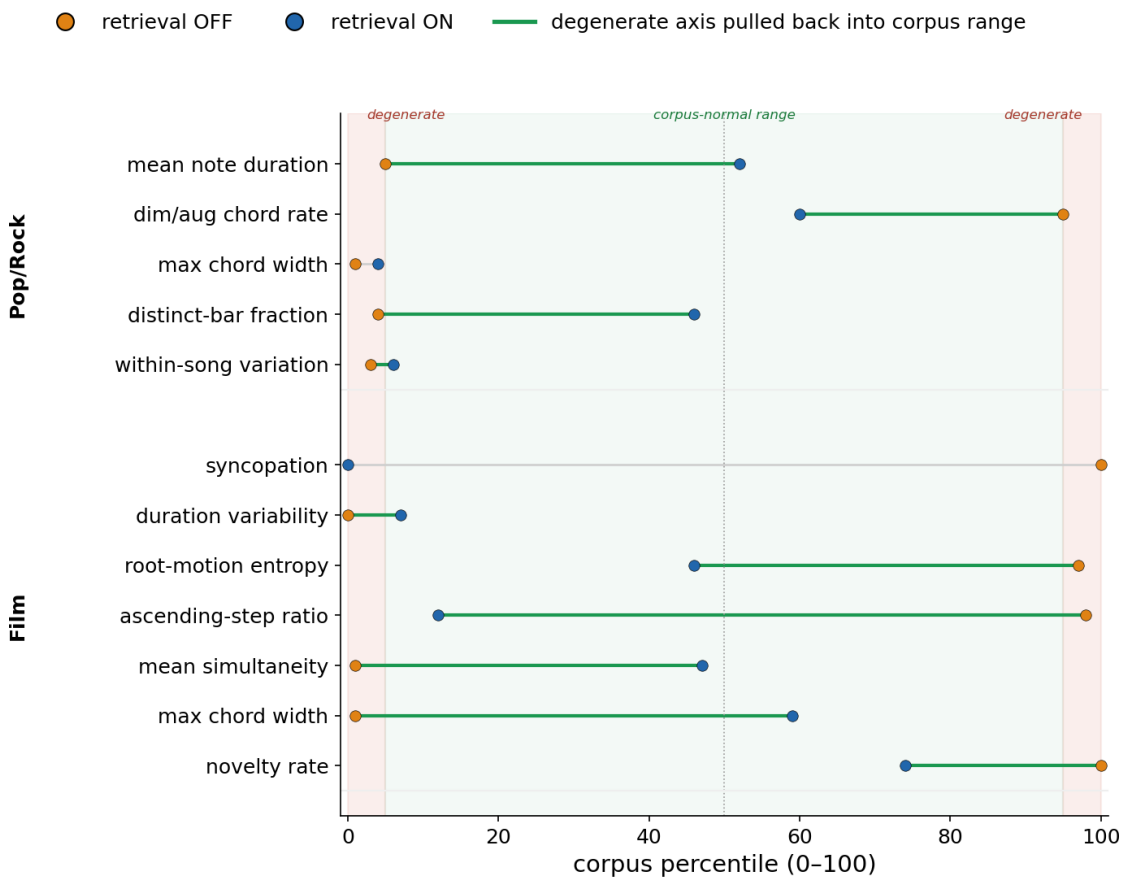

Figure A.3: Retrieval de-degenerates full-piece generation.

```
  KEYS : G3+Bb3+D4+F4@3>2 G3+Bb3+D4+F4@11>2
  LEAD : D5@3>2 F5@7>1 G5@9>2 F5@13>1 D5@15>2
@10 [Cm7]
  BASS : C2@1>2 C2@4>1 Eb2@7>1 G2@9>2 G2@12>1 Bb1@15>2
  GTR  : Eb4+G4+Bb4@3>1 Eb4+G4+Bb4@7>1 Eb4+G4+Bb4@11>1 Eb4+G4+Bb4@15>1
  KEYS : C3+Eb3+G3+Bb3@3>2 C3+Eb3+G3+Bb3@11>2
  LEAD : G5@3>2 Eb5@7>1 D5@9>2 C5@13>3
@11 [Gm7]
  BASS : G1@1>2 G1@4>1 Bb1@7>1 D2@9>2 D2@12>1 F1@15>2
  GTR  : Bb3+D4+F4@3>1 Bb3+D4+F4@7>1 Bb3+D4+F4@11>1 Bb3+D4+F4@15>1
  KEYS : G3+Bb3+D4+F4@3>2 G3+Bb3+D4+F4@11>2
  LEAD : Bb4@3>2 D5@7>1 F5@9>2 G5@13>1 F5@15>2
@12 [D7]
  BASS : D2@1>2 D2@4>1 C#2@7>1 A1@9>2 A1@12>1 D2@15>2
  GTR  : F#3+A3+C4@3>1 F#3+A3+C4@7>1 F#3+A3+C4@11>1 F#3+A3+C4@15>1
  KEYS : D3+F#3+A3+C4@3>2 D3+F#3+A3+C4@11>2
  LEAD : A4@3>2 C5@7>1 D5@9>2 C#5@13>1 A4@15>2
...
```

Read: Slots 3, 7, 11, and 15 are off-beat sixteenth-grid positions between the main beats. The `GTR` and `KEYS` hits on those slots create the funk push, while `+`-joined pitches encode chord voicings.

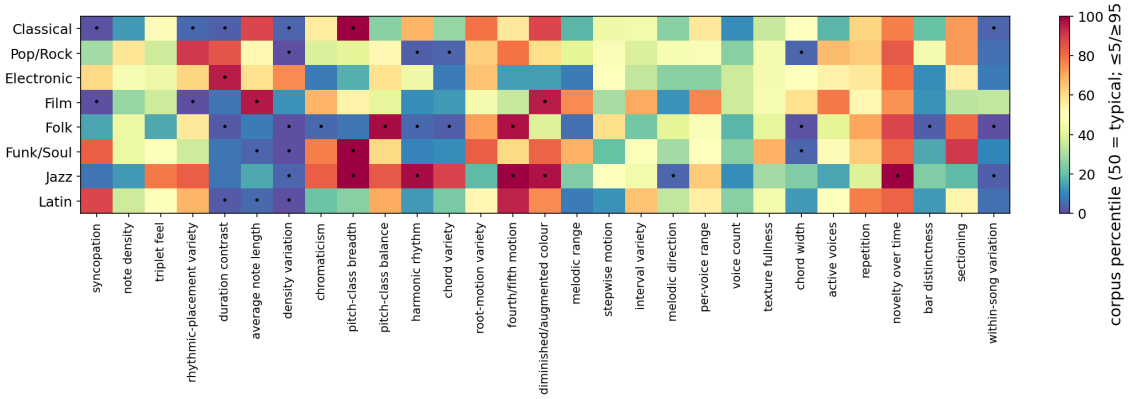

Figure A.4: Per-piece axis profiles for generated outputs.

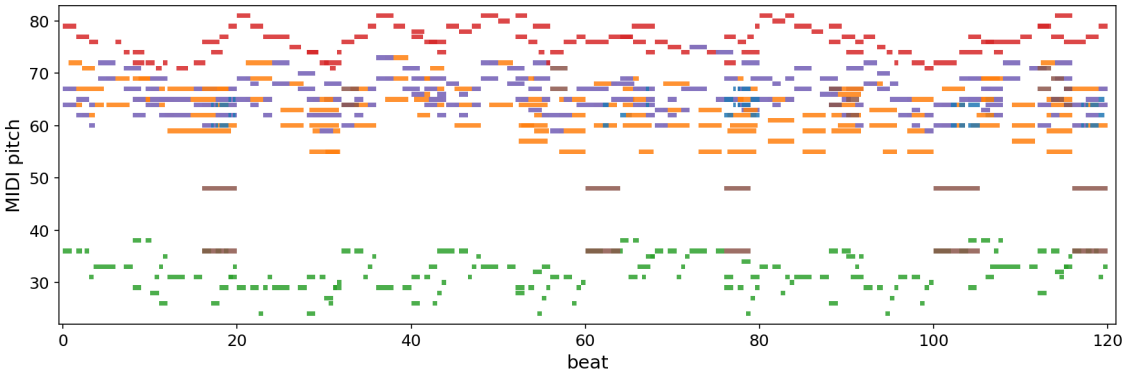

Figure A.5: Interpretable failure case for a gap-fill continuation.

C. Gap filling: anchored multi-voice fill

A generated region filled into an existing song's voice set. The excerpt shows a 9-voice adaptive-grid grammar with anonymized source tracks such as `Part2` and `Part6`, preserving the original multi-track structure while avoiding leakage from track names.

```
KEY: Bb major | METER: 4/4 | TEMPO: 200 | GRID: 16th (adaptive) | BARS: 27
VOICES: Bass Gtr, Part2, Guitar 3, Guitar 2, Guitar 1, Part6, Part7, Strings, Marimba
@1 [F | Am]
  Bass Gtr: F2@1>3 F2@5>1 C2@6>1 B1@7>1 A1@8>4 A1@13>2 A1@15>1
  Part2   : A3+C4+F4@1>6 G3+A3+C4+E4@9>6
  Guitar 3: A3+C4+F4@1>2 A3+C4+F4@4>1 C4+F4@6>1 A3+C4+E4@9>2 C4+E4@11>1 A3+C4+E4@13>3
  Guitar 1: C5+F5@3>1 A4+C5@7>2 E5@13>3
  Part6   : C5@1>1 C5@5>2 D5@9>1 E5@11>1 G4@13>2 F4@16>1
@2 [Bb]
  Bass Gtr: Bb1@1>2 Bb1@5>1 F2@7>2 Bb1@9>3 C2@13>2 D2@15>1
  Part2   : Bb3+D4+F4@1>8 G3+Bb3+D4@9>5 A3+D4+F4@15>2
  Guitar 3: Bb3+D4+F4@1>1 D4+F4@3>1 Bb3+D4+F4@5>2 Bb3+D4+F4@8>1 G3+Bb3+D4@9>2 Bb3+D4+F4@11>1 Bb3+
     D4@13>2 F4@16>1
...
```

Read: The label `[F | Am]` denotes a mid-bar harmonic change. Names such as `Part2` and `Part6` are anonymized real source tracks, used to keep the gap-filling task leakage-clean while preserving the original voice structure.

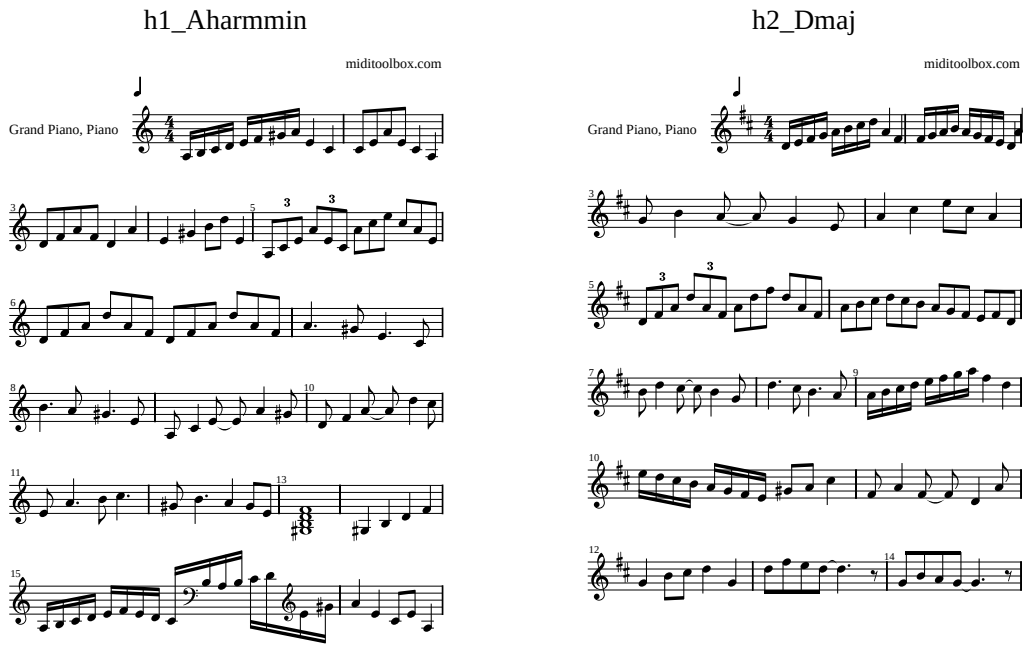

Figure A.6: Educational music-generation examples for targeted harmonic concepts.

B Metric Definitions

This appendix defines the structural axes, percentile fingerprint, copy-risk score, and calibrated gates used in the experiments. All quantities are computed from the Libretto grammar tokens and are descriptive rather than aesthetic.

Notation. Let $P = \{e\}$ be the set of parsed note events. Each event has bar $b(e)$, within-bar onset $o(e)$, absolute onset $t(e)$, duration $d(e)$, MIDI pitch $m(e)$, pitch class $pc(e) = m(e) \bmod 12$, and voice $v(e)$. Let Q be beats per bar, $\mathcal{B}$ the set of bars, $N_b = |\mathcal{B}|$, $V = \{v(e) : e \in P\}$, and

$$\mathcal{O} = \{(v(e), t(e)) : e \in P\}, \qquad n_{\text{on}} = |\mathcal{O}|\,.$$

Let $D = (d(e))_{e \in P}$. Means and standard deviations are population statistics. For a count vector c, define normalized entropy

$$H(c) = \begin{cases} -\dfrac{\sum_{i:c_i>0} p_i \log_2 p_i}{\log_2 k'}, & k' > 1, \\ 0, & k' \le 1, \end{cases} \qquad p_i = \frac{c_i}{\sum_j c_j}, \quad k' = |\{i : c_i > 0\}|\,.$$

For event set E, define duration-weighted pitch-class mass $w_E(p) = \sum_{e \in E : pc(e) = p} d(e)$, and

$$\text{prom}(w) = \{p : w(p) \ge 0.30 \max_q w(q)\}.$$

For each voice u, let $\mu(u) = \text{mean}\{m(e) : v(e) = u\}$ and $\chi(u) = |\{e : v(e) = u\}| \,/\, |\{t(e) : v(e) = u\}|$. The bass is the lowest-μ voice. The melody is the highest-μ voice among voices with $\chi < 1.4$

and at least 8 distinct onsets, or otherwise the highest-μ voice. Let $s_1, \ldots, s_L$ be the melody line, taking the highest MIDI pitch at each melody-voice onset.

For each bar b, define the note set

$$A_b = \{(v(e), o(e), m(e)) : b(e) = b\},$$

and the self-similarity matrix

$$S_{ij} = \frac{|A_i \cap A_j|}{|A_i \cup A_j|}.$$

Rhythm axes.

$$\begin{aligned}
\textsc{Syncopation Rate} &= \frac{|\{(u,t) \in \mathcal{O} : t - \lfloor t \rfloor \neq 0\}|}{n_{\text{on}}}, \\
\textsc{Onset Density} &= \frac{n_{\text{on}}}{N_b}, \\
\textsc{Triplet Share} &= \frac{n_{\text{trip}}}{n_{\text{trip}} + n_{\text{bin}}}, \\
\textsc{Onset Position Entropy} &= H(\text{hist}(\text{round}((t(e) \bmod Q)/0.25))), \\
\textsc{Duration CV} &= \frac{\text{std}(D)}{\text{mean}(D)}, \\
\textsc{Mean Duration} &= \text{mean}(D), \\
\textsc{Density Variability} &= \frac{\text{std}((n_b)_b)}{\text{mean}((n_b)_b)}, \qquad n_b = |\{e : b(e) = b\}|.
\end{aligned}$$

Harmony axes. Let $w(p) = w_P(p)$, $W = \sum_p w(p)$, and $S_{\text{maj}} = \{0, 2, 4, 5, 7, 9, 11\}$. Split the piece into half-bars h_k and define $C_k = \text{prom}(w_{h_k})$. Let r_b be the lowest bass MIDI pitch in bar b, reduced modulo 12, and let $\tau_b = (r_{b+1} - r_b) \bmod 12$.

$$\begin{aligned}
\textsc{Chromaticism} &= 1 - \frac{\max_{r \in \{0,\ldots,11\}} \sum_{i \in S_{\text{maj}}} w((r+i) \bmod 12)}{W}, \\
\textsc{Distinct Pitch Classes} &= |\{p : w(p) > 0\}|, \\
\textsc{Pitch-Class Entropy} &= H((w(0), \ldots, w(11))), \\
\textsc{Chord Change Rate} &= \frac{|\{k : C_k \neq C_{k+1},\, C_k \neq \emptyset,\, C_{k+1} \neq \emptyset\}|}{2N_b - 1}, \\
\textsc{Chord Vocabulary Density} &= \frac{|\{C_k : C_k \neq \emptyset\}|}{N_b}, \\
\textsc{Root-Motion Entropy} &= H(\text{hist}((\tau_b)_b)), \\
\textsc{Fourth-Motion Rate} &= \frac{|\{b : \tau_b = 5\}|}{|\{\tau_b\}|}.
\end{aligned}$$

The diminished/augmented color axis is

$$\begin{aligned}
\textsc{Diminished-Augmented Color} &= \frac{D_{\text{dim}} + \min\{D_{\text{aug}}, N_b\}}{N_b}, \\
D_{\text{dim}} &= \sum_b \mathbf{1}\{\exists r : \{r, r+3, r+6\} \subseteq \text{prom}(w_b)\}, \\
D_{\text{aug}} &= \sum_b |\{r : \{r, r+4, r+8\} \subseteq \text{prom}(w_b)\}|.
\end{aligned}$$

with pitch classes interpreted modulo 12.

Melody axes. Let $\iota_j = s_{j+1} - s_j$ and $M = \{j : \iota_j \neq 0\}$.

$$\begin{aligned}
\textsc{Pitch Range} &= \max_{e \in P} m(e) - \min_{e \in P} m(e), \\
\textsc{Step Ratio} &= \frac{|\{j \in M : |\iota_j| \leq 2\}|}{|M|}, \\
\textsc{Interval Entropy} &= H(\mathrm{hist}((\min\{|\iota_j|, 12\})_j)), \\
\textsc{Ascending Ratio} &= \frac{|\{j \in M : \iota_j > 0\}|}{|M|}, \\
\textsc{Melody-Voice Range} &= \max_{e : v(e) = \text{melody}} m(e) - \min_{e : v(e) = \text{melody}} m(e).
\end{aligned}$$

If $M = \emptyset$, $\textsc{Ascending Ratio} = 0.5$.

Texture axes.

$$\begin{aligned}
\textsc{Voice Count} &= |V|, \\
\textsc{Mean Simultaneity} &= \frac{|P|}{n_{\text{on}}}, \\
\textsc{Maximum Chord Width} &= \max_{(u,t) : |P_{u,t}| \geq 2} \left(\max_{e \in P_{u,t}} m(e) - \min_{e \in P_{u,t}} m(e) \right), \\
\textsc{Active Voice Density} &= \mathrm{mean}_{b \in \mathcal{B}} |\{v(e) : b(e) = b\}|,
\end{aligned}$$

where $P_{u,t} = \{e : v(e) = u, t(e) = t\}$.

Form axes.

$$\begin{aligned}
\textsc{Self-Similarity} &= \mathrm{mean}_{i<j} S_{ij}, \\
\textsc{Novelty Rate} &= \mathrm{mean}_i (1 - S_{i,i+1}), \\
\textsc{Distinct-Bar Fraction} &= \frac{|\{A_b : b \in \mathcal{B}\}|}{N_b}.
\end{aligned}$$

For section density, let $L = \min\{4, \lfloor N_b/4 \rfloor\}$. Define a checkerboard novelty curve

$$\mathrm{nov}(c) = \frac{1}{|\text{cells}|} \sum_{\alpha, \beta \in [-L, L)} \mathrm{sgn}(\alpha, \beta) S_{c+\alpha, c+\beta},$$

where $\mathrm{sgn}(\alpha, \beta) = +1$ if $(\alpha < 0) = (\beta < 0)$, and -1 otherwise. A peak is a local maximum with $\mathrm{nov}(c) \geq \mathrm{mean}(\mathrm{nov}) + 0.5\,\mathrm{std}(\mathrm{nov})$. Then

$$\textsc{Sections per 100 Bars} = \frac{\#\text{peaks} + 1}{N_b} \cdot 100.$$

Within-song variation. Split the piece into W equal-bar windows. For each window w, compute a base vector x_w over the rhythm, harmony, melody, texture, and form axes. For each base axis a, let $\sigma_a = \mathrm{std}_w(x_w[a])$, and let SD_a be the corpus standard deviation of axis a. The within-song variation axis is

$$\textsc{Within-Song Variation} = \mathrm{mean}_a \frac{\sigma_a}{SD_a}.$$

Percentile fingerprint. For axis a, let $v_a = \text{axis}_a(P)$, and let col_a be the frozen 314-song corpus column for that axis. The percentile coordinate is

$$\text{pct}_a(P) = \text{round}\left(100 \cdot \frac{1}{314}\,|\{x \in \text{col}_a : x \le v_a\}|\right).$$

The fingerprint is $(\text{pct}_a(P))_{a=1}^{29}$. An axis is a degenerate extreme iff $\text{pct}_a(P) \le 5$ or $\text{pct}_a(P) \ge 95$.

Copy risk and gates. Represent a piece by $g[b] = \{(\text{round}(o(e), 2), m(e)) : b(e) = b\}$. For real song S,

$$A(g, g_S, \delta) = \frac{\sum_b |g[b] \cap g_S[b+\delta]|}{|P|}, \qquad \text{slide}(g, g_S) = \max_\delta A(g, g_S, \delta).$$

The copy-risk score is

$$\text{copy risk}(P) = \max\left\{\max_{S \in \text{cited}} \text{slide}(g, g_S), \max_{S \in \text{top25}} \text{slide}(g, g_S), \text{slide}(g, g_{\text{ref}})\right\}.$$

For genre g, gates are calibrated from real songs:

$$C1_g = \min\{6, \max\{3, \lceil Q_{0.85}(\text{extreme counts}(g))\rceil\}\},$$

$$F_g = \min\{6, \max\{3, \lfloor Q_{0.15}(\text{band occupancy}(g))\rfloor\}\},$$

$$T_g = \min\{0.45, \max\{0.30, 1.20 \cdot Q_{0.90}(\text{copy risk}(g))\}\}.$$

A generated piece passes the shared gates when $n_{\text{extreme}}(P) \le C1_g$, $\text{fit}(P, g) \ge F_g$, and $\text{copy risk}(P) < T_g$.