Self-Similarity Matrices and Localized Attention for Chorus Recognition: A Data-Efficient Music Information Retrieval Approach

José Daniel Luna Mena

Leiria, September of 2025

Self-Similarity Matrices and Localized Attention for Chorus Recognition: A Data-Efficient Music Information Retrieval Approach

Master's degree in Data Science

José Daniel Luna Mena

Project Report under the supervision of Professor Ricardo Malheiro

Leiria, September 2025

Originality and Copyright

This project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Partial reproduction of this document is authorized, provided that the Author is explicitly mentioned, as well as the study cycle, i.e., Master's degree in Data Science, 2023/2025 academic year, of the School of Technology and Management of the Polytechnic Institute of Leiria, and the date of the public presentation of this work.

Dedication

This project is dedicated to my family: my parents, José and Wilma; my siblings, Wilma and Marco; my niece, Matilda, and her mother, Lia; and my uncles, Marco and Miriam. Your unwavering support, encouragement, and trust were vital in helping me complete my studies abroad. This achievement is not mine alone.

Acknowledgments

I sincerely thank my advisor, Professor Ricardo Malheiro, and the professors of Informatics department for their guidance. I am also grateful to the Politécnico de Leiria and to the city of Leiria, which has been very welcoming during my studies. I also wish to thank my colleagues and friends for their support throughout this journey, and my family for always being there for me.

This work is funded by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of project MERGE (Music Emotion Recognition - Next Generation) - PTDC/CCI-COM/3171/2021.

Abstract

This project presents an efficient approach to chorus recognition in English song lyrics that achieves state-of-the-art performance with significantly fewer resources than existing methods. We developed a Bidirectional Long Short-Term Memory (BiLSTM) model with localized attention mechanisms, trained on only 780 songs compared to the 25,000+ songs typically used in Music Information Retrieval research.

Our approach addresses class imbalance through comprehensive stabilization techniques and leverages nine feature views capturing structural, semantic, and rhythmic patterns via self-similarity matrices. Through systematic experimentation, we demonstrate that chorus detection relies primarily on local contextual patterns rather than global structural awareness, with head self-similarity features (line beginnings) proving most critical for segmentation.

The BiLSTM + Attention model achieves 78.2% Macro F1 at the line level, matching Watanabe & Goto's (2020) performance with 100,000+ songs and significantly exceeding Fell et al.'s (2018) 67.4% F1 with 25,000 songs. For boundary detection, the model achieves 59.6% F1 for exact boundaries and 74.7% F1 with ±2 tolerance.

The research demonstrates that strategic data curation, comprehensive feature engineering, and targeted optimization can compete effectively with resource-intensive approaches, showing that local pattern recognition outperforms complex global modeling strategies in specialized domains like lyric analysis.

Keywords: lyric segmentation, chorus detection, attention mechanisms, self-similarity matrices, local pattern recognition.

Contents

List of Figures

List of Tables

List of Appendix

List of Abbreviations and Acronyms

BiLSTM	Bidirectional Long Short-Term Memory
CISUC	Centre for Informatics and Systems of the University of Coimbra
CMU	Carnegie Mellon University
CNN	Convolutional Neural Networks
CSV	Comma-Separated Values
CTC	Connectionist Temporal Classification
DTW	Dynamic Time Warping
ECE	Expected Calibration Error
EDU	Elementary Discourse Unit
ESTG	School of Technology and Management
FCT	Foundation for Science and Technology
JSON	Javascript Object Notation
JSONL	JSON lines
LCS	Longest Common Subsequence
LLM	Large Language Model
LoRA	Low-Rank Adaptation
LSR	Lyric Segmentation and Retrieval
MER	Music Emotion Recognition
MERGE	Music Emotion Recognition - Next Generation
MEVD	Music Emotion Variation Detection
MIR	Music information retrieval
NLP	Natural language processing
NLTK	Natural Language Toolkit
OCR	Optical Character Recognition
POS	Part of Speech
SOTA	State of the art
SSM	Self-Similarity Matrix
TPE	Tree-structured Parzen Estimator
URL	Uniform Resource Locator

1. Introduction

Music Information Retrieval (MIR) is a research field dedicated to developing methods that allow computers to analyze, classify, and retrieve information from music. Its applications span from large-scale recommendation systems, such as those used by Spotify, to specialized tasks like Music Emotion Recognition (MER), which seeks to identify the emotions conveyed by songs. For many of these applications, two dimensions are particularly important: audio and lyrics. Audio provides information about timbre, rhythm, and melody, while lyrics provide information about structural, semantic, stylistic characteristics. Equally crucial is the recognition that different sections of a song, such as verses, choruses, and bridges, often contribute differently to how music is perceived, remembered, or recommended. Understanding and segmenting these structures is therefore central to advancing MIR tasks.

Such segmentation has broader implications for music information retrieval (MIR) and music analysis. Understanding how verses and choruses function within lyrics can support applications ranging from structural music analysis to affective computing and recommendation systems. Moreover, a purely text-based approach contributes to existing literature by offering a lightweight alternative to multimodal methods, which often demand large-scale datasets and significant computational resources.

Despite the progress in multimodal approaches, their reliance on large-scale datasets and computationally demanding pipelines limits their accessibility and reproducibility. Similarly, the few existing text-only approaches often assume the availability of tens or hundreds of thousands of annotated songs, which restricts their applicability in low-resource settings. Yet, lyrics inherently encode structural and semantic cues that can signal boundaries between verses and choruses without the need for audio or multimodal data.

1.1. Problem statement and motivation

Lyrics segmentation, the task of dividing a song into structural units such as verses and choruses, is a fundamental problem in music information retrieval (MIR). Accurate

segmentation provides a foundation for applications including structural music analysis, lyric-based search, music emotion recognition (MER), and recommendation systems.

Text-only approaches offer an attractive alternative, as lyrics themselves encode semantic and structural cues that indicate where transitions between verses and choruses occur. However, the few text-based methods that have been proposed share a critical drawback with multimodal systems: they remain heavily data-dependent. Current state-of-the-art methods often assume the availability of tens or hundreds of thousands of annotated songs, making them impractical in low-resource scenarios.

This situation highlights a clear gap in the literature. Despite the potential of lyrics as a self-contained modality, there is still no established method that achieves accurate segmentation with minimal data and computation. The ability to address this challenge carries clear practical value. For example, segmentation can provide finer insights into how emotions are distributed across a song, enabling systems to detect whether the affective content of a chorus aligns with that of the overall lyrical content. Similarly, in large-scale applications such as music recommendation, the computational cost of processing entire audio and lyric streams for millions of new songs each day is prohibitive; efficient text-only segmentation could significantly increase the scalability of such tasks.

These considerations highlight both the scientific significance and the practical relevance of developing a data-efficient, text-only method for lyric segmentation. By exploring structural and semantic cues encoded in lyrics themselves, it becomes possible to design approaches that are not only accurate but also accessible in low-resource scenarios, broadening the applicability of segmentation systems in music information retrieval.

1.2. Objective

The goal of this project is to develop a data-efficient and computationally efficient method for segmenting song lyrics into two structural classes: verse and chorus. Unlike approaches that rely on audio features or multimodal strategies, this work focuses exclusively on text-based information, allowing us to explore the structural and semantic patterns encoded in lyrics alone. To achieve this, we design a modular and reproducible methodology that combines dataset construction, feature extraction, and the implementation of neural models, aiming to remain competitive with large-scale systems while requiring significantly fewer resources.

1.3. Research Questions

To guide the investigation, this project is structured around the following research questions:

- What computational methods are employed for segmenting lyrics using text alone?
- How effectively are these methods detecting structural elements such as verses, choruses?
- What are the most effective techniques for segmenting lyrics into musical parts (e.g., choruses, verses)?
- How do specific lyric components (e.g., choruses) influence model performance?

1.4. Contributions and limitations

This project makes several contributions: it introduces a modular pipeline for text-only lyric segmentation that integrates structural and semantic features through self-similarity matrices (see Appendix 10). It demonstrates that competitive segmentation performance can be achieved with fewer than one thousand annotated songs, addressing the challenge of data efficiency; it provides comparative analyses of multiple neural architectures (CNN, BiLSTM with attention, fine-tuned small transformer) under consistent evaluation metrics, and it offers a documented dataset construction process that supports reproducibility and future extensions.

At the same time, the study faces limitations. The dataset remains relatively small compared to large-scale corpora used in related work, and it focuses exclusively on English lyrics. Furthermore, the segmentation task is restricted to binary classification (verse vs. chorus), leaving out other structural elements such as bridges or intros.

1.5. Methodology And Resources

To address the research problem, we propose a modular pipeline for text-based lyric segmentation. The pipeline is organized around four main tasks:

- **Dataset construction:** Collect and preprocess song lyrics using web scraping techniques, followed by cleaning, segmentation, and storage in a standardized JSON Lines (JSONL) format.

- **Feature extraction:** Represent lyrics using multiple semantic and structural dimensions, condensing each song into specialized feature vectors that capture both surface-level and deep contextual information.
- **Model development:** Implement and compare three neural architectures, a convolutional neural network (CNN), a Bidirectional Long Short-Term Memory (BiLSTM) network with attention, and a parameter-efficient fine-tuned transformer using Low-Rank Adaptation[1] (LoRA) on Gemma-3 1B. The training pipeline is designed to be highly customizable, with configuration files that enable systematic exploration of hyperparameters, loss functions, feature combinations, and preprocessing strategies.
- **Evaluation and validation:** Train models on the constructed dataset, validate on held-out data, and analyze results with particular attention to data efficiency, generalization, and class imbalance.

This design provides a modular and resource-efficient training pipeline, enabling systematic experimentation with different features and architectures while maintaining a focus on data efficiency.

1.5.1. Dataset Construction

Objective: Collect and preprocess a clean, labelled dataset of song lyrics segmented into verses and choruses.

Activities:

- Song selection was restricted to English-language tracks, with duplicate entries removed to maximize coverage. The intent was to build a dataset representative of widely recognized popular music while maintaining consistent structural patterns (verse/chorus). Further analysis of artist distribution, stanza structure, and genre diversity is presented in Chapter 3.
- Collect lyrics using web scraping techniques on the Genius[2] website.
- Extract metadata including artist name, song title, full lyrics, and structural annotations.

[1] Low-Rank Adaptation (LoRA): A parameter-efficient fine-tuning method for large language models. It introduces small trainable low-rank matrices into specific layers of a pretrained model, enabling adaptation to new tasks with significantly fewer parameters.
[2] Is the world's biggest collection of song lyrics and musical knowledge.

- Preprocess and filter lyrics, normalize text (encoding, casing, punctuation), strip bracketed metadata (e.g., "[Chorus]", "(x2)"), remove artifacts and empty lines, harmonize labels (map all non-chorus to *verse*), discard malformed or incomplete songs, and deduplicate entries.
- Generate a comprehensive JSON file containing metadata and multiple levels of annotation, including line-level class, stanza-level class, raw class labels, and normalized labels (all non-chorus sections mapped to verse).
- Create a training-specific JSONL file with lyrics represented line by line and labels encoded as binary values (0 = verse, 1 = chorus).
- Song selection emphasized culturally influential and widely recognized tracks across decades and genres, ensuring both diversity and clear verse–chorus structures representative of popular music.

Tools and Technologies: Python, Selenium for web scraping, JSON libraries for data formatting.

1.5.2. Feature Extraction

Objective: Represent lyrics with rich semantic and structural information suitable for machine learning models.

- We draw inspiration from prior work on chorus detection in lyrics (Watanabe & Goto, 2020) and implement nine complementary features. These include contextual embeddings and word embeddings to capture semantic meaning, as well as multiple self-similarity matrices (SSMs) that describe different aspects of repetition and structure: head SSM, tail SSM, string SSM, line syllable SSM, syllable pattern SSM, phonetic SSM, and part-of-speech (POS[3]) SSM. Together, these features provide a multi-view representation of each song, combining surface-level similarity with deeper linguistic and rhythmic cues that help distinguish between verses and choruses.
- Construct per-line vectors and optional global descriptors.

[3] POS: part-of-speech tagging, the process of labeling words in a text with their grammatical categories (e.g., noun, verb, adjective, adverb).

Tools and Technologies: Python (NumPy, Pandas), spaCy (POS tagging), pretrained embeddings (word2vec, sentence transformers), phonetic libraries, and custom SSM extraction scripts.

1.5.3. Model Development

We selected three models that represent the primary approaches in the literature on text segmentation and related sequence labeling tasks. Convolutional Neural Networks (CNNs) have been widely adopted for their ability to capture local n-gram style patterns efficiently, making them a strong baseline in segmentation tasks (Fell et al., 2018). Bidirectional Long Short-Term Memory (BiLSTM) networks with attention mechanisms remain a standard approach for modeling sequential dependencies, especially when boundary detection requires contextual awareness across longer spans (Watanabe, 2018). More recently, transformer-based architectures have set the state of the art in numerous natural language processing tasks (Lukasik et al., 2020) In this work, we explore a lightweight alternative through parameter-efficient fine-tuning (LoRA) of Gemma-3 1B, which allows us to test whether large pre-trained language models can deliver strong segmentation performance under resource constraints. Together, these three families of models represent the dominant paradigms in current text segmentation research, ensuring that our comparisons cover both established and cutting-edge approaches.

Objective: Implement and compare multiple neural architectures for efficient lyrics segmentation.

Activities:

- Implement three architectures:
  - Convolutional Neural Network (CNN).
  - Bidirectional Long Short-Term Memory (BiLSTM) with attention and a boundary-aware loss function.
  - Parameter-efficient fine-tuned transformer (Gemma-3 1B) using Low-Rank Adaptation (LoRA).
- Configure models through a modular framework that supports hyperparameter sweeps, custom loss functions, and alternative feature combinations.

Tools and Technologies: PyTorch, configuration management (YAML/JSON/JSONL).

1.5.4. Evaluation and validation

Objective: Assess segmentation performance with a focus on both line-level accuracy and boundary-level quality, while highlighting data efficiency and robustness.

Activities:

- Per-line evaluation: Measure standard classification metrics (accuracy, precision, recall, F1) at the line level, including class-specific F1 for verses and choruses to analyze imbalance.
- Segmentation evaluation: Compute boundary-aware metrics that directly assess structural segmentation quality, including boundary F1 with different tolerance windows (±1, ±2 lines), *Pk*[4], and *WindowDiff*.[5] We also report line *F1*, chorus-specific *F1*, and verse-specific *F1* for a more detailed breakdown.
- Comparative evaluation: Compare results against state-of-the-art implementations reported in the literature, focusing on relative performance with significantly less training data.
- Ablation studies: Conduct systematic ablations to evaluate the contribution of individual feature groups, preprocessing strategies, and loss functions.

Tools and Technologies: Python (scikit-learn, NumPy), Matplotlib for visualization, custom evaluation scripts for boundary metrics.

1.5.5. Expected outcomes

- A clean and structured dataset of lyrics in a standardized format, suitable for experimentation with segmentation tasks.
- A set of feature representations that capture semantic, phonetic, and structural properties of lyrics from multiple perspectives.

[4] Pk metric (Probability of error): An evaluation measure for text segmentation. It calculates the probability that two units of text are incorrectly classified as being in the same segment or in different segments. Lower values indicate better segmentation quality.

[5] WindowDiff: A segmentation evaluation metric designed to address limitations of Pk. It applies a sliding window across the text to check whether the number of boundaries inside the window matches between the reference and the proposed segmentation.

- Baseline comparisons of neural architectures, providing insights into the relative strengths and limitations of CNNs, BiLSTMs with attention, and small LLMs[6] in a text-only setting.
- A modular and configurable training pipeline that supports systematic testing of different features, preprocessing strategies, and hyperparameter settings.
- Evaluation results across several metrics, offering an overview of model performance at both line and boundary levels, and illustrating the effect of limited training data.

1.6. Outline

The remainder of this project is organized as follows. Chapter 2 presents a systematic literature review of text-based lyric segmentation methods and their limitations. Chapter 3 describes the dataset construction process and comprehensive feature engineering, including structural, semantic, and phonetic feature extraction. Chapter 4 develops and evaluates various segmentation models, comparing BiLSTM, CNN, and fine-tuned LLM architectures through systematic experimentation. Chapter 5 concludes by presenting key findings on local context dependency, architectural insights, and the demonstration that state-of-the-art performance can be achieved with modest computational resources. The chapter also discusses the limitations of the current approach and outlines future work directions, including computational scaling, dataset expansion, multi-class segmentation extensions, and advanced architectural approaches, such as Mixture of Experts frameworks, for improved ensemble performance.

[6] Large Language Models (LLM): An AI model trained on massive text datasets to generate, summarize, translate, and answer questions in human language.

2. Systematic Literature Review

Segmenting music lyrics into meaningful structural components, such as verses and choruses, is crucial for various music information retrieval tasks. Historically, this process was often performed manually, requiring experts or annotators to identify structural boundaries by hand (Malheiro, 2016). While effective on a small scale, manual segmentation is time-consuming, subjective, and impractical when applied to the large collections of songs that modern MIR systems must process. This reliance on manual effort has underscored the importance of developing automatic segmentation methods. Although many approaches rely heavily on audio features, recent advances in text analysis techniques open new opportunities for purely text-based segmentation methods. However, the existing literature on lyric segmentation remains fragmented, lacking a comprehensive synthesis of purely textual methodologies. This systematic literature review aims to consolidate existing studies, identify gaps, and evaluate the efficacy of text-based segmentation methods applied specifically to song lyrics.

The primary objective of this systematic review is to synthesize the current state of text-based lyric segmentation techniques and to analyze the effectiveness of computational methodologies, such as neural networks and semantic embedding approaches, specifically applied to the task of lyric segmentation.

This review addresses the research questions presented in Chapter 1, focusing on text-based lyric segmentation and its computational methodologies.

2.1. Search Strategy and Implementation

This systematic literature review was conducted across four distinct databases: IEEE Explore, Semantic Scholar, Scispace, and Google Scholar. To ensure comprehensive coverage while maintaining search precision, four specialized search strings organized by subtopic were developed:

Lyric segmentation and structural analysis: (“Lyric Segmentation" OR "Text Segmentation in Lyrics" OR "Structural Analysis of Lyrics" OR "Chorus Detection in Lyrics" OR "Verse Segmentation in Lyrics" OR "Repetition Patterns in Lyrics")

Machine Learning for Lyric and Structural Analysis: ("Machine Learning" OR "Deep Learning" OR "Neural Networks") AND ("Lyric Segmentation" OR "Lyric Structure Classification" OR "Lyric Part Identification" OR "Chorus Detection in Lyrics")

Feature Extraction and Structure Analysis: ("Lyric Feature Extraction" OR "Semantic Features in Lyrics" OR "Syntactic Patterns in Lyrics") AND ("Lyric Segmentation" OR "Lyric Structure Analysis")

Identification of Musical Structures in Lyrics: ("Structural Components in Lyrics" OR "Verses in Lyrics" OR "Choruses in Lyrics" OR "Bridges in Lyrics") AND ("Lyric Structure Detection" OR "Text-based Song Structure")

The broad scope of this investigation necessitated dividing the search queries by topic. The first query focused specifically on structural segmentation of lyrics into verses, choruses, and repetition patterns. This approach was designed to capture studies addressing not only musical lyrics, but also similar structure analysis found in poems, which shares many characteristics relevant to segmentation techniques.

The second search string specifically targeted technological approaches by incorporating machine learning, deep learning, and neural network terminology alongside key lyric segmentation terms. Building upon this foundation, the third query delved deeper into feature extraction methodologies, a critical aspect of segmentation that would guide the selection of appropriate analytical tools for identifying structural elements.

The final query took a more generalized approach to identifying lyrical structures without specifying technological domains. This broad search aimed to gather foundational information about lyrical structures and the defining characteristics of segmentation boundaries, the conceptual lines that separate distinct structural elements within lyrics.

2.2.Search Refinement

Initial queries returned numerous irrelevant results, particularly from works limited to audio-only signal processing, image-based segmentation, or multimodal analyses not concerned with lyrical structure. To refine the search in line with our focus on chorus detection, exclusion terms were systematically applied to filter out studies that did not incorporate lyrical content. For example, search strings included clauses such as:

NOT ("audio-only" OR "image" OR "instrument")

This ensured the inclusion of studies, such as verses and choruses. Databases such as IEEE Xplore and Semantic Scholar support Boolean operators (AND, OR, NOT), allowing precise queries that emphasize terms like "lyrics," "chorus detection," and "segmentation." In contrast, Scispace and Google Scholar required manual keyword entry due to limited Boolean functionality, where keyword combinations were used directly to target works on lyric-based structural segmentation rather than low-level audio processing.

2.3.Synthesis Methods

Prior to conducting the searches, clear criteria were established to guide the selection of relevant literature:

Inclusion criteria:

- **IC1:** Papers discussing text segmentation methods, with a preference for those applicable to structured or repetitive texts (e.g., lyrics, poems, scripts).
- **IC2:** Papers specifically addressing structural analysis of music lyrics (e.g., identifying verses, choruses, bridges).
- **IC3:** Papers exploring computational methods (e.g., NLP, machine learning, rule-based systems) for text or lyric segmentation.
- **IC4:** Include papers on audio–lyric synchronization that explicitly address structural boundaries or lyrical sections (e.g., verse–chorus transitions, line boundaries, or section onsets).
- **IC5:** Published between 2013 and 2025.
- **IC6:** Written in English and accessible in full text.

Exclusion criteria:

- **EC1:** Studies focused solely on audio-based song segmentation (e.g., using pitch, tempo).
- **EC2:** Papers on image text segmentation (e.g., OCR-based segmentation).
- **EC3:** Papers where the full text was unavailable or behind an inaccessible paywall.

These criteria were applied throughout the screening process to ensure systematic and consistent selection of relevant studies.

2.4.Search Results and Screening Process

The initial database search yielded a total of 13,838 records (see Table 2.1). Given the high volume of results from Google Scholar (13,580), we limit the screening to the first 3 pages, and a pre-screening limit was established to restrict Google Scholar results to the most relevant entries (first few pages per query), resulting in the removal of 13,783 records prior to the manual screening phase.

This process left 53 distinct documents for title screening. After reviewing titles against the inclusion criteria, 26 records were excluded, leaving 27 documents for abstract analysis. The abstract screening phase resulted in the exclusion of 9 additional papers that did not meet the specific criteria regarding text-only segmentation or English language availability.

Consequently, 18 papers were sought for full-text retrieval and detailed eligibility assessment. Upon in-depth review, 10 studies were excluded primarily due to a reliance on multimodal approaches or a focus on classification rather than segmentation. This resulted in a final selection of 8 studies included in this review.

Search Topic	Total Found	Total Analyzed	Total Selected	Final Selection
Lyric Segmentation (QS1)	3208	22	12	5
Machine Learning & Deep Learning (QS2)	3448	11	6	2
Feature Extraction (QS3)	6108	14	5	1
Musical Structures (QS4)	1074	6	4	0
Total	13838	53	27	8

Table 2.1 - Search Results

The search results, detailed in Table 2.1, show that QS3 on feature extraction yielded the highest number of initial results, comprising 44% of all screened literature. In contrast, Query 4 (QS4) on musical structure in lyrics yielded the fewest results, accounting for only 8% of the information found. However, after the screening process, the distribution shifted

significantly. QS1, which employed specific terminology directly related to the research focus, ultimately contributed 42% of the final selection of papers.

2.5.Selection Methodology

The consideration and screening phases followed a systematic three-step approach:

Title screening: Initially, all documents retrieved from the search databases underwent a rigorous screening process based solely on their titles. Given the substantial volume of search results, particularly those originating from Google Scholar, a pragmatic limitation was imposed: screening was restricted to the first few pages of results. This decision was methodologically justified, as documents retrieved beyond these initial pages typically exhibited limited relevance to the core research topic. Only those documents whose titles contained explicit or closely related keywords were advanced to the subsequent, preliminary "Abstract screening" phase.

Abstract screening: The selected documents were then screened again using their abstracts. At this stage, the inclusion criteria emphasized strictly textual segmentation studies, immediately excluding any studies that were vague, multimodal, or not in English.

Full-text screening: Finally, documents that passed abstract screening were subjected to a detailed review of their introduction and conclusion sections. Documents unavailable in full-text form were immediately excluded (see Figure 2.1).

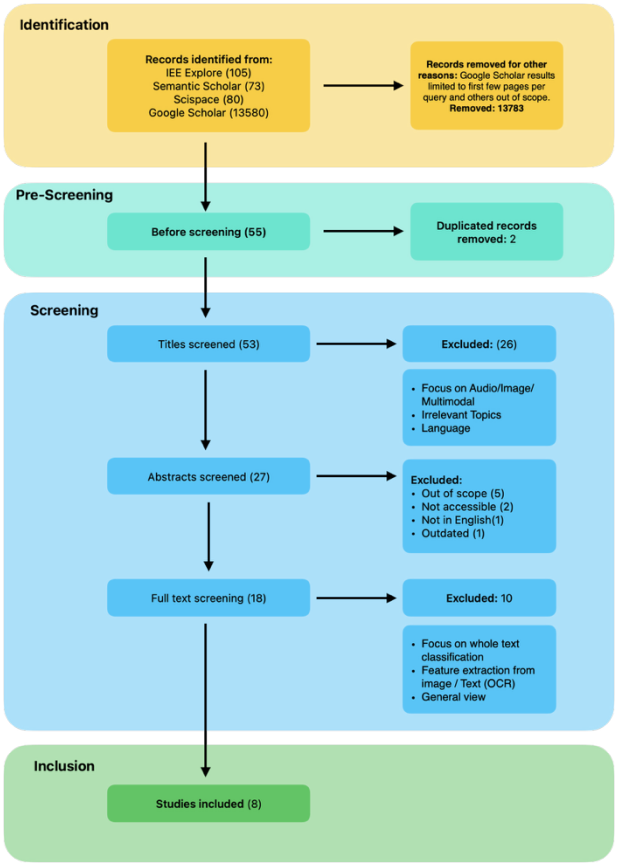


Figure 2.1 - PRISMA Flow Diagram

2.6.Search Strategy Validation and Sensitivity

Lighter or simplified versions of each query were tested to ensure that the designed query strings were neither overly restrictive nor unintentionally excluded relevant papers. These simplified queries typically involved reducing complexity by removing some OR operators or adding quotation marks to enforce exact keyword matching. This step aimed to verify whether the strictness of the queries significantly limited the search results.

2.7.Certainty Assessment

A key observation repeatedly emerged throughout the search process: many papers consistently appeared across multiple queries. While occasionally, less restrictive queries uncovered additional interesting papers, most of these additional results were ultimately outside the precise scope of this research (focusing instead on related yet distinct topics such as lyric classification, multilingual song analysis, or structural segmentation of poetry rather than lyrics). This consistency reinforced confidence in the comprehensiveness of the initial

search approach, suggesting that the queries effectively captured the core literature relevant to the intended research objectives.

2.8. Key Studies and Emerging Themes

The comprehensive screening process resulted in 27 papers selected for abstract review. Upon closer examination of the abstracts and subsequent full texts, 18 papers were deemed appropriate for full-text analysis. Ultimately, 8 studies were selected for the final qualitative synthesis, with 10 papers excluded primarily due to their reliance on multimodal approaches or a focus on whole-text classification rather than linear or boundary segmentation (see Figure 2.1).

2.8.1. Text-Only, Line-Level Segmentation

We define text-only lyrics segmentation as a line-level sequence labelling problem, where the goal is to identify structural boundaries (e.g., verse-to-chorus transitions) based solely on textual content. Prior work highlights two closely related subtasks: boundary detection (Baratè, Luca, & Santucci, 2013), which identifies lines that mark the end of a section, and chorus span detection, which labels contiguous lines as chorus sections. To prevent models from exploiting shortcuts like layout artifacts, state-of-the-art pipelines strip empty lines, explicit tags (e.g., "(chorus)"), and repeat markers, compelling reliance on intrinsic textual signals such as repetition and semantics. For instance, one key study addresses chorus detection in plain text without labels or empty lines, integrating repetition-aware structural features with linguistic embeddings from word2vec and context2vec (Watanabe & Goto, 2020). This approach underscores the language-independent nature of repetition patterns in lyrics, achieving strong cross-lingual transfer (e.g., Japanese-trained models performing well on English). Experimental results demonstrate that combining structural (e.g., self-similarity matrices) and linguistic features outperforms baselines, with the model formulating the task as predicting chorus/not-chorus status per line using a neural sequence labelling architecture.

Another semantics-driven approach processes lyrics by extracting basic formatting and applying rules for fault-tolerant reconstruction, outputting labelled sections based on content similarity. It emphasizes identifying recurrent groups without assigning specific labels like "verse" or "chorus," instead using symbols (A-B-A) to denote repetitions. It discusses common song structures in popular music (e.g., verse-chorus-bridge) (Baratè et al., 2013).

2.8.2. Lyrics-Specific, Text-Only Segmentation

We define text-only lyric segmentation as a line-level sequence labeling problem, where the goal is to identify structural boundaries (e.g., verse–chorus transitions) using only the lyric text without relying on audio or layout metadata. Within this formulation, prior work highlights two closely related subtasks: boundary detection, which aims to identify the lines that mark the end of a lyrical section, and chorus span detection, which labels contiguous lines as belonging to the chorus. To prevent models from exploiting trivial layout artifacts, state-of-the-art pipelines deliberately strip empty lines, explicit markers such as "(chorus)" and repeat annotations (e.g., "x2"), thereby forcing reliance on intrinsic textual cues, including lexical repetition, phonetic patterns, syntactic templates, and semantic coherence.

Early work in this area explored rule-based and semantics-driven approaches. One study by Baratè et al. (2013) proposed an algorithm that parses lyrics into lines and reconstructs sections using formatting cues (e.g., carriage returns \r) combined with similarity rules for fault-tolerant reconstruction. Rather than assigning explicit section labels such as "verse" or "chorus," their method outputs abstract symbolic structures (e.g., A–B–A), capturing repetition-based macrostructures common in popular music. This framework emphasizes identifying recurrent groups without requiring genre-specific annotations, and the authors note that such symbolic representations align with widely observed organizational schemes such as verse–chorus–bridge forms.

Subsequent studies shifted from symbolic reconstruction to feature-based and neural models that directly exploit textual repetition patterns. A landmark contribution by Watanabe & Goto (2020) introduced a chorus detection method for lyrics text that integrates repetition-aware structural features with semantic embeddings (word2vec and context2vec). Their pipeline deliberately removes formatting shortcuts (such as empty lines and explicit chorus tags) to ensure that detection relies solely on intrinsic lyric properties. Crucially, they demonstrate the language-independent nature of repetition patterns in lyrics: models trained on Japanese corpora transfer effectively to English lyrics, underscoring the universality of textual repetition as a signal of a chorus. Their experiments also show that combining structural features (e.g., self-similarity matrices, SSM) with linguistic embeddings substantially outperforms baselines, with the task formulated as per-line classification of "chorus" vs. "non-chorus" using a neural sequence labeling architecture.

Building on this insight, convolutional neural networks (CNNs) have been applied to the detection of lyrics macrostructure. A paper by Fell et al. (2018) designed a CNN-based system that leverages self-similarity matrices to capture repeated motifs and jointly infer complete song structures. Their implementation defines segmentation as classifying each line as a segment border, using a large feature set of approximately 240 descriptors. These include three SSM views, string similarity via normalized Levenshtein distance[7], phonetic similarity via the Double Metaphone algorithm, and lexicon-structural similarity, which combines bigram overlap and POS bigrams, augmented with traditional features such as character counts, TF-IDF n-grams[8], and part-of-speech n-grams. Evaluated on English lyrics from large corpora, including the Music Lyrics Database (102,802 songs) and the WASABI corpus (743,939 songs), their CNN demonstrated strong performance across genres and motivated applications in music information retrieval systems, such as search engines that allow structural queries over lyrics.

In parallel, researchers have pursued large-scale data generation strategies to overcome the scarcity of annotated lyric corpora. One notable study generated training data for 100,772 songs by aligning lyrics with audio-based chorus detection systems, then transferring those annotations back into the text domain. Their neural models combine multi-view structural features (nine SSMs derived from string similarity, phonetic similarity via the CMU Pronouncing Dictionary, part-of-speech similarity via NLTK[9], semantic embeddings from word2vec and context2vec, and rhythmic cues such as syllable counts via dynamic time warping) with linguistic features, including averaged word embeddings and sequence embeddings. Results confirm that chorus sections are indeed detectable from text alone, with repetition-based cues proving robust across languages and genres.

Across these works, several themes consistently emerge. First, repetition is the central organizing principle in lyrics segmentation. Whether captured through SSMs, longest common subsequences, or symbolic A–B–A structures, repeated units form the backbone of computational models. Second, multi-view fusion (integrating string, phonetic, syntactic,

[7] Levenshtein distance: A string similarity metric that calculates the minimum number of single-character edits (insertions, deletions, substitutions) required to transform one string into another. Commonly used in text processing, error correction, and natural language tasks.
[8] TF-IDF n-grams: term frequency–inverse document frequency representation applied to contiguous word sequences (n-grams), weighting them by how frequent they are in a document compared to their rarity across the corpus.
[9] NLTK (Natural Language Toolkit): a popular Python library for NLP. Provides tokenizers, POS taggers, and utilities used for text preprocessing.

semantic, and rhythmic similarity) provides resilience against real-world variability, such as paraphrased choruses, typos in web-scraped lyrics, or orthographic variation. Third, these approaches reveal domain-specific challenges: genres like rap, which often feature low literal repetition, remain difficult for repetition-driven models, and reliable supervised data continues to be a bottleneck for robust evaluation. Metrics also matter, while F1 scores for boundary detection are standard, near-miss tolerances (±1 or ±2 lines) are often applied implicitly, acknowledging the annotation ambiguities inherent in lyrical text.

Together, these developments establish a clear trajectory: from early symbolic rule-based systems through feature-rich CNNs to cross-lingual neural sequence models, all converging on the insight that repetition remains the most powerful signal for text-only, line-level lyric segmentation.

2.8.3. Audio-Based Alignment for Lyric Segmentation

Whereas text-only segmentation relies on semantic and repetition cues, audio-based segmentation exploits the acoustic realization of lyrics to reveal higher-level structural boundaries such as verses and choruses. This perspective treats segmentation as an extension of lyrics-to-audio alignment, where temporal precision is critical: small errors can blur the distinction between a repeated chorus hook and the tail of a verse. Singing introduces unique challenges compared to speech, including expressive variation in pitch, elongated vowels, and dense instrumental accompaniment Cheng et al. (2025). At the same time, these musical attributes provide powerful cues: repeated pitch contours and rhythmic regularity often signal chorus material, while dynamic range and timbral variation help distinguish contrasting sections.

Recent work in alignment has demonstrated how audio-informed models can achieve the temporal precision necessary for segmentation. Stoller et al. (2019) introduced an end-to-end *Wave–U–Net–based*[10] system that predicts character probabilities directly from polyphonic audio, trained using Connectionist Temporal Classification (CTC) loss. Building on this, Durand et al. (2023) proposed a contrastive learning framework that embeds lyrics and audio into a shared space, achieving under 0.2 second error on the *Jamendo benchmark*[11]

[10] Wave-U-Net is a 1D U-Net-style convolutional architecture that operates directly on raw time-domain signals (originally audio) to produce per-sample outputs (e.g., separated sources or enhanced speech).
[11] Jamendo benchmark: A publicly available dataset of real-world polyphonic music with time-aligned lyrics, widely used to evaluate lyrics-to-audio alignment systems under challenging, multi-instrument conditions.

and demonstrating cross-lingual robustness. Cheng et al. (2025) further reduced imprecision by introducing a masked frame-wise cross-entropy loss, leveraging partial phoneme labels at word onsets and offsets to reach median alignment errors as low as 0.041 seconds. At this resolution, alignment accuracy supports reliable detection of short and repeated chorus phrases, a critical component of segmentation.

Other approaches explicitly integrate musically meaningful features to guide boundary detection. Huang et al. (2022) showed that modeling pitch jointly with phoneme recognition sharpens lyric onsets, since note attacks often coincide with new syllables. This auxiliary signal not only improves alignment but also provides chorus-specific cues, as chorus sections typically exhibit more stable and repeated pitch trajectories compared to verses.

In practice, aligned audio data can be fed into higher-level models (such as sequence labelers or classifiers) that aggregate word or line-level predictions into structural categories like verse or chorus. This layered pipeline allows segmentation systems to move beyond textual repetition, grounding decisions in performance-specific cues. For example, a system could identify a repeated lyric line in the text, then verify its chorus status by detecting consistent melodic repetition in the audio.

Looking forward, audio-based methods open pathways for expanding Lyric Segmentation and Retrieval (LSR) beyond purely textual features. Future segmentation models may fuse linguistic repetition patterns with audio-derived cues such as pitch stability, rhythmic regularity, and timbral contrast. By uniting textual and acoustic evidence, such hybrid approaches promise more robust, language-independent verse/chorus segmentation across diverse musical styles.

2.9. Discussion

The current state of text-based lyric segmentation is represented by several significant contributions that demonstrate the field's evolution from traditional methods to advanced neural approaches. Landmark research established foundational neural network models for chorus detection using pre-trained embeddings, while subsequent work advanced discourse segmentation through transformer architectures with cross-segment attention. The prevalence of LSTM-based approaches highlights both the effectiveness and limitations of these models in handling diverse textual structures. Semantic analysis remains central to the field, with early work utilizing similarity metrics, such as Levenshtein distance, for boundary

detection. Recent surveys document the transition toward embedding-based methods and large language models, positioning contemporary research within a rapidly evolving landscape that increasingly favors neural architectures over traditional rule-based approaches. These studies collectively demonstrate the field's progression from simple similarity measures to sophisticated contextual understanding, though challenges remain in model flexibility and evaluation standardization.

A key limitation of this research stems from its specific focus on lyrics segmentation techniques, particularly identifying boundaries between verses and choruses within song lyrics. While exploring the literature, many studies on general text segmentation were considered but ultimately excluded, as they primarily addressed whole-text classification tasks, such as determining a song’s genre based on its lyrics, rather than within-text segmentation, which involves detecting structural boundaries inside the text itself.

Another important clarification relates to the deliberate focus on studies involving English lyrics. Several innovative segmentation approaches were not incorporated because they were implemented in other languages, particularly Chinese, which would have introduced additional variables into the analysis framework.

Moreover, during the final stages of screening, it became apparent that certain essential keywords and terminologies, such as "discourse segmentation" and "sentence segmentation," had initially been overlooked. Recognizing these terms late in the screening process highlighted their critical role in understanding and effectively implementing core segmentation concepts, suggesting potential pathways for future research to explore.

2.10. Final Selection

The final corpus of selected studies encompasses diverse approaches to text-based lyric segmentation, ranging from traditional computational linguistics techniques to cutting-edge deep learning architectures. This literature provides a solid foundation for understanding both the historical development of the field and its current frontiers, enabling the identification of gaps and opportunities for further research into purely text-based approaches to lyric structure analysis.

3. Dataset Construction

The effectiveness of any lyric segmentation model depends critically on the quality and representativeness of its training data. Building a reliable dataset required not only collecting enough songs but also ensuring that the materials captured diverse lyrical structures, balanced annotation quality, and compatibility across sources. To this end, the construction process began from a seed list of 1,000 English-language songs, filtered to avoid repetition, which provided a broad and culturally representative starting point for the scraped corpus. From there, two complementary datasets were integrated: a large corpus of scraped lyrics that emphasized breadth and diversity, and a smaller curated validation set designed to provide high-quality reference annotations. Following the collection, systematic preprocessing steps were applied to standardize formats and normalize labels. This chapter details the sources of the datasets, the automated collection pipeline, and the preprocessing and standardization strategies employed to construct the final corpus.

3.1. Data Sources and Collection Strategy

The dataset compilation involved two distinct sources that served complementary purposes in the research design. The primary source consisted of 125 professionally curated songs provided by the supervising professor, which were subsequently reserved exclusively for validation purposes due to their superior annotation quality. The secondary source comprised 1,000 popular and influential English songs, collected through systematic web scraping, which formed the foundation of the training dataset.

The initial pool of 1,000 songs was compiled under two basic criteria: songs had to be in English and not repeated within the dataset. Although no explicit filters were applied for time period or genre, the list naturally included a wide range of decades and styles, reflecting the cultural prominence of popular Western music. This broad, if imperfect, coverage provided diversity in lyrical structure, while subsequent filtering and preprocessing ensured consistency for segmentation tasks. Chapter 3.4 presents detailed statistics on artist distribution, stanza length, and structural balance, offering insight into the dataset's representativeness and the generalizability of the models trained on it.

3.2. Automated Data Collection Pipeline

The automated data collection pipeline was engineered to efficiently acquire and preprocess lyric data from online sources, forming a critical foundation for the dataset used in this study. This pipeline integrates modular components to streamline the extraction, validation, and refinement of song lyrics, ensuring scalability and reliability while minimizing manual intervention. By leveraging web-based resources and custom algorithmic techniques, the process addresses the challenges of inconsistent data availability and structural variability, ultimately supporting the development of a robust dataset tailored for chorus recognition analysis. The following subsections detail the implementation of web scraping and the quality control measures that underpin this approach.

3.2.1. Web Scraping Implementation

The web scraping implementation was designed to efficiently gather lyric data from Genius.com, forming the backbone of the automated data collection pipeline. The process began with the compilation of an initial list of 1,000 songs, selected to represent a diverse set of English-language tracks that have had significant cultural and lyrical influence. This list was generated using an automated ranking tool prompted to identify influential songs across genres and eras, providing a broad foundation for the dataset (see Fig. 3.2 for artist distribution). A custom web scraping script was then developed to query the Genius.com website for each song entry systematically.

To clarify the segmentation process, we examined how Genius structures its lyric pages. As described on the platform’s official site, lyrics are collaboratively transcribed by the community and later reviewed or verified by editors and, in some cases, by the artists themselves. This rigorous editorial workflow not only enhances reliability but also produces a normalized, widely accepted format for the published lyrics. Leveraging this standardization, we powered our parser with the site's consistently reliable structure, which held true across diverse song types and enabled straightforward extraction of any lyric into our desired initial raw format.

Finally, ensure accurate song identification and prevent mismatched annotations, the system implemented a dual similarity matching algorithm using Python's *difflib*[12] library. For each

[12] difflib: a Python standard library module that provides tools for comparing sequences, commonly used to generate text differences, similarity ratios, and unified or context diffs.

search result, the algorithm computed separate similarity scores for both artist names and song titles, subsequently combining these into a unified similarity metric. Only results exceeding the predetermined threshold were retained, with their corresponding URLs stored for subsequent lyric extraction.

3.2.2. Quality Control and Error Handling

The pipeline incorporated comprehensive error handling mechanisms, flagging problematic entries when songs were not found, similarity scores fell below the threshold, search results were empty, or detail URLs were inaccessible, achieving a success rate of 98.6% and producing 948 raw text files with song metadata, lyrics, and structural segmentation labels, which were subsequently filtered and cleaned to 780 files by removing songs without segments and addressing overall parsing issues, resulting in a more robust and consistent dataset.

The modular design proved essential for managing rate limiting and resolving step-specific issues without requiring the entire collection process to be restarted. Each stage generated checkpoints for subsequent processing phases, ensuring robust data collection despite the inherent volatility of web scraping operations.

3.3. Data Preprocessing and Standardization

After collection, the raw lyrics required systematic preprocessing to ensure structural consistency and interoperability across different data sources. Preprocessing served two main purposes: to transform heterogeneous lyric files into a uniform schema that preserved relevant metadata and structural annotations, and to harmonize labels and representations in a way that supported binary classification between verse and chorus.

In addition to the scraped corpus, a curated validation dataset was incorporated into this study. This dataset originates from the CISUC research group at the University of Coimbra and forms part of the FCT-funded MERGE project, where a new bimodal corpus for Music Emotion Variation Detection (MEVD) is being manually constructed by a team of annotators. Although not yet published, these resources provide a reliable, high-quality reference for validation, complementing the larger scraped dataset and strengthening the overall experimental design.

3.3.1. Structural Formatting

Following initial collection, the raw data underwent comprehensive preprocessing to establish a consistent structural format. A JSON-based schema was implemented, with each song represented as an object containing metadata (name, artist, original filename) and a hierarchical list of stanzas.

Each stanza entry preserved both the original label from source metadata and a standardized parsed label. The parsing process systematically removed special characters and normalized all non-chorus labels to "verse" while preserving chorus designations. This binary classification approach, while simplified, aligns with the project's primary objective of chorus detection.

3.3.2. Validation Dataset Processing

The professor-provided dataset was distributed in ".ass" karaoke format, reflecting its alignment with audio in the MERGE project. To repurpose it for lyric segmentation, a custom parser extracted line-level annotations into CSV files with two fields: section type (chorus or verse) and lyric text. Non-chorus categories were standardized to "verse," preserving chorus labels without alteration. The resulting CSVs were then mapped into the same JSON schema used for the scraped corpus, ensuring full compatibility for feature extraction and subsequent model training.

3.3.3. Dataset Composition and Filtering

The combined preprocessing pipeline produced 780 songs in the training dataset after filtering out problematic entries including unlabeled songs or those lacking sufficient stanza structure. Including the 125 validation songs, the final corpus consisted of 905 songs with detailed structural annotations at both the stanza and line levels.

3.4. Dataset Analysis and Characteristics

The success of any lyric segmentation model depends critically on the quality of its dataset and the richness of its feature representations. Building a corpus suitable for training required not only gathering a sufficiently large and diverse collection of songs, but also ensuring that these lyrics were processed into a clean, consistent, and structurally informative format. This chapter details the end-to-end pipeline designed for that purpose. We begin with dataset

construction, describing how raw lyrics were collected, filtered, and standardized into a binary verse/chorus schema. Next, we analyze the characteristics of the dataset, highlighting its distributional properties and structural representativeness. Finally, we present the feature engineering strategy, including the design of nine complementary feature views and their transformation into self-similarity matrices (SSMs).

3.4.1. Label Distribution Analysis

Initial label analysis revealed significant diversity in original annotations, with many variants incorporating artist names or detailed descriptors (e.g., "[Chorus: Noodle & 2D]"). Despite preprocessing efforts, 12% of labels were classified as "other" due to their heterogeneous nature (see Figure 3.1). The standardized binary classification resulted in a distribution of 68.5% verses and 31.5% choruses, reflecting the inherent structural characteristics of popular music where verses typically outnumber choruses.

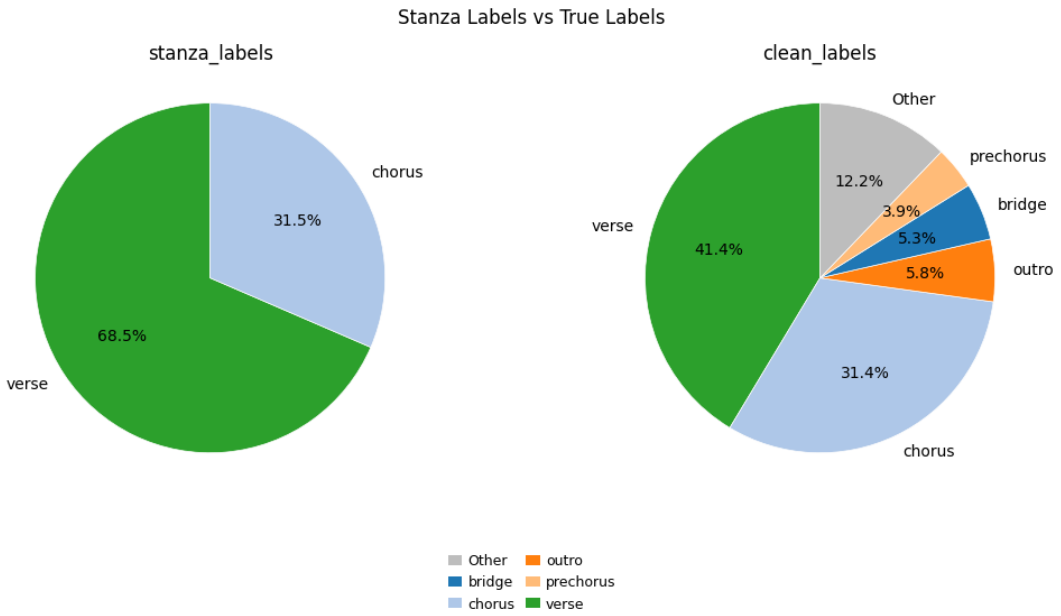


Figure 3.1 - Stanza Labels Distribution

3.4.2. Artist Representation

The artist distribution analysis reveals a critical aspect of dataset quality for machine learning applications. While Bob Dylan (23 songs) and The Beatles (20 songs) represent the highest frequencies in the corpus, the rapid decline to single-digit representation for remaining artists demonstrates effective bias mitigation. Specifically, 18 of the top 20 artists contribute 5-8

songs each, with most of the 905-song corpus distributed across hundreds of different artists (see Figure 3.2).

This distribution pattern is crucial for lyrics segmentation because different artists employ distinct structural approaches. The relatively low concentration prevents the model from overfitting to specific artistic conventions while still providing sufficient examples of consistent structural patterns within individual artist repertoires.

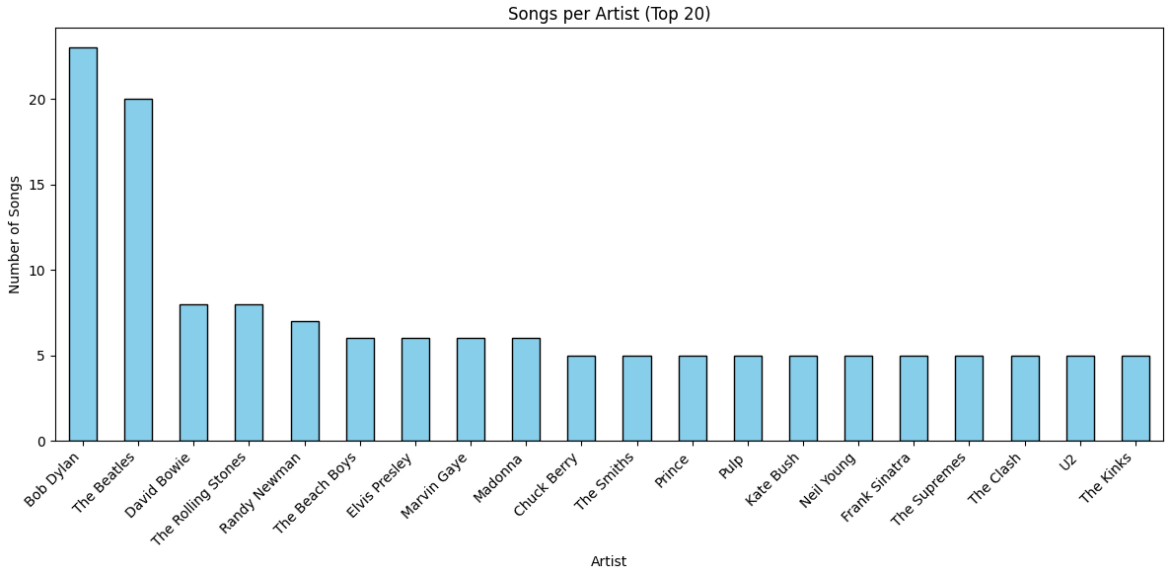


Figure 3.2 - Song Per Artist

3.4.3. Content and Length Distribution

The line count distribution exhibits a strong right-skewed pattern with most songs (approximately 75%) containing fewer than 100 lines, while maintaining representation across the full spectrum up to 500+ lines. This distribution reflects the natural variation in song complexity, providing the segmentation model with exposure to both typical and edge cases (see Figure 3.3 A).

The stanza count distribution follows a more concentrated pattern, with the peak occurring at 6-8 stanzas per song (approximately 275 songs combined at the peak). This concentration, spanning approximately 6-8 stanzas, aligns with the standard popular music structure (intro-verse-chorus-verse-chorus-bridge-chorus-outro), suggesting that the dataset accurately represents conventional songwriting patterns. The presence of songs with 1-5 stanzas covers simpler structures, while the tail extending to 30+ stanzas ensures the model encounters complex, multi-section compositions (see Figure 3.3 B).

Importantly, this distribution provides balanced training data across structural complexity levels, preventing bias toward either overly simplistic or excessively complex song structures.

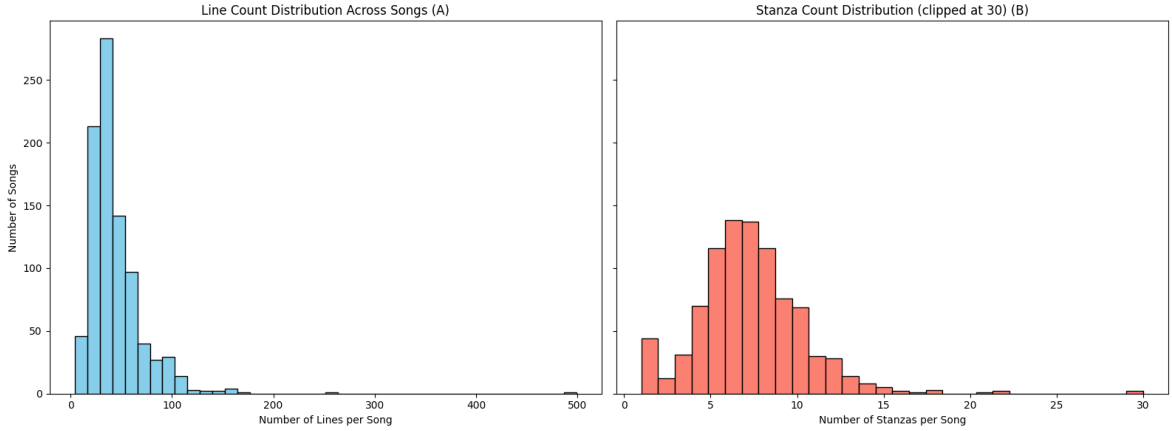


Figure 3.3 - Number of Lines / Stanzas per song

3.4.4. Structural Pattern Analysis

Stanza Length Distribution: The dataset exhibited a right-skewed distribution of average stanza lengths, with a median of approximately 5.5 lines per stanza and representation across the full range from 2 to 17 lines. This variability provides adequate examples of both typical song structures and edge cases, supporting robust model generalization across diverse structural patterns (see Figure 3.4).

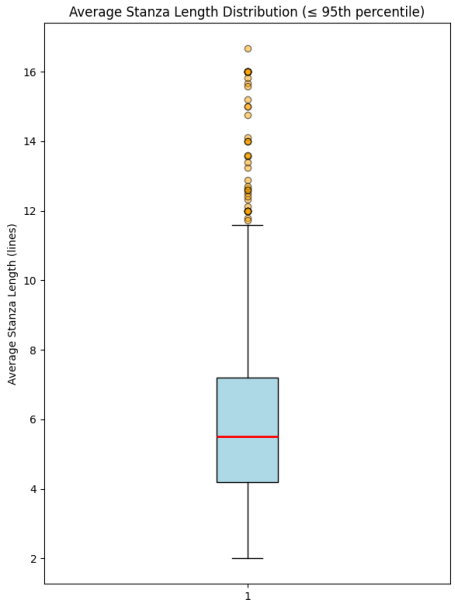


Figure 3.4 - Average Stanza Length Distribution

Verse-Chorus Balance: Structural analysis revealed reasonably balanced distributions across songs, with most containing 2-6 verses and 0-4 choruses. The consistency of verse-to-chorus ratios across songs of varying complexity (stabilizing around 65-70% verses, 30-35% choruses) suggests this proportion represents a structural equilibrium in popular music (see Figure 3.5).

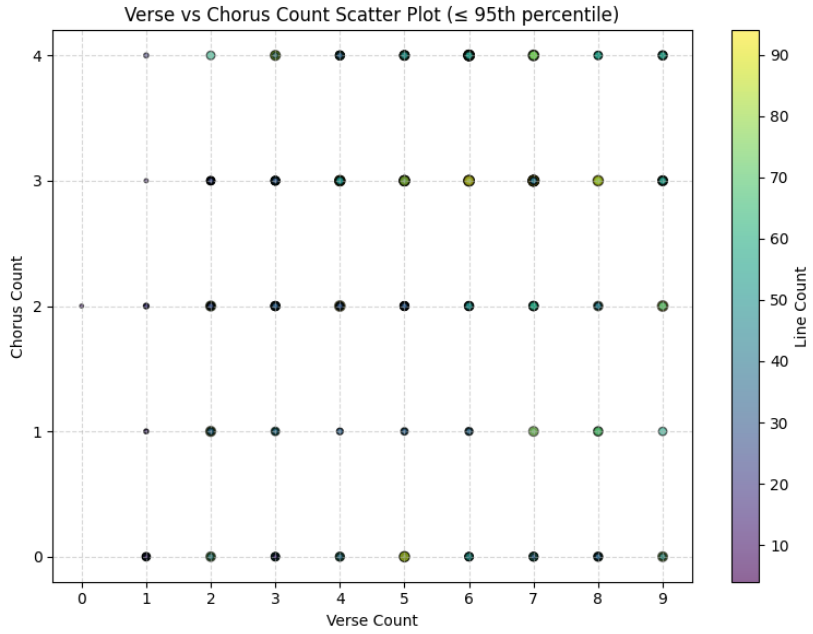


Figure 3.5 - Verse vs Chorus Scatter Plot

Structural Patterns: Analysis of the top 15 song structure patterns demonstrated comprehensive coverage of common organizational schemes in popular music, from simple verse-only arrangements to complex alternating patterns. This diversity provides the segmentation model with exposure to varied organizational patterns typical of the genre (see Figure 3.6).

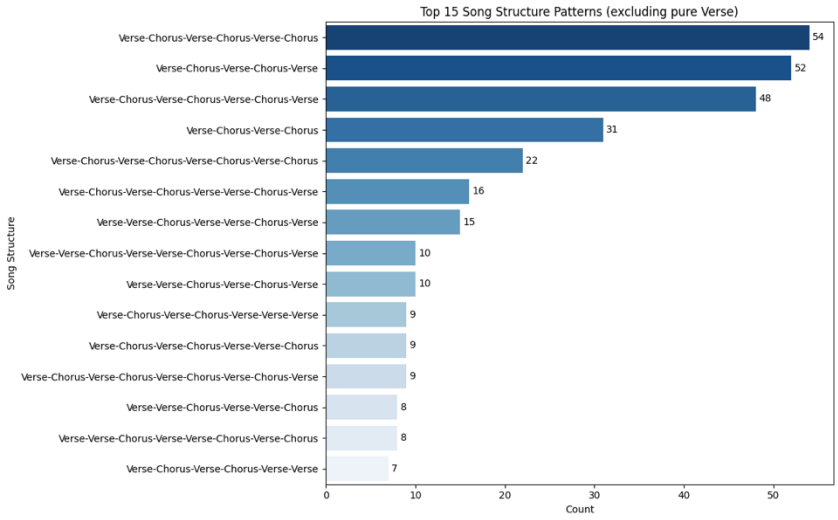


Figure 3.6 - Song Structure Patterns

3.4.5. Dataset Validation and Quality Assessment

The final dataset demonstrates an effective representation of critical structural features essential to lyrics segmentation, despite being substantially smaller than comparable projects that utilize datasets 100 times larger (see Table 3.1). The balanced distribution of stanza lengths, consistent verse-to-chorus ratios across songs of varying complexity, and comprehensive coverage of common structural patterns provide sufficient diversity for robust model training.

This structural representativeness, rather than raw dataset size, appears to be a key factor enabling competitive performance despite scale limitations. The systematic collection methodology, comprehensive quality control measures, and thorough analytical validation establish confidence in the dataset's suitability for training effective lyrics segmentation models.

Dataset	**Song Count**
WASABI corpus	743,939
MLDB (Music Lyrics Database)	102,802
Project Lyrics Dataset	780

Table 3.1 - Dataset Comparison

4. Feature Engineering Strategy

Feature engineering constitutes a critical component in developing an efficient model for chorus extraction from text-only lyrics, particularly when operating within the constraints of our 900-song dataset. Drawing from prior work on lyrics segmentation (Baratè et al., 2013), which serves as a foundational prerequisite for identifying choruses as repeated sections, we extract nine features, referred to as "views" in Watanabe & Goto (2020), at the line level. These views enable us to compute nine self-similarity matrices (SSM), facilitating the detection of repetitions and boundaries essential for chorus identification while achieving comparable performance to systems trained on datasets exceeding 100,000 songs.

Our feature engineering strategy transforms raw lyric text into meaningful representations that capture the structural, semantic, and repetitive patterns inherent in songs. This approach enables our models to achieve good data efficiency through strategic feature design, demonstrating that comprehensive feature engineering can overcome the limitations of small datasets through information extraction rather than relying solely on data volume.

The feature extraction process begins with standardized preprocessing of the lyrics text, including tokenization and normalization of casing and punctuation to ensure consistency across the diverse song formats present in our dataset. From this foundation, we compute nine distinct feature views at the line level, each contributing specialized information to construct comprehensive self-similarity matrices that quantify relationships between lyric lines.

4.1. Pre-processing and Implementation

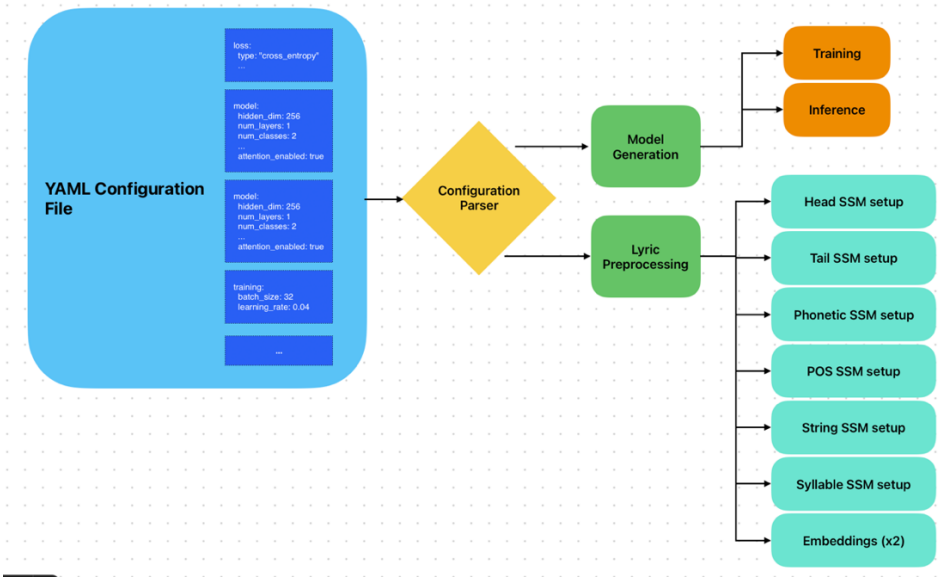


Figure 4.1 Preprocessing Pipeline

To prepare lyrics for feature computation, we apply targeted pre-processing steps that build upon the initial cleanup performed during dataset construction. These steps are designed to minimize noise while preserving relevant textual characteristics for the nine feature views, with configurable settings that control aspects such as punctuation retention and case sensitivity. This ensures the features focus on intrinsic lyric patterns rather than artifacts from source formatting (see Figure 4.1).

Normalization is handled through Unicode standardization and the collapsing of multiple spaces, with optional and selective lowercasing applied. For example, enabled for string similarity and phonetic similarity views to standardize comparisons, but disabled for embedding-based views to retain contextual nuances. Punctuation management follows configurable rules, allowing for retention in structure-sensitive views, such as part-of-speech similarity, or removal/stripping for similarity-focused computations to avoid introducing artificial differences in metrics, such as Levenshtein distance. Following feature extraction, self-similarity matrix values undergo per-song normalization via z-score[13] or min-max[14] methods post-computation, ensuring scale consistency across songs of varying lengths and preventing biases in downstream modelling. This normalization is configurable per view,

[13] Z-score normalization: Rescales a feature so its average is zero and spread is one. Useful when features look roughly bell-shaped; not bounded to a fixed range.
[14] Min-Max Scaling: Linearly maps a feature to a fixed range, typically 0–1. Preserves ordering but can be skewed by outliers.

with options to enable or disable it and select the method (e.g., min-max for bounded scaling or z-score for statistical standardization), adapting to the specific distributional properties of each feature's similarity outputs.

Any remaining bracketed or inline metadata, such as “[chorus]” or “(x2)”, is stripped to prevent models from learning shortcuts based on explicit labels, thereby forcing them to rely on textual content for repetition detection. Tokenization proceeds at the word level without depending on sentence boundaries, maintaining the line as the core analytical unit and aligning with lyric structure. For out-of-vocabulary handling, just like with word vector similarity, context vector similarity, and phonetic similarity, unknown tokens are ignored during averaging. All features are computed at the per-line level, treating each lyric line as an independent row for self-similarity matrix construction. This approach leverages the natural phrasing in lyrics to facilitate the accurate detection of repetitions and boundaries.

Each feature view incorporates configurable similarity metrics to compute pairwise line relationships, enabling adaptation to specific dataset characteristics and optimization during experimentation. For context vector similarity, available options include cosine (default) or dot product. Word vector similarity supports cosine (also default), dot product, or Euclidean distance. String similarity, head similarity, and tail similarity share options such as word overlap, Jaccard[15] index, or Levenshtein edit distance, including token-overlap variants. Phonetic similarity encompasses binary exact rhyme matching, edit distance or Levenshtein, sequence matching via longest common subsequence, and soft or weighted variants like weighted edit distance or soft-rhyme scoring. Part-of-speech similarity offers longest common subsequence, position-index-aware matching, or combined metrics that integrate the longest common subsequence with position weighting. Word syllable count similarity provides Levenshtein edit distance on syllable lists, cosine similarity on pattern vectors, or a weighted combination of both. Line syllable count similarity encompasses ratio-based similarity using syllable count ratios. These cosine similarities on count vectors, or mixed modes, integrate the ratio and cosine approaches. These configurations are set per view to optimizing specific repetition patterns, with defaults selected based on empirical performance in preliminary tests. Additionally, a high similarity threshold is configurable

[15] Jaccard similarity: Measures overlap between two sets by comparing shared items to all unique items combined. Common for token-based text similarity; ranges from 0 (no overlap) to 1 (identical).

per view for calculating repetition density during summarization, allowing fine-tuned detection of strong matches tailored to each feature's scale.

Additionally, pre-processing configurations, including punctuation settings, normalization flags, similarity metric choices, and high similarity thresholds, were persisted in YAML files, allowing for the full regeneration of the feature set. This comprehensive approach not only enhances reliability but also supports iterative experimentation and validation throughout the project development, as seen in Figure 3.7.

4.2. Nine Feature Views and Similarity Measures

To capture the diverse ways choruses, repeat and contrast with verses, we designed nine complementary feature views. These span structural, rhythmic, and semantic dimensions, each represented through self-similarity matrices (SSMs) and summarized into line-level descriptors. The multi-view approach provides richer evidence than any single metric, combining literal matches, phonetic echoes, rhythmic regularities, and semantic parallels to support robust chorus detection.

4.2.1. Structural Pattern Features

We implement five structural views that capture textual repetition patterns via self-similarity matrices (SSMs), then summarize each matrix per line into compact 12-D descriptors (statistics, local context, and position). These views provide robust detection of literal and structural repetition without relying on a single metric.

String Similarity: Overall textual similarity computed on normalized lines using a selectable metric, word overlap (default for efficiency), Jaccard, or normalized Levenshtein. This view effectively captures literal or near-literal repeats typical of chorus phrases. We build the line-to-line SSM and summarize each row into features such as min/max similarity, high-similarity ratio, neighbor/edge similarities, and position indicators, enhancing sensitivity to recurring strings.

Head Similarity: Exact matching on the first k words with optional configuration to change the k number or to make it case-sensitive or not. Surfacing repeated openings that often anchor sections and signal verse/chorus onsets. The binary head match is intentionally simple and stable, producing a clean repetition cue that complements softer similarity views.

Tail Similarity: Exact matching on the last *k* words (optional punctuation removed) to highlight rhyme and line endings, a reliable indicator of refrains across popular genres. As with heads, this binary tail match yields an interpretable signal of line-ending repetition patterns that often structure choruses.

Phonetic Similarity: *CMUdict*[16]-based phonetic signatures for rhyme/alliteration. We compare line-final or line-initial phonetic signatures using configurable similarity methods (binary equality, normalized edit distance over phoneme sequences, or sequence-match). This view captures sound-level repetition even when spelling varies and can optionally apply per-song normalization to the summarized features.

Part-of-Speech Similarity: employs grammatical structure comparison through POS tag sequences using *NLTK*, utilizing three *tagset*[17] options: Universal[18], Penn Treebank[19], and a simplified version with nine categories [20](NOUN, VERB, ADJ, ADV, DET, PREP, PRON, CONJ, WH, with all others labelled as OTHER), to model syntactic templates that recur across repeated lines, thereby revealing structural cohesion beyond exact wording. This approach supports multiple methods, including a combined measure of tag overlap, count overlap, and positional agreement, as well as Longest Common Subsequence (LCS), position-wise comparison, or Jaccard similarity. A simplified version provides a high-level abstraction that enhances broad pattern detection.

4.2.2. Rhythmic Pattern Features

Two rhythmic views target metrical regularities without dynamic time warping, focusing on syllable counts to expose line-internal and line-level rhythmic consistency that aligns with section structure.

Word Syllable Count Similarity: Per-word syllable count sequences are extracted with a lightweight heuristic implementation and compared using normalized Levenshtein or cosine

[16] CMUdict (Carnegie Mellon Pronouncing Dictionary): maps English words to phoneme sequences (with stress). Standard resource for phonetic features in NLP/MIR.
[17] Tagset: A predefined collection of part-of-speech (POS) tags used to label words in a text based on their grammatical categories, such as Universal, Penn Treebank.
[18] Universal**:** 17 coarse-grained, cross-linguistic POS tags (e.g., NOUN, VERB, ADJ).
[19] Penn Treebank: 36 fine-grained POS tags for English (e.g., NN, NNS, VBD).
[20] POS Tags Legend: NOUN: Noun; VERB: Verb; ADJ: Adjective; ADV: Adverb; DET: Determiner; PREP: Preposition; PRON: Pronoun; CONJ: Conjunction; WH: Wh-words (interrogatives and relative pronouns, e.g., who, which, where); OTHER: Catch-all category for punctuation, symbols, and remaining tags.

similarity (optionally combined). This detects rhythmic sameness under lexical variation, a common property of chorus lines that keep meter while changing words.

Line Syllable Count Similarity: Total syllables per line are compared using a ratio/cosine approach and summarized per line. This view emphasizes meter consistency across neighboring lines, helping to surface repeated line-level rhythmic patterns.

4.2.3. Semantic Embedding Features

Two embedding-based views capture semantic relationships that extend beyond surface-level matching. Each of them supports either a compact 12-dimensional SSM-style summary or full-dimensional embeddings, depending on configuration and efficiency goals.

Word Vector Similarity: Cosine or dot similarity between averaged pre-trained word2vec embeddings (GoogleNews-300[21] via Gensim[22]). Averaging yields a bag-of-words representation that captures topical/thematic alignment across lines, identifying semantically consistent phrases that contribute to section cohesion.

Context Vector Similarity: Cosine or dot similarity between contextual embeddings from SentenceTransformer all-MiniLM-L6-v2[23] (384-D). These order-aware embeddings capture broader semantics and phrasal context, enhancing sensitivity to paraphrastic or rephrased chorus lines.

4.2.4. Aggregation and Normalization

For each view, we compute a T×T SSM (T = number of lines) and summarize each row into 12 features (distribution statistics, neighbor/edge similarities, and position). We then concatenate the per-line summaries across active views into a unified T×D feature matrix consumed by the sequence model. Several views expose configurable high-similarity treatments and optional per-song normalization (z-score or min-max) of the 12-D summaries, ensuring scale stability and reproducibility through a custom YAML configuration.

[21] GoogleNews-300: a pre-trained Word2Vec embedding model with 300-dimensional vectors, trained on about 100 billion words from the Google News dataset.
[22] Gensim: an open-source Python library for unsupervised topic modeling and vector space modeling, widely used for implementing Word2Vec, Doc2Vec, and related algorithms.
[23] all-MiniLM-L6-v2: a lightweight sentence-transformer model with 6 transformer layers, optimized for producing dense sentence embeddings; widely used for semantic similarity and clustering tasks due to its balance of efficiency and accuracy.

4.3.Self-Similarity Matrix Construction and Visualization

Each feature view generates a self-similarity matrix where line indices form the axes and similarity scores represent the strength of relationships between lyric lines.

To illustrate the practical application of these views, Figure 4.2 presents a visualization of the nine SSMs computed from a sample song lyric (Creep by Radiohead) in our implementation, based on the methodology in Watanabe & Goto (2020), each heatmap represents one view, with axes corresponding to line indices (0 to 25) and color intensity indicating similarity scores (yellow for high similarity ≈1.0000, dark blue for low ≈0.0000).

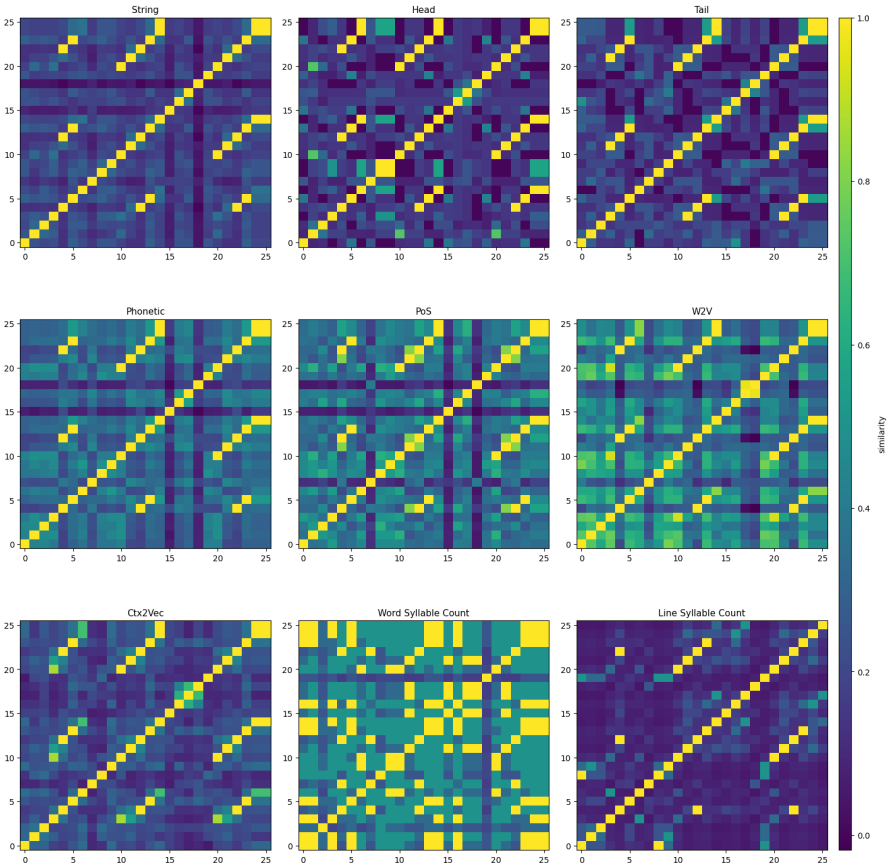


Figure 4.2 Features SSM

The main diagonal appears consistently bright yellow across all matrices, reflecting perfect self-similarity where each line compares to itself. Off-diagonal bright regions highlight repetitive patterns: in the String, Phonetic, and POS views, parallel bright lines and blocks emerge around lines 5–10 and 15–20, indicating sections with high literal, rhyming, or syntactic repetition, likely corresponding to chorus segments. The embedding-based views display broader grid-like patterns, capturing semantic similarities that may include

paraphrased repetitions, while syllable views reveal checkerboard or striped patterns tied to rhythmic consistency.

This multi-view approach enables comprehensive coverage of literal repetitions, rhythmic consistencies, and semantic similarities essential for distinguishing chorus sections from verses in text-only analysis. The complementary nature of our feature views ensures robust boundary identification, with rhythmic features revealing metrical patterns and semantic features capturing broader thematic repetitions.

4.4. Why these views?

Decades of MIR and recent lyric-specific studies concur that choruses are the most repeated and memorable portions of songs, and that repetition aligns with section boundaries. The nine-view design mirrors this:

Fault tolerance (Baratè et al., 2013): edit-distance views (string/head/tail/phonetic/POS) and DTW views (syllables) tolerate noise, handling typos, casing, missing punctuation, and orthographic variants common in web-scraped lyrics.

Multi-scale coverage: word and sound-level (string/phonetic), phrase edges (head/tail), syntactic templates (POS), local meter (syllables), and topic/semantic coherence (embeddings). This allows detection of varying granularities, from fine-grained word matches to broader thematic links.

Paraphrase capture: embedding views retrieve semantic repeats that string/phonetic metrics miss, helping in genres where chorus instances vary slightly in wording.

This hybrid design prioritizes robustness (low-level cues) and generality (semantic cues), which is particularly valuable at our data scale. In addition, similarity thresholds (e.g., for the high-similarity ratio) and DTW parameters can be tuned during optimization to adapt to genre-specific patterns.

The selection of these nine views is grounded in the methodology established by Watanabe & Goto (2020), who categorize them into structural features (repetition patterns via SSMs) and linguistic features (semantic/syntactic via embeddings), emphasizing that choruses represent the "most repeated and memorable portions" detectable through multi-view SSMs. This approach directly addresses the fundamental challenge of maximizing information

extraction from our 780-song dataset while maintaining robustness across diverse musical genres and lyrical styles.

4.5. SSM Summarization and Dimensionality Management

Directly learning from full SSMs across variable-length songs is computationally heavy. We therefore summarize each SSM row into a fixed 12-dimensional (12-D) vector per line, preserving boundary-relevant statistical and positional information:

Statistical Descriptors: We compute row mean, row-wise maximum, and row standard deviation to capture the overall similarity profile of each line. Additionally, we include the 75th and 90th percentiles of the row to characterize the distribution shape, providing insight into the concentration of high-similarity values.

Repetition Density Analysis: A critical component quantifies the fraction of high similarities in each row using view-specific configurable thresholds (e.g., 0.7 for string similarity, 0.8 as the default for other views). This repetition density measure directly indicates the degree to which a line exhibits repetitive characteristics essential for chorus identification.

Local Context Integration: Immediate contextual relationships through similarity to the previous line and similarity to the following line provide crucial boundary information, enabling detection of transitions between song sections.

Structural Anchoring: Similarities between the first and last lines serve as structural anchors, capturing each line's relationship to the song boundaries and overall compositional structure.

Positional Encoding: Normalized line index and its complement (inverse position) provide absolute positioning awareness within the song, enabling models to incorporate structural expectations based on typical verse-chorus arrangements.

4.6. Embedding-Based Summarization Adaptations

For embedding-based views (word2vec and contextual embeddings), we adapt the twelve-dimensional framework to leverage the unique properties of high-dimensional semantic representations:

Magnitude Statistics: Current embedding magnitude, global maximum magnitude within the song, and standard deviation replace traditional percentile measures, providing more meaningful descriptors for embedding similarity distributions.

Directional Similarities: Previous, next, first, and last line similarities from the embedding SSM maintain consistent local and structural context information across view types.

Position-Weighted Analysis: Forward and inverse position-weighted averages over the SSM row replace simple percentile measures, incorporating temporal progression patterns that prove particularly valuable for semantic similarity interpretation.

Unified Positional Context: Normalized line index and complement maintain consistency with structural views while providing positional awareness.

4.7. Normalization and Scale Management

Our approach avoids global pre-summarization normalization of SSMs, instead exposing per-song normalization of the twelve-dimensional summaries as configurable options within specific views. Views such as phonetic, syllable-pattern, and line-syllable offer z-score or min-max normalization methods, while others operate directly on their native similarity scales. This selective normalization strategy preserves the inherent characteristics of different similarity measures: cosine and dot similarities remain in their natural ranges, binary head/tail signals maintain their {0,1} interpretation, and normalized edit-distance-based similarities preserve their [0,1] bounds.

4.8. Computational Efficiency and Batching

This summarization design compresses rich pairwise structure into per-line descriptors that retain global repetition statistics, local neighborhood cues, structural anchors, and positional context. The result produces a uniform T×12 representation across all active views (where T represents the number of lines), enabling efficient batching operations, robust handling of song length variation, and seamless integration with downstream sequence models without requiring computationally expensive T×T channel stacking.

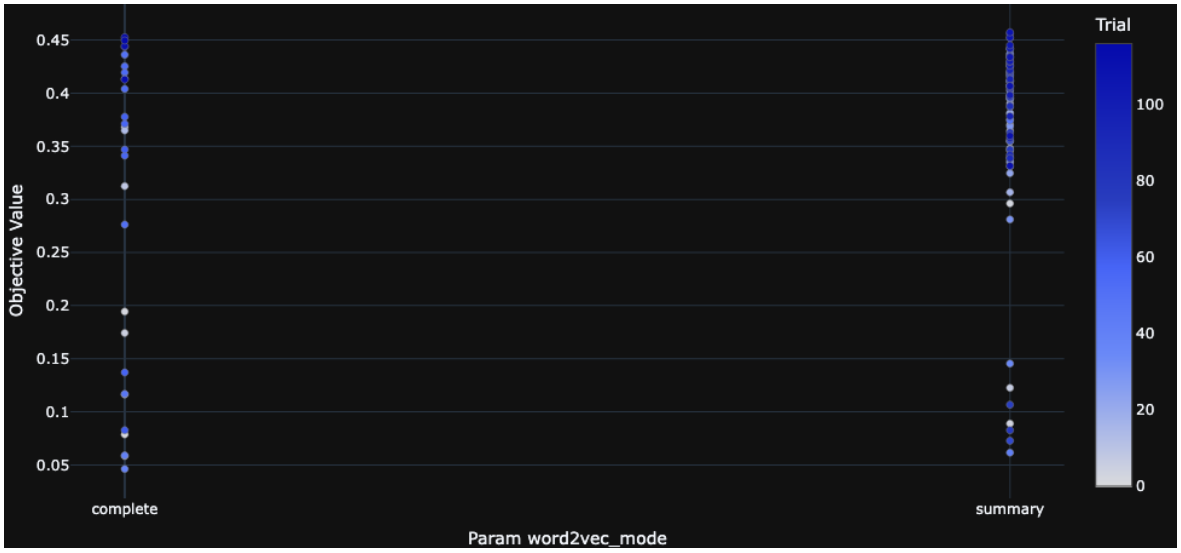
The approach proves particularly advantageous for our 900-song dataset, where efficient processing of variable-length structures becomes essential for extensive hyperparameter optimization and model training procedures.

4.9. Multi-View Performance Validation

Evidence from Watanabe & Goto (2020) demonstrates the superior performance of multi-view approaches in lyrics segmentation tasks. Combining structural and linguistic views in Bi-LSTM architectures yields F-scores of 0.7810 (English songs) and 0.8340 (Japanese songs), substantially outperforming structural-only approaches (0.7790 EN, 0.8120 JA) and linguistic-only methods (0.5740 EN, 0.5520 JA). This validates our comprehensive feature selection strategy and demonstrates the critical value of multi-view SSMs in handling variable songs.

4.10. Data Efficiency and Optimization Results

Critical to our project contribution, hyperparameter optimization via *Optuna*[24] trials revealed that summarized 12D feature representations consistently outperformed full-dimensional embeddings as seen in Figure 4.3 across extensive experiments, proving essential for our data efficiency goal by reducing noise and enhancing generalization in variable-length lyrics, thus improving model performance without necessitating larger datasets.



[24] Optuna: An open-source hyperparameter optimization framework that automates the tuning of machine learning models using efficient algorithms like Tree-structured Parzen Estimator (TPE) and Bayesian optimization.

Figure 4.3 Word2Vec Summary VS Complete

This validation confirms that our feature engineering strategy effectively captures essential segmentation patterns while eliminating noise, with specific parameters like similarity thresholds and weighting schemes proving to be among the most influential factors in model performance along with general training settings such as learning rate or dropout (see Figure 4.4). The finding is crucial for our data efficiency, as it emphasizes that fine-tuning the computation and thresholding of self-similarity views maximizes the extraction of repetitive patterns from limited data, enhancing boundary detection without requiring expanded datasets or additional computational overhead.

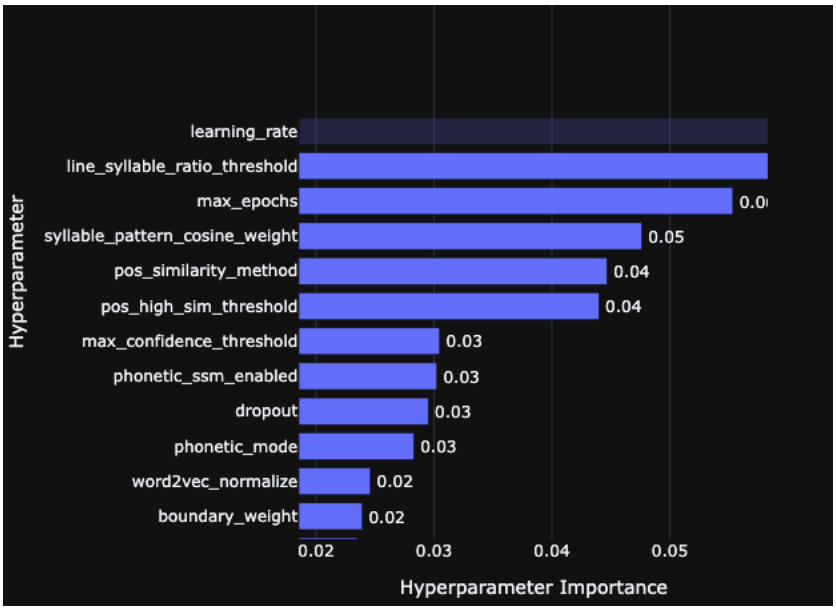


Figure 4.4 - Hyperparameter Importance

As depicted in Figure 4.5, which presents a histogram of boundary F1-score distributions alongside a Spearman correlation analysis with boundary F1-scores, the data highlights the impact of various parameters, with *syllable_pattern_cosine_weight* showing the strongest negative correlation (corr ≈ -0.4), suggesting its significant influence on performance, while *line_syllable_ratio_threshold* and *pos_high_sim_threshold* also exhibit notable positive correlations, and parameters like *word2vec_normalization* and *phonetic_ssm_enabled* display weaker or mixed effects. This analysis suggests that feature-specific parameters, particularly those related to syllable patterns and POS similarity, play a significant role in optimizing segmentation performance, complementing the influence of general settings on the model's F1-score distribution.

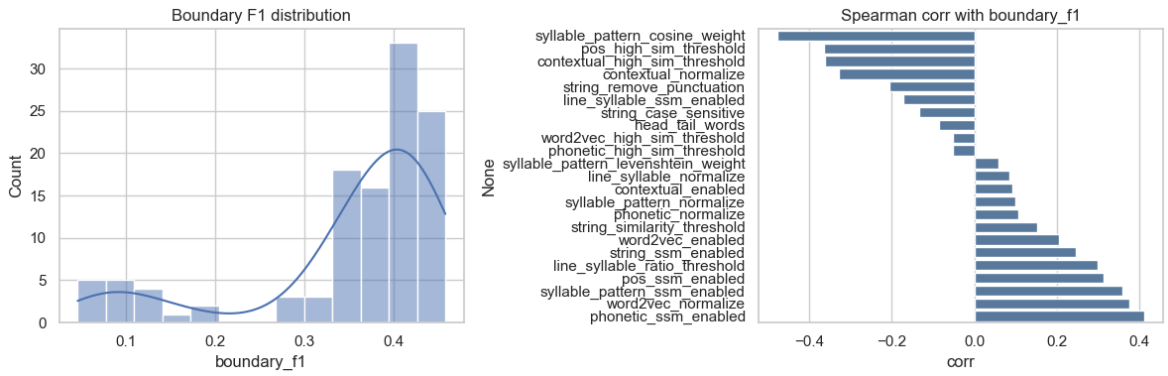


Figure 4.5 - Feature Hyperparameter Correlation

Our hyperparameter optimization via Optuna trials demonstrated that configurations emphasizing tunable feature metrics, such as combined similarity methods and balanced weights, dominated the best-performing setups, validating both our multi-view feature engineering approach and the strategic prioritization of repetition-sensitive parameters over architectural depth.

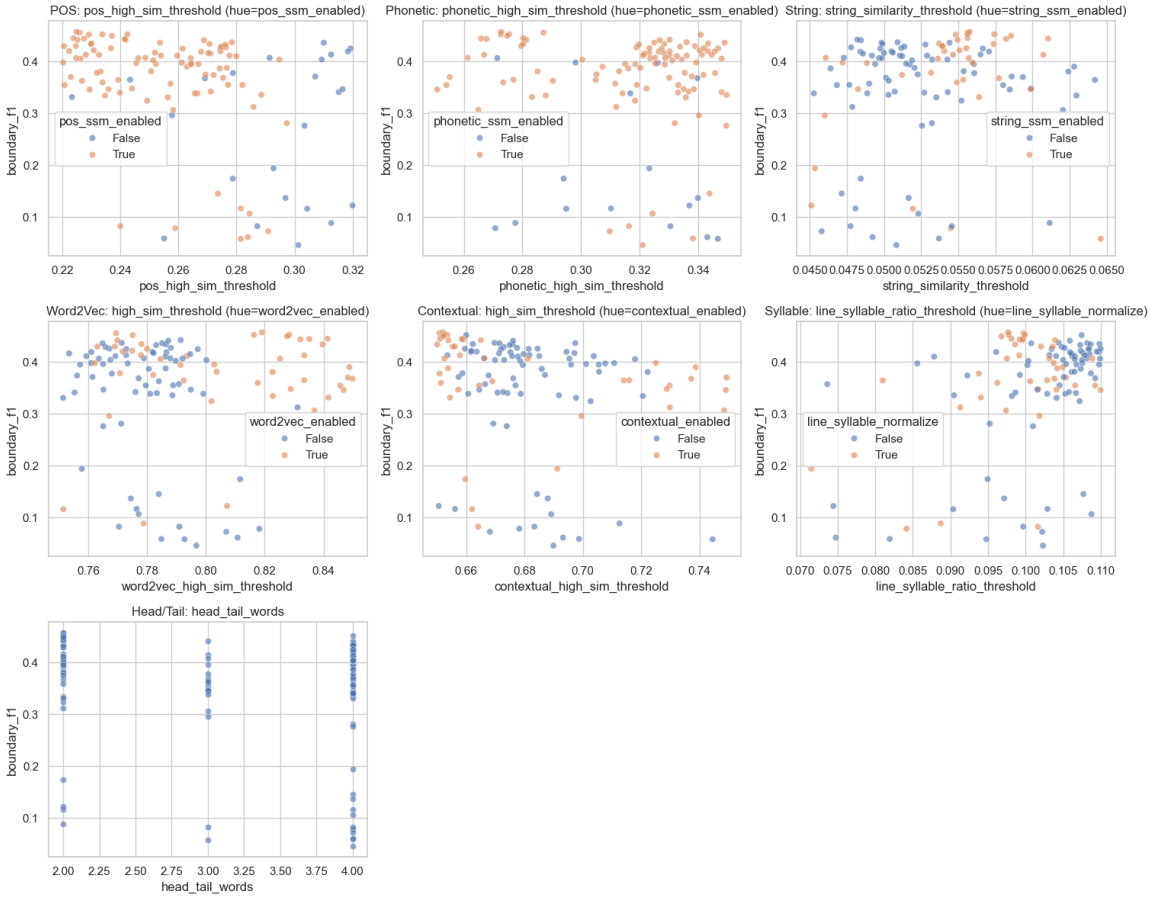


Figure 4.6 - Correlation Between Boundary-F1 and feature configs

Building on these parameter influence patterns, we conducted a comprehensive analysis of the scatter plots in Figure 4.6 to examine how individual feature-specific thresholds shaped our boundary F1-score optimization across our 115 Optuna trials. These visualizations extend the correlation insights by revealing the operational ranges where each parameter achieves optimal performance, with threshold spans carefully selected, such as 0.22–0.32 for part-of-speech (POS) and 0.76-0.84 for Word2Vec, based on preliminary trials that balanced computational efficiency with meaningful boundary detection sensitivity. The performance landscape shows that F1-scores below 0.2 indicate inadequate segmentation due to the metric's strict penalization of even single-line prediction offsets, while scores approaching 0.4-4.5 represent substantial improvements toward reliable boundary identification, reinforcing our data-efficient approach's ability to achieve competitive results with just 780 songs compared to models requiring datasets exceeding 100,000 tracks.

The scatter plot analysis reveals distinct behavioral patterns across feature categories. Syntactic and lexical features, POS (*pos_high_sim_threshold*), Phonetic (*phonetic_high_sim_threshold*), and String (*string_similarity_threshold*) consistently benefit from self-similarity matrix (SSM) activation, with orange points (SSM enabled) clustering toward higher F1-scores, particularly evident in POS performance ranges of 0.28–0.32. This aligns with the strong positive correlation observed for *pos_high_sim_threshold* in our earlier analysis, confirming that repetition-sensitive syntactic patterns serve as reliable boundary indicators.

Semantic embedding features present more complex optimization dynamics. Word2Vec (*word2vec_high_sim_threshold*) demonstrates the correlated effects anticipated by our multi-view approach, where SSM activation enables other hyperparameters to compensate for individual limitations, ultimately producing peak F1-scores up to 0.32. Contextual embeddings (*contextual_high_sim_threshold*) and Syllable patterns (*line_syllable_ratio_threshold*) show broader performance distributions with modest improvements at higher thresholds when SSM is active, reflecting the nuanced interactions between high-dimensional representations and repetition detection mechanisms.

Structural features, exemplified by Head/Tail word matching (*head_tail_words*), maintain consistent peak performance across threshold ranges of 2.25-4.50, demonstrating robustness independent of SSM configuration. This stability validates structural cues as reliable

baseline components within our ensemble architecture, complementing the variable sensitivity observed in semantic features.

These threshold optimization patterns confirm our hypothesis that strategic feature combination outperforms architectural complexity for data-constrained scenarios. The evidence suggests a tiered detection system, where SSM-enhanced features provide primary boundary signals, semantic embeddings contribute to contextual refinement through carefully calibrated interactions, and structural patterns offer a consistent fallback detection. This multi-layered strategy, validated across 115 optimization trials, demonstrates how targeted feature engineering achieves competitive segmentation performance while maintaining data efficiency advantage that distinguishes our approach from conventional deep learning methods requiring massive training corpora.

5. Lyric Segmentation Methodologies and Evaluation

This chapter presents a detailed investigation into the development and evaluation of advanced neural architectures for lyric segmentation, a crucial task in music information retrieval that aims to identify structural boundaries, such as verses and choruses, within song lyrics. Building on the theoretical foundations and literature review established in previous chapters, this chapter focuses on the practical implementation and comparative analysis of two primary approaches: fine-tuning a small-scale large language model (Gemma-1B-IT) and designing custom neural networks, specifically a Bidirectional Long Short-Term Memory (Bi-LSTM) model with attention mechanisms and a Convolutional Neural Network (CNN).

We picked Gemma-1B-IT as our transformer-based architecture because it represents a balance between capability and efficiency. Although our initial plan was to fine-tune the 4B parameter version, hardware constraints made the 1B parameter model a more feasible choice. Importantly, Gemma-1B retains the advantages of modern pretraining while remaining lightweight enough for experimentation on limited resources, aligning with the goals of this project to achieve data-efficient segmentation without massive computational power. According to Google's official documentation, Gemma models are designed to support a wide variety of text generation tasks while offering developer-friendly model sizes, including the 1B variant optimized for constrained environments Google (2025).

The study addresses the unique challenges posed by lyrical texts, including repetitive patterns, stylistic variations, and ambiguous transitions, which complicate accurate boundary detection. A comprehensive multi-metric evaluation framework is introduced, incorporating standard metrics such as Micro and Macro F1, alongside custom boundary-focused metrics like Boundary F1, WindowDiff, and PK, to provide a robust assessment of model performance across diverse dimensions. Despite training on a relatively small dataset of 780 songs, our proposed models achieve competitive performance with prior approaches that leveraged significantly larger corpora. This chapter elucidates the methodologies, experimental designs, and key findings, emphasizing the superiority of localized pattern recognition over global or boundary-specific strategies. These insights contribute to the

broader understanding of text segmentation in semi-structured domains and lay the groundwork for scalable, efficient lyric segmentation systems.

To evaluate the performance of our lyric segmentation models, including the fine-tuned Gemma-1B-IT, Bi-LSTM with attention, and CNN, we developed a comprehensive set of metrics to assess both line-level accuracy and boundary detection precision. The following section outlines the design and application of these evaluation metrics, which guide model development and enable comparisons with state-of-the-art approaches.

5.1. Validation and metrics

To evaluate the performance of our lyric segmentation models, this study employs a comprehensive and systematic evaluation framework that integrates both standard and custom-designed metrics. This framework ensures a robust assessment of model effectiveness in identifying verse and chorus boundaries within lyrical texts, addressing the limitations of traditional metrics in capturing the nuances of segmentation accuracy. The metrics are consistently applied across all models to facilitate fair comparisons with state-of-the-art approaches (e.g., Watanabe) and to guide critical aspects of model development, including selection, hyperparameter tuning, and learning rate scheduling.

Standard Metrics

Macro F1 (Line-level F1)

The Macro F1 score calculates the F1 score for each class (verse and chorus) independently, then takes their unweighted average. This approach gives equal importance to both verse and chorus performance, regardless of their frequency in the dataset. Macro F1 is particularly valuable for addressing class imbalance, a common challenge in lyric segmentation where verses often outnumber choruses. By averaging per-class F1 scores, this metric ensures that poor performance on the minority class (choruses) is not overshadowed by good performance on the majority class (verses), providing a balanced assessment of the model's ability to identify both segment types correctly.

Micro F1

Complementing Macro F1, the Micro F1 score measures classification accuracy by pooling true positives, false positives, and false negatives across all lines to compute a single

precision and recall score. For binary classification tasks like verse/chorus segmentation, Micro F1 is equivalent to overall accuracy. This metric treats each line prediction equally, regardless of class, making it heavily influenced by the performance of the majority class.

Boundary F1

To overcome the limitations of line-level metrics in evaluating segmentation boundaries, we designed custom metrics specifically tailored to assess transition accuracy. These metrics, configured at the start of each training session, enhance the evaluation framework by focusing on the precise identification of segment boundaries.

The Boundary F1 metric assesses the precision and recall of predicted segment boundaries (e.g., the start and end lines of verses and choruses) in relation to ground-truth annotations. To account for varying levels of strictness, we compute Boundary F1 at three tolerance levels:

- Strict Mode (0 Tolerance): Penalizes any mismatch, requiring exact alignment with ground-truth boundaries.
- ±1 Tolerance: Allows a one-line window of correctness, accommodating minor deviations in boundary placement.
- ±2 Tolerance: Permits a two-line window, further relaxing the criteria to capture near-miss boundaries.

This metric provides a granular assessment of the model's ability to pinpoint structural transitions, critical for accurate lyric segmentation.

WindowDiff (WD)

WindowDiff assesses the probability that two positions, separated by a fixed distance (k units), have inconsistent segmentation labels compared to the ground truth. Lower WD scores indicate superior performance, as this metric penalizes near-miss boundaries and systematic errors while being more forgiving than exact matching. WD is particularly effective in capturing the structural coherence of predicted segments.

PK Metric

The PK metric, a probabilistic error measure, evaluates the likelihood of errors in determining whether two positions belong to the same segment. Like WindowDiff, lower

PK values signify better segmentation performance, with heightened sensitivity to missing or extraneous boundaries. This metric complements WD by focusing on the consistency of segment assignments across the text.

The multi-metric framework, comprising Micro F1, Macro F1, Boundary F1 (with 0, ±1, and ±2 tolerances), WindowDiff, and PK Score, is systematically applied during model validation. This approach overcomes the shortcomings of relying solely on line-level F1 scores, which may mask deficiencies in boundary detection. By integrating these metrics, we ensure a comprehensive assessment of model behavior, enabling informed decisions in model selection, hyperparameter optimization, and overall performance evaluation. This framework not only facilitates comparisons with prior work but also provides actionable insights into the strengths and limitations of our proposed architecture, as detailed in the subsequent sections of this chapter.

5.2. Model Architectures and Approaches

Having established our comprehensive evaluation framework, we now present the implementation and validation results of our segmentation models. Each approach was evaluated using the complete metric suite described above, enabling direct performance comparisons across architectures. The following sections detail the design, training process, and experimental outcomes for each model variant.

5.2.1. LLM-Based Segmentation

To address the lyric segmentation task, we explored the fine-tuning of a small-scale large language model, specifically the instruction-tuned Gemma-1B-IT with 1 billion parameters. This model was selected for its reported proficiency in generating structured responses, which is critical for producing consistent and parseable segmentation metadata in *JSON* format. The fine-tuning process, however, presented significant computational challenges, necessitating adaptations to accommodate hardware constraints.

Fine-Tuning Process

The fine-tuning process utilized Low-Rank Adaptation (LoRA) with the MLX framework, chosen to mitigate the high resource demands of training a large language model on limited hardware. Initially, we attempted to fine-tune the 4-billion-parameter version of Gemma, but

memory constraints rendered this infeasible. To maintain architectural consistency and compatibility with our pipeline, we pivoted to the 1-billion-parameter Gemma-1B-IT model.

Training on the full dataset of 780 songs was also constrained by memory limitations, preventing completion even at a modest 5 epochs. To address this, we split the dataset, training on a subset of 315 lyrics. This adjustment enabled the fine-tuning process to complete within a reasonable timeframe without exceeding memory or GPU limits. The resulting model successfully generated structured segmentation metadata in JSON format, identifying verse and chorus boundaries with line-number ranges.

Structured Response Optimization

The initial design of the structured response included both the segmentation metadata and the corresponding lyric text for each segment. However, this approach significantly increased token usage, as the model effectively duplicated the input lyrics in its output, quickly exhausting the context window. For example, the input comprised an initial prompt, a sample output, and the raw lyrics, while the output doubled token consumption by reiterating the lyrics for each segment. To optimize token efficiency, we revised the training data to instruct the model to return only line-number ranges (e.g., {"type": "verse", "start_line": 1, "end_line": 4}) instead of full lyric text as seen in Figure 5.1. This modification substantially reduced token counts, enabling the model to process longer inputs within its context window. However, it introduced a minor issue were, in rare cases, the model generated overlapping segment boundaries. To address this, we implemented a post-processing step that resolved overlaps by adjusting boundaries to the last common line of each section. This ensured consistent and non-overlapping segment predictions.

Post-fine-tuning, the model demonstrated reliable performance in generating structured JSON responses, with no parsing issues observed. Inference times ranged from 10 to 15 seconds per song, depending on lyric length, reflecting the computational demands of processing structured outputs. Due to the requirement for complete JSON responses for accurate parsing, streaming was not utilized, as partial outputs could not be reliably interpreted.

Data Preparation

To prepare the dataset for fine-tuning the Gemma-1B-IT model using Low-Rank Adaptation (LoRA) within the MLX framework, we transformed our lyric dataset into a JSONL format,

which is compatible with the framework's requirements. Each entry in the JSONL dataset consists of an input prompt and the expected structured response. The prompt input includes a system instruction, a practical example of the desired JSON output, and the raw lyrics to be segmented as seen in Figure 4.1.

```
[INST] Identify verse and chorus sections in this song. Return line boundaries
in JSON format.

Example:
Lyrics: "First verse line\\nSecond verse line\\nFirst chorus line\\nSecond
chorus line"
Response: {{"segments": [{{"type": "verse", "start_line": 1, "end_line": 2}},
{{"type": "chorus", "start_line": 3, "end_line": 4}}]}}

Rules:
- Line numbers start at 1
- Each segment needs: type, start_line, end_line
- Only two types: "verse" or "chorus"
- Return ONLY the JSON, no explanations

Song lyrics:
{raw_lyrics}
[/INST]
```

Figure 5.1 - System Prompt Gemma 1B

On the system prompt, we used [INST] [/INST]; these special characters serve as structural delimiters, enabling the instruction-tuned Gemma-1B-IT model to distinguish user instructions from its generated responses. These tokens, integral to the model's instruction-tuning phase, ensure that the model interprets text within [INST] tags as queries requiring a structured JSON response. Maintaining this format in the fine-tuning dataset was critical to preserving the model's instruction-following capabilities while adapting it to the lyric segmentation task. The system prompt was iteratively refined through trial and error to minimize token usage while ensuring clarity and specificity, avoiding overly broad or generic instructions.

LoRA Fine-Tuning Configuration

The fine-tuning process was orchestrated using a YAML template within the MLX LoRA framework, providing precise control over critical parameters such as model specification, training seed, number of epochs, LoRA layers, rank values, and optimization settings. Given the significant memory and computational constraints of our training environment, we adopted a highly conservative configuration strategy to prioritize resource efficiency over training speed and performance optimization. To minimize memory usage, we set the batch size to 1 and limited training to 5 epochs, ensuring completion within our hardware

limitations. Gradient checkpointing was enabled to reduce memory demands during backpropagation, thereby allowing for the efficient use of available resources.

To further optimize our constrained environment, we applied LoRA adaptation to only two layers with a rank of 4, significantly reducing the number of trainable parameters. We deliberately excluded performance-enhancing features, such as mixed-precision training, weight decay, and warm-up steps, as these would have increased memory overhead beyond our system's capacity. Logging was configured to occur infrequently, every 100 steps, and model checkpoints were saved only every 500 steps to minimize resource consumption. Additionally, we restricted evaluation during training progress checks to a maximum of 64 tokens to conserve memory. While this conservative approach may have led to slower convergence and potentially lower adaptation quality compared to more resource-intensive configurations, it enabled successful fine-tuning of the 1-billion-parameter Gemma-1B-IT model on hardware incapable of supporting full-parameter fine-tuning. The training process, conducted over several hours, utilized a subset of 315 lyrics from the original 780-song dataset, as described previously.

Preliminary Validation Results

Upon completion of training, we conducted a validation test using a curated validation dataset to compare the performance of the fine-tuned model against the base Gemma-1B-IT model. Despite overall low performance for both models, the fine-tuned model exhibited notable improvements, achieving a 50% increase in certain line-level metrics (e.g., Micro F1) and a 16% improvement in boundary-level metrics (e.g., Boundary F1). These gains underscore the effectiveness of the LoRA fine-tuning approach in enhancing the model's lyric segmentation capabilities, even under constrained conditions. Figure 5.2 provides a detailed breakdown of these results, comparing the Normal (base) and Fine-Tuned models across key classification and segmentation error metrics, with notable improvements in F1 scores for both verses and choruses, as well as reduced segmentation errors like Pk and WindowDiff.

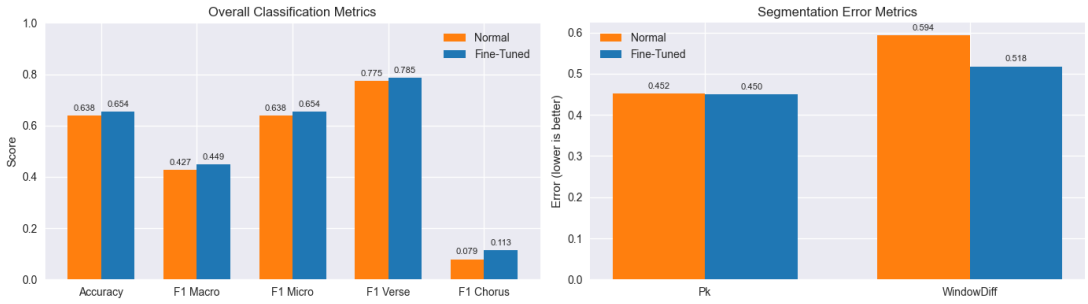


Figure 5.2 - Fine Tune vs Normal LLM

Performance Analysis and Limitations

The fine-tuned Gemma-1B-IT model demonstrated notable improvements over the base model, achieving a 16% enhancement in boundary-level metrics, specifically Boundary F1 (see Figure 5.3). However, the overall performance remained suboptimal, particularly in chorus detection, where both the base and fine-tuned models exhibited a bias toward classifying segments as verses. This is evidenced by the lower Chorus F1 scores compared to Verse F1 scores when evaluated against ground-truth labels (See Figure 5.4). Specifically, in our core metric, Boundary F1, the fine-tuned model achieved a score of 0.47 (strict mode), compared to 0.59 for the best model in our study, with similar trends observed at ±1 and ±2 tolerance levels (0.52 versus 0.74 at ±2 tolerance).

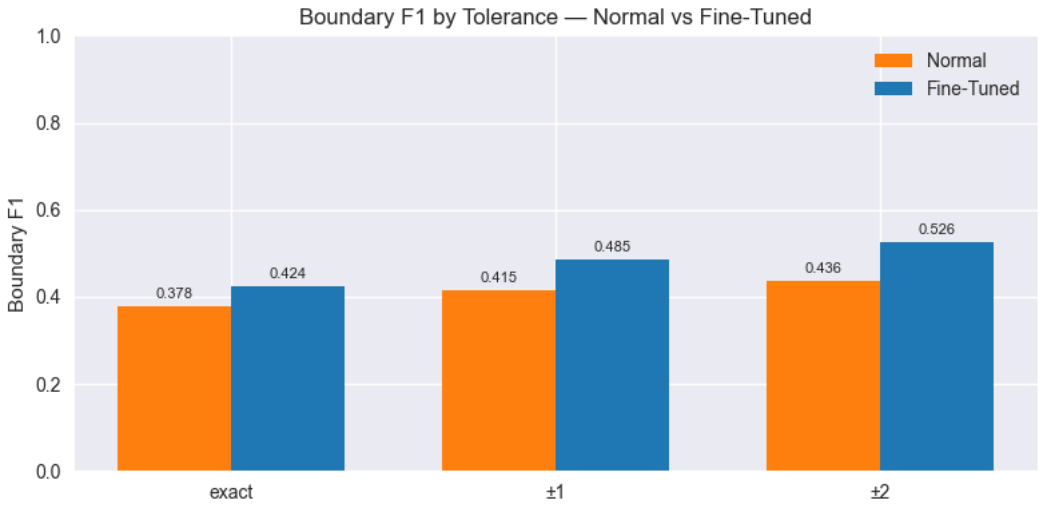


Figure 5.3 - Fine Tune vs Normal LLM Boundary Metrics

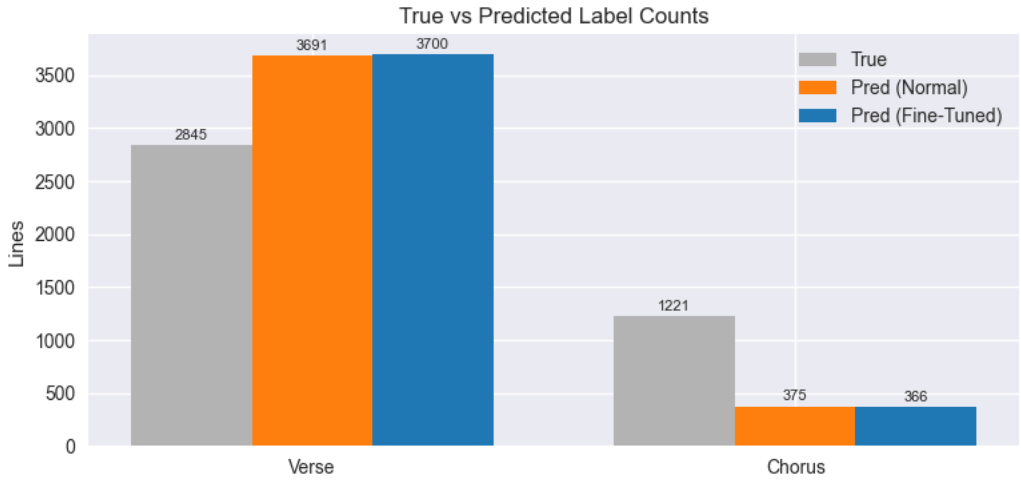


Figure 5.4 - Fine Tune & Normal, Predicted vs True Label

The suboptimal performance can be attributed to the constrained training setup, which was limited to 5 epochs and minimal LoRA configurations due to hardware limitations. Despite these constraints, the observed improvements suggest potential for further optimization with access to more robust computational resources. A key challenge in evaluating the model's performance lies in distinguishing whether its predictions rely on patterns learned during fine-tuning or on knowledge retained from its pre-trained state, even when tested on unseen songs. This ambiguity complicates the assessment of the model's generalization capabilities.

Given these limitations and the need for greater control over training dynamics and feature extraction, we transitioned to exploring custom neural network architectures for lyric segmentation. We hypothesized, based on our previous research, that architectures such as Bi-LSTM with attention mechanisms and convolutional neural networks (CNNs) could offer improved performance by allowing more precise tuning of model parameters and feature representations tailored to the segmentation task. The subsequent sections detail these alternative approaches and their comparative performance.

5.2.2. Neural Network Architectures: Design and Implementation Challenges

To advance our lyric segmentation efforts beyond the fine-tuned Gemma-1B-IT model, we explored two state-of-the-art neural network architectures: a Convolutional Neural Network (CNN) and a Bidirectional Long Short-Term Memory (Bi-LSTM) model with attention mechanisms, inspired by established approaches in text and lyric segmentation. These architectures were selected for their complementary strengths: CNNs excel at capturing local patterns, such as repetitive motifs and phrase-level regularities (Fell et al., 2018), while Bi-

LSTM models with attention mechanisms effectively model long-range sequential dependencies and focus on discriminative features for segment boundary detection (Watanabe & Goto, 2020). Unlike the LLM approach, these custom architectures provided full control over the training loop, loss functions, and feature extraction, enabling precise optimization tailored to the challenges of lyric segmentation, such as class imbalance and boundary detection accuracy.

The design of the CNN leveraged local receptive fields to detect structural cues like rhyme schemes and repetitive phrases within our dataset of 780 songs, while the Bi-LSTM incorporated attention mechanisms to prioritize contextual relationships critical for identifying verse-chorus transitions. However, initial experiments with basic CNN and LSTM configurations revealed significant training challenges that impeded stability and convergence. A primary issue was class imbalance, evident in the dataset's distribution analysis (see Figure 3.4), where verses significantly outnumbered choruses, causing early models to collapse into predicting only the majority class (verses). Attempts to mitigate this by assigning higher weights to the minority class (choruses) led to overcompensation, resulting in models predominantly predicting choruses.

Diagnosis of Training Instability

Further analysis identified a focal loss problem stemming from the imbalanced dataset, characterized by saturated logits that produced maximum probability scores of 1.0, indicating overconfident predictions on noisy lyrical data. This overconfidence, driven by extreme logit values (extremely large positive or negative values in raw model outputs) prior to softmax normalization, caused probabilities to snap to categorical extremes, exacerbating class collapse and gradient explosion. These challenges necessitated a robust stabilization framework, detailed in the following section, to ensure effective training and meaningful segmentation performance for both architectures.

5.2.3. Training Instability: Diagnosis and Stabilization Framework

To mitigate training instability and class imbalance, we developed a comprehensive stabilization framework comprising five complementary techniques. These were explored as part of experimental configurations to stabilize optimization dynamics while preserving the ability to experiment with model architecture to address the focal loss problem.

- **Label Smoothing:** We applied label smoothing to soften target distributions, reducing the likelihood of overconfident predictions by discouraging full certainty for any class. This technique mitigated saturated logits and promoted balanced learning, with the smoothing parameter determined through iterative experimentation.
- **Maximum Probability Thresholding:** A real-time monitoring mechanism was implemented to halt training if the model's maximum prediction probability exceeded a predefined threshold. This safeguard effectively prevented "chorus collapse" observed during manual testing, with the threshold value established through experimentation.
- **Real-Time Class Distribution Monitoring:** To counter class imbalance, we actively monitored the predicted class distribution during training, implementing dynamic confidence thresholds to penalize predictions falling below a minimum chorus rate of 0.0553 or exceeding a maximum chorus rate of 0.7888, as derived from the dataset-specific verse-to-chorus ratios in our tuned configuration. These thresholds, initially set arbitrarily during early trials (e.g., 7%–85%), were refined using the true values from the enhanced BiLSTM model with boundary-aware loss, ensuring robust representation of the minority chorus class and preventing model collapse. This approach, supported by the emergency_monitoring parameters (min_chorus_rate: 0.0553, max_chorus_rate: 0.7887), effectively balanced the distribution and enhanced segmentation performance.
- **Gradient Clipping and Logit Clamping:** Gradient clipping and logit clamping were integrated as safety mechanisms to prevent significant parameter updates that could reinforce majority-class bias. Gradient clipping restricted the magnitude of parameter updates to mitigate runaway updates, while logit clamping constrained the raw model outputs to a stable range, preventing extreme values that would lead to saturated probabilities. These techniques stabilized training dynamics and preserved learning signals for the minority class.
- **Entropy Regularization:** An entropy regularization term, weighted by an experimentally tuned lambda parameter, was introduced to penalize low-entropy predictions. This encouraged predictive uncertainty, maintaining a consistent learning signal even at high confidence levels.

- **Cosine Scheduler:** We incorporated a cosine learning rate scheduler to dynamically adjust the learning rate during training, starting with a high rate and gradually decreasing it following a cosine annealing curve.

The integration of these measures, alongside reduced focal loss parameters, successfully prevented training stagnation and collapse. The final training configuration adopted a binary cross-entropy loss augmented with label smoothing, entropy regularization, and real-time monitoring mechanisms, providing a robust foundation for subsequent experimentation with CNN and Bi-LSTM architecture. This stabilization framework proved transferable, as the same techniques were later applied with success in both the CNN and the final Bi-LSTM with attention model.

It is worth noting that all hyperparameter values associated with this framework were initially established through iterative manual experimentation. These values were subsequently fine-tuned using Optuna, as detailed in Hyperparameter Optimization on hyperparameter optimization.

5.2.4. Bi-LSTM Model with Attention Mechanism

After stabilizing the training loop mechanism and implementing several structural features for testing and project organization, we developed a new Bi-LSTM model with an updated architecture. This version integrated the core features that solved the earlier training problems and introduced two key extensions: an attention mechanism, which has proven highly effective in text segmentation tasks (Lukasik et al., 2020) and a new loss function designed not only for line-by-line categorization but also for explicit boundary detection.

To support this design, we developed a configuration engine that enabled us to manage and experiment with multiple setups for class weights, regularization parameters, and other critical values for handling class imbalance. The system was built on a YAML-based file structure, enabling us to save strong configurations while continuing to iterate on new experiments. This ensured both flexibility and reproducibility, as promising results could be preserved without limiting further exploration.

This modular approach later proved essential when integrating Optuna, which utilized the same configuration system to systematically search for the best parameter settings. In this way, we combined manual experimentation with automated optimization, ensuring that the

Bi-LSTM architecture could be tuned effectively while maintaining a clear experimental record.

Attention Mechanism

Given that our dataset consisted of long sequences of lyrics, adding an attention mechanism to the *Bi-LSTM* architecture was a natural next step. Previous work in text segmentation has shown that attention-based models can substantially improve performance by enabling the network to focus on the most relevant parts of the sequence while reducing the effects of noise (Lukasik et al., 2020), and the general formulation of attention has become a cornerstone of neural sequence modeling (Vaswani et al., 2017).

In recurrent models, one of the main limitations is the bottleneck of compressing long sequences into a single hidden representation, which often results in information loss. Attention mitigates this by allowing the model to dynamically assign weights to different positions in the sequence, effectively deciding where to focus on each step.

Formally, an attention function maps a query and a set of key–value pairs to an output, where both the query and the keys are compared through a compatibility function. The resulting weights are used to compute a weighted sum of the values. This consistent structure of Query (Q), Key (K), and Value (V) allows the network to selectively highlight the most informative parts of the input for the task at hand.

In the context of lyric segmentation, this means the model can highlight words and phrases that are most indicative of verse–chorus boundaries, while down-weighting irrelevant or noisy tokens. Queries serve as the "search requests" for relevant cues, keys define the positions searched across the sequence, and values provide the content to be aggregated according to learned importance. By doing so, the model is better able to capture the structural transitions that characterize lyrics (Vaswani et al., 2017).

Our implementation of attention was fully integrated into the project's modular configuration system. We introduced options to vary the number of attention heads, enabling experiments with different levels of parallel attention. We also included attention dropout to reduce overfitting, extending the regularization strategies that had already proven effective in earlier Bi-LSTM experiments. Importantly, the attention module could be toggled on or off at the configuration level, ensuring that its contribution could be isolated and directly measured in the segmentation pipeline.

Multi-Head Self-Attention configuration

The attention module in our Bi-LSTM architecture was based on the standard multi-head self-attention formulation, inspired by the transformer encoders in the BERT-based segmentation models presented in *"Text Segmentation by Cross Segment Attention" by* Lukasik et al. (2020). To preserve sequence order awareness, we applied sinusoidal positional encodings to the input embeddings prior to projection into Query (Q), Key (K), and Value (V) representations, consistent with BERT's design. The model then employed scaled dot-product attention with masking to handle variable-length sequences, followed by residual connections and layer normalization to stabilize the optimization process.

The multi-head structure divided the representational space across parallel attention heads, with each head operating at reduced dimensionality while maintaining the same overall computational complexity as a single-head setup. This design allowed the network to capture multiple attention patterns simultaneously, enriching its ability to detect structural cues in lyrics. Our implementation was conceptually aligned with the BERTLARGE configuration (24 layers, 1024 dimensions, 16 heads) and the BERTBase configuration (12 layers, 768 dimensions, 12 heads) described in Lukasik et al. (2020) providing a clear point of comparison with established baselines in the segmentation literature.

Specialized attention variants

Beyond standard self-attention, we developed two specialized variants tailored to lyric segmentation tasks, drawing on approaches that emphasize local boundary detection and cross-segment analysis in text segmentation (Fell et al., 2018; Frohmann et al., 2024; Ghinassi et al., 2024; Lukasik et al., 2020). These adaptations were motivated by the observation that transitions between verses and choruses are often signaled by repeated patterns and semantic shifts rather than explicit punctuation. By incorporating ideas such as handling variable contexts and probabilistic boundary modeling, we adapted attention mechanisms to focus on lyric-specific repetitiveness and macrostructures, thereby improving the detection of segment breaks in songs.

The localized attention variant preserved the computational foundation of the baseline but restricted the receptive field to fixed windows (e.g., 128 word-pieces per side) around each position. This encouraged the model to attend primarily to nearby tokens, supporting boundary detection while maintaining efficiency and respecting padding constraints. The

design was inspired by methods that process limited local contexts around candidate breaks to detect semantic shifts, as seen in cross-segment attention models for document segmentation (Lukasik et al., 2020). This principle is also emphasized in surveys highlighting the importance of local cues for avoiding overfitting to domain-specific markers (Ghinassi et al., 2024). In the lyrical domain, such localized focus is essential for capturing macrostructural transitions, where short, repeated patterns often mark verse–chorus boundaries (Fell et al., 2018). Sliding-window attention techniques for handling noisy or variable-length sequences further informed this design, particularly when working with incomplete or corrupted lyric transcriptions Frohmann et al. (2024).

The boundary-aware attention variant introduced an auxiliary prediction module for explicit boundary modeling. This module consisted of a two-layer feedforward network with ReLU activation and dropout, applied to attention input features after positional encoding, to generate per-token boundary probabilities. These probabilities were then transformed into symmetric bias matrices by averaging across token pairs, which were added to the scaled attention scores before the softmax operation. This design biased the attention mechanism toward regions likely to contain structural transitions, while still retaining global attention scope. The approach was motivated by probabilistic boundary detection methods in neural segmentation models Nicholls & Tanaka (2021), and it provided a direct way of guiding attention toward critical verse–chorus shifts.

Attention Mechanism Performance Analysis

Experimental results confirmed the substantial benefits of incorporating attention into our lyric segmentation model. As shown in Figure 5.5, attention-enabled configurations consistently outperformed attention-disabled models across all evaluation metrics. For metrics where higher values indicate better performance (Boundary F1 and Line F1), attention-enabled models achieved clearly higher medians with narrower distributions, while attention-disabled models clustered at lower ranges. For error-based metrics where lower values are preferred (Pk and WindowDiff), the opposite pattern was observed: attention-enabled models showed substantially lower error rates, with compact distributions close to the lower bound, whereas attention-disabled models exhibited higher variance and much weaker results. Together, these findings demonstrate that attention not only improves segmentation accuracy but also stabilizes performance across different evaluation dimensions.

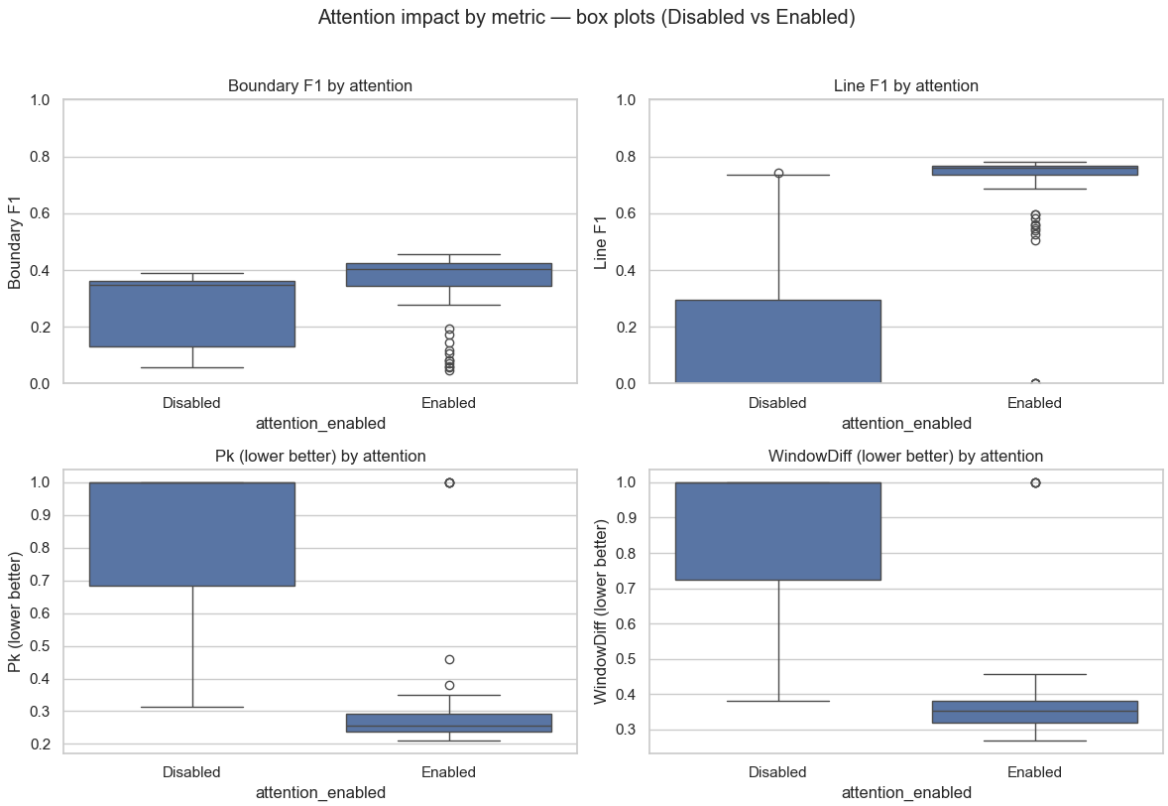


Figure 5.5 - Attention Enabled vs Disabled

Hyperparameter Interactions and Optimization

Analysis of the interaction between attention dropout and learning rate revealed critical insights into training dynamics (see Figure 5.6). The dropout values reported here correspond specifically to the attention mechanism. Models trained with lower attention dropout values (0.15–0.20) consistently achieved higher Boundary F1 scores, indicating that preserving more of the learned attention weights was preferable to aggressive regularization.

Learning rate analysis showed that moderate values (0.0007–0.0009, visualized in green and yellow) produced optimal Boundary F1 scores of approximately 0.40–0.45 when paired with these lower dropout settings. This finding underscores the need to balance stability with sufficient optimization momentum to avoid over-regularization. In contrast, the lowest learning rates (0.0001–0.0003, represented by violet and dark blue colors) consistently yielded the poorest performance, with Boundary F1 scores dropping as low as 0.06–0.18. As the learning rate increases from these minimal values, performance gradually improves, demonstrating the critical importance of sufficient optimization momentum for effective model training.

.

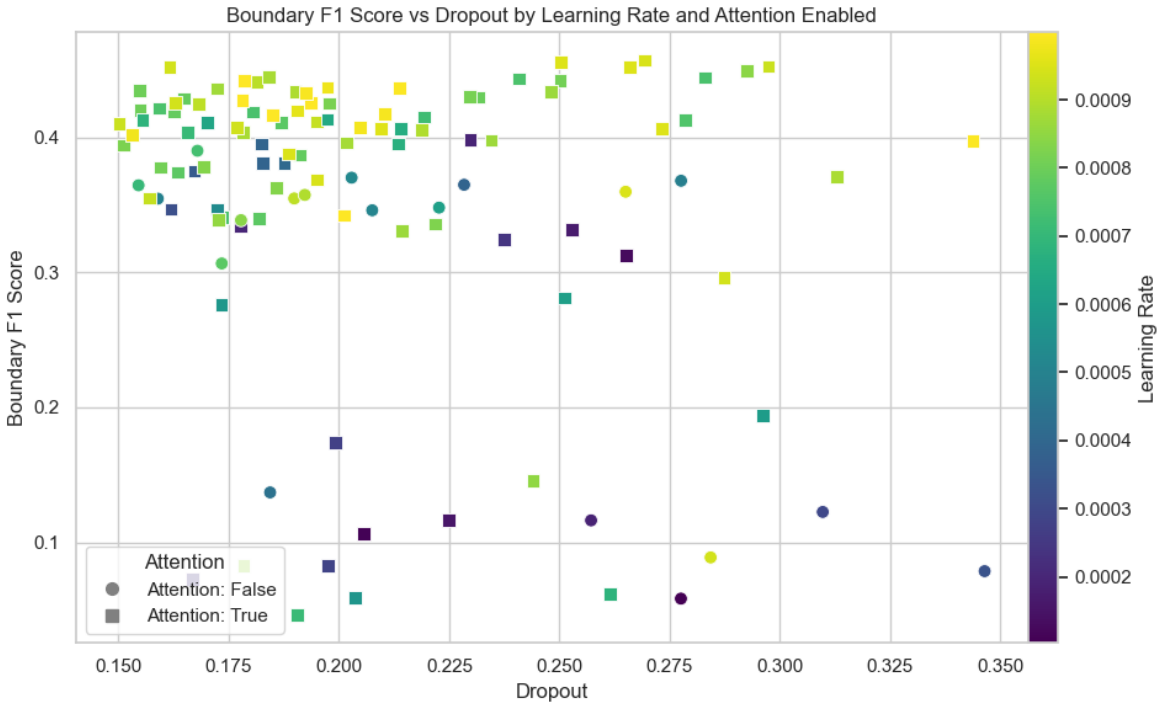


Figure 5.6 - Boundary F1 Score vs Dropout by Learning rate and in Attention Mechanism

Attention variant comparison

Comparison of the three attention variants yielded clear differences in segmentation performance. Localized attention achieved the highest median Boundary F1 score (~0.45), followed by standard self-attention (~0.39) and boundary-aware attention (~0.34) (see Figure 5.7). This hierarchy suggests that local contextual information within constrained windows is more discriminative for verse–chorus segmentation than either unrestricted global dependencies or explicit boundary prediction mechanisms.

The superior performance of localized attention reflects the nature of lyrical structure: verses and choruses are often distinguished by short-range patterns such as rhyme schemes, meter, and repeated phrases. These cues are most effectively captured within limited textual neighborhoods. By contrast, the boundary-aware attention mechanism, despite its explicit focus on transition modeling, appeared to introduce additional complexity without a corresponding performance benefit for this domain.

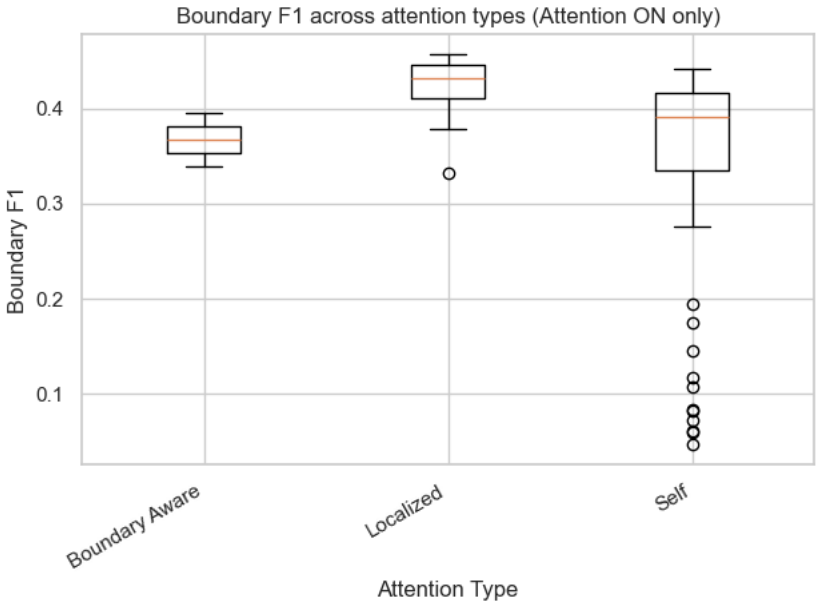


Figure 5.7 - Boundary F1 Across Attention Type

Implications for Lyric Structure Understanding

Together, these results provide empirical evidence that lyric segmentation is fundamentally a local pattern recognition task. The consistent advantage of localized attention supports the hypothesis that verses and choruses can be distinguished primarily through immediate textual cues rather than long-range dependencies. This finding highlights the significance of rhyme, repetition, and short-context semantics in shaping musical text structures, reinforcing the role of local linguistic and structural signals as key features for successful segmentation.

5.2.5. Loss Function Implementation

As we moved to attention-based architecture, it became clear that the segmentation task could benefit from more specialized optimization strategies. While our initial token-level cross-entropy loss performed adequately with BiLSTM models, we hypothesized that attention-enabled architectures might benefit from loss functions explicitly designed to capture boundary detection and segment coherence. This motivated us to design and evaluate a boundary-aware cross-entropy loss in conjunction with the established token-level formulation, to test whether direct boundary optimization would yield improvements in our new framework.

Loss function design

Our design is centered on balancing token-level classification accuracy with explicit encouragement of well-formed boundaries. To this end, we implemented two complementary loss functions:

- **Token-level cross-entropy with anti-collapse features (baseline):** This version extended standard binary cross-entropy by integrating safeguards developed in our earlier experiments, including label smoothing to prevent overconfident predictions, class weighting to mitigate imbalance, and entropy regularization to avoid single-class collapse.
- **Segmentation-aware cross-entropy (experimental):** This formulation placed additional emphasis on tokens near predicted boundaries, effectively weighting them more heavily during optimization. The goal was to encourage the model to refine its decision-making around transition points, thereby improving structural segmentation without compromising overall classification quality.

Both loss functions inherited the stability mechanisms proven effective in our previous training pipelines. By systematically comparing these implementations, we were able to test whether an explicit boundary focus would provide measurable advantages over robust token-level optimization alone.

Token-Level Cross-Entropy Implementation

The baseline implementation utilized standard cross-entropy loss, supplemented with several enhancements to stability and performance. Label smoothing was applied to soften target distributions and reduce brittle, overconfident predictions. Class weighting adjusted per-class importance to address the verse–chorus imbalance observed in our dataset. The loss computation also included masking for padded and invalid positions, ensuring that only valid tokens contributed to the average, thereby avoiding bias from alignment artifacts. Finally, an optional entropy regularization term was subtracted, a multiple of the mean entropy of the predicted distributions, which encourages predictive uncertainty and prevents collapse into single-class outputs.

Boundary-Aware Cross-Entropy Enhancement

As an alternative, we developed a segmentation-focused loss function designed to emphasize boundary detection. This boundary-aware variant extended the baseline with three specialized components:

- **Boundary weighting:** increased loss weights at label transitions, directly optimizing for accurate boundary identification.
- **Segment consistency:** penalized within-segment logit variance to encourage stability across tokens in the same section, reducing over-fragmentation of verses and choruses.
- **Confidence penalty:** applied quadratic penalties to excessively confident predictions, improving calibration and preventing extreme logit saturation.

Comparative Loss Function Analysis

To evaluate the effectiveness of different loss functions, we conducted a comparative analysis between the token-level cross-entropy loss and the boundary-aware loss, with results visualized in Figure 5.8. This figure presents performance metrics for two models: one trained with token-level loss and another with boundary-aware loss, each assessed across line-level F1 scores and boundary F1 scores with varying tolerances.

- **Line-level F1 Scores:** The left panels display line-level F1 metrics, including Macro F1, Micro F1 (accuracy), and class-specific F1 for Verses and Choruses. For the token-level loss model (top left), Macro F1 reached 0.782, Micro F1 0.813, Verse F1 0.864, and Chorus F1 0.701, indicating strong overall classification performance with a slight edge for verses. The boundary-aware loss model (bottom left) improved slightly, with Macro F1 at 0.779, Micro F1 at 0.816, Verse F1 at 0.869, and Chorus F1 at 0.609, suggesting robust token-level accuracy but a notable drop in chorus detection precision.
- **Boundary F1 Scores with Tolerance:** The right panels show boundary F1 scores, comparing exact matches (tolerance 0) and relaxed tolerances (±1 and ±2 lines). The token-level loss model achieved an exact Boundary F1 of 0.596, improving to 0.716 with ±1 tolerance and 0.747 with ±2 tolerance, reflecting better boundary precision. In contrast, the boundary-aware loss model scored 0.485 for exact matches, rising to

0.664 with ±1 tolerance and 0.709 with ±2 tolerance, indicating less accuracy in precise boundary placement despite gains at higher tolerances.

Despite the theoretical advantage of boundary-aware loss in targeting segment transitions, the empirical results revealed that its added complexity introduced competing optimization pressures, leading to inconsistent boundary detection (0.485 vs. 0.596 for exact F1). The token-level cross-entropy loss, enhanced with stability mechanisms such as dynamic thresholding and regularization, proved more effective, offering a simpler yet well-balanced objective. Consequently, we adopted this approach as our primary loss function, demonstrating its superiority in achieving precise segmentation under constrained conditions.

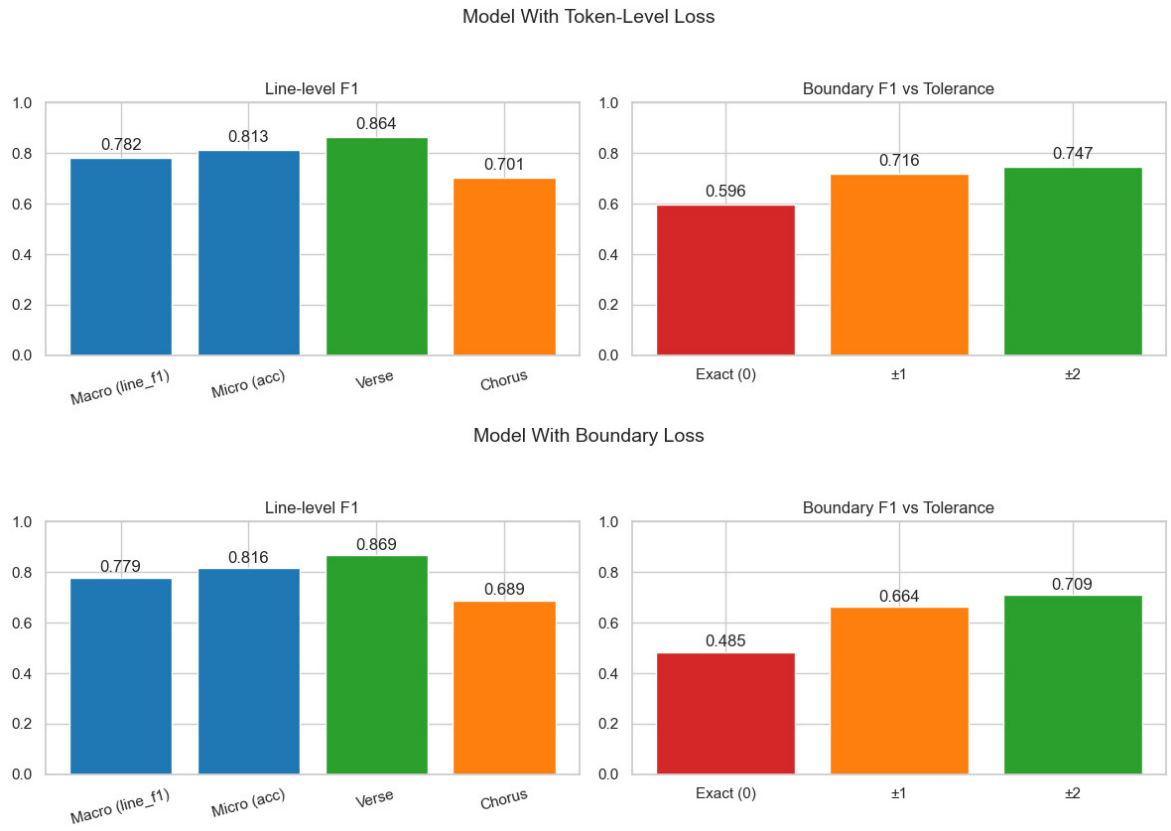


Figure 5.8 - Token Level Loss vs Boundary Loss

Convergent Evidence for Local Pattern Recognition

This outcome reinforces a consistent pattern observed throughout our architectural experiments: explicit boundary-focused mechanisms consistently underperformed simpler, locally focused approaches. Just as localized attention outperformed boundary-aware attention by concentrating on immediate contextual patterns rather than explicit boundary

prediction, our loss function evaluation demonstrates that token-level optimization provides more effective learning signals than boundary-specific penalties. Both findings point to the same underlying principle that lyric segmentation is fundamentally a local pattern recognition task, where the distinguishing features between verses and choruses emerge from immediate textual neighborhoods rather than global structural awareness. The convergence of these results across different model components suggests that successful lyric segmentation systems should prioritize local feature extraction and token-level accuracy over sophisticated boundary detection mechanisms, challenging the intuitive assumption that segmentation tasks necessarily benefit from boundary-aware optimization strategies.

5.2.6. Hyperparameter Optimization

After establishing stable training dynamics and identifying effective architectural strategies, we turned to systematic hyperparameter optimization. Our modular YAML-based configuration engine allowed seamless integration with Optuna, enabling us to move beyond manual trial-and-error and explore parameter spaces systematically. A total of 115 optimization trials were conducted, targeting learning rate, batch size, dropout values (for both BiLSTM and attention mechanisms), label smoothing coefficients, and entropy regularization strength.

Local vs Global Pattern Recognition

A consistent theme across our experiments was the superiority of localized approaches over explicitly global or boundary-focused methods. Localized attention outperformed both standard self-attention and boundary-aware attention by focusing on immediate contextual patterns. At the same time, token-level cross-entropy loss surpassed boundary-aware loss functions in precision and stability.

This convergence highlights a fundamental principle of lyric segmentation: local pattern recognition provides stronger learning signals than global modeling or explicit boundary optimization. The result aligns with the structural nature of lyrics, which rely on short-range cues such as rhyme, meter, and refrains rather than extended narrative coherence. In our dataset of 780 songs, macrostructural transitions were consistently signaled by repetitive local markers. Global sequence modeling often dilutes these cues by introducing long-range

dependencies, which reduces segmentation accuracy, particularly in noisy or ambiguous cases.

These findings reinforce earlier work noting that lyrical macrostructures emerge from local repetitions rather than global narrative arcs in Fell et al. (2018), and that segmentation benefits from exploiting these repetitions without requiring explicit boundary mechanisms Baratè et al. (2013). More broadly, this suggests that lyric segmentation models should prioritize architectures emphasizing local feature extraction, challenging the assumption that transformer-based global models are optimal for all sequential tasks. The principle may extend to other semi-structured, repetitive texts such as poetry or scripts.

Model Selection and Performance Optimization

Optuna-based optimization yielded two optimal configurations that reflected different trade-offs between performance and efficiency:

- **Robust model:** trained for 15 epochs in ~27 minutes, delivering the highest overall performance across metrics, particularly in chorus F1 scores and boundary detection at multiple tolerance levels.
- **Lite model:** trained for the same epoch count in ~13 minutes, achieving competitive scores with substantially reduced computational requirements.

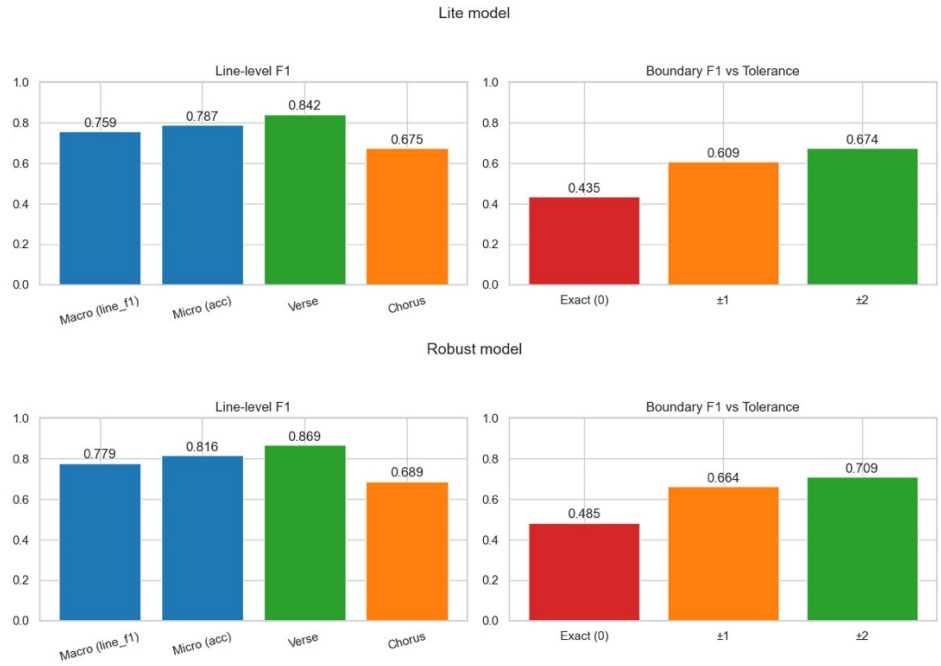


Figure 5.9 - Robust Model vs Lite Model

While the robust model became our primary implementation due to its superior accuracy, the lite model demonstrated that competitive performance can be achieved under tighter resource constraints (see Figure 5.9). The modest performance differential between the two highlights the architecture's sensitivity to hyperparameter tuning, suggesting that targeted optimization strategies could further narrow the gap.

This dual outcome underscores both the scalability potential of our approach and the importance of considering computational efficiency alongside segmentation accuracy. By balancing these trade-offs, lyric segmentation systems can be adapted for both high-performance research settings and resource-limited practical deployments.

5.2.7. Feature Engineering and Performance Analysis

While architectural choices and hyperparameter optimization established a stable and competitive segmentation framework, the effectiveness of the system also depended heavily on the features provided to the model. To better understand their individual and combined contributions, we conducted a systematic feature ablation analysis. Each feature was selectively enabled and disabled across multiple runs, and performance was measured primarily using Boundary F1 to assess segmentation accuracy. This process revealed a clear hierarchy of feature effectiveness for lyric segmentation.

Feature-Level Evidence for Local Pattern Focus

Figure 4.10 compares boundary detection performance (Boundary F1 Score) across different uses of head and tail self-similarity features, including configurations where each is enabled individually, disabled, or combined. The boxplots illustrate variation in performance across models, while the scatterplot shows the effect of word-level features in combination with positional self-similarity.

The most substantial evidence emerged from positional self-similarity features. Disabling head self-similarity (capturing similarity at the beginnings of lines) caused Boundary F1 scores to drop sharply across model configurations (see Figure 4.10). In contrast, tail self-similarity (capturing similarity at the ends of lines) offered minimal discriminative power on its own, though modest gains appeared when combined with head features. This asymmetry

indicates that verse–chorus transitions in English lyrics are primarily signaled by how lines begin rather than how they end.

The gap between head-enabled (median Boundary F1 ≈ 0.40) and head-disabled configurations (≈ 0.25) demonstrates that effective local context depends not just on proximity but also on specific positional cues (see Figure 5.10). Even within the short windows that proved most informative, the beginnings of lines carried disproportionate importance for segmentation.

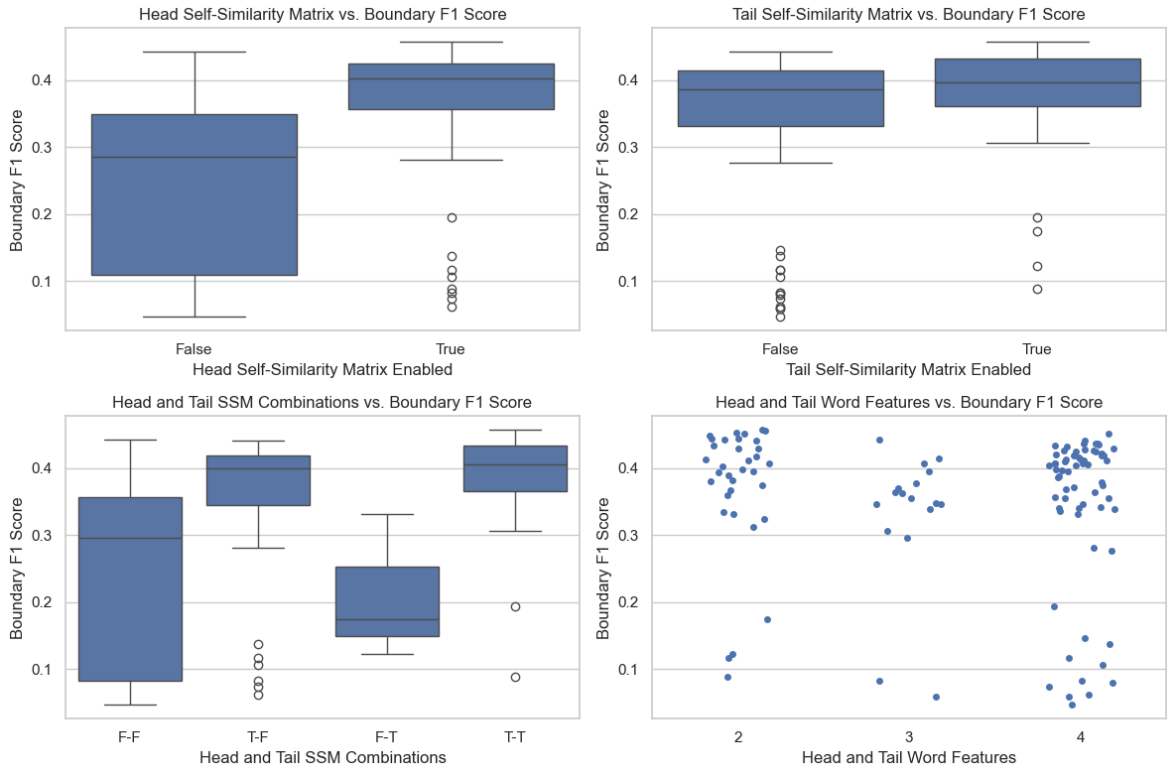


Figure 5.10 - Head and Tail SSM

Complementary Phonetic Patterns

In Figure 5.11, phonetic similarity features highlighted the acoustic dimension of local pattern recognition. Rhyme-based similarity (line endings) achieved markedly higher performance (Boundary F1 ≈ 0.38) than alliteration-based similarity (line beginnings, ≈ 0.17). This suggests that while semantic discrimination hinges online-initial tokens, phonetic discrimination depends on online-final sounds.

Combining rhyme and alliteration features produced only marginal gains over rhyme alone, indicating that the acoustic signature of verse–chorus transitions reside primarily in ending sounds. Together with the positional results, this suggests a complementary division: semantic beginnings and phonetic endings jointly define the most reliable cues for segmentation.

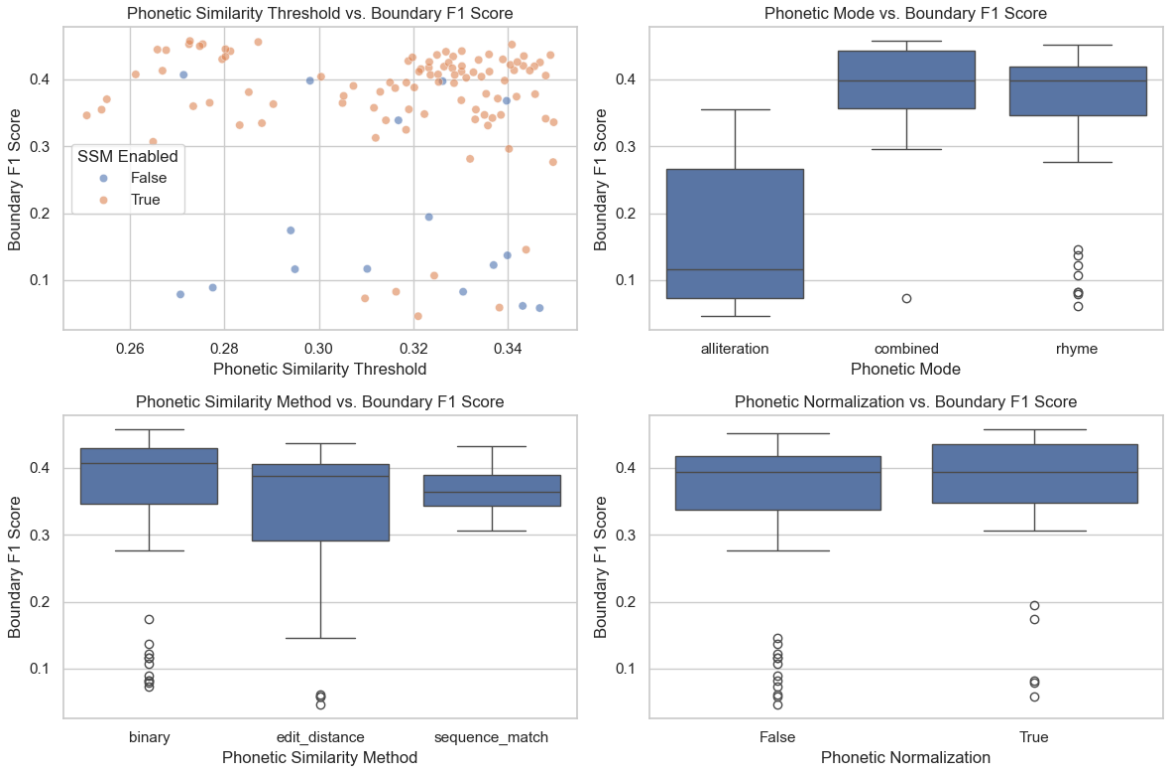


Figure 5.11 - Phonetic SSM vs Boundary F1 Score

High-Impact Features

The most influential features were Head SSM and Phonetic SSM, each boosting median Boundary F1 scores by 0.10–0.15 compared to their disabled variants (see Figure 5.12). Additional contributions came from POS SSM and Syllable Pattern SSM, which provided steady improvements across models. These findings confirm that syntactic regularities and phonetic rhythms convey a meaningful signal beyond the raw content of words.

Moderate-Impact Features

Word2Vec and Contextual embeddings yielded smaller yet consistent gains, typically raising Boundary F1 by 0.02-0.05. While not as dramatic as positional or phonetic cues, these embeddings captured complementary semantic information that generalized across different datasets and configurations, reinforcing their value as auxiliary features.

Low-Impact and Counter-Productive Features

Not all features improved performance. String SSM showed negligible discrimination, likely because lyrics reuse varied vocabulary even within similar sections, reducing the effectiveness of direct string matching. Notably, Line Syllable SSM consistently yielded degraded results (the only feature to do so). This counterintuitive outcome suggests that syllable counts across entire lines introduce noise, as they may vary within verses or choruses while remaining stable across sections, obscuring true boundaries.

Convergent Local Evidence

Taken together, these findings reinforce our architectural results: lyric segmentation is driven by local pattern recognition. The most effective features capture specific aspects of immediate textual neighborhoods, while broad or global similarity measures provide little benefit. By focusing on line-initial semantics and line-final phonetics, models exploit the precise cues that distinguish verses from choruses within short contexts.

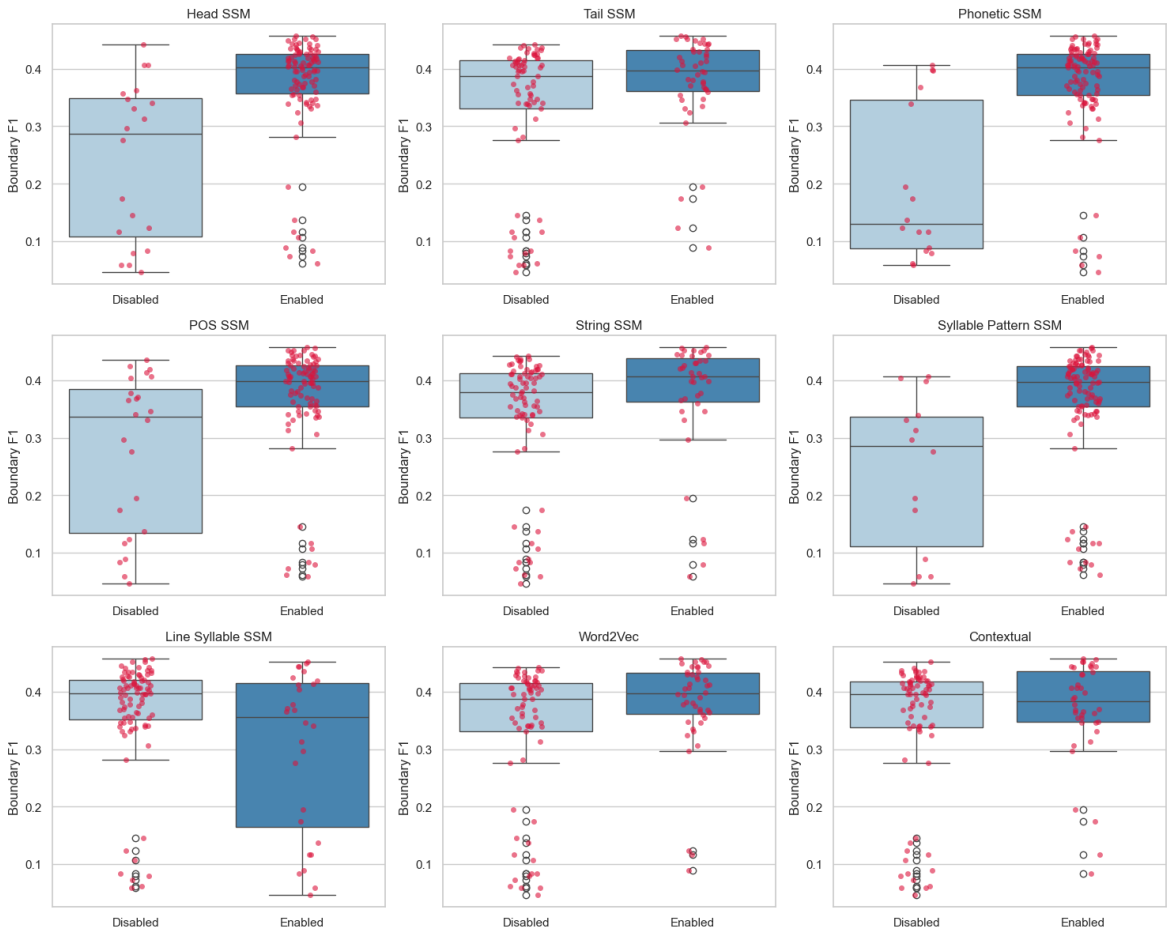


Figure 5.12 - Feature Presence Impact on Boundary F1

5.2.8. Model Calibration

One recurring challenge we identified throughout training was the model's tendency toward overconfident predictions. Even after stabilization measures such as label smoothing and entropy regularization, probability outputs often clustered near extremes. While this behavior did not always degrade segmentation accuracy, it reduced the trustworthiness of probability estimates.

To address this issue, we applied post-hoc calibration methods that adjust the mapping between model logits and predicted probabilities without altering the underlying classifier. Two techniques were evaluated: temperature scaling and Platt scaling. Both are lightweight and computationally efficient, making them well-suited to resource-constrained settings.

Figure 4.13 illustrates the effect of calibration on the Expected Calibration Error (ECE), a metric that measures the alignment between predicted probabilities and observed accuracy. High ECE values indicate that a model's predicted confidence does not match its true correctness rate, while lower values reflect more reliable probability estimates.

Both calibration methods substantially reduced ECE: from approximately 0.048 before calibration to 0.027 with temperature scaling, and from 0.048 to 0.032 with Platt scaling. Although accuracy metrics remained unchanged, these improvements indicate that our probabilities have become significantly more reliable indicators of segmentation confidence.

By combining calibration with the training-time stability framework, we achieved a model that not only segments lyrics accurately but also produces probability outputs that can be trusted and acted upon. This step enhances the practical usability of the system in settings where calibrated confidence is as important as raw accuracy.

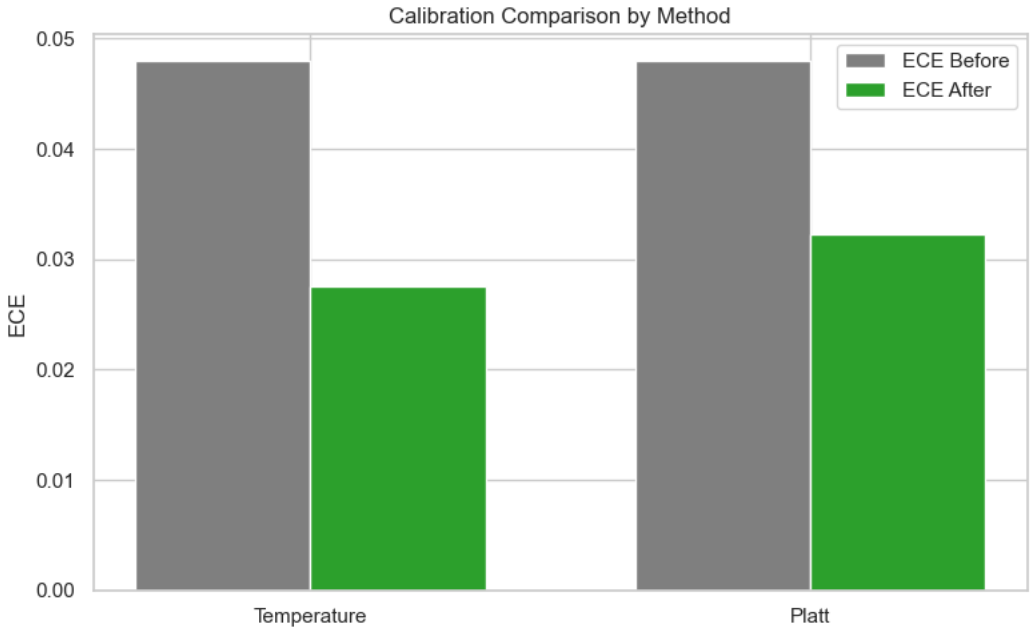


Figure 5.13 - Calibration Comparison (ECE)

5.2.9. Comparisons with State of the Art

Our proposed Bi-LSTM with localized attention achieves competitive performance across multiple evaluation metrics when compared to existing approaches for lyric segmentation. On the line-level segmentation task, the model achieved a Macro F1 score of 78.2% and a Micro F1 score of 81.3%.

Compared to Fell et al. (2018), who reported a segment-level F1 of 67.4% using their SSMall + n-grams model on the MLDB dataset of 25,000 songs (Fell et al., 2018), our approach improves performance by more than ten percentage points. This gain highlights the effectiveness of Bi-LSTM with localized attention in capturing repetitive lyrical structures.

More recently, Frohmann et al. (2024) reported macro-F1 scores between 77% and 79% with their SAT+LoRA framework across diverse musical genres, peaking at 79.3% in high-repetitiveness contexts. Our Macro F1 of 78.2% is directly competitive with these state-of-the-art results, demonstrating robustness across lyrical domains. Similarly, Watanabe & Goto (2020) achieved an F-measure of 78.1% for chorus detection using a Bi-LSTM trained on more than 100,000 songs. Our line-level F1 closely aligns with theirs, indicating strong generalization for identifying lyrical boundaries. Earlier convolutional approaches in *Lyrics Segmentation: Textual Macrostructure Detection Using Convolutions* by Fell et al. (2018)

delivered competitive results but did not exceed our performance, further validating our hybrid architecture.

A key distinction of our contribution lies in the training dataset size. While prior studies leveraged very large corpora (25,000 songs for Fell, over 100,000 songs for Watanabe, see Table 3.1), and large multi-genre corpora for Frohmann et al. (2024), our model achieved competitive, state-of-the-art performance with only 780 songs. Achieving comparable accuracy with an order of magnitude fewer examples.

Boundary-based evaluations reflected the additional complexity of lyric data. Our Boundary F1 score reached 0.596 for exact matches, 0.716 with a ±1 tolerance, and 0.747 with a ±2 tolerance. These increasing values reflect the irregularity of lyrical boundaries, where minor annotation deviations are common. Error-based metrics further highlighted this challenge: our Pk was 24.7% and WindowDiff 30.1%, substantially higher than long-document benchmarks. For example, in the text *A Neural Model For Text Segmentation* the authors reported an F1 of 57.7% on Wiki-727K with a hierarchical Bi-LSTM (Nicholls & Tanaka, 2021), and Lukasik et al. (2020) achieved a Pk of 0.07 on the Choi dataset using a cross-segment BERT model (*Text Segmentation by Cross Segment Attention*). These much lower error rates emphasize the unique difficulty of lyrical segmentation, where repetitions and stylistic variations obscure transitions in ways not observed in encyclopedic or synthetic corpora.

It is important to note that these comparisons are not direct replications of prior models on our dataset. The most directly comparable benchmarks are those from Fell et al. (2018), Frohmann et al. (2024), and Watanabe & Goto (2020), as each explicitly targeted lyric segmentation tasks. In contrast, our evaluation used a smaller corpus with distinct annotation criteria, making numerical comparisons approximate but still informative for contextualizing progress.

Taken together, these results demonstrate that our Bi-LSTM with localized attention achieves competitive performance for lyric segmentation, matching or exceeding earlier systems on line-level F1 metrics despite a dramatically smaller training set. This efficiency reinforces the potential of our method for real-world applications where annotated data is scarce, with line-level segmentation emerging as the most reliable and transferable measure of progress in this domain ass seen in our comparison table (see Table 4.1).

Model / Study	Dataset Size	Task	Metric (F1)	Notes
Our BLSTM + Attention model	780 songs	Boundary and line level segmentation	Boundary F1: 59.6% (exact) Macro F1 (Line Level): 78.2%	Small Dataset
Fell et al. (2018)	25,000 (MLDB)	Segment-level	67.4% (F1)	Baseline on MLDB corpus
Frohmann et al. (2024)	Large multi-genre	Verse in genre Segmentation	77–79% (Macro-F1)	Peak 79.3% in high repetitiveness
Watanabe & Goto (2020)	100,000+ songs	Chorus detection	78.1% (Macro-F1)	Chorus-specific task

Table 5.1 - Model Comparison

5.2.10. CNN-based Model

While our Bi-LSTM with localized attention achieves competitive state-of-the-art performance in lyric segmentation, alternative neural architectures may capture complementary aspects of lyrical structure. Recurrent models excel at modeling sequential dependencies, but convolutional approaches have also proven effective in this domain. Fell et al. (2018), for example, demonstrated that convolutional models can detect lyrical macrostructures by leveraging their strength in identifying local textual patterns (Fell et al., 2018).

Given that lyrics often rely on short, repetitive motifs and phrase-level regularities, convolutional neural networks (CNNs) offer a natural alternative for modeling these cues. In the following section, we introduce our CNN-based model, describe its architectural design, and evaluate its performance in comparison to our Bi-LSTM system and prior convolutional approaches in lyric segmentation.

Architectural Validation of Local Processing

The CNN architecture implements local pattern recognition through its convolution operations, which process lyrics within sliding windows of neighboring tokens. This mechanism naturally aligns with the structural cues of lyrics that define verse–chorus transitions.

Our experiments showed that the CNN achieved competitive Boundary F1 (see Figure 5.14) scores (0.48–0.49), confirming that convolutional local processing is effective for lyric segmentation. Interestingly, performance was nearly identical across both token-level and boundary-aware loss functions, suggesting that CNNs benefit from their inherent stability in local feature extraction, regardless of explicit boundary optimization.

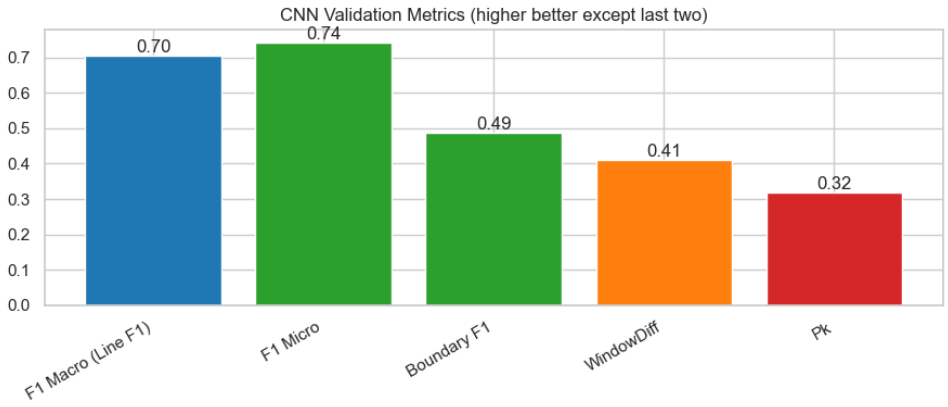


Figure 5.14 - CNN Validation Metrics

Distinctive Feature Preferences

Feature-level analysis revealed a contrast between CNNs and Bi-LSTMs in how they exploit local information. While Bi-LSTM models consistently relied on head self-similarity features (line beginnings), the CNN favored tail self-similarity matrices (line endings), in combination with Word2Vec and contextual embeddings. This difference suggests that CNNs may emphasize line-final cues and semantic context, whereas recurrent architectures prioritize line-initial signals.

These divergent preferences illustrate that multiple architectures can implement local pattern recognition, but they may emphasize different aspects of the text. For CNNs, the combination of tail features and dense semantic embeddings reflects an alternative pathway to capturing the same underlying principle.

Comparative Performance and Efficiency

Although the CNN did not surpass our best Bi-LSTM configurations, its Boundary F1 of ~0.488 places it within the competitive range of our local approaches. CNNs also offer notable computational efficiency, as convolution operations process local receptive fields in parallel rather than sequentially, making them attractive for deployment scenarios requiring faster inference.

Overall, the CNN results support our central finding: lyric segmentation is successful when models prioritize local pattern recognition. Whether implemented through attention windows in Bi-LSTMs or convolutional filters in CNNs, effective segmentation emerges from exploiting immediate textual neighborhoods rather than global dependencies. CNN's performance, achieved with a comparatively lightweight architecture, provides convergent evidence that the local-first principle generalizes across fundamentally different neural paradigms.

5.2.11. Interpretability Analysis and Case Study

Understanding why lyric segmentation models succeed or fail requires examining specific predictions against ground truth annotations. This section presents a detailed interpretability analysis of our BiLSTM + Localized Attention model through three representative case studies, each illustrating different failure modes that reveal the boundaries of local pattern recognition approaches.

Case Study Selection Methodology

Four songs that were not present on our train or validation sets were selected to represent distinct categories of segmentation challenges:

- Hotel California (The Eagles) - Complex multi-verse narrative structure
- Zombie (The Cranberries) - Multi-section structure with pre-chorus and post-chorus
- Fly Me to the Moon (Frank Sinatra) - Jazz standard with semantic similarity across sections
- Creep (Radiohead) - Verse–chorus structure with strong lyrical contrast and emotional escalation.

Each case study follows the same analytical framework: identification of specific error patterns and mapping these errors to theoretical limitations identified in our research, the first three case studies are unsuccessful attempts for the model, and the last one (Creep) a successful one, also an example of the way the model consumes these lyrics can be seen in Appendix 1.

Case Study 1: Hotel California

Successful First Chorus Detection: The model correctly identified the first chorus occurrence with the distinctive "*Welcome to the Hotel California*" refrain. This validates our findings about repetition-based pattern recognition and the effectiveness of Head SSM

features in detecting chorus-specific opening phrases (e.g. check Appendix 2 for true labels and Appendix 3 for predicted labels).

Failed Second Chorus Detection: Despite identical lyrical content in the second chorus (lines 41-48, see Appendix 3), the model failed to classify it as a chorus, instead merging it with surrounding verses. This suggests the model may be learning positional biases rather than pure content-based chorus identification.

Theoretical Implications: The partial success supports our conclusion that "Head self-similarity features (capturing similarity at line beginnings) proved most critical" for detecting chorus sections. However, the missed second chorus indicates that local pattern recognition may be influenced by song position rather than pure repetition detection.

Case Study 2: Zombie

Complete First Chorus Miss: The model failed to detect the first occurrence of "*In your head, in your head, Zombie, zombie, zombie-ie-ie*" as chorus content (see Appendix 5), instead classifying it as verse. This represents a fundamental failure in chorus-specific pattern recognition.

Misattributed Second Chorus: The model detected chorus content in the second half but misplaced the boundary, starting the chorus prediction in the pre-chorus section rather than the actual chorus.

Multi-Section Confusion: The presence of pre-chorus and post-chorus sections created false repetition signals. The phrase "*In your head*" appears in both pre-chorus and chorus, confusing the repetition-based detection mechanism that our project identified as central to chorus recognition.

Binary Classification Limitation: Our binary verse/chorus framework forced structurally distinct pre-chorus sections into the verse category, creating noisy training signals that degraded chorus detection accuracy.

Case Study 3: Fly Me to the Moon

On this case study the model failed to detect any chorus content despite clear repetition of "*Fill my heart with song*" sections (see Appendix 7). This represents the most severe form of chorus detection failure.

Preprocessing Impact: The instrumental bridge section was completely removed ([Instrumental]) during preprocessing (see Appendix 6 for true labels and Appendix 7 for predicted labels), as described in our methodology: "Any remaining bracketed or inline metadata, such as '[chorus]' or '(x2)', is stripped to prevent models from learning shortcuts based on explicit labels." This created an artificially extended 13-line chorus section (lines 7-19) when the two chorus instances were concatenated.

The removal of the instrumental bridge and the 13-line continuous chorus section likely exceeded the model's learned expectations for chorus length. Our dataset analysis showed that most songs from our dataset shared a standard music structure with shorter and more concise chorus sections (see Figure 3.5).

Training Data Length Distribution: The model was trained on choruses that typically span 4-8 lines based on popular music conventions. The artificially extended 13-line chorus created by preprocessing may have been interpreted as an anomalously long section that couldn't be a chorus, leading the model to default to verse classification.

Preprocessing-Induced Structural Distortion: The removal of structural markers in this case, "*[Instrumental]*" eliminated crucial contextual information that would have helped the model understand that two separate chorus instances were present, rather than one unusually long section.

Feature Engineering Limitations: Our Self-Similarity Matrix features are designed to detect repetition patterns within typical section lengths. An artificially extended chorus may have diluted repetition signals or created internal similarity patterns that conflicted with chorus detection.

Case Study 4: Creep

Almost Perfect Chorus Recognition: The model successfully identified all three chorus instances with the distinctive "*But I'm a creep, I'm a weirdo*" refrain (see Appendix 9), demonstrating effective application of our repetition-based pattern recognition with a slight tolerance for the starting and ending boundaries for each one.

Head SSM Feature Success: Each chorus begins identically with "*But I'm a creep,*" (see Appendix 8), providing strong head self-similarity signals that our ablation studies identified as most critical for boundary detection. This validates the feature's effectiveness when clear repetitive openings are present.

Robust Repetition Detection: Despite slight variations in chorus length (the final chorus includes additional "*I don't belong here*" repetitions as seen in Appendix 8), the model maintained consistent chorus classification, showing resilience to minor structural variations within repeated sections.

Minor Boundary Displacement: The model slightly expanded the first chorus detection to include the preceding line "*I wish I was special,*" which appears at the end of Verse 1. This demonstrates the local pattern recognition principle, the model detected that this line shares thematic and emotional content with the chorus, even though it's structurally part of the verse.

Bridge Handling Success: The model correctly classified the bridge section ("*She's runnin' out the door...*") as verse, avoiding false chorus detection despite the repetitive "*run, run, run*" elements. This shows the model learned to distinguish between chorus-type repetition and other forms of lyrical repetition.

Validation of Chorus Detection Challenges

The first three case studies reveal systematic patterns in chorus detection failures:

- **Position-Dependent Detection:** Hotel California shows successful first chorus detection but missed repetition, suggesting positional rather than content-based learning
- **Multi-Section Interference:** Zombie demonstrates how structural complexity beyond verse-chorus creates false signals that degrade chorus detection

- **Genre-Specific Blindness:** Fly Me to the Moon reveals complete failure on non-popular music structures

While the last case study (Creep) demonstrates that the BiLSTM + Localized Attention model can succeed when repetitive head patterns are strong and structurally consistent, the contrast with the first three cases highlights its boundaries. Successful detection depends heavily on conditions aligning with the model's inductive biases: short, repeated openings and stable chorus length distributions. When songs deviate from these conventions like long narrative structures, multi-section complexity, or genre-specific departures from pop patterns the model tend to struggle.

General Interpretability Insights

Together, these case studies underline three critical insights about model interpretability in lyric segmentation:

- Local feature dependency: Head self-similarity features provide robust signals, but the model lacks global awareness to consistently detect repeated choruses across distant positions.
- Structural bias: Pre-chorus and post-chorus sections expose the limitations of binary labeling, forcing heterogeneous structures into a verse/chorus dichotomy that introduces confusion.
- Preprocessing sensitivity: The Fly Me to the Moon failure demonstrates how aggressive preprocessing can distort true section boundaries, suggesting future work must preserve more structural context.

Implications for Model Development and Data Scale

These findings imply that while localized self-similarity is highly effective in mainstream pop contexts, broader generalization requires careful extensions. The binary verse/chorus framework forces structurally distinct elements such as pre-chorus or bridge sections into a single "verse" class, creating noise that the model cannot easily resolve. Multi-class labeling schemes, global attention mechanisms, or hybrid approaches that fuse textual repetition with semantic or even audio-derived cues could help mitigate these weaknesses.

At the same time, it is important to recognize that these shortcomings may not stem solely from the architecture itself. The model was trained on fewer than one thousand songs, a

fraction of the data available in large-scale lyric corpora. A richer and more diverse training set, including jazz standards, extended ballads, and multi-section pop compositions, might allow the BiLSTM + Localized Attention model to internalize subtler distinctions between chorus-like repetition and other recurring structures. In this sense, interpretability analysis highlights both architectural refinements and the possibility that simply scaling the training data could significantly improve robustness, extending the model's reach beyond the verse–chorus dichotomy without abandoning its data-efficient design.

6. Conclusions and Future Work

The main objective of this project was to investigate whether text-only methods could achieve accurate and data-efficient segmentation of song lyrics, with a particular focus on chorus recognition. Specifically, the research sets out to design a modular pipeline that combines dataset construction, feature extraction, and neural architectures to address the limitations of current approaches, which often rely on large, annotated datasets or multimodal inputs. By pursuing this objective, the goal was to evaluate whether structural and semantic cues encoded in lyrics themselves could provide a reliable foundation for segmentation in low-resource scenarios.

This research demonstrates that structural and semantic features can encode complementary information, significantly enhancing chorus recognition in English song lyrics. Given the inherent class imbalance in lyrical structures, developing robust frameworks to address this imbalance proves crucial for preventing model collapse and ensuring reliable performance across different song sections.

Our experiments with a modest dataset of 780 songs reveal that Bidirectional Long Short-Term Memory (BiLSTM) models achieve the best overall performance. Remarkably, our approach not only matches state-of-the-art metrics in music information retrieval for English songs but surpasses them in some key areas, demonstrating that effective chorus detection can be achieved without massive datasets or computational resources.

When considering industry-standard evaluation practices that typically allow for boundary tolerance, our results become even more compelling. The ±1 and ±2 tolerance metrics (71.6% and 74.7% respectively) reflect more realistic performance expectations in practical applications. Remarkably, even our strictest exact boundary performance is competitive with state-of-the-art approaches in Music Information Retrieval: our line-level performance matches Watanabe & Goto's (2020) chorus-specific BiLSTM model (78.1% Macro-F1) that used over 100,000 songs, while our approach achieves this with only 780 songs. This demonstrates that effective chorus detection can be achieved without massive datasets or computational resources.

6.1. Key Findings on Local Context Dependency

A fundamental finding of this work is that chorus detection relies heavily on local and near-neighboring contextual patterns rather than global structural awareness. This conclusion is supported by converging evidence from multiple experimental approaches:

Architectural Evidence: Our experimentation with boundary detection mechanisms consistently showed that explicit boundary-focused approaches underperformed simpler, locally focused methods. Localized attention mechanisms outperformed boundary-aware attention by concentrating on immediate contextual patterns, while token-level optimization provided more effective learning signals than boundary-specific penalties.

Feature-Level Evidence: Systematic feature ablation analysis revealed a clear hierarchy of effectiveness that reinforces the local pattern principle. Head self-similarity features (capturing similarity at line beginnings) proved most critical, with their removal causing sharp drops in Boundary F1 scores from approximately 0.40 to 0.25 across model configurations. In contrast, tail self-similarity features offered minimal discriminative power independently, demonstrating that verse-chorus transitions in English lyrics are primarily signaled by how lines begin rather than how they end.

Complementary Pattern Discovery: The analysis uncovered a sophisticated division of labor within local contexts. While semantic discrimination relies on line-initial tokens (head features), phonetic discrimination depends on line-final sounds, with rhyme-based similarity achieving significantly higher performance (Boundary F1 $\approx$ 0.38) than alliteration-based similarity ($\approx$ 0.17). This suggests that effective segmentation requires both semantic beginnings and phonetic endings working in concert.

High-Impact Feature Identification: Head SSM and Phonetic SSM emerged as the most influential features, each boosting median Boundary F1 scores by 0.10-0.15. Additional contributions from POS SSM and Syllable Pattern SSM confirmed that syntactic regularities and phonetic rhythms add a meaningful signal beyond raw word content.

These converging results challenge the intuitive assumption that segmentation tasks necessarily benefit from boundary-aware optimization strategies. Instead, they demonstrate that distinguishing features between verses and choruses emerge from immediate textural neighborhoods with specific positional and acoustic characteristics, suggesting that

successful lyric segmentation systems should prioritize local feature extraction and token-level accuracy.

6.2. Architectural Insights and Model Behavior

Our comparative analysis revealed distinct behavioral patterns across different architectures. BiLSTM models consistently outperformed both CNN and fine-tuned LLM approaches within our constrained environment, characterized by limited hardware and dataset size. Notably, feature-level analysis uncovered complementary strengths:

- BiLSTM models consistently relied on head self-similarity features, focusing on line beginnings and prioritizing line-initial signals
- CNN models favored tail self-similarity matrices (line endings) combined with Word2Vec and contextual embeddings, emphasizing line-final cues and semantic context

These divergent preferences illustrate that while multiple architectures can implement local pattern recognition effectively, they emphasize different textual aspects, potentially offering complementary insights for ensemble approaches.

6.3.Implications for the Field

This work demonstrates that achieving state-of-the-art performance in chorus segmentation (78.2% Macro-F1, matching large-scale approaches) does not require massive datasets or extensive computational resources. Our success, achieved with only 780 songs (compared to the 25,000+ songs typically used in MIR research), stems from our focused attention to training stability, data curation, feature engineering, and systematic iteration on different combinations of these customizations. This finding has significant implications for researchers working with limited resources, suggesting that thoughtful methodology and targeted optimization can effectively compete with resource-intensive approaches.

Our results contribute to a growing understanding that, in specialized domains such as lyric analysis, carefully designed local pattern recognition can be more effective than complex global modeling strategies, opening new directions for efficient and accessible music information retrieval systems.

6.4. Future Work

While this work achieves strong results under constraints, several avenues warrant exploration to address limitations and extend contributions. First, scaling computational resources could enhance all models: fine-tuning larger LLMs (e.g., beyond Gemma-1B-IT) on expanded datasets, increasing epochs beyond 5, and using batch sizes >1 to mitigate suboptimal convergence and overfitting risks observed in our 315-song fine-tuning subset. This might also incorporate perceptual or musical metrics to better align evaluations with human auditory experience, where our chorus F1 scores were consistently lower than verses (potentially due to class imbalance).

To improve generalization, future studies could expand the dataset beyond 780+125 songs, incorporating larger benchmarks like WASABI or MLDB, while testing for robustness across genres, as suggested by structural pattern analysis (Fig. 3.6). Extending binary labeling to multi-class segmentation (e.g., including bridges, intros, and outros) could enrich structural understanding, addressing the simplification in our approach.

A notable contribution from this project lies in the identification of 15 diverse song structure patterns (Fig. 3.6), which could be transformed into boolean features indicating the presence of any of these structures (e.g., verse-chorus alternation or complex schemes). Future work should explore the effectiveness of these features in tasks such as Music Emotion Recognition (MER) classification, potentially revealing their utility in capturing affective or stylistic nuances across different musical contexts.

Architecturally, developing a Mixture of Experts (MoE) framework by freezing our BiLSTM and CNN weights and training a router layer could fuse their complementary local strengths (BiLSTM's head-focused near-context with CNN's tail and semantic emphasis) for superior ensemble performance. Additionally, exploring boundary F1 improvements (our 0.47 vs. baseline 0.59) through advanced attention variants or hybrid audio-text integration could make the method more competitive. Overall, these directions could broaden the pipeline's applicability to multilingual lyrics or real-time MIR systems.

7. Bibliography

Baratè, A., Luca, L. A., & Santucci, E. (2013). A semantics-driven approach to lyrics segmentation. *Proceedings - 8th International Workshop on Semantic and Social Media Adaptation and Personalization, SMAP 2013*, 73–79. https://doi.org/10.1109/SMAP.2013.15

Cheng, T., Nakano, T., & Goto, M. (2025). *Improving Lyrics-to-Audio Alignment Using Frame-wise Phoneme Labels with Masked Cross Entropy Loss*. https://www.dafx.de/paper-archive/2025/DAFx25_paper_15.pdf

Durand, S., Stoller, D., & Ewert, S. (2023). Contrastive Learning-Based Audio to Lyrics Alignment for Multiple Languages. *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096725

Fell, M., Nechaev, Y., Cabrio, E., Gandon, F., & Kessler, F. B. (2018). Lyrics Segmentation: Textual Macrostructure Detection using Convolutions. *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, 2044–2054. https://aclanthology.org/C18-1174/

Frohmann, M., Sterner, I., Vulić, I., Minixhofer, B., & Schedl, M. (2024). *Segment Any Text: A Universal Approach for Robust, Efficient and Adaptable Sentence Segmentation*. http://arxiv.org/abs/2406.16678

Ghinassi, I., Wang, L., Newell, C., Purver, M., & Jožef Stefan, I. (2024). Recent Trends in Linear Text Segmentation: A Survey. *Findings of the Association for Computational Linguistics: EMNLP 2024*, 3084–3095. https://doi.org/10.18653/v1/2024.findings-emnlp.174

Google. (2025, August 14). *Gemma 3 model overview*. https://ai.google.dev/gemma/docs/core

Huang, J., Benetos, E., & Ewert, S. (2022). Improving Lyrics Alignment Through Joint Pitch Detection. *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 451–455. https://doi.org/10.1109/ICASSP43922.2022.9746460

Lukasik, M., Dadachev, B., Papineni, K., & Simões, G. (2020). Text Segmentation by Cross Segment Attention. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 4707–4716. https://doi.org/10.18653/v1/2020.emnlp-main.380

Malheiro, R. (2016). *Emotion-based Analysis and Classification of Music Lyrics*. University of Coimbra.

Nicholls, A. S., & Tanaka, G. (2021). *A Neural Model for Text Segmentation*. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/reports/final_reports/report001.pdf

Stoller, D., Durand, S., & Ewert, S. (2019). *End-To-End Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model*. https://arxiv.org/abs/1902.06797

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 5998–6008. https://doi.org/10.48550/arXiv.1706.03762

Watanabe, K. (2018). *Modeling Discourse Structure of Lyrics* [Tohoku University, Graduate School of Information Sciences]. https://www.cl.ecei.tohoku.ac.jp/publications/2018/kento_dthesis.pdf

Watanabe, K., & Goto, M. (2020). A Chorus-Section Detection Method For Lyrics Text. *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020)*, 315–359. https://archives.ismir.net/ismir2020/paper/000088.pdf

Glossary

CMUdict (Carnegie Mellon Pronouncing Dictionary): Maps English words to phoneme sequences (with stress). Standard resource for phonetic features in NLP/MIR.

F1 metric (F1 score): A performance measure that combines precision (the proportion of correct positive predictions among all positive predictions) and recall (the proportion of correct positive predictions among all actual positives) into a single score. It is the harmonic mean of precision and recall, ranging from 0 to 1, where 1 indicates perfect performance.

Jaccard similarity: Measures overlap between two sets by comparing shared items to all unique items combined. Common for token-based text similarity; ranges from 0 (no overlap) to 1 (identical).

Jamendo benchmark: A publicly available dataset of real-world polyphonic music with time-aligned lyrics, widely used to evaluate lyrics-to-audio alignment systems under challenging, multi-instrument conditions.

Levenshtein distance: A string similarity metric that calculates the minimum number of single-character edits (insertions, deletions, substitutions) required to transform one string into another. Commonly used in text processing, error correction, and natural language tasks.

Low-Rank Adaptation (LoRA): A parameter-efficient fine-tuning method for large language models. It introduces small trainable low-rank matrices into specific layers of a pretrained model, enabling adaptation to new tasks with significantly fewer parameters.

min-max scaling: Linearly maps a feature to a fixed range, typically 0–1. Preserves ordering but can be skewed by outliers.

NLTK (Natural Language Toolkit): A popular Python library for NLP. Provides tokenizers, POS taggers, and utilities used for text preprocessing.

Optuna: An open-source hyperparameter optimization framework that automates the tuning of machine learning models using efficient algorithms like Tree-structured Parzen Estimator (TPE) and Bayesian optimization.

Pk metric (Probability of error): An evaluation measure for text segmentation. It calculates the probability that two units of text are incorrectly classified as being in the same segment or in different segments. Lower values indicate better segmentation quality.

Self-Similarity Matrices (SSM): A matrix representation where the axes correspond to the sequence of lyric lines, and each cell represents the similarity score between two lines. It is used to visualize and identify structural patterns and repetitions, such as choruses, within a song.

Stanza: A number of lines or verses forming a division of a song or poem, and agreeing in meter, rhyme, number of lines, etc., with other divisions.

Tagset: A predefined collection of part-of-speech (POS) tags used to label words in a text based on their grammatical categories, such as Universal, Penn Treebank.

WindowDiff: A segmentation evaluation metric designed to address limitations of Pk. It applies a sliding window across the text to check whether the number of boundaries inside the window matches between the reference and the proposed segmentation.

Z-score normalization: Rescales a feature so its average is zero and spread is one. Useful when features look roughly bell-shaped; not bounded to a fixed range.

Appendixes

Appendix 1 - Example JSONL entry from the annotated dataset


```
{
"id":"training_data_0402",
"lines":[
"Comin' out of my cage and I've been doin' just fine","Gotta,
gotta be down because I want it all","It started out with a kiss,
how did it end up like this?","It was only a kiss, it was only a
kiss","Now I'm falling asleep and she's calling a cab","While he's
having a smoke and she's taking a drag","Now they're goin' to bed
and my stomach is sick","And it's all in my head, but she's
touching his","Chest now","He takes off her dress now","Let me
go","And I just can't look, it's killing me","And taking
control","Jealousy","Turning saints into the sea","Swimming
through sick lullabies","Choking on your alibis","But it's just
the price I pay","Destiny is calling me","Open up my eager
eyes","'Cause I'm Mr. Brightside","I'm comin' out of my cage and
I've been doin' just fine","Gotta, gotta be down because I want it
all","It started out with a kiss, how did it end up like
this?","(It was only a kiss) It was only a kiss","Now I'm falling
asleep and she's calling a cab","While he's havin' a smoke and
she's taking a drag","Now they're goin' to bed and my stomach is
sick","And it's all in my head, but she's touching his","Chest
now","He takes off her dress now","Let me go","'Cause I just can't
look, it's killing me","And taking control","Jealousy","Turning
saints into the sea","Swimming through sick lullabies","Choking on
your alibis","But it's just the price I pay","Destiny is calling
me","Open up my eager eyes","'Cause I'm Mr. Brightside","I
never","I never","I never","I never"
],
"labels":[0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0]
}
```


Appendix 2 - Case Study 1, Hotel California True Labels

```
[verse] On a dark desert highway
[verse] Cool wind in my hair
[verse] Warm smell of colitis
[verse] Rising up through the air
[verse] Up ahead in the distance
```

[verse] I saw a shimmering light
[verse] My head grew heavy and my sight grew dim
[verse] I had to stop for the night
[verse] There she stood in the doorway
[verse] I heard the mission bell
[verse] And I was thinkin' to myself
[verse] "This could be Heaven or this could be Hell"
[verse] Then she lit up a candle
[verse] And she showed me the way
[verse] There were voices down the corridor
[verse] I thought I heard them say
[chorus] "Welcome to the Hotel California
[chorus] Such a lovely place (Such a lovely place)
[chorus] Such a lovely face
[chorus] Plenty of room at the Hotel California
[verse] Her mind is Tiffany-twisted
[verse] She got the Mercedes Benz, uh
[verse] She got a lot of pretty, pretty boys
[verse] That she calls friends
[verse] How they dance in the courtyard
[verse] Sweet summer sweat
[verse] Some dance to remember
[verse] Some dance to forget
[verse] So I called up the Captain
[verse] "Please bring me my wine"
[verse] He said, "We haven't had that spirit here
[verse] Since 1969"
[verse] And still those voices are callin'
[verse] From far away
[verse] Wake you up in the middle of the night
[verse] Just to hear them say
[chorus] "Welcome to the Hotel California
[chorus] Such a lovely place (Such a lovely place)
[chorus] Such a lovely face
[chorus] They livin' it up at the Hotel California
[chorus] What a nice surprise (What a nice surprise)
[chorus] Bring your alibis"
[verse] Mirrors on the ceiling
[verse] The pink champagne on ice, and she said
[verse] "We are all just prisoners here
[verse] Of our own device"
[verse] And in the master's chambers
[verse] They gathered for the feast

[verse] They stab it with their steely knives
[verse] But they just can't kill the beast
[verse] Last thing I remember, I was
[verse] Running for the door
[verse] I had to find the passage back
[verse] To the place I was before
[verse] "Relax," said the night man
[verse] "We are programmed to receive
[verse] You can check out any time you like
[verse] But you can never leave"

Appendix 3 - Case Study 1, Hotel California Predicted Labels

[verse] On a dark desert highway
[verse] Cool wind in my hair
[verse] Warm smell of colitis
[verse] Rising up through the air
[verse] Up ahead in the distance
[verse] I saw a shimmering light
[verse] My head grew heavy and my sight grew dim
[verse] I had to stop for the night
[verse] There she stood in the doorway
[verse] I heard the mission bell
[verse] And I was thinkin' to myself
[verse] "This could be Heaven or this could be Hell"
[verse] Then she lit up a candle
[verse] And she showed me the way
[verse] There were voices down the corridor
[chorus] I thought I heard them say
[chorus] "Welcome to the Hotel California
[chorus] Such a lovely place (Such a lovely place)
[chorus] Such a lovely face
[chorus] Plenty of room at the Hotel California
[verse] Any time of year (Any time of year)
[verse] Her mind is Tiffany-twisted
[verse] She got the Mercedes Benz, uh
[verse] She got a lot of pretty, pretty boys
[verse] That she calls friends
[verse] How they dance in the courtyard
[verse] Sweet summer sweat
[verse] Some dance to remember
[verse] Some dance to forget
[verse] So I called up the Captain
[verse] "Please bring me my wine"

[verse] He said, "We haven't had that spirit here
[verse] Since 1969"
[verse] And still those voices are callin'
[verse] From far away
[verse] Wake you up in the middle of the night
[verse] Just to hear them say
[verse] "Welcome to the Hotel California
[verse] Such a lovely place (Such a lovely place)
[verse] Such a lovely face
[verse] They livin' it up at the Hotel California
[verse] What a nice surprise (What a nice surprise)
[verse] Bring your alibis"
[verse] Mirrors on the ceiling
[verse] The pink champagne on ice, and she said
[verse] "We are all just prisoners here
[verse] Of our own device"
[verse] And in the master's chambers
[verse] They gathered for the feast
[verse] They stab it with their steely knives
[verse] But they just can't kill the beast
[verse] Last thing I remember, I was
[verse] Running for the door
[verse] I had to find the passage back
[verse] To the place I was before
[verse] "Relax," said the night man
[verse] "We are programmed to receive
[verse] You can check out any time you like
[verse] But you can never leave"

Appendix 4 - Case Study 2, Zombie True Labels

[verse] Another head hangs lowly
[verse] Child is slowly taken
[verse] And the violence caused such silence
[verse] Who are we, mistaken?
[pre-chorus] But you see, it's not me, it's not my family
[pre-chorus] In your head, in your head, they are fightin'
[pre-chorus] With their tanks and their bombs and their bombs and their guns
[pre-chorus] In your head, in your head, they are cryin'
[chorus] In your head, in your head
[chorus] Zombie, zombie, zombie-ie-ie
[chorus] What's in your head, in your head?
[chorus] Zombie, zombie, zombie-ie-ie-ie, oh

[post-chorus] Doo, doo, doo, doo
[post-chorus] Doo, doo, doo, doo
[post-chorus] Doo, doo, doo, doo
[post-chorus] Doo, doo, doo, doo
[verse] Another mother's breakin'
[verse] Heart is takin' over
[verse] When the violence causes silence
[verse] We must be mistaken
[pre-chorus] It's the same old theme, since 1916
[pre-chorus] In your head, in your head, they're still fightin'
[pre-chorus] With their tanks and their bombs and their bombs and their guns
[pre-chorus] In your head, in your head, they are dyin'
[chorus] In your head, in your head
[chorus] Zombie, zombie, zombie-ie-ie
[chorus] What's in your head, in your head?
[chorus] Zombie, zombie, zombie-ie-ie-ie
[chorus] Oh-oh-oh-oh, oh-oh-oh, eh-eh-oh, ya-ya

Appendix 5 - Case Study 2, Zombie Predicted Labels

[verse] Another head hangs lowly
[verse] Child is slowly taken
[verse] And the violence caused such silence
[verse] Who are we, mistaken?
[verse] But you see, it's not me, it's not my family
[verse] In your head, in your head, they are fightin'
[verse] With their tanks and their bombs and their bombs and their guns
[verse] In your head, in your head, they are cryin'
[verse] In your head, in your head
[verse] Zombie, zombie, zombie-ie-ie
[verse] What's in your head, in your head?
[verse] Zombie, zombie, zombie-ie-ie-ie, oh
[verse] Doo, doo, doo, doo
[verse] Doo, doo, doo, doo
[verse] Doo, doo, doo, doo
[verse] Doo, doo, doo, doo
[verse] Another mother's breakin'
[verse] Heart is takin' over
[verse] When the violence causes silence
[verse] We must be mistaken
[verse] It's the same old theme, since 1916
[chorus] In your head, in your head, they're still fightin'

[chorus] With their tanks and their bombs and their bombs and their guns
[chorus] In your head, in your head, they are dyin'
[chorus] In your head, in your head
[chorus] Zombie, zombie, zombie-ie-ie
[chorus] What's in your head, in your head?
[chorus] Zombie, zombie, zombie-ie-ie-ie
[chorus] Oh-oh-oh-oh, oh-oh-oh, eh-eh-oh, ya-ya

Appendix 6 - Case Study 3, Fly Me To The Moon True Labels

[verse] Fly me to the moon
[verse] Let me play among the stars
[verse] And let me see what spring is like
[verse] On a-Jupiter and Mars
[verse] In other words, hold my hand
[verse] In other words, baby, kiss me
[chorus] Fill my heart with song
[chorus] And let me sing forevermore
[chorus] You are all I long for
[chorus] All I worship and adore
[chorus] In other words, please be true
[chorus] In other words, I love you
[instrumental]
[chorus] Fill my heart with song
[chorus] Let me sing forevermore
[chorus] You are all I long for
[chorus] All I worship and adore
[chorus] In other words, please be true
[chorus] In other words, in other words
[chorus] I love you

Appendix 7 - Case Study 3, Fly Me To The Moon Predicted Labels

[verse] Fly me to the moon
[verse] Let me play among the stars
[verse] And let me see what spring is like
[verse] On a-Jupiter and Mars
[verse] In other words, hold my hand
[verse] In other words, baby, kiss me
[verse] Fill my heart with song
[verse] And let me sing forevermore
[verse] You are all I long for
[verse] All I worship and adore
[verse] In other words, please be true

[verse] In other words, I love you
[verse] Fill my heart with song
[verse] Let me sing forevermore
[verse] You are all I long for
[verse] All I worship and adore
[verse] In other words, please be true
[verse] In other words, in other words
[verse] I love you

Appendix 8 - Case Study 4, Creep True Labels

[verse] When you were here before
[verse] Couldn't look you in the eye
[verse] You're just like an angel
[verse] Your skin makes me cry
[verse] You float like a feather
[verse] In a beautiful world
[verse] I wish I was special
[verse] You're so fuckin' special
[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?
[chorus] I don't belong here
[verse] I don't care if it hurts
[verse] I wanna have control
[verse] I want a perfect body
[verse] I want a perfect soul
[verse] I want you to notice
[verse] When I'm not around
[verse] You're so fuckin' special
[verse] I wish I was special
[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?
[chorus] I don't belong here
[chorus] Oh, oh
[bridge] She's runnin' out the door
[bridge] She's runnin' out
[bridge] She run, run, run, run
[bridge] Run
[verse] Whatever makes you happy
[verse] Whatever you want
[verse] You're so fuckin' special
[verse] I wish I was special

[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?
[chorus] I don't belong here
[chorus] I don't belong here

Appendix 9 - Case Study 4, Creep Predicted Labels

[verse] When you were here before
[verse] Couldn't look you in the eye
[verse] You're just like an angel
[verse] Your skin makes me cry
[verse] You float like a feather
[verse] In a beautiful world
[chorus] I wish I was special
[chorus] You're so fuckin' special
[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?
[chorus] I don't belong here
[verse] I don't care if it hurts
[verse] I wanna have control
[verse] I want a perfect body
[verse] I want a perfect soul
[verse] I want you to notice
[verse] When I'm not around
[verse] You're so fuckin' special
[chorus] I wish I was special
[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?
[chorus] I don't belong here
[chorus] Oh, oh
[verse] She's runnin' out the door
[verse] She's runnin' out
[verse] She run, run, run, run
[verse] Run
[verse] Whatever makes you happy
[verse] Whatever you want
[chorus] You're so fuckin' special
[chorus] I wish I was special
[chorus] But I'm a creep
[chorus] I'm a weirdo
[chorus] What the hell am I doin' here?

```
[chorus] I don't belong here
[chorus] I don't belong here
```

Appendix 10 - GitHub Repository (Models & Dataset)

To ensure reproducibility and transparency, the full implementation of the models, training pipeline, and annotated dataset used in this project are publicly available. The repository includes the BiLSTM and CNN architectures, the custom boundary-aware loss functions, hyperparameter optimization setup, and the curated lyric dataset.

- GitHub repository: https://github.com/rsmal-ipl/SEEEM-TM
- Last commit used for this project: 8c49468193b1137134e5470a21909407247dea7f