Final Project Writeup

601.466/666 - Information Retrieval and Web Agents

Jonathan Lee (jlee844@jh.edu, 601.666)
Ernie Chu (schu23@jhu.edu, 601.666)

Johns Hopkins University

***Summary**: This project explores the application of information retrieval (IR) techniques to the domain of song lyrics. It consists of two main components: 1) a large-scale, up-to-date lyric search engine utilizing TF-IDF vector modeling and longest common subsequence (LCS) for efficient retrieval, and 2) a classification system that categorizes lyrics into eight distinct emotion labels. The goal is to assess the effectiveness of traditional and semantic IR methods in handling lyrical content, both for search relevance and emotional understanding. Together, these components demonstrate the versatility of IR techniques in music lyric analysis.*

Lyric Search Engine: https://ernestchu.github.io/lyric-search

Project Repository: https://github.com/ernestchu/ir-project

May 11, 2025

1 User and code instruction

1.1 Lyric search engine

Our main Lyric Search App, `https://ernestchu.github.io/lyric-search`, provides one straightforward function, which lets user look up songs based on partial lyrics, artist, or album name, etc. Notably, the backend is written in Python and served on a free server, so it may take a while to retrieve the relevant lyrics.

1.2 Lyric mood classification

Understanding the high-level mood of a song based on its lyrics can enable more personalized user experience in music recommendation system. In light of the absence of publicly available large-scale lyric mood labels, we leverage GPT [1] to generate mood label for each lyric based on eight mood classes [5]. The collected data is further used for training classifiers. Our code is composed of three parts:

1. Generating training data using OpenAI API (`/lmc/lyrics_emotion_annotation_gpt.py`)
2. Preprocessing data (`/lmc/make_lyrics_emotion_dataset.py`)
3. Training and evaluating classifier (`lyrics_emotion_classification.py`)

More details can be found in Section 4.3.

2 Achievements and contributions

- Created a search engine webpage
- Classification model of 8 emotion class of emotions, reached 65 percent of f1 weighted score on models

3 Limitations and future directions

- Search Engine: In our lyric search engine, document ranking is heavily influenced by term frequency (TF). Given the extensive coverage of our lyric database, certain phrases frequently recur across songs. For instance, when searching for the lyric "from sea to shining sea," a song that repeatedly uses the phrase "shining sea" may receive a disproportionately high relevance score, potentially outranking the correct song containing the full query. To address this, we incorporate the longest common subsequence (LCS) between the query and retrieved lyrics as a secondary measure of relevance. However, computing LCS is computationally expensive ($O(mn)$) and requires full-text access, so it is only applied to the top 1,000 candidates selected based on TF scores. This introduces the risk of missing the truly relevant lyric if it is not initially included in that subset. One potential improvement is to impose an upper bound on TF to limit the influence of overly repeated terms.

[1] https://platform.openai.com

- Mood Classification:For the lyrics classification, we only retrieve and classify english song lyrics, we can further process non english songs as an improvement. This would require introduction of other NLP topics such as machine translation that utilizes back translation and sentence alignment to catch emotion labels.
- Mood Classification: For this project we used the openai gpt-3.5-turbo api to create the groundtruth labels only considering the raw lyrics text. The features used to classify the mood of each song also only utilizes the pure text lyrics. As an improvement and future direction, we can further expand the project by analyzing the actual music audio to classify the songs using models such as wave2vec to make the classification more robust. We can use feature such as distinct instrument, tempo, frequency range, incombination with the feautres of the text lyrics to make the classificatioin task more generalizable and robust [1].

4 Methods and evaluation results

4.1 Data preparation

We use the lyric database from `https://lrclib.net`. We used `/make_datasets.py` to convert the 11.7 GB database dump at 2025/03/19 to the Hugging Face datasets format,[2] and serve it at `https://huggingface.co/datasets/ernestchu/lrclib-20250319` for the ease of further processing of the lyric data.

4.2 Lyric search engine

To index the entire lyric database, we use `/lse/backend/prepare_tfidf_sim.py` to perform tokenization, bag-of-words conversion, TF-IDF vectorization, and optional Latent Semantic Indexing, with the help of `BERT` multilingual tokenizer [3] and `GENSIM`.[4] All of the resulting artifacts (models, corpus representation, etc) are uploaded to `https://huggingface.co/ernestchu/lrclib_gensim_assets` for later uses.

4.2.1 Backend

At the core of our lyric search engine, we retrieve top 1000 relevant lyrics by comparing the user's query against the entire lyric database in vector space using cosine similarity. Since the bag-of-words preprocessing discards the ordering of the texts, we further rank the retrieved lyrics by the length of the Longest Common Subsequence between user's query and each relevant lyric. The concrete implementation can be found in `/lse/backend/app.py`, and the program is running on the free server provided by Hugging Face Spaces [5] at `https://huggingface.co/spaces/ernestchu/lyric-search`, providing a minimal user interface and APIs.

[2] https://huggingface.co/docs/datasets
[3] https://huggingface.co/google-bert/bert-base-multilingual-uncased
[4] https://radimrehurek.com/gensim/index.html
[5] https://huggingface.co/spaces

LSI	LCS	Top-1	Top-5	Top-10	Top-20	Top-50
✗	✗	0.136	0.293	0.359	0.428	0.524
✗	✓	0.146	0.353	0.462	0.545	0.642
✓	✗	0	0	0	0	0
✓	✓	0	0	0	0	0

Table 1: Quantitative ablation study for our search engine. Retrieval success rate in top-k documents. LSI means Latent Semantic Indexing and LCS means Longest Common Subsequence.

4.2.2 Frontend

The web app is written in Vue.js [6], served on Github Pages, and can be found under `/lse/frontend`.

4.2.3 Quantitative evaluation

To systematically examine the contribution of each design choice in our search engine, we perform an quantitative ablation study. We first use `/lse/random_1k.py` to sample 1000 lyrics from our lyric database. For each song, we randomly select a snippet of lyrics and use our engine to search with the snippet and evaluate whether the correct song appears within the top-k ranked search results as implemented in `/lse/evaluate.py`. The results in Table 1 shows that the use of Longest Common Subsequence solidly improves the overall performance, and the semantic dense representation brought by Latent Semantic Indexing does not benefit the lyric search engine at all.

4.3 Lyric mood classification

4.3.1 Data annotation

Our feature extraction process involves several text preprocessing steps to enhance the quality of textual data. This includes removing meaningless words such as stop words and identifying emotionally salient terms using emotion lexicons [4, 8, 2]. We also extract named entities to capture specific references within the text. For similarity-based retrieval, we utilize both keyword-level and sentence-level similarity measures to improve matching accuracy. Specifically, we use Stanza, a multilingual NLP toolkit for lemmatization, POS tagging, and named entity recognition [6].

4.3.2 Coding Instruction

1. Run `/lmc/lyrics_emotion_annotation_gpt.py` to generate emotion cluster label and emotion word for the song that has plain lyrics This generates `emotion_labels.json` and `raw_lyrics.json`.

2. Run `/lmc/make_lyrics_emotion_dataset.py` to perform data preprocessing and upload the dataset to HF. Preprocess steps are as follows:

   (a) Remove exact duplicates and the lyrics that differs only by punctuation but is essentially the same song using cosine similarity

[6] https://vuejs.org

(b) Remove non english songs according to the lyrics

(c) Remove all punctuations for lyrics

(d) Get song metadata, split the dataset into train/dev/test and upload to HF

After these two steps, there are 37473 annotated samples by GPT and 20229 unique English songs after preprocessing.

4.3.3 Training classifier

- We tested multiple permutations of lyrics (text) preprocessing techniques paired with classification models
- Text preprocessing: Used Stanford's Stanza model to remove stop words and select keywords with word properties and named entity recognition Stanza [6]. We extracted specific entities such as names, geographical location, Countries and place, events, facilities, and products. we also lemmatize words, and filter part of speech such as noun, adjective, and verb. In short, a word in a song lyrics is only tokenized if it matches one of the extracted named entities, or has a POS tag indicating meaningful content, specifically nouns, verbs, adjectives, adverbs, interjections, or proper nouns.
- Lyrics vector creation:
  - TF-IDF: Represents lyrics as sparse vectors where each term is weighted by its frequency in a song and its rarity across the corpus. Effective for capturing important keywords but ignores word order and context.
  - Word2Vec: Learns dense word embeddings based on context windows using Skip-gram or CBOW. Captures semantic similarity between words, allowing generalization, but aggregates word vectors without sentence structure.
  - Doc2Vec: Extends Word2Vec by learning a unique vector for each document. This method can preserve some global semantic information about the entire text.
  - SBERT (Sentence Transformer): A sentence-level embedding model that fine-tunes BERT with a Siamese architecture. Provides dense representations of full sentences or documents, ideal for semantic tasks like mood classification [7].
  - LDA Topic Vector: A generative topic model that represents each document as a mixture of latent topics. Useful for capturing abstract themes in lyrics, though not optimized for short texts.
- Introduced emotion lexicons to capture emotion vectors:
  - NRC: Maps words to 8 basic emotions (joy, anger, fear, etc.). Helps introduce interpretable emotional features [4].
  - VADER lexicon + valence score: A lexicon-based tool tuned for sentiment analysis, especially on social media. Each word has a valence (positivity/negativity) score, supporting affective feature extraction [2].
- Models:
  - Logistic Regression: A linear model for binary or multiclass classification. Interpretable and effective with dense or sparse vectorized text data.

- Naive Bayes: A simple probabilistic classifier assuming feature independence. Fast and works well with high-dimensional text data like TF-IDF.
- Random Forest: An ensemble of decision trees that improves robustness and reduces overfitting. Handles non-linear relationships and categorical features.
- Support Vector Machine (RBF Kernel, Linear Kernel): A strong linear/non-linear classifier that finds the optimal separating hyperplane. Both RBF and Linear kernels tested for best performance.
- Voting stacked model: (SVM + Logistic Regression + Random Forest): An ensemble model that combines predictions from multiple classifiers to improve robustness and generalization.

Feature	Classifier	Accuracy	Precision	Recall	F1
sbert	svm	65.6	64.5	65.6	64.4
sbert	logreg	65.2	64.1	65.2	64.0
sbert	svm_linear	65.1	64.1	65.1	64.0
sbert	voting	64.9	63.7	64.9	63.7
sbert	xgb	62.6	61.1	62.6	61.0
sbert	lgbm	62.2	60.8	62.2	60.6
tfidf	voting	60.2	58.9	60.2	57.6
word2vec	svm_linear	58.6	57.1	58.6	56.6
tfidf	logreg	59.4	56.2	59.4	56.2
word2vec	logreg	57.6	56.5	57.6	55.8
word2vec	lgbm	57.6	56.2	57.6	55.6
word2vec	voting	58.3	57.4	58.3	55.5
tfidf	svm_linear	58.1	55.9	58.1	55.5
doc2vec	logreg	56.4	55.5	56.4	55.4
doc2vec	voting	57.5	55.5	57.5	55.4
doc2vec	svm_linear	56.5	55.3	56.5	55.3
tfidf	lgbm	57.4	55.9	57.4	55.1
word2vec	xgb	56.1	54.7	56.1	53.9
doc2vec	lgbm	55.9	53.4	55.9	53.5
doc2vec	xgb	55.2	52.8	55.2	52.7
tfidf	xgb	55.5	53.9	55.5	52.7
sbert	rf	57.3	54.8	57.3	51.5
word2vec	rf	55.0	54.8	55.0	50.0
doc2vec	svm	53.8	47.5	53.8	47.4
tfidf	rf	53.0	58.1	53.0	47.2
sbert	nb	44.3	55.0	44.3	47.1
tfidf	svm	53.8	52.3	53.8	46.8
doc2vec	rf	52.2	50.0	52.2	46.5
word2vec	svm	48.4	38.5	48.4	42.0
word2vec	nb	36.9	51.6	36.9	41.4
doc2vec	nb	37.7	48.8	37.7	39.3
tfidf	nb	32.4	35.7	32.4	32.9

Table 2: Quantitative evaluation for lyric mood prediction, sorted by F1 score in descending order.

4.3.4 Quantitative Evaluation

Table 2 presents the results of the lyrics mood classification task. The best performance was achieved using SBERT-based sentence embeddings in combination with a support vector machine (SVM) classifier. Mood classification often depends on capturing the holistic meaning of entire sentences, making SBERT a more effective feature extractor than traditional term vector representations, which may overlook semantic context.

5 Samples and screenshots

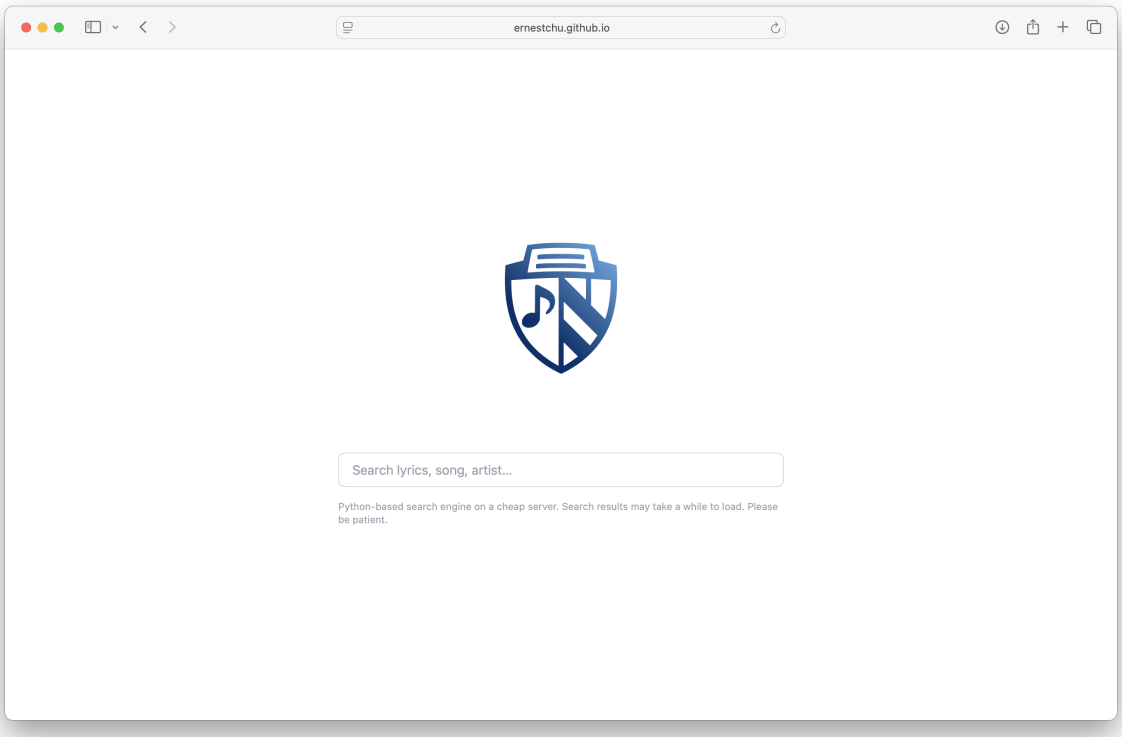

Figure 1: Welcome page for our lyric search.

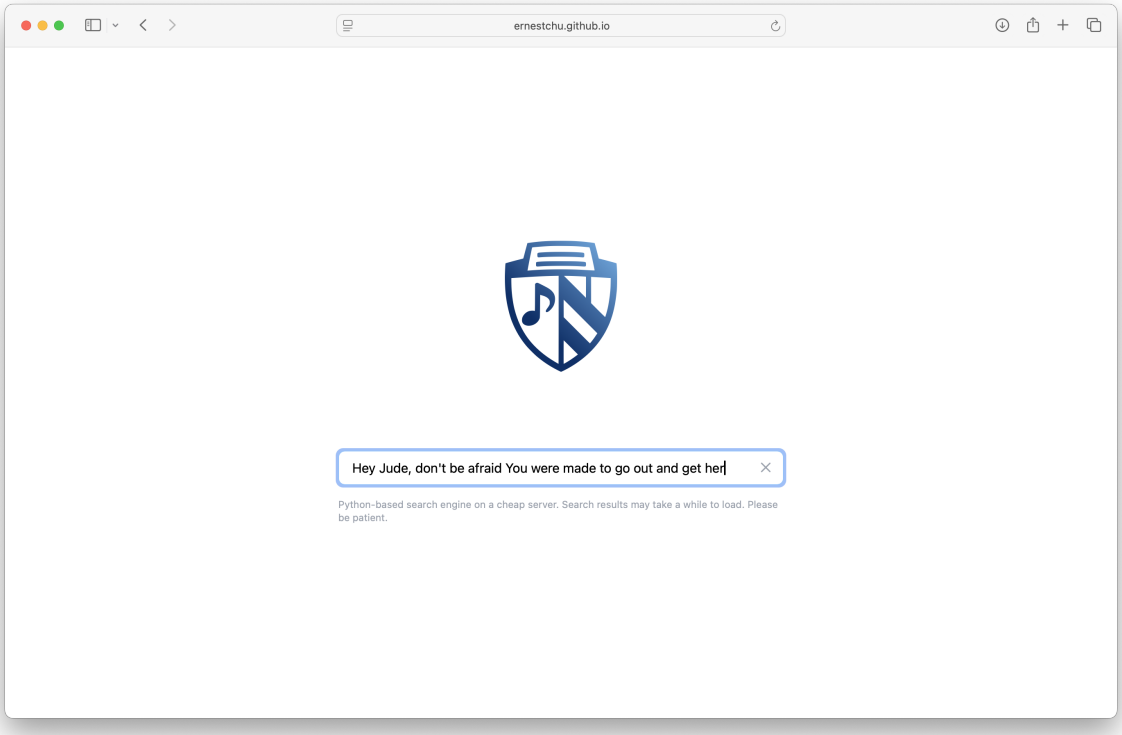

Figure 2: Entering search query.

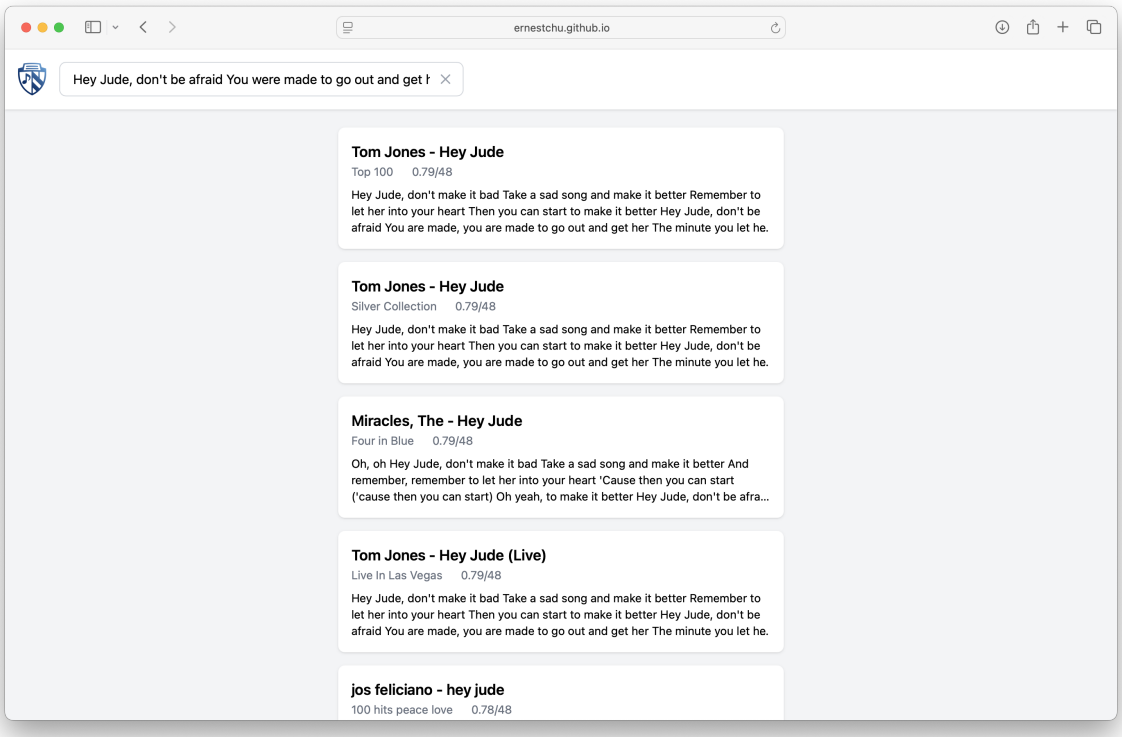

Figure 3: Top search results.

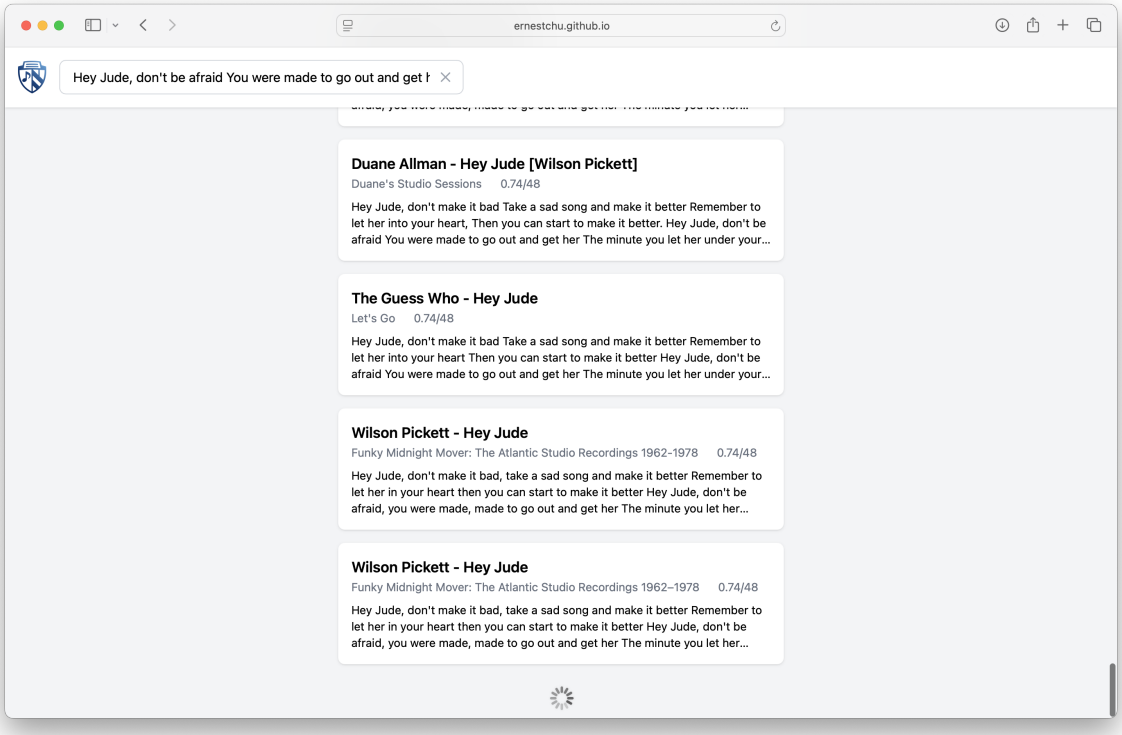

Figure 4: On-demand results loading when scrolling to the bottom.

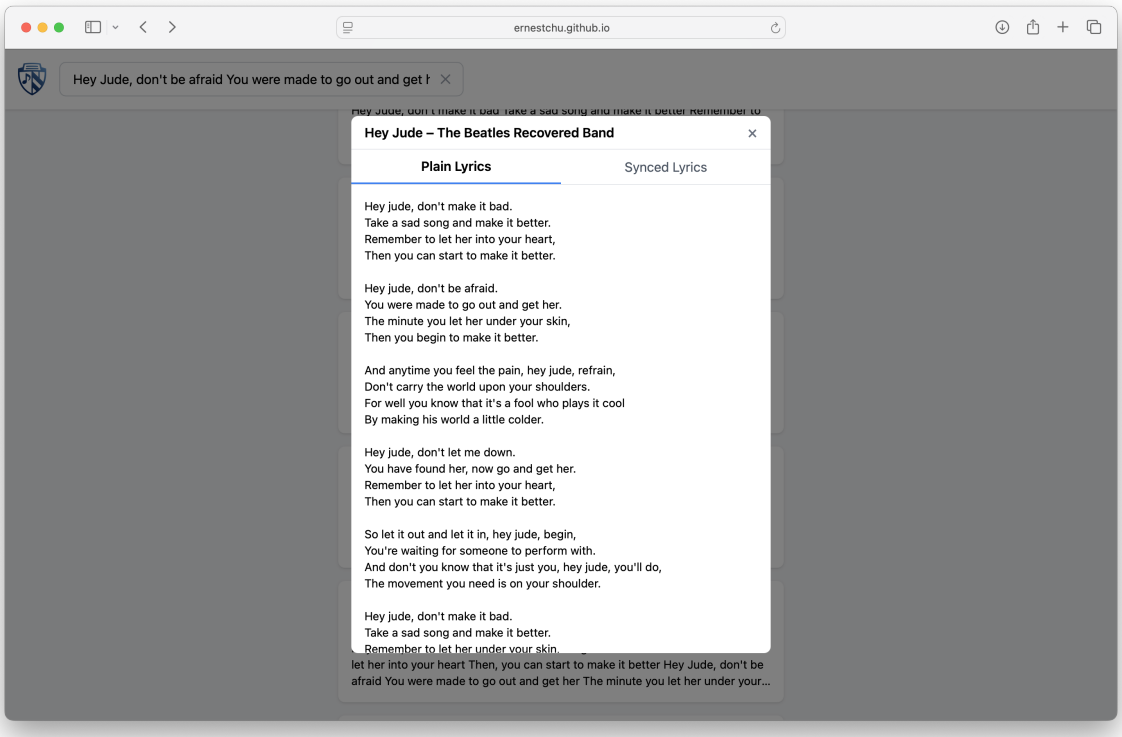

Figure 5: Lyrics viewing.

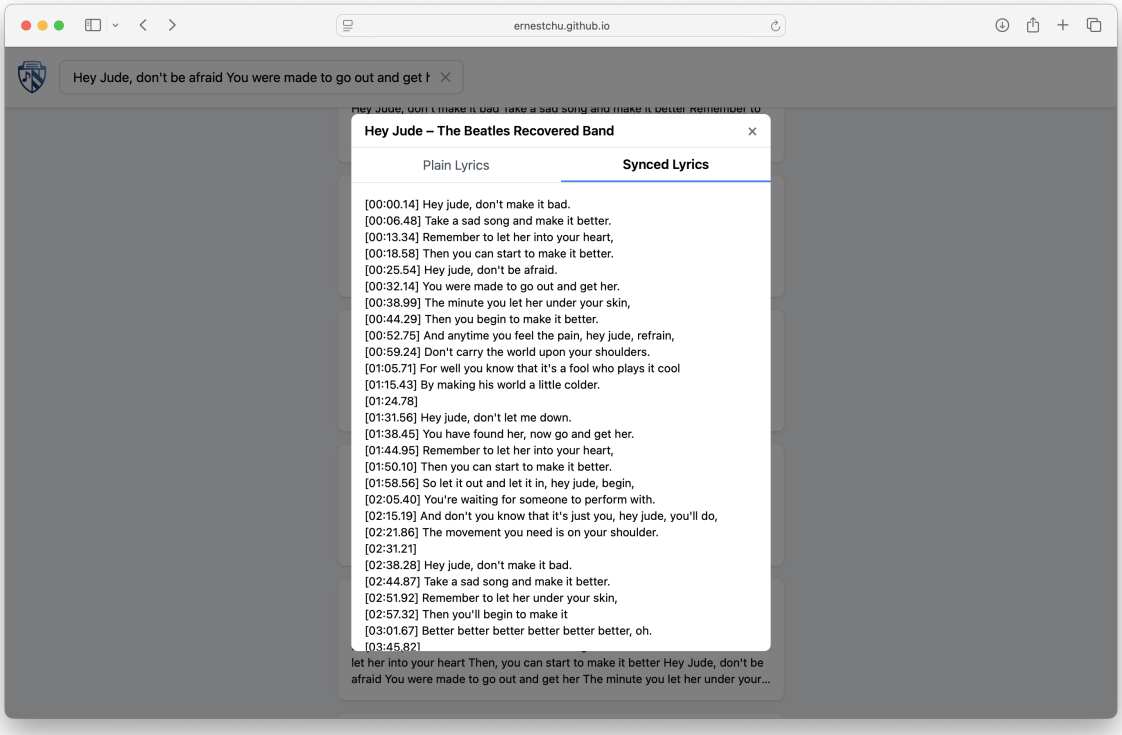

Figure 6: Lyrics viewing with timestamps.

References

[1] Romain Delbouys, Romain Hennequin, Mathieu Piccoli, Juan Royo-Letelier, and Manuel Moussallam. Music mood detection based on audio and lyrics with deep neural net. In *ISMIR*, pages 756–763, 2018.

[2] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225. AAAI, 2014.

[3] Cyril Laurier, Olivier Lartillot, Tuomas Eerola, and Markus Schedl. Exploring the limits of the classical approach to music emotion recognition from lyrics. In *ISMIR*, pages 345–350, 2008.

[4] Saif M. Mohammad and Peter D. Turney. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465, 2013.

[5] Loreto Parisi, Simone Francia, Silvio Olivastri, and Maria Stella Tavella. Exploiting synchronized lyrics and vocal features for music emotion detection. *arXiv preprint arXiv:1901.04831*, 2019.

[6] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108. Association for Computational Linguistics, 2020.

[7] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

[8] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, 45(4):1191–1207, 2013.