# Using Language Models to Increase Accessibility of Japanese Vocaloid Songs

**Student**: Alyssa Traylor

**Class:** Machine Translation

## 1. Introduction

Music connects people across cultural and language barriers. With the increasing accessibility of international music through streaming platforms and social media, more people become enamored with the music from different nations. As an American, there is a specific genre of music from overseas that I think found an interesting way into Western culture: Vocaloid. "Vocaloid" is a term used to refer to both a type of voice synthesizer software and the genre of music describing songs made with the software. Here is a brief explanation on the software (Traylor 2024):

> *"The software is created by gathering vocal samples from a volunteer and storing these samples into a voicebank. A user inputs words and melody into the software, which is used to tweak the voicebank samples in order to synthesize singing."*

Vocaloid – originally developed in 2004 in Japan – found its way into western countries in the late 2000s, and has since continued to grow, leading to world tours featuring the virtual singers. From this description, one may be curious what songs are featured on these tours. Vocaloid songs are created by independent producers and posted online; there is no specific artist or band who makes all Vocaloid songs. Furthermore, these artists don't have any obligation to translate their songs into any given language in order to boost sales like traditional artists do. How do foreign audiences get into Vocaloid?

The answer to both of these questions is that it depends on the fans. Crypton Future Media asks permission from the most popular artists to use their songs on tours, or holds public competitions to find new songs to include. Fans of songs, or unofficial translators that work with the artists, create the English (or other language) translations of the lyrics to share with a wider audience. The community thrives off fan involvement in order to keep going.

Even with the help of fans, there's a significant number of songs that never receive an English translation, making it harder for overseas fans to discover similar songs they may like, or for translators to discover new projects to work on. My goal is to create a program that increases the ability for Vocaloid listeners to discover new songs, using a lyrics-based recommendation system. I try basing recommendations on, separately, the original Japanese lyrics, and then the English translations (for songs that have them).

## 2. Literature Review

Due to time constraints, I wasn't able to finish my write-up of the literature review, but I consulted Ko et.al (2022)[1] and Datta[2] for my work.

## 3. Methods

In order to increase the discoverability of Vocaloid songs with no translation, I decided upon a lyrics-based recommendation system. The ideal user experience is simple: the listener provides the name of a song that they enjoy, and the system supplies a list of semantically similar songs to peruse, along with sample translations.

[1] https://doi.org/10.3390/electronics11010141
[2] https://medium.com/data-science/lyrec-a-song-recommender-that-reads-between-the-lyrics-eca5ea2ae8c8

## 3.1 Data Processing

Of course, a song corpus is necessary to generate recommendations. I use the same corpus as in my previous work (Traylor 2024), which was collected from vocadb.net, an unofficially maintained catalogue for Vocaloid songs. The returned JSONs from VocaDB contain songs organized by ID, containing various metadata such as the song name, artist, and popularity rating. However, not all songs have transcribed lyrics in the database, and not all songs are Japanese-original. Since the purpose of this project is to increase accessibility for those who can't understand Japanese – and because the size of the data would be unfeasibly large otherwise – I only want to examine songs that are originally Japanese.

So, the data needs to be pruned to include only entries with lyrics. This process was completed in my original paper with an estimated 80,000 valid songs to analyze. However, upon re-inspection, the JSON data was not cleaned properly. Formatting errors and a few bugs meant that several of the entries with no lyrics were retained in the "clean" dataset. I refined the dataset using Python in my local computing environment, then produced spreadsheets with specific information for each song. The retained data is song ID; song title; artist name; publish date; user rating score; Japanese lyrics; and the most recent English translation if one exists. Figure 1 shows a preview of the data from 2005 to 2009. Across over 300,000 songs examined from 2005 to the end of 2023, the actual count of valid songs for this experiment is 66,539. Out of those songs, 7,791 of them had at least one English translation.

## 3.2 Creating Embeddings

Next, I needed a way to capture the semantic quality of the lyrics. The simplest way to do this is to create text embeddings for each set of lyrics. While it may be sufficient to use the embeddings for Japanese lyrics, I wanted to see if the results differed when the embeddings of English translations were used for suggestions as well. Thus I stored the embeddings for both, when possible.

I looked for a simple transformer model which had good performance on both English and Japanese texts. After browsing the HuggingFace embeddings leaderboard[3], I decided on the multilingual-e5-large-instruct[4] model. The model size is 500M parameters, and the embedding size is 1024; it truncates text to at most 512 tokens, although it is unlikely for any entry in the data to surpass this limit. The encoding process was straightforward, as I just used the SentenceTransformer class from HuggingFace to encode each set of lyrics. When training on GPU, the English lyrics took around 9 minutes to encode, while the Japanese lyrics took around 70 minutes.

## 3.3 Searching for Similar Embeddings

Finally, after creating the embeddings, I needed an efficient way to search for similar ones. After looking online, I was directed towards Facebook AI Similarity Search (FAISS)[5], which provides an efficient vector search algorithm given a matrix of embedding vectors. The process involves specifying the similarity metric to compare embeddings with, then creating an index with the embedding vectors. Our dataset is very large, so I went with a variant of the k-means algorithm, the IndexIVFFlat. It performs an approximate nearest neighbor search, which is sufficient for this project.

I also needed to ensure that the program could retrieve the correct embedding based on the query. This part wasn't too difficult; the entries are organized by row number in the song

[3] https://huggingface.co/spaces/mteb/leaderboard?source=post_page-----eca5ea2ae8c8
[4] https://huggingface.co/intfloat/multilingual-e5-large-instruct
[5] https://github.com/facebookresearch/faiss

dataframe, so after finding the row with the correct song title, I used the embedding at the same row in the final matrix and used it as the query to the search.

## 4. Experiments and Results

Due to time constraints, I wasn't able to finish my write-up of the experiments performed to check the similarity; you can perform further testing in the GitHub supplied with this paper.

## 5. Conclusion

In this project, I presented a lyrics-based recommendation system for Japanese-original Vocaloid songs. Experiments with embeddings based on translation and on the original lyrics produced similar results, but the small size of the former dataset indicates the need for more translation work.

One possible way to ameliorate this issue is to encourage more translators to work on Vocaloid songs, which I hoped to encourage with this project. Another way is to improve machine translation specifically for songs. It's often the case that song translations do not translate lines one-to-one. In other words, line number 5 in a translation often does not need a direct translation of line number 5 in the original song, and is instead a rearrangement of the lines or a reworded interpretation. Machine translation for songs should look holistically at chunks of lyrics to determine the best way to maximize both fluency and accuracy.

In future extensions to this project, I would like to create a visual interface for searching for songs. Right now, the user has to directly type in the exact name of the song that they're using to search, and can only do this through testing on segments of code. Ideally, a user would be able to begin their search using the name of the song, use an autocomplete to select the song they’re looking for, and have the relevant information presented in a neat visual format. Unfortunately due to other obligations I was not able to implement this within the timeframe of this project, but I find this to be the most pertinent extension.

It is also worth investigating whether integrating other information into the embeddings would result in more accurate results. For example, instead of only creating embeddings on lyrics, the artist name and publish date are also included in the training so that works by the same artist are more closely clustered together.

## Appendix

### Honor Code

*This paper represents my own work in accordance with university regulations.*

- Alyssa Traylor