Cross-Lingual Singing Voice to Singing Voice Translation

Automatic Translation of Lyrics From A Real Singing Voice to A Synthetic Singing Voice

Silas Antonisen: santon18@student.aau.dk

Project Report
University of Aalborg
Department of Electronic Systems

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems
Frederk Bajers Vej 7
9220 Aalborg Øst, DK
https://www.es.aau.dk/

Project Title:
Cross-Lingual Singing Voice to Singing Voice Translation

Project Theme:
Singing Voice Synthesis, Music Information Retrieval

Project Period:
Aug 2023 - Jan 2024

Author:
Silas Antonisen

Supervisor:
Ivan Lopez Espejo

Page number: 36

Completed: 04-01-2024

Abstract

While cross-lingual synthesis has emerged in Singing Voice Synthesis (SVS), its usage is limited due to language barriers in songwriting. To ease the language barrier, this work presents an attempt at English to Japanese lyrics translation by fine-tuning. This is done jointly with an overarching cascaded solution for singing voice to singing voice translation. The task at hand is to translate an English singing voice into a synthesized Japanese voice with lyrics and aligned melody. Such a system is successfully created using state of the art methods, but evaluations show that the translation model creates poorly singable lyrics. Moreover, a deeper understanding of the Japanese language is needed to create accurate yet singable singing voice to singing voice translations.

Table of Contents

1 Introduction **1**

2 State of The Art in Singing Voice Synthesis and Information Retrieval **2**

2.1 Singing Voice Synthesis 3

2.2 Automatic Lyrics Transcription 5

2.3 Phoneme Level Lyrics Alignment 7

2.4 Vocal Melody Extraction 8

2.5 Lyrics Translation 9

3 Implementation **11**

3.1 Singing Voice in English 12

3.2 Lyrics Transcription 12

3.2.1 Transformers 13

3.3 Phoneme Alignment 15

3.4 Syllable Concatenation 17

3.5 Automatic Lyrics Translation 19

3.6 Conversion of Kanji to Hiragana 22

3.7 Vocal Melody Extraction 23

3.8 Synthesized Singing in Japanese 24

4 Test and Results **26**

5 Conclusion **29**

6 Future Work **30**

Bibliography **32**

1 | Introduction

Singing Voice Synthesis (SVS) has in recent years become capable of extremely human like expression in both the commercial [1] [2] and open source [Liu et al., 2022] [Yamamoto et al., 2022] field. With the capabilities of AI voices to sing songs in languages they have never been trained on, the musical expression in which they can operate manifolds.

However, while the AI voice is capable of synthesizing songs in foreign languages, the musician behind the facade is still limited to writing songs in his native language or making covers of foreign songs.

With rapid development in multi-lingual machine translation [Tang et al., 2020] [Team et al., 2022] and the mainstreaming of Large Language Models (LLMs) [Radford et al., 2018] [Touvron et al., 2023], the question arises to whether such technology can be utilized for assisting musicians in writing cross-lingual songs.

While AI voices in SVS systems are becoming better at automatic tuning, i.e. creating smooth and musical intonations and transitions between notes, one of the most labour intensive parts about SVS it the creation of human like or otherwise beautifully musical tuning. Letting an SVS musician use his own voice for note creation and tuning will enable an alternative workflow than manual editing in the editor.

Therefore this work presents a joint approach at cross-lingual lyrics creation and automatic creation of notes along with their tuning, based on the musicians own voice.

Since Japanese, English and Chinese by far are the most popular languages in the SVS music scene, they will be the the languages in focus. Since there has been prior examples at English to Chinese lyrics translation [Guo et al., 2022] [Li et al., 2023], this work will focus on English to Japanese.

The structure of this report will in chapter 2 analyze the components required for such a system. Chapter 3 goes into how the system is created. chapter 4 conducts a small scale mean opinion score test with Japanese natives. Chapter 5 concludes the work and chapter 6 goes in depth with shortcomings and potential work to fix them.

[1] `https://dreamtonics.com/synthesizerv/`
[2] `https://www.vocaloid.com/en/`

2 | State of The Art in Singing Voice Synthesis and Information Retrieval

Possibly the most related field of research to this work, is S2ST (speech-to-speech translation). At the current state of S2ST, there are two approaches to the problem. The conventional way of handling the problem is with a cascaded system consisting of a STT (Speech-To-Text) module, a MT (Machine-Translation) module and a TTS (Speech-To-Text) module [International Telecommunication Union, 2016]. Alternatively, there has been rapid progress in end-to-end solutions for direct S2ST using sequence-to-sequence models [Jia et al., 2019] and LLMs (Large Language Models) [Rubenstein et al., 2023]. The use of LLMs for speech understanding and audio generation show promising results in both translation and voice synthesis with capabilities of preserving features such as speaker identity and intonation. However, while such system are becoming great competitors to cascaded system and will possibly soon overtake, they are very scarce in their public availability due to end-to-end S2ST systems being a technology in very early stages of development. as of August 2023, Meta AI publicly released SeamlessM4T [Seamless Communication et al., 2023] on GitHub [1]

S2ST systems offer much less control than a cascaded system, and at the current time it is unknown how well they will work for applications outside general speech generation, i.e. high quality singing-voice translation as intended with this project. Direct S2ST systems are highly sophisticated systems in early development by large research teams such as Google and Meta AI, so while creating an end-to-end solution for direct sing-to-sing translation could be an end-goal to pursue, it has been deemed beyond the scope of this work.

Therefore, the focus will be to create a cascaded system for sing-to-sing translation. By doing so, there will be a higher level of control over the entire pipeline and there will be more focus on researching what capabilities a system must have to generate high quality sing-to-sing translation along with what limitations and intricacies there might be.

In the remainder of this chapter, the modules required for a cascaded sing-to-sing solution are explored in depth.

[1]`https://github.com/facebookresearch/seamless_communication`

2.1 Singing Voice Synthesis

One of the first pioneers to commercialize and mainstream singing synthesis was Yamaha Corporation with their engine Vocaloid [Kenmochi and Ohshita, 2007]. Vocaloid is comprised of three parts; a score editor, a singer library (often referred to as a voicebank) and a synthesis engine. The score editor is an environment in which musical notes with an onset, duration and lyrics can be created along with capabilities for automating features such as note transitions and vibrato. The lyrics are automatically broken down into syllables and phonemes by a dictionary. The voice bank is a library of samples recorded from a real human singer, mostly as diphones. All phonetic combinations must be covered for all pitches in the desired range, which is reported to be approximately 2000 samples per pitch. These voicebanks are not created by Yamaha, but by third party companies who are licensed to do so. The synthesis engine uses the score made in the editor to select the necessary samples from the voicebank, and automatically concatenates them.

In 2008 UTAU [2] came to rise as a free competitor to Vocaloid where users could make and share their own custom made voicebanks. Since 2013 UTAU stopped getting updates, so OpenUTAU [3] emerged as an unofficial open source UTAU successor.

With much development in deep learning, AI voicebanks/engines became the next generation of singing voice synthesis. Usually, they consist of an acoustic model to generate acoustic features (e.g. mel-spectrograms) from a musical score (i.e notes and lyrics) and a vocoder to transform acoustic features into wave forms [Nakamura et al., 2019][Hono et al., 2021]. Diffsinger [Liu et al., 2022] shows remarkable results using shallow diffusion for acoustic modeling, and is an open source solution for users to create AI voicebanks [4]. NNSVS [Yamamoto et al., 2022] embraces the cascaded nature of SVS pipelines, and creates an open source toolkit where users are free to integrate their own acoustic models, vocoders, time lag models and duration models for lyrics to phoneme alignment. The baseline system whom they themselves propose reaches the current state of the art with a mean opinion score of 3.86 out of 5. NNSVS is targeted at research, so a plugin called ENUNU [5] enables the usage of NNSVS voicebanks in the UTAU editor for music recreation. Here is an example of an original musical score written in the UTAU editor using a high quality voicebank created in NNSVS with several singing styles (standard/sweet/rock) [6].

in 2018 a new commercial singing voice synthesis engine called Synthesizer V [7] emerges as the next popular alternative to Vocaloid. Synthesizer V started out by releasing concatenated voicebanks in the Vocaloid style, but introduced AI voicebanks in 2020 which are by themselves and licensed third party companies. In 2021 cross-lingual synthesis was enabled for AI voicebanks which has since expanded to support Japanese, English, Chinese, Cantonese and Spanish. Cross-lingual synthesis enables an AI voicebank to synthesize singing in a language which it has not been trained, i.e. a voicebank trained on Japanese data can synthesize English singing.

[2]http://utau2008.xrea.jp/
[3]https://www.openutau.com/
[4]https://github.com/MoonInTheRiver/DiffSinger
[5]https://github.com/oatsu-gh/ENUNU
[6]https://www.youtube.com/watch?v=1V41mghsxIU
[7]https://dreamtonics.com/synthesizerv/

Synthesizer V has since become famous in the SVS music production community for their high quality AI voicebanks. Here is an example of a song created with a synthesizer V AI voicebank [8]

The latest release of Vocaloid [9] in 2022 also adds AI voicebank support, however Synthesizer V is more renowned for their AI voicebanks, while Vocaloid remains the most popular choice by far for concatenated voicebanks. While The concatenated approach to SVS is far from as natural and human-like as AI voicebanks, it is an artistic choice of creating beautiful synthesized voiced instead of replicating the human voice. Here is an example of a popular song made with Vocaloid [10]

Synthesizer V supports scripting in their editor to programmatically create and automate musical scores as well as control any parameter in the engine. Vocaloid does not support scripting, neither does UTAU/OpenUTAU natively. Considering scripting capabilities, the AI voicebank quality, and a well made editor, Synthesizer V makes for a good development environment in this sing-to-sing work. Synthesizer V will therefore be the choice for SVS in this work. Vocaloid might serve as a cumbersome engine without a programmable interface, as it will solely rely on external files (e.g generated MIDI files) which can be dragged and dropped in the editor or manual editing. NNSVS might however be a very good candidate for future works as it is free and open source. NNSVS will allow for full control of the synthesis engine and custom made voicebanks. NNSVS can be used with recipes closely related to Kaldi recipes [11]. Musical scores can be in MusicXML format or UST (UTAU Sequence Text File), but UTAU via ENUNU itself will not provide a programmable interface to automatically generate UST files.

[8]https://www.youtube.com/watch?v=OMKomCQ_KSA
[9]https://www.vocaloid.com/en/vocaloid6/
[10]https://www.youtube.com/watch?v=e1xCOsgWGOM
[11]https://nnsvs.github.io/recipes.html

2.2 Automatic Lyrics Transcription

Traditionally, the different works regarding ASR (Automatic Speech Recognition) has often revolved around acoustic modelling with CTC (Connectionist Temporal Classification), HMM (hidden markov model), and language modeling. State Of the Art solutions in ASR are maintained in the Kaldi ASR Toolkit [12]. However, the singing domain encompasses different challenges than the speech domain, as phoneme recognition and keyword spotting often are unsatisfactory when using ASR models due to altered properties such as alternate pronunciation, duration and vibrato [Kruspe, 2016]

While ALT (Automatic Lyrics Transcription) has not gotten the same level of attention as ASR, a few works have been published the last couple of years. Naturally, ALT has been merged with Kaldi based approaches such as [Demirel et al., 2020] using dilated convolutional neural networks for monophonic audio recordings, i.e. recordings of isolated singing with no background music. This approach reports a WER (Word Error Rate) of 14.96% on the Dsing Test dataset [Dabike and Barker, 2019]. Sing! 300x30x2 is a part of the DAMP database [13], and consist of the 300 most popular songs sung by both males and females in the 30 most famous countries in the Smule karaoke app [14]. In order to make Sing! 300x30x2 into an ASR oriented dataset, the dataset was cleaned, curated and temporally realigned with the lyrics, resulting in the Dsing ASR dataset [Dabike and Barker, 2019]

Transcribing lyrics from polyphonic music is a more complicated task to solve, with a proposed solution being MSTR-net [Demirel et al., 2021] which is a variant of a Multistreaming Time-Delay Neural Network. While The Dsing dataset is the typical benchmark for monophonic singing, DALI [Meseguer-Brocal et al., 2018] is the leading dataset for polyphonic music with aligned lyrics. DALI consist of 5358 songs collected from YouTube with automatically created and synchronized lyrics and notes at four levels of granularity. MSTRE-Net utilizes both Dsing and DALI for robust training. MSTRE-Net did not surpass the previous solution using dilated convolutional networks as it obtained a WER of 15.38% on Dsing Test, but it did perform much better than previous attempts at DALI Test. However, while the results on DALI Test are comparatively impressive, it got a WER of 42.11% which is not suitable for a robust sing-to-sing solution.

As transformers are becoming more and more popular especially for LLMs (Large Language Models) [Devlin et al., 2018] [Radford et al., 2018] as well as other fields such as computer vision [Dosovitskiy et al., 2020], Wav2Vec [Baevski et al., 2020] has also become a popular choice for ASR. One of the major advantages of such transformer models is that they don't solely rely on supervised training, but can expand to a larger domain of non-labeled data for self-supervised training. Transformers are also able to train faster than RNNs and LSTMs as there are no recurrent elements. With faster training and non-labeled data combined with the scalability allowed my modern hardware, transformers have become the State of The Art within most fields of NLP (Natural Language Processing).

[12]`https://github.com/kaldi-asr/kaldi`
[13]`https://ccrma.stanford.edu/damp/`
[14]`https://www.smule.com/`

One of the difficulties of training an ALT system is the scarcity of high quality labeled data for supervised training. Therefore, the pretrained Wav2Vec model was attempted for ALT by transfer learning to the singing domain by first fine tuning on speech data with CTC loss and then fine tuned on Dsing and DALI with a hybrid CTC/Attention ALT head. while this approach is aimed at monophonic singing, advances in source separation [Défossez, 2022] allow for isolated singing from polyphonic music. Transfer learning of Wav2Vec reaches a WER of 12.99% on Dsing Test and 30.85% on DALI Test with source separation, setting the state of the art for both test sets. However, we still see a WER above 30% which is not suitable for implementation in a cascaded sing-to-sing solution, suggesting such a solution might only work with monophonic voice recording.

As of 2023, Whisper [Radford et al., 2022] is the current state of the art ASR model from OpenAI which was released the same year as the previously mentioned transfer learned Wav2Vec for ALT. With the update to Whisper-Large-V3 in November in 2023, it is the most popular ASR model on HuggingFace Model Hub in December 2023 [15]. While there are no empirical studies on Whisper's capabilities in the singing domain such as a WER for Dsing and DALI, personal tests show promising results that suggest a lower WER than previously discussed ALT models. Whisper surpasses all previous ASR in amount of training data with 680K hours of non-standardized labeled speech used for large-scale weak supervision, which they claim produces a robust model across datasets and domains. While personal test suggest that it generalized better for the ALT domain than previous ALT models, an empirical study must be made to state any claim. This will be left for future work, along with the possibility of fine tuning Whisper for the speech domain, as further discussed in chapter 6. As Whisper seems to produce high quality transcriptions from singing, and it is very simple to implement and control through the HugginFace API, it will be the current choice for ALT.

[15] `https://huggingface.co/openai/whisper-large-v3`

2.3 Phoneme Level Lyrics Alignment

Alongside the lyrics, alignment is needed to get the length of words, or more specifically the length of phonemes are needed to obtain the rhythm and pronunciation during singing.

Classic ASR approaches such as the ones presented in Kaldi usually optimze w.r.t phoneme posteriors. To extract the phoneme alignments, audio frames get classified into phonemes, as can be seen in MSTRE-net [Demirel et al., 2021] by training a GMM(Gaussian Mixture Model)-HMM model on "singer adaptive features" [Anastasakos et al., 1996] and applying forced alignment [Gales et al., 2008].

While Whisper generates high quality transcriptions, it is an end-to-end solutions which operates by predicting BPE (Byte-Pair Encoding) tokens which usually are either complete words or a set of graphemes. Word level timestamps are possible with Whisper either through the HuggingFace API or external solutions building upon the functionality of Whisper [Bain et al., 2023]. Hence, to generate phoneme level alignments for Whisper's transcriptions, it is necessary to use a solutions dislocated from Whisper.

Assuming high quality lyrics are generated by Whisper, arguably the best solution for the current problem is text-informed phoneme level lyrics alignment [Schulze-Forster et al., 2021]. The text-informed aligner shows competitive results for word-level alignment compared to the state of the art [Gupta et al., 2019], but will not serve as the best solution for that purpose. While MSTRE-net does not share a measure on the error rate of its phoneme alignments, the Montreal Forced Aligner [McAuliffe et al., 2017] is used to train a Kaldi based baseline GMM-HMM model to test up against the text-informed aligner, which share similarities with MSTRE-net's approach. For solo singing, the text-informed aligner achieved a PCAS (Percentage of Correctly Aligned Segments) score of 85.94% on the NUS-48E CORPUS [Duan et al., 2013] compared to the baseline at 77.94%. However, no study has been done on the performance of a lyrics aligner which has also trained on "singer adaptive feature" compared to text informed alignemt. While it would be interesting future work to conduct such a test, the given state of the art results for phoneme-level lyrics alignment and the ease of implementing the model [16], makes it the current choice for the sing-to-sing solution.

[16]`https://github.com/schufo/lyrics-aligner`

2.4 Vocal Melody Extraction

SVT (Singing Voice Transcription) is the task of extracting note events from a singing voice. SVT can commonly be broken into sub-tasks such as pitch detection, onset detection, offset detection and sequence-level modelling, but by far the most researched sub-task of SVT is frame level vocal melody extraction. SVT is not a very well defined problem and it is thus extremely difficult to create a humanly annotated dataset. VOCANO [Hsu and Su, 2021] tries to make a clearer definition of SVT and leverages unlabeled data through semi-supervised learning. As the state of the art in SVT, VOCANO has been integrated in Omnizart [Wu et al., 2021], a toolkit for music transcription.

While there for certain is progress in the SVT field, what exactly the boundaries of a note is, is not very accurately defined, and thus their transcription is inconsistent with what an implementation such as this one might need. instead, by using frame level pitches along with phoneme alignment, the onset and duration of a phoneme will define a note.

Vocal melody extraction itself is however a more well defined task. Praat [17] is a popular toolkit for all sorts of speech analysis. For vocal melody extraction, their standard method is by acoustic periodicity detection [Boersma, 2000] and might work well in many implementations. However, some manual labor is needed to reproduce good results, such as setting variables to match the vocal range and intensity. Jitter is also sometimes a problem alongside the handling of background noise.

For better recognition of the desired signal, polyphonic vocal melody extraction has similarly been approached by carefully engineered approaches to the notion of salience by audio prepossessing, salience function computation, fundamental frequency tracking and voicing decisions [Dressler, 2011] [Salamon and Gomez, 2012]. As described in [Bittner et al.], These approaches are heuristic in their nature and are limited by the data they weer designed for, so they approach the problem with a fully data driven classification problem which outputs the frame-level likelihood for each pitch.

Neural networks have been applied to vocal melody extraction with great success, e.g using CNNs for semantic segmentation [Lu and Su, 2018] [Chen et al., 2019], semi-supervised training [Kum et al., 2020] and graph modelling [Gra, 2023].

[Lu and Su, 2018] has been implemented in omnizart [Wu et al., 2021] to seamlessly extract vocal melodies in MIDI pitches. Considering how well developed vocal melody extraction is, this will serve as an excellent candidate and will therefore be used in this work for a sing-to-sing solution. [Chen et al., 2019] is open source and will most likely also be an excellent solution. These two methods have not been directly compared, but both show state of the art results on several datasets.

[17] `https://www.fon.hum.uva.nl/praat/`

2.5 Lyrics Translation

lyrics translation is a more involved task than direct translation. A more poetic understanding of the language and culture at both the source and target side is needed to make quality translations. This makes word choice and placement within the sentence more difficult, as the translation also need to fit a rhythm and melody. Apart from semantics and melody alignment, rhyme is a big part of poetry and song lyrics, along with a sense of "musical color" which allows for an artist to bring life to the song by allowing compromises and creative recreation [Franzon, 2015]

While neural networks have been greatly used for general machine translation [Bahdanau et al., 2014] [Vaswani et al., 2017], lyrics translation is still in much need for research. The first attempt at neural poetry translation [Ghazvininejad et al., 2018] achieved acceptable translations 78.2% of the time, based on human evaluations. Lyric translations between western languages can often be viewed as poetry translation, but it is not so much the case when e.g. translating to a tonal language such as Chinese.

GagaST (Guided AliGnment for Automatic Song Translation) [Guo et al., 2022] attempts to solve song translation as constrained text translation, by constrained beam search. A problem in lyrics translations is the very low accessibility to paired translated songs. Therefore, GataST adopts self supervised learning on mono lingual songs in English and Chinese after first training a transformer for general translation, and finally fine tuning on a small dataset of paired English-Chinese song translations [18]. These paired translations are not necessarily singable. The decoder is then constrained according to three properties, length alignemt, intra-syllable alignment and inter-syllable alignment. The length constraint is to make sure that the translation either has exactly the same amount of syllables as the input, or exactly as many syllables as there are notes in the musical score. Intra-syllable is to stay consistent with the tonal shape for a syllable in case it is attached to more than one note, as Chinese is a tonal language. Inter syllable alignment is to make sure that the transition between notes are correct, as the same set of syllables might have a very different meaning with different tones.

LTAG (Lyrics-Melody Translation with Adaptive Grouping) [Li et al., 2023] argues that a rule based system is too rigid of a solution for the delicate nature of lyrics. Their approach consist of note embeddings which get fed to a encoder-decoder transformer together with lyrics, which is trained to generate a sequence of tokens dependant on both the embeddings and lyrics. The note embeddings and translated lyrics are then passed to a separate lightweight neural network which aligns the lyrics with the notes. To evaluate the result, they synthesize the generated lyrics and notes are synthesized by DiffSinger [Liu et al., 2022]. As no dataset exist with paired lyrics and lyrics alignment, they create a small dataset themselves with manual annotations. For training they create another lyrics translator with a length constraint similar to GagaST which backtranslates lyrics and automatically generate a melody for the backtranslation [Yu et al., 2019].

[18]`https://lyricstranslate.com/`

This will create a noisy source but accurate target. comparing GagaST and LTAG in MOS tests shows the folowing on a scale from 0 to 5:

- LTAG scoring 0.05 higher than GagaST in MOS-T (intelligibility and naturalness) when translating from English to chinese, and 0.13 higher when translating from Chinese to English.
- LTAG scores 0.19 higher than GagaST in MOS-S (singability) when translating from English to Chinese. Since They synthesized with a DiffSinger voicebank trained for Chinese synthesis, MOS-S is not presented for Chinese to English.
- LTAG scoring 0.04 higher than GagaST in MOS-Q (overall quality) when translating from English to Chinese. Since They synthesized with a DiffSinger voicebank trained for Chinese synthesis, MOS-Q is not presented for Chinese to English.

LTAG shows better results than GagaST and has a high chance at generally translating better between any language pair as it doesn't rely itself on any constrains. The problem however is collecting a dataset. If using backtranslation for generating training data, constrains will still be necessary specific to the target language anyway. None of the datasets nor recipes for LTAG are public, and reproducing such a system or emproving it is beyond the scope of this work.

Reproducing a system such as GagaST, i.e. fine tuning a transformer on lyrics data and applying language specific constrains, is a simpler approach proven to work well on Chinese. Therefore This work will take a similar approach for a sing-to-sing translation pipeline.

3 | Implementation

This proposed solution for a Singing Voice to Singing Voice Translator, is a cascaded system of state of the art models in Singing Voice Synthesis and Singing Voice Information Retrieval. The structure of the proposed pipeline can be seen in figure 3.1. In the remainder of this chapter, the individual modules of the pipeline will be explained in detail.

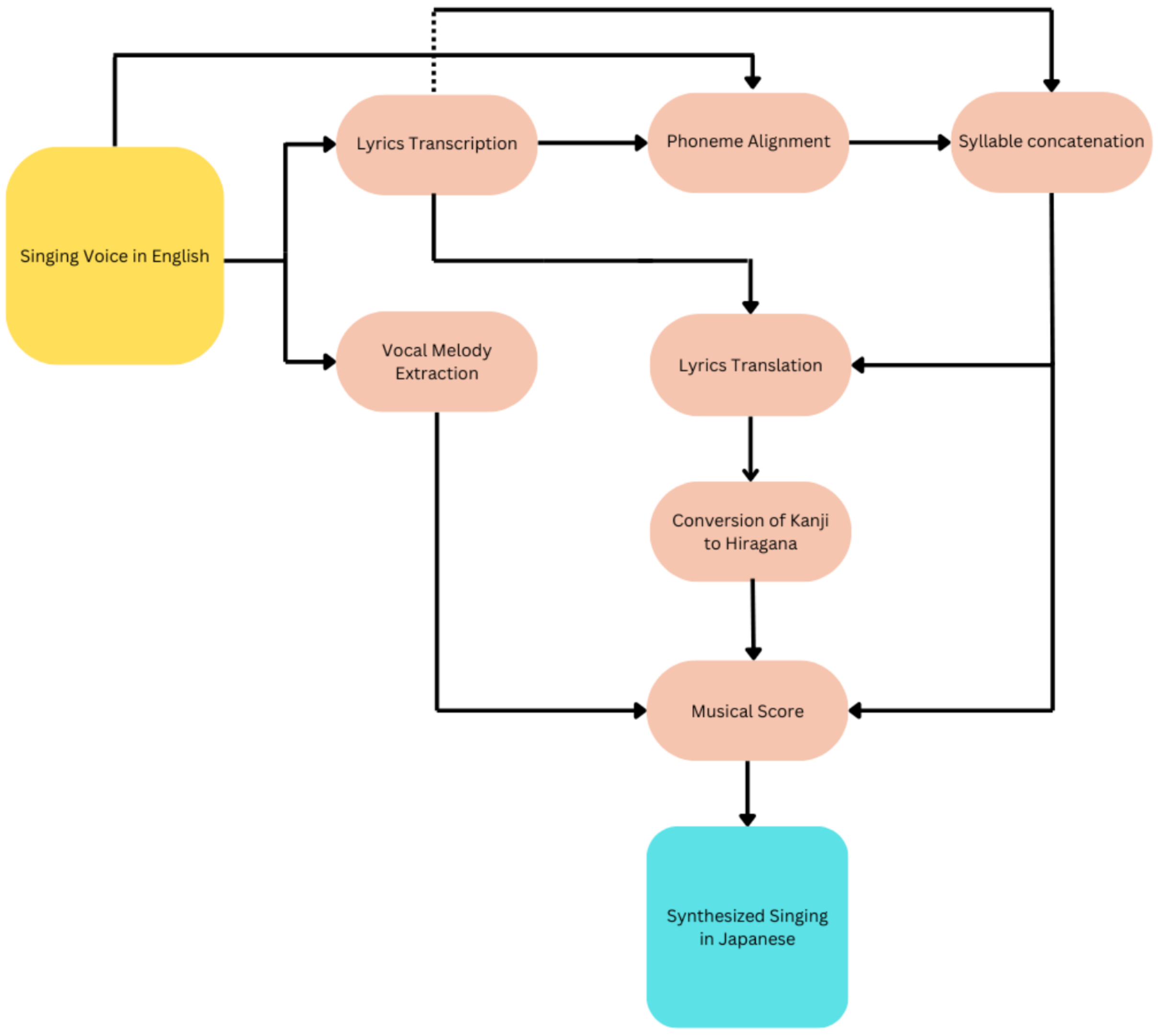


Figure 3.1. Overview of The Pipeline

3.1 Singing Voice in English

This is the input for which the pipeline has been designed for English singing. This pipeline is at this time built for monophonic recordings of singing. However, it has not been tested how well it functions on polyphonic music with or without source separation using e.g. [Défossez, 2022]

3.2 Lyrics Transcription

Due to its popularity, streamlined usage and capabilities for generalization in speech domains, Whisper [Radford et al., 2022] is the model of choice in this pipeline for Automatic lyrics transcription.

Whisper is an ASR system with the goal to create a speech recognizer which generalizes well to different domains (e.g multi-lingual speech transcription, language detection and translation) without the need of fine-tuning on a specific dataset as fine-tuning can often times be a disadvantage. With this goal in mind, Whisper focuses on the capabilities of large-scale weak supervision, and not the creation of a a new model. Whisper is based on the encoder-decoder transformer [Vaswani et al., 2017]. See section 3.2.1 for more information on transformers. For training Whisper, they create one the larges labeled ASR datasets ever, at 680.000 hours of labeled speech data.

Whisper is trained in a multitask training format. The encoder input is mel-spectrograms which get passed through a Conv1D network. The Decoder's task is to jointly predict a sequence of tokens arching over several tasks, e.g. the transcription, language token, voice activity detection.

In this pipeline, the pretrained Whisper-Large-V3 is used [1] which is the largest checkpoint available. The model is run on CPU for inference as it takes up to much memory for a 6GB GPU, but can very well run on 16GB of RAM with relatively low inference time (about 15s for a 30s audio recording). However, a chunking algorithm will be necessary if longer recordings than 30s are desired. By chunking every 30s of audio, Whisper uses about 15GB of memory.

Whisper has not been fine-tuned in anyway, but is used straight out of the box. The literature doesn't indicate that Whisper has been trained on any singing data, so it is unknown what fine-tuning Whisper might lead to. DALI [Meseguer-Brocal et al., 2018] might be a candidate for fine-tuning, as Whisper has supposedly been trained on data with background music. However, their own literature suggest that fine-tuning might not be a positive thing to do, and is counter intuitive to what Whisper was designed for.

Here are a few examples of transcriptions by Whisper:

Example 1: Audio can be found here [2]

- "So I heard you found somebody else And at first I thought it was a lie I took all my things that make sounds The rest I can deal without"

[1]`https://huggingface.co/openai/whisper-large-v3`

[2]`https://drive.google.com/file/d/1lxO6ybG9CFsgGVaqge2v8RfuncHRl_oB/view?usp=drive_link`

Example 2: Audio can be found here [3]

- "If I go crazy then will you still call me Superman? If I'm alive and well, will you be there to hold in my hand? I'll keep you by my side with my superhuman might, Kryptonite."

Example 3: Audio can be found here [4]

- "I heard there was a secret chord that David played and it pleased the Lord, but you don't really care for music, do you? It goes like this, the fourth, the fifth, the minor fall, and the major lift, the baffled king composing Hallelujah."

Example 4: Audio can be found here [5]

- "Fly me to the moon, let me play among the stars. Let me see what spring is like on Jupiter and Mars. In other words, hold my hand In other words Baby kiss me"

Example 5: Audio can be found here [6]

- "Don't try to make yourself remember Darling, don't look for me I'm just a story you've been told So let's pretend a little longer Cause when we're gone, everything goes on"

These same 5 examples will be used for the rest of the tests throughout the report.

3.2.1 Transformers

The transformer is a sequence-to-sequence model first proposed in "Attention is All You Need" [Vaswani et al., 2017] as an encoder-decoder architecture. An essential partner to the transformer is the tokenizer. A tokenizer maps entries from a vocabulary to a number which serve as the input for the transformer. The input layer of the transformer, i.e. embedding layer, embeds these tokens with a very simple neural network which is trained jointly with the entire transformer. Each token get embedded independently, but with the exact same weights in the network. This allows for sequences of variable length.

Since a sentence can have very different meanings depending on the position of each word, a positional encoding gets added to each embedding. Several methods exist, however Attention is All You Need implements alternating sine and cosine encoding. The positional embeddings are given by alternating sines and cosines in decreasing frequencies. The more tokens there are, the more sines and cosines will be used. See figure 3.2. These encodings are simply added to the embeddings

Self Attention is a mechanism to measure the similarity between words. This is useful for the transformer to understand context, e.g. what object is being referred to by the word "it" in the sentence "the dog came to greet me and it was so happy"?

Self attention is calculated by three metrics, queries Q, keys K and values V. Q is simply the outcome of multiplying the positionally encoded embedding with some weights to get a set of

[3]https://drive.google.com/file/d/1Tk-1Hr5RCAnkI74fvvmB8rjWE9Jt-rM1/view?usp=drive_link
[4]https://drive.google.com/file/d/1WsVMsLk4VO3MJ6CBEiUCh2SRm1nBrOur/view?usp=drive_link
[5]https://drive.google.com/file/d/13h1njgErkobjDZL9l0j8FIAf0B7aI8Rd/view?usp=drive_link
[6]https://drive.google.com/file/d/1xGnYbPU_SDo3-0E4DWmpZ41e5Lzm07vi/view?usp=drive_link

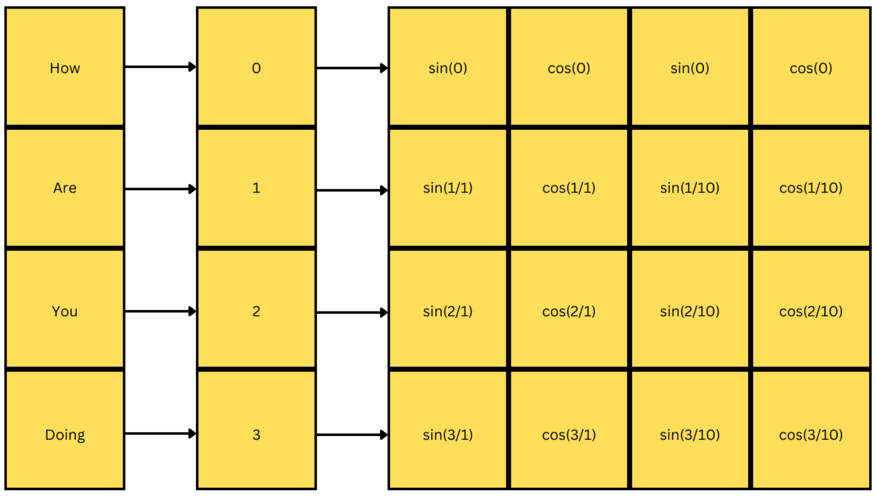


Figure 3.2. Illustration Of Positional Embedding

Q values. Similarly, a set of K values are created from other weights. Then, a K set is also generated for another word, and thus by calculating the dot product between Q and all Ks, the similarity between the Q word and the K word can be measured, including itself. When encoding a word, each similarity score will be ranked to how much influence they will have on encoding the given word, which is done with the softmax function to have them on a scale from 0 to 1. Similarly to Q and K, V is created by multiplying the embeddings with some weights. V is used to scale the dot product bewtween Q and Ks. The softmax outputs which are scalled by V are then the final self-attention scores for the word.

So the procedure is to calculate Q and make self-attention values by comparing its similarity to every K in the sequence. Thus, to get the self-attention for the next word , it is simply doing the same procedure with a new query. All weights are the same. This is however only a single self-attention cell. In Attention is All You Need, they make a stack of 8 cells with their own unique weights that run in parallel, called multi-head attention.

All of these calculations can be done in parallel, making it very scalable for parallel computing with GPU servers. Calculating the self-attention values of a word does not depend on any other self-attention values, which is a major reason for the popularity of transformers in large-scale data driven solutions.

The positional encoded embeddings are added to the self-attention values as residual connections to not loose positional data in the self-attention mechanism. This creates the foundation of the encoder in a transformer.

The decoder uses the same mechanisms as the encoder, however by embedding the target sequence. By inputting a "Start Of Sequence" token, the transformer will predict the next token, which is then input as the next token in the decoder. This procedure will continue until it produces a "End Of Sequence" token. While self-attention makes the encoder and decoder compare similarities between words, the encoder and decoder is also connected for encoder-decoder attention. This is to make sure that the decoder keeps track of the words from the encoder with high self-attention scores. The procedure is the same as in self-attention; compare similarities between Q (i.e. the decoder output) and all Ks (i.e. the encoder outputs).

To get an overview of the transformer, see figure 3.3

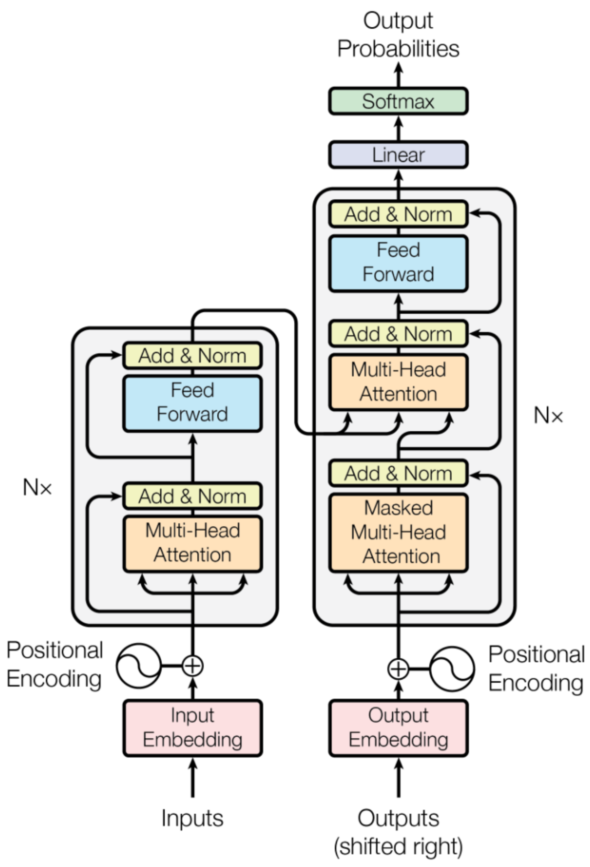


Figure 3.3. Original transformer from Attention is All You Need [Vaswani et al., 2017]

3.3 Phoneme Alignment

Given the quality of lyrics transcripts by Whisper, it can be taken advantage of with text informed phoneme-level lyrics alignment [Schulze-Forster et al., 2021]. This lyrics aligner is designed with the goal of assisting a model at singing voice separation. The models are however run sequentially, and it is possible to run the aligner alone.

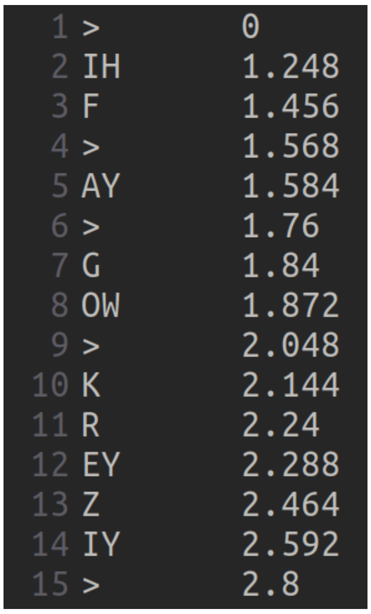
It starts with a text encoder and an audio encoder. The text encoder consist of a BLSTM (Bidirectional Long Short-Term Memory) [Hochreiter and Schmidhuber, 1997] which turns a phoneme sequence into hidden phoneme representations H. The audio encoder generates the STFT (Short Time Fourier Transform) of the audio signal comprised of both the singing voice and accompaniment. the transformed audio signal is fed through a fully connected layer and a tanh function followed by two BLSTM layers, which generates the the hidden audio features G.

The aligner then aligns H and G. One audio fra matches exactly one phoneme, whereas one phoneme can align to multiple audio frames. It is assumed that no phonemes are skipped, and they are pronounced in a specific order.

Singing has too much variance in phonetic pronunciation due to wider range of pitch etc. so they force monotonic alignment with a hybrid of attention and DTW (Dynamic Time Warping). The attention mechanism is useful for measuring the similarity between all elements in H and G, i.e. how likely is it that phoneme m was pronounced in frame n. With the obtained attention values, DTW is used to find the alignment which result in the optimal patch of minimum distance between phoneme and audio frame features.

The aligner is trained on 100 songs from the MUSDB dataset [Rafii et al., 2017] dataset which have been human annotated with phonetic alignments.

When using the aligner, a VAD (voice activity detection) threshold must be set, which defaults to 0. Choosing a VAD threshold is ambiguous, but with loud and clear singing, and little to no background noise, a VAD of 0 often works. The phonemes phonetics used are from the CMU (Carnegie Mellon University) pronunciation dictionary, and will therefore not always transcribe the exact pronunciation that was sung, but rather the standard pronunciation. See figure 3.4 for examples of alignments.



```
>       0
IH      1.248
F       1.456
>       1.568
AY      1.584
>       1.76
G       1.84
OW      1.872
>       2.048
K       2.144
R       2.24
EY      2.288
Z       2.464
IY      2.592
>       2.8
```

Figure 3.4. Example of aligned phonemes. the alignment to phonemes are in seconds and are denoted as onsets. ">" denotes the onset of a pause between phonemes

3.4 Syllable Concatenation

According to the literature [Guo et al., 2022], singing is usually done in a syllable wise manner, however the Japanese language is structured by mora, and not syllables. Mora are similar to syllables as a unit of speech, but are more specifically defined as the rhythm of speech, meaning that each mora has the same length in pronunciation. As such, mora is a timing based system. Japanese is made of three writing system. Hiragana and Katakana have different symbols but the exact same pronunciations of their 46 characters. There are five vowel sound あ, い, う, え, お (a, i, u, e, o) which each represent one mora. They can be combined with a consonant such as k to make か, き, く, け, こ (ka, ki, ku, ke, ko) which also makes one mora. However a long vowel such as "kaa" will represent two mora and be written as かあ in hiragana. In that way, hiragana and katakana characters represent one mora each.

the thrid writing system, kanji, are letters adopted from the Chinese alphabet and convey a meaning. Kanji will often not have a one true pronunciation in sentences, but will indeed depend on the context. However, their pronunciation is true to the other alphabets, and as such their pronunciation can be written in hiragana and katakana. 雨 for instance is pronounced with the two hiragana あめ (ame) which means rain.

for this work, the length constrain inspired by [Guo et al., 2022], is defined as minimum one Japanese mora per English syllable, and at maximum an amount of mora equal to syllable count + some threshold for allowed surplus. However, this approach has some flaws which will be discussed in section 6

By looking up English words in the CMU dictionary with the cmudict python wrapper [7], their pronunciations can be obtained. The phonetics are consistent with the lyrics aligner since it also uses the CMU dictionary. Pronunciations from the CMU dictionary also has indicators for vowel sounds which are the building blocks of syllables.

For this work, an algorithm was developed to look up an English word and concatenate the phonemes into syllables, based on simple rules in the following order:

- if phoneme p is a vowel sound, create new syllable entry
- if phoneme p is not a vowel sound, append it to the last generated syllable, or start a new one if there is none.
- If phoneme p+2 is a vowel sound, create new syllable entry
- proceed to next phoneme

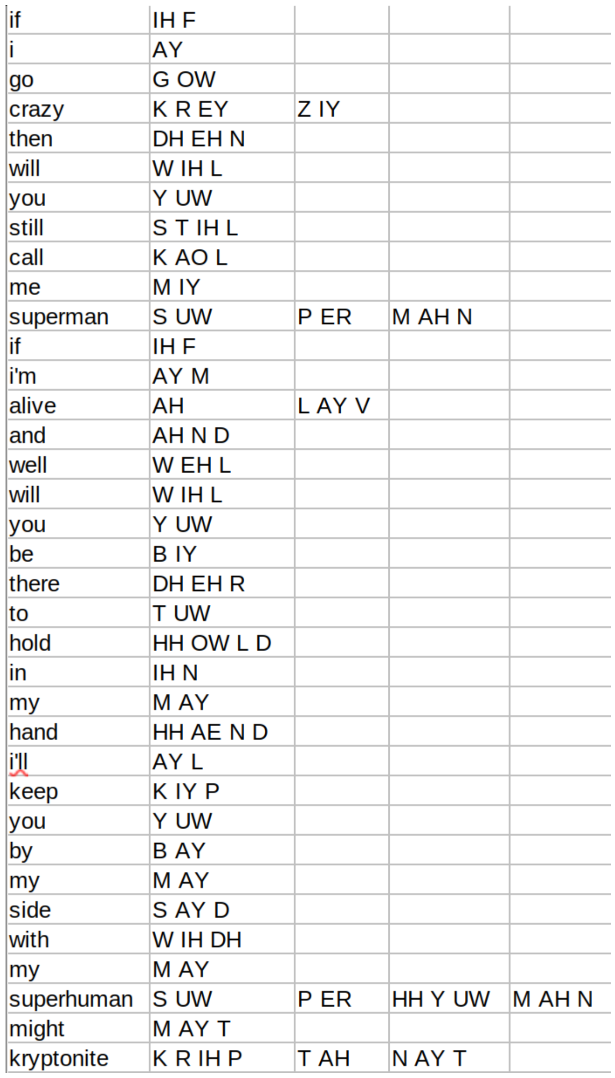
Since the sound Y is sometimes inconsistent with being categorised as a vowel sound or not, the algorithm adapts to surrounding syllables in order to shift over spare consonants:

- if first entry of syllable S is Y and the last entry of S-1 is not a vowel sound, and S-1 has at least 3 entries, then move the last entry of S-1 to be the first entry of S

Some of these methods are specific to encountered errors in syllable creation, so more errors might be encountered when not using a fully dynamic solution, but from personal experiments this solution is very consistent with the vowel dictionary in Synthesizer V.

[7] `https://github.com/prosegrinder/python-cmudict`

Since the phonemes are the exact same as in the lyrics alignment, the timings can be concatenated as well to generate notes with onsets and durations for each syllable. However, some pronunciation information is lost when using the overall syllable duration instead of the individual phoneme durations. See figure 3.5



if	IH F			
i	AY			
go	G OW			
crazy	K R EY	Z IY		
then	DH EH N			
will	W IH L			
you	Y UW			
still	S T IH L			
call	K AO L			
me	M IY			
superman	S UW	P ER	M AH N	
if	IH F			
i'm	AY M			
alive	AH	L AY V		
and	AH N D			
well	W EH L			
will	W IH L			
you	Y UW			
be	B IY			
there	DH EH R			
to	T UW			
hold	HH OW L D			
in	IH N			
my	M AY			
hand	HH AE N D			
i'll	AY L			
keep	K IY P			
you	Y UW			
by	B AY			
my	M AY			
side	S AY D			
with	W IH DH			
my	M AY			
superhuman	S UW	P ER	HH Y UW	M AH N
might	M AY T			
kryptonite	K R IH P	T AH	N AY T	

Figure 3.5. Example of syllables concatenated from phonemes. Syllables are split into cells, and in each syllable the phonemes are separated by white-space

3.5 Automatic Lyrics Translation

Inspired by the design of GagaST [Guo et al., 2022], this implementation of ALT will focus on the fine-tuning of a translation model on lyrics data. While it would likely be ideal to use a massive LLM such as ChatGPT [Radford et al., 2018] or LLaMA [Touvron et al., 2023], it is not feasible to run, let alone train, these models without access to large quantities of remote compute power. Therefore, a more compact solution is desired. As the research conducted in this work is mostly interested in the question of how a model can be trained for automatic lyrics translation, the choice of model might not be the most important part.

No Language Left Behind-200 (NLLB-200) [Team et al., 2022] is a model made for multi-lingual translation especially for low resource languages. NLLB-200 has achieved state of the art result over a wide range of translation directions. While their baseline model is quite large at 54b parameter which makes it infeasible for use in this pipeline, they have also released it at several smaller checkpoint. NLLB-200-600M with 600M parameters is a lightweight model which is also very popular on HuggingFace [8] Japanese is however not considered a low level language as it serves as one of the most abundant monolingual corpora for NNLB which is also true for mBART-50 [Tang et al., 2020], another very popular multilingual translator on HuggingFace [9]. As the two primary choices for open source multilingual translation in lightweight models, both NLLB and mBART will most likely be good candidates for this pipeline; especially if there is a drive to expand to more languages than English and Japanese in the future.

For this implementation NLLB-200-600M is used. NLLB is a sequence-to-sequence model based on the transformer [Vaswani et al., 2017], see section 3.2.1 for more details on this architecture. The main contribution of NLLB is the 6creation of datasets and recipes to create large-scale training datasets including low resource languages.

In [Guo et al., 2022] they argue that they would ideally train on paired lyrics, but such data doesn't exist in any substantial quantity between English and Chinese. They instead pretrain their model for general translation on the WMT14 dataset [10] containing 29.6M English to Chinese sentence pairs. They then perform self-supervised learning on 12.4M lines of Chinese monolingual lyric lines and 109M English Lyric lines. In the end they fine-tune the model for the lyrics translation task using 140K lines of low quality lyrics translation.

in [Li et al., 2023] they create a small human annotated paired lyrics dataset of a couple throusnad verses. For training they automatically backtranslate monolingual songs and automatically generate melodies for bakctranslated songs to facilitate supervised training. It is not stated how many songs were undergone this procedure, but they argue that having a noisy source in translation is not much of a problem with high quality targets, as it will learn to generate singable lyrics from poorly written lyrics.

Assuming NNLB is properly pretrained, the remaining task is to fine-tune on lyrics. Considering the pursuit of supervised training and success in using noisy source language data, There might be data in paired English and Japanese which are adequate.

[8]`https://huggingface.co/facebook/nllb-200-distilled-600M`
[9]`https://huggingface.co/facebook/mbart-large-50-many-to-many-mmt`
[10]`https://huggingface.co/datasets/wmt14`

Japan has a huge entertainment media in animation shows which is very popular in the western word. Demon Slayer: Mugen Train for instance was the highest grossing movie of 2020 reaching over $507 million at the worldwide box office [11]. Every Japanese animation show has an intro song and an outro song which change about once per season or movie. Typically, these songs english subtitles. These are most likely not very consistently singable, but one might argue that human made subtitles might be of higher quality than backtranslated ones. However, there is no evidence for such a statement.

Nonetheless, the web was scraped of anime lyrics with their original Japanese lyrics and English translations. After cleaning the data by removing pairs that don't have the same amount of lines on both sides or is inconsistent with the tokenization process, there is a total of 212.698 lines, 191.429 of which are used for training.

Training was performed on Kaggle [12] with a P100 GPU, facilitating 16GB of memory. To max out on memory, the training was done with a batch size of 32 using the the adafactor optimizer, as Adam took up too much memory. Gradient accumulation was also enabled to reduce memory overhead but it will however slow down training time. The starting learning rate was the HuggingFace Transformers standard at 2e-4, and for regularization weight decay was used. There was experimentation with data augmentation through random word swapping, synonym substitution and word deletion. However, it did not make it to the final revision of training the model due to inconsistencies with the tokenization process, but might very well be something to add in the future. Considering the dataset size it will most likely be better to scrape more data first instead, as the web has not been thoroughly scraped for Japanese Animation lyrics and other Japanese songs.

On figure 3.6 the results of training can be seen. Notice how there is a spike at the 18th epoch, this is due to the last 8 epochs having s higher starting learning rate at 2e-4. This was done to see if anything could be done to the flat-lining of the evaluation loss, but ended up overfitting intead. With the minuscule changes to the evaluation loss, the lowest loss achieved was 2.1208 at the 14th epoch.

As loss is the only objective measure present, the checkpoint at the 14th epoch will serve as the model for testing.

For inference, beam search is applied to guide the translation. The number of beams are set to 45 as that is what comfortably runs on 16GB of RAM. As discussed in section 3.4, it is not known what the pronunciation of a kanji is before it has been transformed to its pronunciation in context. How this conversion is done is discussed in section 3.6. Therefore, as many outputs as possible are generated in the beam search, in this case 45. After the kanji conversion to hiragana, the sentence with an amount of mora closest to the amount of syllables in the english sentence is chosen as desired output. If there are more than one sequence at such a length, the one with the highest likely token sequence according to the beam search is selected.

To guide the beam search, a bias towards shorter outputs is added, as the outputs have a tendency of being too long. In an attempt to preserve them meaning as much as possible in translation, top_k=100 and top_p=0.96 is used. However, this might not necessarily be the best idea for singability [Franzon, 2015] as it will limit the "artistic freedom" of the model.

[11]`https://www.imdb.com/title/tt11032374/`
[12]`https://www.kaggle.com/`

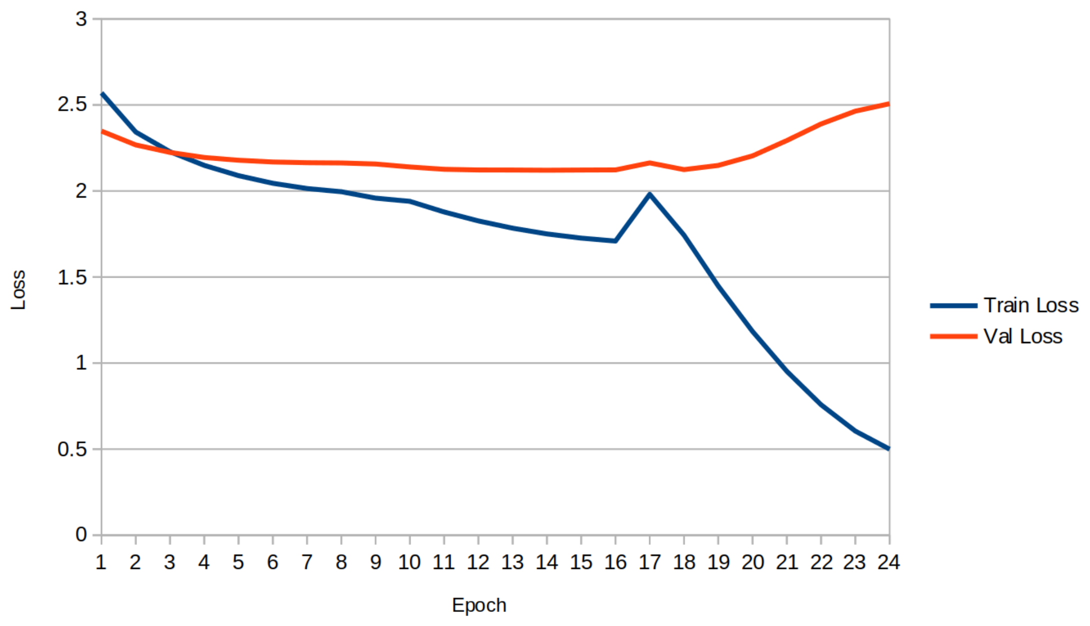


Figure 3.6. The results of fine-tuning NLLB-200-600M on Japanese Animation intro/outro songs

Here are some examples of outputs, given the inputs from Whisper

Example 1

- So I heard you found somebody else And at first I thought it was a lie I took all my things that make sounds The rest I can deal without
- 他に誰かがいたと聞いた最初は嘘だと思ってた音が出るものを全て取って残した

Example 2

- If I go crazy then will you still call me Superman? If I'm alive and well, will you be there to hold in my hand? I'll keep you by my side with my superhuman might, Kryptonite.
- 気が狂ったらまだスーパーマン生きていれば抱きしめてくれるの僕の超人力クリプトナイトで

Example 3

- I heard there was a secret chord that David played and it pleased the Lord, but you don't really care for music, do you? It goes like this, the fourth, the fifth, the minor fall, and the major lift, the baffled king composing Hallelujah.
- ダビデが弾いた秘密の和音神様に甘えて音楽なんていうか四番五番マイナーでメジャーリフト戸惑う王がハレ ルヤをぐ

Example 4

- Fly me to the moon, let me play among the stars. Let me see what spring is like on Jupiter and Mars. In other words, hold my hand In other words Baby kiss me
- 月に飛ばして星の中で遊ぼうよ木星火星で春を見せてよつまり手を握ってよ

Example 5

- Don't try to make yourself remember Darling, don't look for me I'm just a story you've been told So let's pretend a little longer Cause when we're gone, everything goes on
- 思い出さないでダーリン探さないで君に話された物語だからもう少し気取って居なくなるから

3.6 Conversion of Kanji to Hiragana

Conversion of Kanji is in this work done with pykakasi [13], a NLP library for conversion between Japanese writings romanization of Japanese text.

Converting a kanji to hiragana as an isolated event will often not give an accurate result as it lacks context. An option would be to convert token wise from the lyrics translation output, but that is also not very consistent as tokens can sometimes so sub-words and again, kanji that can't be converted on their own are a singular token. Thus, what is done in this pipeline is simply passing the entire sentence to to pykakasi, for full context. However a problem with this approach is that it relies on a well structured sentence were every context is correct. An alternative could be an algorithm that can recognize and split words.

Due to attempts at standardizing the tokenization process of noisy data with non-Japanese tokens (such as spaces, commas, question marks and exclamation marks), there are no punctuatuins or break in sentences. This might lead to unnatural structure of a Japanese sentence, which in return can make it difficult for pykakasi to convert kanji.

Here is an example of the converted kanji. Note that katakana are also turned into hiragana since the entire sentence is passed to pykakasi:

- Original: 気が狂ったらまだスーパーマン生きていれば抱きしめてくれるの僕の超人力クリプトナイトで
- Converted: きがくるったらまだすーぱーまんいきていればだきしめてくれるのぼくのちょうじんちからくりぷとないとで

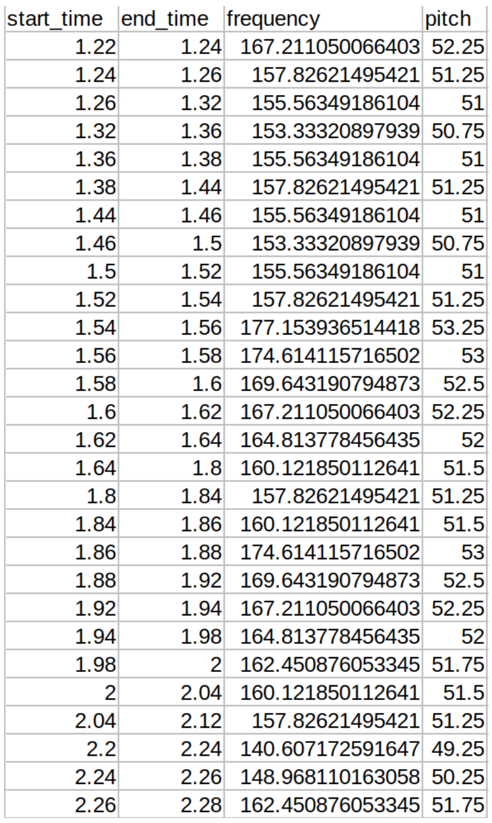
[13] `https://pykakasi.readthedocs.io/en/latest/?badge=latest`

3.7 Vocal Melody Extraction

Vocal melody extraction is in this pipeline done with [Lu and Su, 2018]. This vocal melody extractor takes inspiration from a successful technique often used in computer vision [Chen et al., 2017], semantic segmentation. They present a solution for extracting a vocal melody with a deep convolutional neural network with dilated convolutions as a semantic segmentation tool. Symbolic data (e.g MIDI) makes it easier to make a dataset of melodies, so they develop an adaptive model based on the progressive neural network [Rusu et al., 2016] for cross-domain parameter sharing, with the goal of learning to shift the symbolic domain to the audio domain.

the framework is integrated in omnizart [Wu et al., 2021] for streamlined usage. When generating the melody, the output is presented both in the symbolic domain with pitches and the audio domain with frequencies. The melody has a resolution of 2ms. An example can be seen at figure 3.7

Here is an example of an extracted melody [14] given this recording as input [15] As you might notice, there is a slight bug in the vocal melody extraction framework. Given a singing input, the few last moments of the melody are rendered completely monotone. To counteract this, i end a recording with a little hum, such that the hum will get rendered monotone instead of the singing melody.



start_time	end_time	frequency	pitch
1.22	1.24	167.211050066403	52.25
1.24	1.26	157.82621495421	51.25
1.26	1.32	155.56349186104	51
1.32	1.36	153.33320897939	50.75
1.36	1.38	155.56349186104	51
1.38	1.44	157.82621495421	51.25
1.44	1.46	155.56349186104	51
1.46	1.5	153.33320897939	50.75
1.5	1.52	155.56349186104	51
1.52	1.54	157.82621495421	51.25
1.54	1.56	177.153936514418	53.25
1.56	1.58	174.614115716502	53
1.58	1.6	169.643190794873	52.5
1.6	1.62	167.211050066403	52.25
1.62	1.64	164.813778456435	52
1.64	1.8	160.121850112641	51.5
1.8	1.84	157.82621495421	51.25
1.84	1.86	160.121850112641	51.5
1.86	1.88	174.614115716502	53
1.88	1.92	169.643190794873	52.5
1.92	1.94	167.211050066403	52.25
1.94	1.98	164.813778456435	52
1.98	2	162.450876053345	51.75
2	2.04	160.121850112641	51.5
2.04	2.12	157.82621495421	51.25
2.2	2.24	140.607172591647	49.25
2.24	2.26	148.968110163058	50.25
2.26	2.28	162.450876053345	51.75

Figure 3.7. Example of frequencies and pitches transcribed. Start_time and end_time are in seconds, giving a resolution of 2ms between data points

[14] `https://drive.google.com/file/d/131Ar3TUJzu5o6rftinYoZDRik0eZzToU/view?usp=drive_link`
[15] `https://drive.google.com/file/d/1Tk-1Hr5RCAnkI74fvvmB8rjWE9Jt-rM1/view?usp=drive_link`

3.8 Synthesized Singing in Japanese

The requirements Synthesizer V is a musical score and lyrics. For this implementation, the musical score is defined by a melodic contour extracted from a singing voice and a sequence of notes. A specific pitch is not assigned to the note from the melody, but are instead all assigned to a standard pitch of 60 (the musical note C4). The extracted melody contour is then used to automate the variance from 60 across the timeline.

When phonemes are used to define notes as a means to reproduce the original singing in its original language, a fairly high quality synthesis is produced, based on personal opinion. An example can be heard here [16]. A visual representation of the phoneme notes and melody can be seen at figure 3.8

Concatenating the phonemes into syllables still produces an intelligible and singable synthesis, but is slightly degraded in quality. The is most likely due to miss-alignments between the melody and the phoneme transitions, as the synthesizer will have to handle the timings itself when the input is a syllable. An example can be heard here [17]. A visual representation of the syllable notes and melody can be seen at figure 3.9

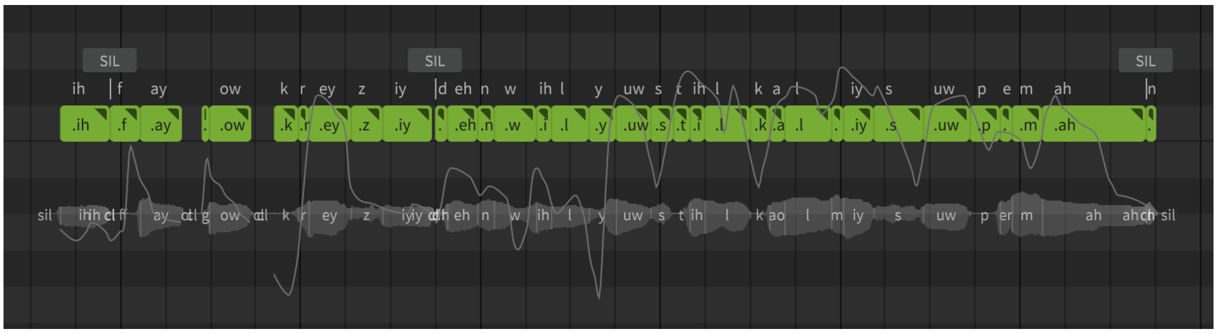

SIL
SIL
SIL
ih f ay ow k r ey z iy d eh n w ih l y uw s t ih l k a l iy s uw p e m ah n
.ih .f .ay .ow .k .r .ey .z .iy .eh .n .w .i .l .y .uw .s .t .i .l .k .a .l .iy .s .uw .p .m .ah
sil ih f ay g ow k r ey z iy dh eh n w ih l y uw s t ih l k ao l m iy s uw p er m ah sil

Figure 3.8. Notes defined by phonemes. Pitch automation by pitch contour

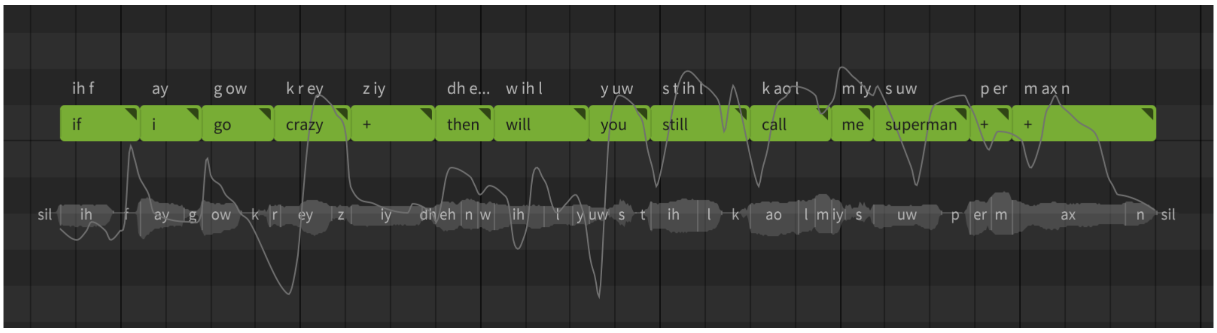

ih f ay g ow k r ey z iy dh e... w ih l y uw s t ih l k ao l m iy s uw p er m ax n
if i go crazy + then will you still call me superman + +
sil ih f ay g ow k r ey z iy dh eh n w ih l y uw s t ih l k ao l m iy s uw p er m ax n sil

Figure 3.9. Notes defined by syllables. Pitch automation by pitch contour

The Japanese lyrics generation is constrained by a combination of a wide beam search and a a narrow search for a sequence with a length equal to the syllables in the English lyrics. An example of inputting Japanese in the syllable notes can be seen on figure 3.10. At the current time, there has not been developed a standard way of assigning characters to notes that are

[16]https://drive.google.com/file/d/1XWUhmM047su_oQEf0S1xPBBCbjjafAzQ/view?usp=drive_link
[17]https://drive.google.com/file/d/1xW3VHa4Sz2xn3KVoVTjWnMANg-1Uau8Q/view?usp=drive_link

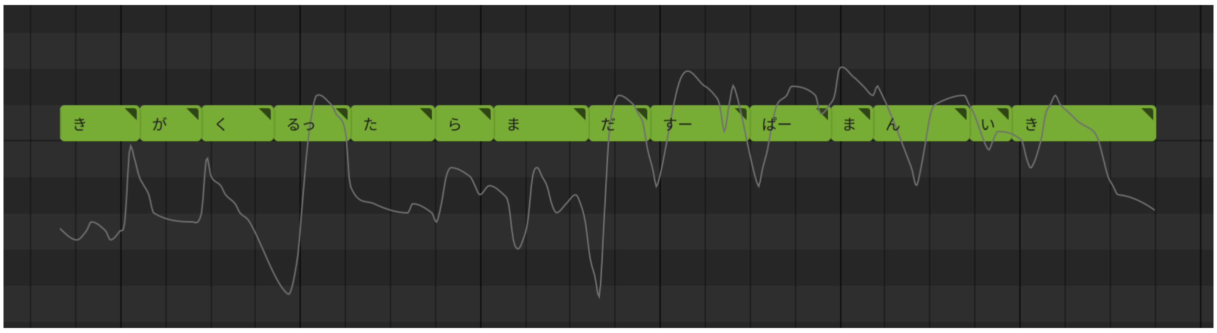


Figure 3.10. Notes defined by syllables with Japanese lyrics. Pitch automation by pitch contour

in surplus. Therefore they it is done randomly, which will most likely result in words getting separated incorrectly.

Lua [18] is the scripting language used to automate functionalities of Synthesizer V such as note generation and vocal contour.

Here are 5 examples on the generated vocal synthesis by Synthesizer V with a musical score defined by automatically aligned syllables and vocal contour, and automatically translated lyrics from English to Japanese captured by ASR.

Example 1: The 1975 - Somebody Else

- Source Audio [19], Synthesized Audio [20]

Example 2: 3 Doors Down - Kryptonite

- Source Audio [21], Synthesized Audio [22]

Example 3: Lenoard Cohen - Hallelujah

- Source Audio [23], Synthesized Audio [24]

Example 4: Frank Sinatra - Fly Me To The Moon

- Source Audio [25], Synthesized Audio [26]

Example 5: Porter Robinson - Everything Goes On

- Source Audio [27], Synthesized Audio [28]

[18]`https://www.lua.org/`
[19]`https://drive.google.com/file/d/1lxO6ybG9CFsgGVaqge2v8RfuncHRl_oB/view?usp=drive_link`
[20]`https://drive.google.com/file/d/1doqLm7CJe1sIiaJx2-L6mmnTPKb1v610/view?usp=drive_link`
[21]`https://drive.google.com/file/d/1Tk-1Hr5RCAnkI74fvvmB8rjWE9Jt-rM1/view?usp=drive_link`
[22]`https://drive.google.com/file/d/16IwYlrt_9sp0zc9VIc_TFh4RlzL2XejI/view?usp=drive_link`
[23]`https://drive.google.com/file/d/1WsVMsLk4VO3MJ6CBEiUCh2SRm1nBr0ur/view?usp=drive_link`
[24]`https://drive.google.com/file/d/16Vr5ZaQZQ2n8kZoQ6IIkz0D3JBblohpD/view?usp=drive_link`
[25]`https://drive.google.com/file/d/13h1njgErkobjDZL9l0j8FIAf0B7aI8Rd/view?usp=drive_link`
[26]`https://drive.google.com/file/d/17B4jnJKOABiWLPht7d9fO1BYkVFhci8H/view?usp=drive_link`
[27]`https://drive.google.com/file/d/1xGnYbPU_SDo3-0E4DWmpZ41e5Lzm07vi/view?usp=drive_link`
[28]`https://drive.google.com/file/d/1woF8pdjD8IQW3HV-EdaeA93TkSh3cqIX/view?usp=drive_link`

4 | Test and Results

To evaluate the quality of the automatically generated Japanese singing voice synthesis presented 3.8, two Japanese people was asked to evaluate them on 6 questions with a 5-point scale from very poor to very good. As to have some comparative measure, the Japanese people were also asked to evaluate the same songs generated with the baseline NLLB-200-600M model with no fine-tuning which can be found here [1] [2] [3] [4] [5]. The questions were these:

- How much sense do the lyrics make?
- How natural is the Japanese used in the lyrics?
- How well is the meaning of the original lyrics preserved?
- How singable are the generated lyrics?
- How well is the lyrics and melody aligned?
- What is the overall quality of the generated Japanese singing?

the feedback was as follows for the fine-tuned model, represented as the average answer.

The 1975 - Somebody Else

- Q1: 2.5
- Q2: 3
- Q3: 3
- Q4: 2.5
- Q5: 2.5
- Q6: 2

3 Doors Down - Kryptonite

- Q1: 2
- Q2: 1.5
- Q3: 2
- Q4: 2.5
- Q5: 2.5
- Q6: 2

Leonard Cohen - Hallelujah

- Q1: 1
- Q2: 1
- Q3: 1.5
- Q4: 1.5

[1] https://drive.google.com/file/d/19h0N3fFijxUx7hLewINdTh0awXpK-fdk/view?usp=drive_link
[2] https://drive.google.com/file/d/16L0jK4S_iTR5zxxHfspIel1cG6l9k4NN/view?usp=drive_link
[3] https://drive.google.com/file/d/17ePZi9jH-5KGo5iCAFpGEEfy1kUE97hp/view?usp=drive_link
[4] https://drive.google.com/file/d/1Tgxbqo-Mb7V4oBZNgPl_cCLmbXYmzu6m/view?usp=drive_link
[5] https://drive.google.com/file/d/1u1tslXrXjyavKEt2tHng-14L5mZeDqq9/view?usp=drive_link

- Q5: 1.5
- Q6: 1

Frank Sinatra - Fly Me To The Moon

- Q1: 3.5
- Q2: 4
- Q3: 2.5
- Q4: 2.5
- Q5: 3
- Q6: 3

Porter Robinson - Everything Goes On

- Q1: 4
- Q2: 4
- Q3: 2.5
- Q4: 3
- Q5: 3
- Q6: 3.5

the feedback was as follows for the baseline model, represented as the average answer.

The 1975 - Somebody Else

- Q1: 2
- Q2: 4
- Q3: 2
- Q4: 3
- Q5: 3
- Q6: 3

3 Doors Down - Kryptonite

- Q1: 4.5
- Q2: 4.5
- Q3: 4.5
- Q4: 2.5
- Q5: 1.5
- Q6: 2.5

Leonard Cohen - Hallelujah

- Q1: 3.5
- Q2: 3.5
- Q3: 2
- Q4: 2.5
- Q5: 3.5
- Q6: 2.5

Frank Sinatra - Fly Me To The Moon

- Q1: 2
- Q2: 3.5
- Q3: 1.5
- Q4: 3
- Q5: 2.5
- Q6: 2

Porter Robinson - Everything Goes On

- Q1: 2
- Q2: 3
- Q3: 1.5
- Q4: 4
- Q5: 4
- Q6: 3

which means that the fine tuned model got the mean opinion scores:

- Q1: 2.6
- Q2: 2.7
- Q3: 2.3
- Q4: 2.4
- Q5: 2.5
- Q6: 2.3

and the baseline model got the mean opinion scores:

- Q1: 3.3
- Q2: 3.7
- Q3: 2.3
- Q4: 3
- Q5: 2.9
- Q6: 2.6

Even though there were only two test subject, the data indicates that the fine-tuning of the model did not help, as the baseline model performs better at quite a large margin.

A few comment were written by one of the participant. Regardless of translation model, some key takeaways for future improvement is that the model often doesn't complete the translation, which could possibly be due to length constraints. There are Unnatural cutting of words, but that is to be expected with no standard for separation of words for either concatenation of characters into single notes nor any determination to when it is natural to have a break in the sung lyrics. Some of the sentences are way to long considering the melody, causing very rushed and unnatural singing. Opposite meaning in translation occurs as key words are not spotted. Kanji sometimes pronounced incorrectly. Sometimes the translation is very literate and sound very "stiff".

The participant suggest that the cutting of words should be the first step in more natural Japanese lyrics, as it is not very intelligible when pauses are made in the middle of words or they are pronounced to quickly to hear.

5 | Conclusion

The work presented in this report first explores and conducts a state of the art analysis in the art of singing voice synthesis and singing voice information retrieval. After concluding what technologies are necessary for reproducing and autonomously understanding the human singing voice, a cascaded pipeline is proposed for autonomous singing voice to singing voice translation. Singing voice to singing voice translation is the procedure of translating the audio of a human singing voice in some language into audio of a singing voice in some other language. In this proposal, that is the translation of an English singing voice into a synthesized Japanese singing voice. In essence, the pipeline consist of an automatic speech recognizer, a phoneme level lyrics aligner, a vocal melody extractor and a singing voice synthesizer. The pipeline is successfully assembled and is capable of creating synthesized Japanese singing on the basis of an English singing voice, however a brief study with Japanese natives show that the system has a far way to go to become a high quality lyrics translator and natural singer.

6 | Future Work

Results in this report show that further understanding of the Japanese language is needed for lyrics translation, as well as for its alignment with a melody for a more natural singing voice. In the presented work, mora were directly mapped to syllables from English singing which themselves lost some alignment information in transition from phonemes. According to the literature [Hayes, 1989], a syllable represent a mora if it is a short vowel, but it represents two mora if it is a long vowel. In English the ending consonants of a stressed syllable represents a mora. And if the syllable onset starts with a consonant, that consonant does not represent any mora.

These are all very useful information about moraic languages that might very well help in improving the alignment of Japanese lyrics

As suggested by Japanese natives during evaluation, it is very important to semantically group the sung characters with neighbouring characters that jointly make up word. A song will sound unnatural and be less intelligible if there are prolonged breaks in words, or if one sentence has a word that belonged in the previous sentence.

As for pronunciation of kanji, it is yet to soon to tell if the error lies in the tools used, simply just because of poorly generated data.

As the alignment of a Japanese singing voice will most likely not be the same as an English singing voice, the vocal melody will most likely not completely match either. An idea could be to guide the vocal contour. When investigating a vocal contour, it is common that there will be deep valleys and steep hills for brief moments just as the singer transition from one word to another. Such information which is based on incidents might be useful in correcting for another language. An alternative would be to not directly use the vocal contour for for automation, but use it to classify the pitches of notes.

Once a more well defined singing voice to singing voice system is defined, it is essential to make a formal mean opinion score test with a good amount of people.

The evaluations indicate that no meaningful positives came from fine tuning NLLB-200-600M [Team et al., 2022] on a set of around 200.000 lines of noisy translations. Therefore, further research in the state of art in machine translation is needed to find the optimal solution for the task, as this report did get much involved in it. However, as with most machine learning tasks, the error is not the model but the data. Therefore, further data collection and a better data cleaning protocol might be some of the most crucial tasks for future work.

No test have been conducted on the lyrics alignment capabilities of [Demirel et al., 2021] and its performance versus [Schulze-Forster et al., 2021].

Neither has the lyrics transcription capabilities of Whisper [Radford et al., 2022] been formally investigated. With this purpose, fine-tuning of whisper could be a possibility, however it would not seem as a high priority task considering its apparent performance.

With open source singing synthesis solutions such as [Yamamoto et al., 2022], one might consider adapting to those for possibly more control of the synthesis.

end-to-end solutions for speech-to-speech are also gaining popularity [Seamless Communication et al., 2023] and might as well be considered for adapting to the singing domain.

Considering more abstract approaches to the problem of creating high quality singing voice translations, there could also be options text generation by query or keywords.

Bibliography

, **2023**. *Graph modeling for vocal melody extraction from polyphonic music.* Applied acoustics, 211, article 109491, 2023.

Anastasakos et al., **1996**. T. Anastasakos, J. McDonough, R. Schwartz and J. Makhoul. *A compact model for speaker-adaptive training.* Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96, 1996, `https://ieeexplore.ieee.org/document/607807`.

Baevski et al., **2020**. Alexei Baevski, Henry Zhou, Abdelrahman Mohamed and Michael Auli. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*, 2020, `https://arxiv.org/abs/2006.11477`.

Bahdanau et al., **2014**. Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*, 2014, `https://arxiv.org/abs/1409.0473`.

Bain et al., **2023**. Max Bain, Jaesung Huh, Tengda Han and Andrew Zisserman. *WhisperX: Time-Accurate Speech Transcription of Long-Form Audio*, 2023, `https://arxiv.org/abs/2303.00747`.

Bittner et al. Rachel M Bittner, Justin Salamon, Slim Essid and Juan P Bello . *MELODY EXTRACTION BY CONTOUR CLASSIFICATION.* International Conference on Music Information Retrieval (ISMIR), Sep 2015, Malaga, Spain.

Boersma, **01 2000**. Paul Boersma. *Accurate Short-Term Analysis Of The Fundamental Frequency And The Harmonics-To-Noise Ratio Of A Sampled Sound.* Proceedings of the Institute of Phonetic Sciences, 17, 2000.

Chen et al., **2017**. Liang-Chieh Chen, George Papandreou, Florian Schroff and Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*, 2017, `https://arxiv.org/abs/1706.05587`.

Chen et al., **2019**. Ming-Tso Chen, Bo-Jun Li and Tai-Shih Chi. *CNN Based Two-stage Multi-resolution End-to-end Model for Singing Melody Extraction.* IEEE, 2019. ISBN 9781479981311.

Dabike and Barker, **2019**. Gerardo Roa Dabike and Jon Barker. *Automatic Lyric Transcription from Karaoke Vocal Tracks: Resources and a Baseline System*, 2019, `https://www.isca-speech.org/archive/interspeech_2019/dabike19_interspeech.html`.

Demirel et al., **2020**. Emir Demirel, Sven Ahlbäck and Simon Dixon. *Automatic Lyrics Transcription using Dilated Convolutional Neural Networks with Self-Attention.* IEEE. 2020 International Joint Conference on Neural Networks (IJCNN), 2020. ISBN 1728169267, `https://arxiv.org/abs/2007.06486`.

Demirel et al., oct 2021. Emir Demirel, Sven Ahlbäck and Simon Dixon. *MSTRE-Net: Multistreaming Acoustic Modeling for Automatic Lyrics Transcription.* ISMIR, 2021, https://doi.org/10.5281/zenodo.5624643.

Devlin et al., 2018. Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2018, https://arxiv.org/abs/1810.04805.

Dosovitskiy et al., 2020. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit and Neil Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* 2020, https://arxiv.org/abs/2010.11929.

Dressler, 01 2011. Karin Dressler. *An Auditory Streaming Approach for Melody Extraction from Polyphonic Music.* Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, pages 19–24, 2011.

Duan et al., oct 2013. Zhiyan Duan, Haotian Fang, Bo Li, Khe Sim and Ye Wang. *The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech.* 2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2013, pages 1–9, 2013.

Défossez, 2022. Alexandre Défossez. *Hybrid Spectrogram and Waveform Source Separation*, 2022, https://arxiv.org/abs/2111.03600.

Franzon, 10 2015. Johan Franzon. *Three dimensions of singability. An approach to subtitled and sung translations. In: Text and Tune. On the Association of Music and Lyrics in Sung Verse. (Eds: Teresa Proto, Paolo Canettieri & Gianluca Valenti). Peter Lang, 2015. ISBN 978-3-0343-1560-9. Pages 333-346.* 2015, https://www.researchgate.net/publication/315596638_Three_dimensions_of_singability_An_approach_to_subtitled_and_sung_translations_In_Text_and_Tune_On_the_Association_of_Music_and_Lyrics_in_Sung_Verse_Eds_Teresa_Proto_Paolo_Canettieri_Gianluca_Valenti_.

Gales et al., 2008. Mark Gales, and Steve Young. *"The application of hidden Markov models in speech recognition.* 2008.

Ghazvininejad et al., jun 2018. Marjan Ghazvininejad, Yejin Choi and Kevin Knight. *Neural Poetry Translation.* Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 2018, https://aclanthology.org/N18-2011.

Guo et al., may 2022. Fenfei Guo, Chen Zhang, Zhirui Zhang, Qixin He, Kejun Zhang, Jun Xie and Jordan Boyd-Graber. *Automatic Song Translation for Tonal Languages.* Findings of the Association for Computational Linguistics: ACL 2022, pages 729–743, 2022, https://aclanthology.org/2022.findings-acl.60.

Gupta et al., 2019. Chitralekha Gupta, Emre Yılmaz and Haizhou Li. *Automatic Lyrics Alignment and Transcription in Polyphonic Music: Does Background Music Help?*, 2019, https://arxiv.org/abs/1909.10200.

Hayes, **1989**. Bruce Hayes. *Compensatory Lengthening in Moraic Phonology.* Linguistic Inquiry, 20, 253–306, 1989, http://www.jstor.org/stable/4178626.

Hochreiter and Schmidhuber, **1997**. S. Hochreiter and J. Schmidhuber. *Long short-term memory.* Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

Hono et al., **2021**. Yukiya Hono, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku and Keiichi Tokuda. *Sinsy: A Deep Neural Network-Based Singing Voice Synthesis System.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29, 2803–2815, 2021, https://arxiv.org/abs/2108.02776.

Hsu and Su, **oct 2021**. Jui-Yang Hsu and Li Su. *VOCANO: A note transcription framework for singing voice in polyphonic music.* ISMIR, 2021, https://doi.org/10.5281/zenodo.5624383.

International Telecommunication Union, **July 2016**. International Telecommunication Union. *ITU-T F.745: Functional requirements for network-based speech-to-speech translation services.* 2016.

Jia et al., **2019**. Ye Jia, Ron J. Weiss, Fadi Biadsy, Wolfgang Macherey, Melvin Johnson, Zhifeng Chen and Yonghui Wu. *Direct speech-to-speech translation with a sequence-to-sequence model.* 2019, http://arxiv.org/abs/1904.06037.

Kenmochi and Ohshita, **2007**. Hideki Kenmochi and Hayato Ohshita. *VOCALOID - commercial singing synthesizer based on sample concatenation.* Proc. Interspeech 2007, pages 4009–4010, 2007.

Kruspe, **2016**. Anna M. Kruspe. *Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing.* International Society for Music Information Retrieval Conference, 2016, https://zenodo.org/doi/10.5281/zenodo.1417552.

Sangeun Kum, Jing-Hua Lin, Li Su and Juhan Nam, *2020*. Sangeun Kum, Jing-Hua Lin, Li Su and Juhan Nam. Semi-supervised learning using teacher-student models for vocal melody extraction. In *Proc. International Society of Music Information Retrieval Conference (ISMIR)*, 2020.

Li et al., **2023**. Chengxi Li, Kai Fan, Jiajun Bu, Boxing Chen, Zhongqiang Huang and Zhi Yu. *Translate the Beauty in Songs: Jointly Learning to Align Melody and Translate Lyrics*, 2023, https://arxiv.org/abs/2303.15705.

Liu et al., **2022**. Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen and Zhou Zhao. *DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism*, 2022, https://arxiv.org/abs/2105.02446.

Lu and Su, **2018**. Wei Tsung Lu and Li Su. *Vocal Melody Extraction with Semantic Segmentation and Audio-symbolic Domain Transfer Learning.* 2018, http://ismir2018.ircam.fr/doc/pdfs/286_Paper.pdf.

McAuliffe et al., **2017**. Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner and Morgan Sonderegger. *Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi.* 2017. doi: 10.21437/Interspeech.2017-1386.

Meseguer-Brocal et al., nov 2018. Gabriel Meseguer-Brocal, Alice Cohen-Hadria and Geoffroy Peeters. *DALI: A Large Dataset of Synchronized Audio, Lyrics and notes, Automatically Created using Teacher-student Machine Learning Paradigm.* ISMIR: Proceedings of the 19th International Society for Music Information Retrieval Conference, 2018, `https://zenodo.org/record/1492443`.

Nakamura et al., 2019. Kazuhiro Nakamura, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku and Keiichi Tokuda. *Singing voice synthesis based on convolutional neural networks*, 2019, `https://arxiv.org/abs/1904.06868`.

Radford et al., 2018. Alec Radford, Karthik Narasimhan, Tim Salimans and Ilya Sutskever. *Improving Language Understanding by Generative Pre-Training.* OpenAI, 2018, `https://openai.com/research/language-unsupervised`.

Radford et al., 2022. Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey and Ilya Sutskever. *Robust Speech Recognition via Large-Scale Weak Supervision.* OpenAI, 2022, `https://arxiv.org/abs/2212.04356`.

Rafii et al., dec 2017. Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis and Rachel Bittner. *The MUSDB18 corpus for music separation*, 2017, `https://doi.org/10.5281/zenodo.1117372`.

Rubenstein et al., 2023. Paul K. Rubenstein, Chulayuth Asawaroengchai, Duc Dung Nguyen, Ankur Bapna, Zalán Borsos, Félix de Chaumont Quitry, Peter Chen, Dalia El Badawy, Wei Han, Eugene Kharitonov, Hannah Muckenhirn, Dirk Padfield, James Qin, Danny Rozenberg, Tara Sainath, Johan Schalkwyk, Matt Sharifi, Michelle Tadmor Ramanovich, Marco Tagliasacchi, Alexandru Tudor, Mihajlo Velimirović, Damien Vincent, Jiahui Yu, Yongqiang Wang, Vicky Zayats, Neil Zeghidour, Yu Zhang, Zhishuai Zhang, Lukas Zilka and Christian Frank. *AudioPaLM: A Large Language Model That Can Speak and Listen*, 2023, `https://arxiv.org/abs/1904.06037`.

Rusu et al., 2016. Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu and Raia Hadsell. *Progressive Neural Networks*, 2016, `https://arxiv.org/abs/1606.04671`.

Salamon and Gomez, 2012. Justin Salamon and Emilia Gomez. *Melody Extraction From Polyphonic Music Signals Using Pitch Contour Characteristics.* IEEE Transactions on Audio, Speech, and Language Processing, 20, 1759–1770, 2012. doi: 10.1109/TASL.2012.2188515.

Schulze-Forster et al., 2021. Kilian Schulze-Forster, Clement S. J. Doire, Gaël Richard and Roland Badeau. *Phoneme Level Lyrics Alignment and Text-Informed Singing Voice Separation.* IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29, 2382–2395, 2021, `https://ieeexplore.ieee.org/document/9463691`.

Seamless Communication et al., 2023. Seamless Communication, Loïc Barrault, Yu-An Chung, Mariano Cora Meglioli, David Dale, Ning Dong, Paul-Ambroise Duquenne, Hady Elsahar, Hongyu Gong, Kevin Heffernan, John Hoffman, Christopher Klaiber, Pengwei Li, Daniel Licht, Jean Maillard, Alice Rakotoarison, Kaushik Ram Sadagopan, Guillaume Wenzek, Ethan Ye, Bapi Akula, Peng-Jen Chen, Naji El Hachem, Brian Ellis, Gabriel Mejia Gonzalez, Justin Haaheim, Prangthip Hansanti, Russ Howes, Bernie Huang, Min-Jae Hwang, Hirofumi Inaguma, Somya Jain, Elahe Kalbassi, Amanda Kallet, Ilia Kulikov, Janice Lam, Daniel Li, Xutai Ma, Ruslan Mavlyutov, Benjamin Peloquin, Mohamed Ramadan, Abinesh Ramakrishnan, Anna Sun, Kevin Tran, Tuan Tran, Igor Tufanov, Vish Vogeti, Carleigh Wood, Yilin Yang, Bokai Yu, Pierre Andrews, Can Balioglu, Marta R. Costa-jussà, Onur Celebi, Maha Elbayad, Cynthia Gao, Francisco Guzmán, Justine Kao, Ann Lee, Alexandre Mourachko, Juan Pino, Sravya Popuri, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, Paden Tomasello, Changhan Wang, Jeff Wang and Skyler Wang. *SeamlessM4T: Massively Multilingual & Multimodal Machine Translation*, 2023, https://arxiv.org/abs/2308.11596.

Tang et al., 2020. Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu and Angela Fan. *Multilingual Translation with Extensible Multilingual Pretraining and Finetuning*, 2020, https://arxiv.org/abs/2008.00401.

Team et al., 2022. NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk and Jeff Wang. *No Language Left Behind: Scaling Human-Centered Machine Translation*, 2022, https://arxiv.org/abs/2207.04672.

Touvron et al., 2023. Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave and Guillaume Lample. *LLaMA: Open and Efficient Foundation Language Models*, 2023, https://arxiv.org/abs/2302.13971.

Vaswani et al., 2017. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin. *Attention Is All You Need*, 2017, https://arxiv.org/abs/1706.03762.

Wu et al., 2021. Yu-Te Wu, Yin-Jyun Luo, Tsung-Ping Chen, I-Chieh Wei, Jui-Yang Hsu, Yi-Chin Chuang and Li Su. *Omnizart: A General Toolbox for Automatic Music Transcription*, 2021, https://arxiv.org/abs/2106.00497.

Yamamoto et al., 2022. Ryuichi Yamamoto, Reo Yoneyama and Tomoki Toda. *NNSVS: A Neural Network-Based Singing Voice Synthesis Toolkit*, 2022, https://arxiv.org/abs/2210.15987.

Yu et al., feb 2019. Yi Yu, Abhishek Srivastava and Simon Canales. *Conditional LSTM-GAN for Melody Generation from Lyrics.* ACM Transactions on Multimedia Computing, Communications, and Applications, 17, 1–20, 2019, https://arxiv.org/abs/1908.05551.