Creating Heterogenous Transcription of English and Japanese on a Multilingual Audio File

Yuika Sun
Los Altos High School, CA 94022, USA
yuhyeung@gmail.com

Abstract

Code-switching (CS) is the switching between multiple languages in speech. Current literature on code-switching transcription technology has been successful in translating the CS transcripts into one language but there yet to exist satisfactory technology to accurately transcribe one single speaker's CS audio into CS transcripts. Current speech-to-text (STT) technology, although produces accurate transcriptions, is limited to monolingual sentences. In this paper, the focus is on an unexplored area: expanding STT to English and Japanese CS language transcription without utilizing translation software.

1 Introduction

Bilingualism, the ability to speak two languages, is prevalent—more than half of the world is bilingual. [1] This leads many to code-switch, a phenomenon where one speaks multiple languages interchangeably in spoken sentences.

Accurately transcribing code-switched audio is difficult, as not many software can successfully transcribe a multilingual audio file into heterogenous text correctly.

A program using Heterogenous Language Detection and Re-transcription (HLDR)[1] for multilingual audio files with Japanese and English code-switching was developed to address existing gaps in functionality of current STT software. There exists some audio-to-text transcription software that resolves constituents of this problem—the task of transcribing one language at a time.

For instance, Adobe Premiere Pro's automatic transcription feature immediately auto-transcribes audio. However, when the user code-switches from English and Japanese, the closed captions generate incomprehensible characters. A similar phenomenon is observed in OpenAI's Whisper. After the first 30 seconds of the audio file, the transcription outputs *romaji* for Japanese characters if the speaker code-switches from English to Japanese. If a language like Japanese is specified using a parameter from the beginning, it will completely translate the English sentence into Japanese, regardless of whether the speaker has code-switched or not. However, if the speaker code-switches within the first 30 seconds of the audio file, the software accurately transcribes the Japanese portion of the code-switched audio with the appropriate *katakana*, *kanji*, and *hiragana*, and the English portion of the code-switched audio with correct English words.

Another technology that transcribes spoken language is speech diarization, which involves determining "who spoke when", distinguishing between speech and non-speech sounds, and marking speaker changes within the identified speech. [2] However, speech diarization could not be used for this specific issue of recognizing code-switched language spoken by one single person. Speech diarization requires multiple distinct acoustic features to identify the two borders between CS and non-CS speech. This paper focuses on accurately transcribing CS from one speaker with uniform acoustic features in their speech, making speech diarization an unsuitable technology.

Multilingual transcription technology faces challenges in accuracy due to existing technologies' sole focus on monolingual transcription. [3] Creating STT software with the capability of handling code-switching between English and Japanese will make video subtitles more accessible to more people, reducing the time for video editing and transcription for those who

[1] https://github.com/LearnML-Me/HLDR/

code-switch between English and Japanese in their videos. In addition, creating a STT will open more doors in the world of STT for East Asian languages. For instance, Japanese, Chinese, Korean, and English have mutually borrowed loanwords, allowing for increased efficiency in creating these language models.

2 Method

In the realm of linguistic analysis, a novel concept, "noise" is introduced within the context of code-switching for this paper. Unlike the conventional understanding of noise as auditory interference, in this context, it denotes the unintended linguistic distortions that arise during the transcription of code-switched conversations. In audio files with code-switched language, noise emerges as an inadvertent byproduct. This distortion materializes when transcription software encounters code-switching instances, unable to distinguish between the original language and the interjected foreign language. Consequently, the transcribed output consists of unfamiliar and seemingly nonsensical words in the original language, illustrating the challenges inherent in accurately capturing the complexity of code-switched language. This paper aims to eliminate “noise” from transcripts that deal with code-switched language. “noise” is taken out by a distinction of where to start the and stop a transcription in one language and append the transcription in the second language. To accomplish this task, two English language models and one Japanese model are utilized to find the noise relationship between them and merge them into an English and Japanese heterogenous final transcription. Two English transcriptions are used to find the commas between each phrase. To detect the noise, it is understood that the English translation has a relatively low word error rate (WER). WER is the ratio of substitutions (S), deletions (D), and insertions (I) of words in a transcript to the overall number of spoken or intended words (N_1).

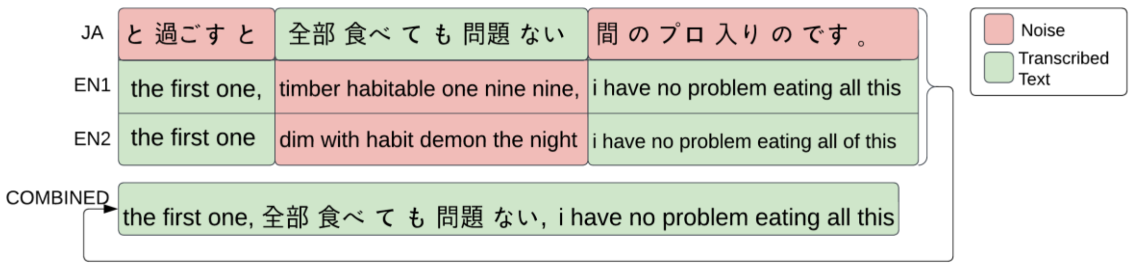

JA
と 過ごす と
全部 食べ て も 問題 ない
間 の プロ 入り の です 。
EN1
the first one,
timber habitable one nine nine,
i have no problem eating all this
EN2
the first one
dim with habit demon the night
i have no problem eating all of this
COMBINED
the first one, 全部 食べ て も 問題 ない, i have no problem eating all this
Noise
Transcribed Text

Figure 1

$$WER = (S + D + I)/N_1$$

Figure 2

[4] Considering this, the two English transcriptions are highly likely to be very similar due to its WER. Therefore, a phrase that has a high WER when comparing the two English transcription files will be detected as “noise”. In HLDR approach, proper capitalization is ignored. Similarly, the same HLDR principle may be used with the Japanese STT software, but an appropriate Japanese STT software other than dictation-kit was not found with a satisfactory WER as of the writing of this paper.

Materials: Julius [5], dictation-kit [6], DeepSpeech [7], running on Ubuntu 22.04 LTS, an audio recording containing both English and Japanese sentences.

3 Results

In this trial a heterogenous code-switched transcription was tested.

Original intended audio: “the first one, 全部 食べ て も 問題 ない, I have no problem eating all of this.”

Phase I:

This phase is to get the raw speech-to-text (STT) text file.

Transcriptions were made through neural networks. Types include Deep Learning Neural Networks (DNN), Recurrent Neural Network (RNN), and Convolutional neural network (CNN).

A difference between the Julius model and DeepSpeech model transcripts Julius transcribed it as “I have no problem eating all this” while DeepSpeech transcribed the audio as “I have no problem eating all of this”. To mitigate this concern, WER was used; there was one word that caused the discrepancy: “of”, with a WER of 12% when compared to the full phrase. When compared to WER in other contexts outside of this paper, such as the WER for Wit (25.87%), a free speech-to-text software. [8] HLDR was used on the third portions of EN1 and EN2 respectively to produce the final output, which is “I have no problem eating all this”. To intelligently choose the most accurate final output based on nuances and semantics of each word using advanced NLP libraries like spaCy and MeCab is left to future research as improvement.

The existing ASR language models were leveraged to process the audio file. On 3 platforms: dictation-kit, English and DeepSpeech, after running the processor binary by specifying a language model, the essential outcome of the recognized speech text was extracted from the result with the combination of Linux commands “grep” and “sed”, the preliminary file JA.txt, EN1.txt, EN2.txt are created respectively. The content is like the following without quotation mark:

- Japanese STT: dictation-kit (JA.txt)

“

と 過ごす と 、 全部 食べ て も 問題 ない 、 間 の プロ 入り の です 。

”

- English STT: Julius (EN1.txt)

“

the first one, timber habitable one nine nine, I have no problem eating all this

”

- English STT: DeepSpeech (EN2.txt)

“

the first one dim with habit demon the knight I have no problem eating all of this

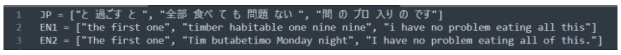
”

Phase II

This phase is the key portion of this article. A Python script is used to process 3 raw transcription files. The basic functionality is like the following:

- Read 3 files and store them as 3 variables
- Tokenize the strings by converting to string lists
- Scan and record the location of comma “,”
- Normalize the strings by removing comma(s)
- Use diff Python library to get the difference of EN1 and EN2 tokenized strings
- Pick up the common words into a new list and identify the English noise section.
- Skip the same amount of noise from the Japanese result and inject the correct Japanese sentence into the English noise section.
- Restore the commas in their recorded locations (from step 3) in the combined string.
- Combine the string list to get the combined string.

If I section out the outputs from dictation-kit, DeepSpeech, and Julius into respective Python lists, I can compare their phrases with each other. (Figure 3)



```
JP = ["と 過ごす と ", "全部 食べ て も 問題 ない ", "間 の プロ 入り の です"]
EN1 = ["the first one", "timber habitable one nine nine", "i have no problem eating all this"]
EN2 = ["The first one", "Tim butabetimo Monday night", "I have no problem eating all of this."]
```

Figure 3

Each list has three items. Focusing on EN1 and EN2, each word inside the first and third items are nearly identical. Conversely, all the words inside of the second item in the list are different. The second item is the portion in which the Japanese was being spoken. This scenario is visualized in Figure 1. In this case, I utilize this pattern in hopes of writing a code that substitutes the inconsistent portion of item 2 from transcription. The code that could implement this is referenced in Figure 3.

4 Discussion

There exist several limitations to the method used in this research paper. First, because this method can be regarded as meta transcription, its quality is variable based on the underlying transcription software’s quality and language model training hours. Its accuracy also depends on the field of the terminology between the spoken words/sentences and the trained model. The WER must be at a minimum.

In the course of time, this problem will gradually be improved and solved, but ultimately it needs a different approach to address the issue at its foundational level. In addition, the quality of the output when utilizing this paper's approach varies heavily on the spoken words and sentences. Correct Japanese punctuation in the transcription such as commas and periods are inconsistent, which will cause the script to have difficulty finding the border of the correct Japanese words and noise coming from English. This can be attributed to the fact that only one Japanese language model has been identified that meets the specified requirements, which include accurate punctuation and a fully functional translation encompassing kanji, katakana, and hiragana. HLDR could be further improved by understanding instances when Japanese kanji is written as hiragana and vice versa, and add the functionality of allowing the user to choose between options.

There also exist several directions for future research on this topic. First is expanding the English transcription service from 2 to many, to get the best accuracy of English transcription. In addition, expanding the Japanese transcription service from 1 to many is imperative to get the most accurate insertion of Japanese code-switched transcription. Furthermore, expanding this method of accurate transcription of code-switched speech requires software libraries spaCy, NLTK, and MeCab for advanced NLP tasks such as semantic analysis. The current strategy in this paper can only transcribe code-switched phrases, not individual words. Semantic analysis will allow for accurate insertion of specific words in text by considering the context of the surrounding sentence. Another limitation is that of the scope of the transcription ability. This paper's approach is adequate at transcribing code-switched speech with one sentence is English, and one is Japanese, etc. However, tasks such as transcribing a spoken sentence containing mixed words instead of mixed phrases between English and Japanese is beyond the capabilities of the software. Finding a solution to this scenario would require deep analysis of the transcription using advanced NLP processing software such as spaCy or NLTK.

Another direction is to statistically adopt the best translation of Japanese. The method would be more diverse if it included more languages, such as recognizing, parsing, and transcribing Chinese and English code-switched sentences. Due to the loanwords that exist between Japanese and Chinese, this method would be most practical. By extension, it is worth considering engineering a program that can process low-resource languages.

5 Conclusion

In this paper, a strategy is introduced to merge English and Japanese transcription software together using two English and one Japanese transcription software and a Python script to find appropriate words to merge them together. Any combination of STT software, whether commercial or open source, is the user's choice and may be used in the HLDR method, opening the door to meta-transcription. Similar to a technology term, Redundant Array of Independent Disks (RAID), which use multiple disks and combine them together to get a better result in terms of speed and availability, [9] a higher number of STT used in the HLDR method is predicted to increase the accuracy and lower the cost of the combined result.

References

[1] Ana Inés Ansaldo, Karine Marcotte, Lilian Scherer, and Gaelle Raboyeau. Language therapy and bilingual aphasia: Clinical implications of psycholinguistic and neuroimaging research. In Journal of Neurolinguistics, Vol. 21, No.6, pp. 1, 2008.

[2] –. ScienceDirect. (Online) (Access Date: 1/3/24.) https://www.sciencedirect.com/topics/computer-science/speaker-diarization.

[3] –. Machine Learning Research. (Online) (Access Date: 1/3/24.) https://machinelearning.apple.com/research/towards-real-world.

[4] –. ResearchGate. (Online) (Access Date: 1/3/24.) https://www.researchgate.net/publication/221488965_An_Empirical_Analysis_of_Word_Error_Rate_and_Keyword_Error_Rate.

[5] A. Lee and T. Kawahara: Julius v4.5 (Online) (Access Date: 1/3/24) https://doi.org/10.5281/zenodo.2530395.

[6] –. Github dictation-kit. (Online) (Access Date: 1/3/24) https://github.com/julius-speech/dictation-kit.

[7] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng Deep Speech: Scaling up end-to-end speech recognition arXiv preprint arXiv:1412.5567v2, 2014.

[8] –. National Library of Medicine. (Online) (Access Date: 1/6/24.) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7256403/.

[9] Christian Zoubek, Sabine Seufert, and Andreas Dewald. Generic RAID reassembly using block-level entropy. In Digital Investigation, Vol. 16, pp. 1, 2016.