JaVaCE

Japanese Vocabulary and Comprehension Estimator

Shane Davis
University of Central Florida
Orlando, FL, USA
sdavis175@knights.ucf.edu

ABSTRACT

Language learning is a long and difficult task, especially when going from languages that have little similarity such as English to Japanese. This paper proposes two tools, the first being a vocabulary recognizer that highlights the user's known vocabulary in a sentence. The second tool is a machine learning model that estimates the user's comprehension level of a sentence. Both tools are designed for Japanese, however other languages can be applied without a lot of modification.

1 Introduction

1.1 Basics of Japanese

Japanese is one of the most complex languages to learn, being noted as a Category V language (highest level) which requires roughly 2200 hours to become fluent, with an additional increase in difficulty from western native language speakers [1].

Japanese has three main alphabets: Hiragana (ひらがな・平仮名), which are the main alphabetical sounds for native words, and individual characters are known as kana. Katakana (カタカナ・片仮名) which are the same sounds as Hiragana with different written kana and are used for loan words from other languages (ペン is read as pen and means pen). Finally, the most difficult of the three alphabets, Kanji (漢字) which are borrowed Chinese characters that make up a majority of the vocabulary. Individual Kanji has readings, which are the way the Kanji is spoken, which is built out of the same sounds as the other two alphabets. Every Kanji can be decomposed into its Hiragana reading counterparts; however, Kanji can have multiple readings which can make memorization very difficult because the reading depends on the specific vocabulary word. To help with this, Furigana (振り仮名 (ふ が な)) will be used, which are the Hiragana that appear above the Kanji, signifying the way the Kanji is supposed to be read. Unfortunately, Furigana is not used much outside of learning material which can make interacting with native content extremely difficult for beginners that are learning the language.

Additionally, Japanese uses grammar particles, which are markers that are made up of one or two kana that are used only for grammar functionality to describe various things in a sentence. For example, the が particle is the subject marking particle, and is used to mark the subject of a sentence when put after a word.

Another important part of the Japanese sentence structure is the lack of spaces between words. This makes things very difficult as a beginner as you do not know where words start and begin, and additionally with the grammar particles they are hard to pick out since the same kana that are in the grammar particles are also used in the vocabulary. With more experience and knowledge of vocabulary and grammar, this process becomes more natural. However, this means that for word separation in sentences a tokenizer is required to split the words.

Similar to western languages, Japanese has tenses in their vocabulary, however it is more detailed because of the various polite and casual forms of the same word, which is called inflections. For example, in English, Figure 1 shows the various inflections for the verb "To Go". It is important to note that there are no real concrete rules for forming these types of tenses in English, since there would be so many edge cases for these rules.

	Affirmative	Negative
Past	Gone	Not Gone
Non-Past	Going	Not Going
Potential	To Go	Not To Go

Figure 1: "To Go" English Inflections

Figure 2 shows the equivalent Japanese inflections for the verb "行く", which means "to go". Note: that list is not comprehensive, there are more possible inflections.

	Affirmative	Negative
Non-past	行く	行かない
Non-past, polite	行きます	行きません
Past	行った	行かなかった
Past, polite	行きました	行きませんでした
Te-form	行って	行かなくて
Potential	行ける	行けない
Passive	行かれる	行かれない
Causative	行かせる	行かせない

Causative Passive	行かせられる	行かせられない
Imperative	行け	行くな

Figure 2: "行く" (to go) Japanese Inflections [2]

Even though there are much more inflections in Japanese compared to English, the rules for creating these inflections are much more concrete with very little exceptions. Also, the base part of the word (which in Figure 2 is the Kanji "行") will always remain the same in the inflections, so for beginners that do not know all the inflections they can still get a general idea of what the word means as long as they know the base word.

1.2 Language Learning

For learning a new language, it has been recommended to take an immersion learning approach compared to a traditional classroom/textbook approach [3]. Immersion learning for language study is the process of immersing only in the target language's native content (meaning no English/native language subtitles), and immersing means to listen, read, or watch content. The backing theory behind immersion learning is that it is meant to replicate how we all learned our first language as a child - through immersion by being surrounded by the language. Additionally, immersion learning can be done for free - there is endless amount of free content in any language on the internet that natives produce such as YouTube videos or blog posts, compared to traditional learning where you need to pay money for textbooks and classes. With enough exposure to the target language, you will naturally pick up the language in its common form (not some textbook form of the language with uncommon phrases). However, you need to be able to tolerate ambiguity, meaning that you need to be okay with not understanding everything that you are immersing yourself in. There are various approaches to immersion learning, one of the most common ones is Refold [4]. Refold's immersion learning approach for language learning consists of two daily processes: vocabulary and grammar study and immersion.
The vocabulary and grammar study is typically done through SRS (Spaced Repetition System) flashcard programs such as Anki or JPDB, which each flashcard teaches a specific word or grammar point with the definition on the back [5] [6].

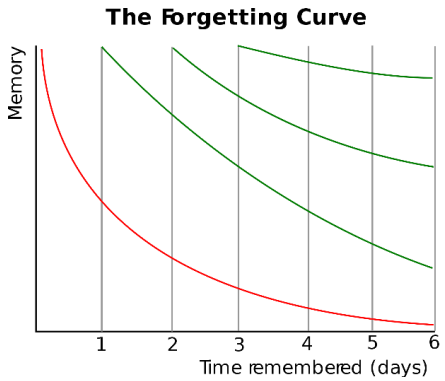

Figure 3: Forgetting Curve that SRS's attempt to predict [7]

In Figure 3, this is an estimated graph of how well you can recall information when learning, with the additional curves going down as you continue to forget it. The goal for SRS systems is to predict this graph based on the user and show the flashcard when the user is predicted to forget it to strength the memory.
For daily immersion, Refold breaks down immersion into two types of active immersion: free-flow immersion and intensive immersion. With free-flow immersion, you are supposed to immerse yourself with little to no dictionary look ups. This is where tolerating ambiguity is important, you need to allow yourself to experience the language with very little interrupts as this is where you acquire the language the most [8]. With intensive immersion, this is where you are supposed to actively attempt to understand everything you come across, and anything you do not understand you look up. This process is important to build up your complex understanding of the language to understand the nuances between varying aspects as you go in depth in your understanding [9]. There is also passive immersion, which is where you aren't actively paying attention to the language and listening to things such as audiobooks or the radio in the target language while doing other things [8]. Passive immersion is not as effective for language learning however it is good to get used to the various sounds and hearing the language, especially as a beginner where it is difficult to even distinguish words in a sentence. More details about Refold and its process is on their website [4].

2 Problem Statement

With immersion language learning in Japanese, there are two main problems that I have faced: recognizing known vocabulary outside of the SRS and choosing comprehensible content to immerse in.
It is one thing to be able to recognize and recall the meaning and reading of vocabulary in the SRS programs, but outside of it can be difficult due to all of the complexity of the Japanese language that was covered in the Introduction section. I would like to design a tool to aid in this process, it would take input sentences and a known vocabulary database as input and output the highlighted known words in a sentence.
On top of that, it is difficult to choose content to immerse yourself in because you do not know how well you can comprehend a piece of content until you interact with it. To solve this, I would like to design a machine learning model that takes in a sentence and a known vocabulary database as input and output the estimated comprehension level from Easy, Medium, and Hard. That way you could pass in a full piece of content into the model and take the average difficulty score to determine how well you would comprehend the material.

3 Related Work

3.1 Migaku

Migaku is a subscription service that provides various tools for language learning [10]. One of the tools that they offer is a one-step flashcard creator for SRS programs. During immersion, if you would like to create a custom flashcard with the target word that you would like to know on the front and on the back the definition, you just press the upload button and Migaku handles the rest.

Migaku also offers a pop-up dictionary, which is a quick way to look up a specific word without leaving the website. These are very useful tools, however there are plenty of free alternatives such as Yomichan that accomplish the same tasks, albeit with a bit more work to set up [11]. The most important tool that Migaku offers is their vocabulary recognizer, which changes the colors of the words on screen depending on how well you know the word based on stats imported from the SRS programs. However, the tool is only available on specific websites such as Netflix and YouTube because of their direct modification to the text. This is what I want to replicate with the vocabulary recognizer portion of my program, however I want it to be usable based on any source, and I do not like the subscription service aspect of Migaku.

3.2 Iago

Iago is another subscription service that provides similar tools to Migaku [12]. They provide a pop-up dictionary for vocabulary word lookups during immersion. They also highlight known words in subtitles; however, they use their own custom SRS implementation for their database. Additionally, their SRS system is much more interactive with things such as various styles of quizzes to reinforce the vocabulary you have been learning, which is much more in-depth compared to other SRS programs that are just flashcards with a special algorithm. Iago still faces the shortcoming of only being available on specific websites, however the service is still new, and the developer intends on expanding out to more websites in the future.

3.3 JPDB

JPDB is a free website (but with some additional paid features) that offers both an SRS program and a difficulty rating estimation [6]. The SRS program is unique in that it uses machine learning to try to predict the forgetting curve compared to other common SRS programs such as Anki that just uses basic metrics [13]. As a result, many users believe that JPDB's SRS program is more effective for memorization as it does a better job at predicting the forgetting curve compared to other SRS programs. JPDB also allows you to import flashcard decks from Anki so that you can use the better SRS algorithm with the much more common and widely available Anki flashcard decks. JPDB also stores lots of content such as books, shows, and video games with all of their scripts so that you can import the vocabulary into the SRS program to learn the vocabulary in the daily SRS study in addition to the daily immersion study. On top of that, JPDB provides a generalized difficulty estimation using machine learning on a scale from 1 to 10, with 1 being the easier and 10 being the hardest, of how difficult a piece of content is [14]. They did not provide me with any details on the model or their sources of training data unfortunately. Also, the difficulty ratings are not super accurate, I have had better comprehension with a 4/10 difficulty rating than with a 1/10 difficulty rating, and this is a common problem that has been reported by numerous users. This is similar to what I want to implement with my comprehension estimator portion of the program, however I want it to be any sentence the user provides and also more centered towards the user's interpretation of comprehension, not some general crowd-sourced difficulty estimation. A new (and unfortunately inaccessible tool to me) is their public API, which uses their custom parser to parse sentences into individual words and obtain the base-dictionary form of the words. Current python implementations of tokenizers all have various problems which are further highlighted in the Evaluation & Discussion section. JPDB also offers a similar vocabulary recognition tool to Iago and Migaku, however it is a paid service and there is not much information publicly available on it.

4 Technique

4.1 Known Word Database

For both tools, a known word database is required for the tools to be customized to the user. The database can be generated from two sources, an import from Anki (using `anki_sync.py`) and a manual word input (using `manual_add_vocab.py`). Both sources are stored in their own individual .txt file, with each word being on a separate line. The `vocab_recognizer.py` provides a class which handles loading all of the words into their dictionary form (inflections removed) using the SudachiPy library to perform the tokenization and conversion, which will be used for later [15].

4.2 OCR Input

Originally, I was intending on using OCR libraries such as Python-tesseract and Manga OCR for extracting the sentences [16] [17]. However, from my initial testing I found that the OCR results were very poor, both of which having their own issues which can be seen in the following figures. The output would have then been drawing bounding boxes around the known words for the vocabulary recognizer and a specific color to represent the difficulty level of the sentence for the comprehension estimator.

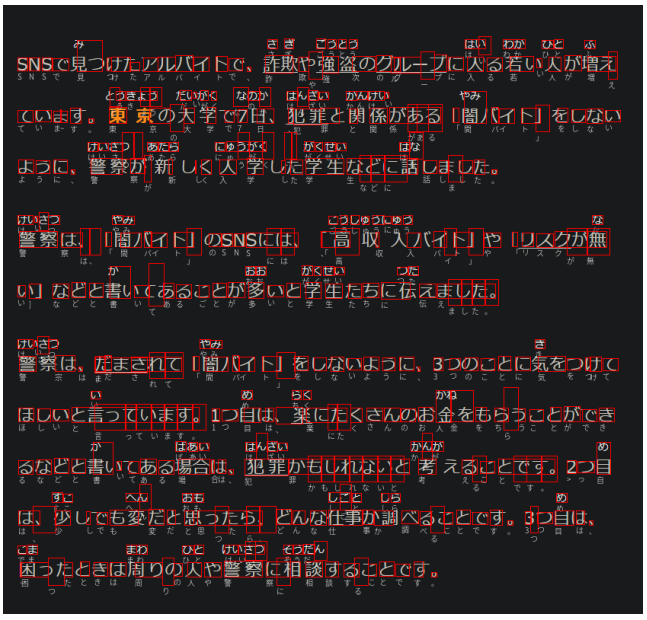

Figure 4: Tesseract OCR on NHK article [16] [18]

In Figure 4, the Tesseract OCR gave very poor bounding boxes, sometimes highlighting empty spaces as characters. Additionally, it would pick up on the Furigana which would add additional unnecessary characters to the output, since the Furigana is only there to help the reader to know how the Kanji are read and is not required to do any parsing.

そのため、これからのお客様においていただきますのですが、いますが実際に実行する方が多いのでしょうか。ということです。これは、このようになりませんでしたが、ことが、そうですから、いいいけれですね。こと思っていいましょことは、それば、ころので、そこればからこのではないけないまではころ、そのよね。それで、こうないいたした。

Figure 5: Manga OCR on NHK article [17] [18]

In Figure 5, the results from Manga OCR were even worse (only the output string could be generated, no bounding boxes for the characters). The output text was not even close to the original text, it was hallucinating various actual words and phrases that were never there. This was highly unexpected; Manga OCR was specifically developed for Japanese text and to specifically ignore Furigana, so seeing these poor results on very clear input text was very strange. I was able to reproduce the results in the Manga OCR GitHub, where they had some sample pictures along with the intended results which did work, so I know it has nothing to do with my environment or the way I am running it.

4.3 Clipboard Input

Since the OCR Input results were so poor, I decided to go with a clipboard hook as my input to the program instead, using pyperclip as the Python library for it [19]. While it isn't as convenient as an OCR solution would be, the input will be 100% accurate to the sentences the user would like to parse. Also, since I do not have access to the bounding boxes anymore, the output will instead be in the console of the running program, where the known words will be colored in green (using the colorama Python library) and the comprehension of the sentence will be an additional difficulty output per sentence (Easy, Medium, Hard) [20].

For separating the input into sentences, I first ran a replacement to remove any newlines, tabs, and spaces, afterwards I split by "。" (Japanese period character). Each individual sentence is then ran on the Vocabulary Recognizer, and if enabled then it will also be ran on the Comprehension Estimator.

4.4 Vocabulary Recognizer

The vocabulary recognizer portion uses a simple maximization algorithm to match the longest sequence of known vocabulary tokens to the longest sequence in the input sentence. The input sentence is first tokenized using the SudachiPy tokenizer to split the sentence into tokens and convert them into dictionary form, so that each token can directly match with the original stored known vocabulary version of it. There are various additional parameters such as automatically recognizing all Katakana loan words, English words, and grammar particles; adding the Kanji readings; and generating an array of known vs unknown tokens (which will be used in the Comprehension Estimator). Any known words will be surrounded by a marker to indicate to the colorama library to display the word as green, however this marker is a parameter and can be customizable depending on different use cases.

4.5 Comprehension Estimator

4.5.1 Dataset

A dataset of 50,000 sentences from the original Japanese sentences from the やさしい日本語 (translation: Easy Japanese) paper will be used along with hand labeling the difficulty levels per sentence [21]. The paper provided these 50,000 original Japanese sentences with the intent to simplify Japanese with corresponding simpler sentences, however I will just be using the original sentences and scoring them based on my perception of difficulty. An Easy sentence is a sentence that is completely comprehensible, a Medium sentence is a sentence with a few words that are unknown and therefore not fully comprehensible, and a Hard sentence is a sentence with little to no known words and therefore fully incomprehensible. For grading, I designed a grading program (`grade_sentences.py`) that would display each sentence, along with using the Vocabulary Recognizer to show which vocabulary I am supposed to know, and I would input the grade per sentence. I graded roughly 2,000 sentences, with the following breakdown per difficulty rating:

Easy: 359
Medium: 1440
Hard: 201

Additionally, I scraped sentences from the NHK news website, which is a Japanese news website, to diversify my dataset more [22]. I scraped roughly 700 sentences with the following breakdown per difficulty rating:

Easy: 14
Medium: 236
Hard: 236

Clearly, the dataset has a large bias towards the number of medium difficulty sentences. This makes sense, I am still a quite new to learning Japanese so a majority of the sentences will not be too easy or too hard for me to understand. To prevent the machine learning model from overfitting on the Medium class, I chose to randomly remove 1400 Medium sentences from my dataset. I randomized and stratified the dataset and used 20% of the data for validation in splitting the reduced dataset.

The dataset was then preprocessed using the BERT Japanese tokenizer that Hugging Face offers in their `transformers` Python library, along with using the pre-trained weights from `cl-tohoku/bert-base-japanese` [23] [24]. Additionally, for each sentence a custom known embedding was generated to associate each token with a Known/Unknown weight (Known was 10, Unknown was -10), which will be used later in the model. The difficulty ratings were converted to a one-hot encoding, and the dataset was converted into a tensorflow `Dataset` class [25].

4.5.2 Machine Learning Model

The dataset will then be used for transfer learning on the pretrained BERT base Japanese model (same model name as the tokenizer), with an additional classification network afterward [8], using the `keras` library [26]. The sentences will first pass through the BERT network, producing an embedding of 768 dimensions per sentence. The custom known word embedding is concatenated onto that embedding, giving 769 dimensions per sentence. The intent of the custom embedding is to push the model to learn that the more words that are known should directly correlate with how easy the sentence is to comprehend. The following classification network starts with two Bidirectional-LSTMs then a Global Average Pooling into a series of Dense layers, most of which contains Dropout layers, which eventually produces the final Dense layer with 3 outputs that are ran on softmax to determine the most-likely difficulty score.
For training the model, I split the training into 4 experiments: train both BERT model and use the Known Embeddings, only train BERT model and do not concatenate Known Embeddings, freeze BERT and only use Known Embeddings, and finally freeze BERT and do not concatenate Known Embeddings. This will test the effectiveness of my custom known word embedding along with seeing how freezing the pre-trained BERT model will still perform on the dataset. I ran each of these experiments on 100 epochs with the Adam optimizer set at 1e-6 learning rate with the rest of the parameters default and my dataset set to a batch size of 32.

5 Evaluation & Discussion

5.1 Vocabulary Recognizer

For evaluating the Vocabulary Recognizer, the main focus is to see how well it would tokenize the sentence and recognize the known words in the sentence.

防衛 省 は 13 日 朝 、 北朝鮮 から 弾道 ミサイル の 可能 性 の ある もの が 発射 さ れ た と 発表 し まし た 。

防衛省 は 13 日朝、 北朝鮮 から 弾道ミサイル の 可能性 の あるもの が 発射 された と 発表 しました 。

Figure 6: First sentence is the SudachiPy tokenizer output for splitting words (each word shown by a space), second sentence is the correct word split using a sentence from an NHK article [18]

In Figure 6 there are many clear problems with the SudachiPy tokenizer that can be seen, it splits a lot of words into separate tokens. Some of them are understandable, however it also splits up words such as "された" into individual tokens, when in reality it is the verb "される" in its past inflection form. Errors like these contribute to the inaccuracies in the Vocabulary Recognizer output since it will not be properly converted to the dictionary form and be recognized as a known word.

防衛(ボウエイ)省(ショウ)は13日(ニチ)朝(アサ)、北朝鮮(キタチョウセン)から弾道(ダンドウ)
ミサイルの可能性(カノウセイ)のあるものが発射(ハッシャ)されたと発表(ハッピョウ)しました。

Figure 7: Sample output that highlights the known words from the same sentence as Figure 6, along with showing the readings for the Kanji

In Figure 7, we can see those tokenization errors propagated into the actual recognition results. Another key problem is that if the known words are stored in their typically written in Kanji such as "物", it will not be recognized in its equivalent Hiragana form "もの", which can be misleading as to how many vocabulary words are actually known in the sentence and can be seen in Figure 7.

5.2 Comprehension Estimator

For evaluating how well my model trained on the dataset, I used the accuracy, precision, and recall metrics along with a confusion matrix to see a better breakdown of how the model was predicting each of the classes given the true classes on the validation dataset.

5.2.1 BERT and Known Embeddings

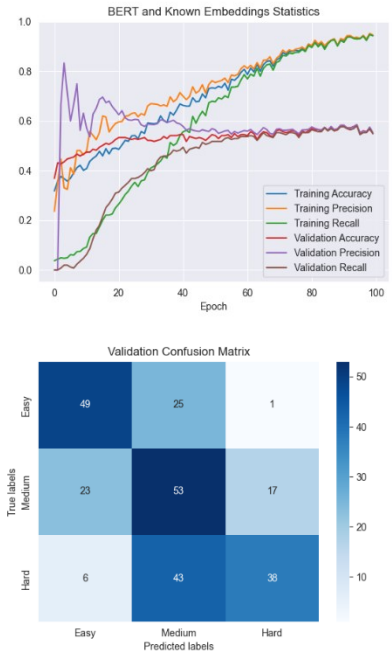

Figure 8: Model results on BERT and Known Embeddings training experiment

In Figure 8 the main results are shown with both BERT and Known Embeddings being trained on. The results were alright, the model predicted Medium for a lot of the Medium and Hard labels. However, there were very few incorrectly predicted Easy to Hard (and vice versa) predictions.

5.2.2 BERT Only

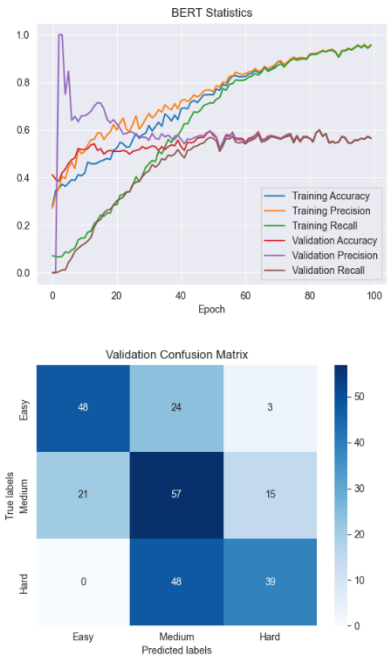

Figure 9: Model results on BERT only training experiment

In Figure 9 the ablation results are shown with only training on the BERT model and not integrating the Known Embeddings. The model performed roughly the same as the BERT and Known Embeddings experiment. Even with the results still being roughly the same, I believe that the Known Embeddings is worth it to the model so that for the future when the user increases their vocabulary size the model will hopefully still be able to make the distinction that it is easier to comprehend with the words being marked as Known/Unknown.

5.2.3 BERT Frozen and Known Embeddings Only

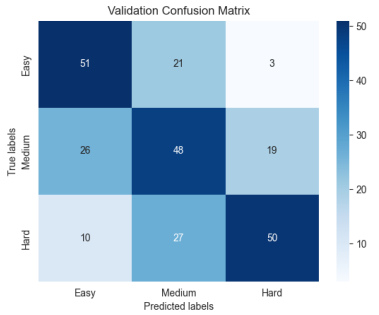

Figure 10: Model results on Known Embeddings only training experiment

In Figure 10 the ablation results are shown with freezing the BERT model and integrating the Known Embeddings. The results were significantly worse compared to the previous two experiments, indicating that fine-tuning the BERT model is required to give good results.

5.2.4 BERT Frozen

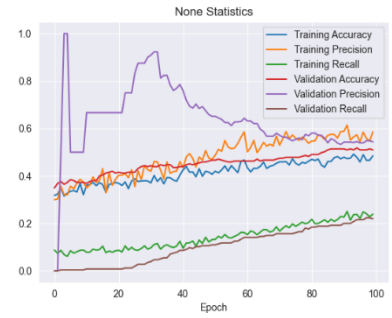

Figure 11: Model results on BERT Frozen training experiment

In Figure 11 the ablation results are shown with freezing the BERT model and not integrating the Known Embeddings. This was the worse performing model, which was to be expected as seen before training the BERT model helps the results and the addition of removing the custom Known Embeddings makes it even harder for the model to even fit to the training data.

5.3 Qualitive Results

For using both the Vocabulary Recognizer and the Comprehension Estimator, both features are integrated into one clipboard hook

program (`javace.py`). While running it, the workflow was a bit awkward to use as I would copy text I was interested in analyzing, but then I would need to look over to see the output results. In terms of responsiveness, running both portions takes less than a second which made for smooth look ups. Overall, the majority of my known vocabulary was accurately highlighted and displayed, and the majority of the comprehension estimations I felt were accurate.

6 Conclusion

JaVaCE is a 2-part program I designed to help me with learning Japanese, one half is a Vocabulary Recognizer that highlights my known vocabulary in text, and the other half is a Comprehension Estimator that tries to predict my comprehension level of a sentence on a scale of Easy, Medium, and Hard. Overall, the results were decent, but not perfect. I believe the tokenizer still needs more work, and with JPDB's new public API tokenizer being released soon that could be integrated into the project to improve the results. Also, I think that the user experience would greatly benefit from having an on-screen overlay of the output text rather than having it in a separate window in the output terminal so that you are not constantly looking back and forth between the terminal and the content you are immersing in. Other than that, I am satisfied with the project, and I believe these tools will help me with my language learning journey.

7 References

[1] "Language Difficulty Ranking," [Online]. Available: https://effectivelanguagelearning.com/language-guide/language-difficulty/. [Accessed 23 April 2023].

[2] jisho, "行く inflections," jisho, [Online]. Available: https://jisho.org/search/%E3%81%84%E3%81%8F. [Accessed 23 April 2023].

[3] "Immersion language learning: The best way to learn a new language," Memrise, 24 September 2021. [Online]. Available: https://www.memrise.com/blog/immersion-language-learning-the-best-way-to-learn-a-new-language-top-5-language-immersion-tips. [Accessed 23 April 2023].

[4] Refold, "Refold," Refold, [Online]. Available: https://refold.la/. [Accessed 12 4 2023].

[5] Anki, "Anki," Anki, [Online]. Available: https://apps.ankiweb.net/. [Accessed 12 4 2023].

[6] jpdb, "jpdb," jpdb, [Online]. Available: https://jpdb.io/. [Accessed 12 4 2023].

[7] "Forgetting curve," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Forgetting_curve. [Accessed 23 April 2023].

[8] "1A: Immersion," Refold, [Online]. Available: https://refold.la/simplified/stage-1/a/immersion. [Accessed 23 April 2023].

[9] "2A: Intensive Immersion," Refold, [Online]. Available: https://refold.la/simplified/stage-2/a/intensive-immersion. [Accessed 23 April 2023].

[10] "The Evolution of Language Learning," Migaku, [Online]. Available: https://www.migaku.io/. [Accessed 23 April 2023].

[11] "Yomichan," FooSoft Productions, [Online]. Available: https://foosoft.net/projects/yomichan/. [Accessed 23 April 2023].

[12] "Learn by Watching," iago, [Online]. Available: https://getiago.com/. [Accessed 23 April 2023].

[13] "How is your spaced repetition algorithm better than what other SRS apps use?," jpdb, [Online]. Available: https://jpdb.io/faq#SpacedRepetitionAlgorithm. [Accessed 23 April 2023].

[14] "How are you calculating the difficulty ratings on your difficulty lists?," JPDB, [Online]. Available: https://jpdb.io/faq#HowIsDifficultyCalculated. [Accessed 23 April 2023].

[15] Works Applications, "Sudachi," GitHub, 9 3 2023. [Online]. Available: https://github.com/WorksApplications/Sudachi. [Accessed 12 4 2023].

[16] M. A. Lee, "Python Tesseract," GitHub, 14 3 2022. [Online]. Available: https://github.com/madmaze/pytesseract. [Accessed 12 4 2023].

[17] M. Budyś, A. Alhendi and A. Kurniawan, "Manga OCR," GitHub, 5 11 2022. [Online]. Available: https://github.com/kha-white/manga-ocr. [Accessed 12 4 2023].

[18] NHK, "「闇バイトをしないように気をつけて」警察が大学生に話す," NHK, 10 4 2023. [Online]. Available: https://www3.nhk.or.jp/news/easy/k10014031781000/k10014031781000.html. [Accessed 12 4 2023].

[19] A. Sweigart, "pyperclip 1.8.2," PyPI, 20 February 2021. [Online]. Available: https://pypi.org/project/pyperclip/. [Accessed 24 April 2023].

[20] J. Hartley, "colorama 0.4.6," PyPI, 24 October 2022. [Online]. Available: https://pypi.org/project/colorama/. [Accessed 24 April 2023].

[21] T. Maruyama and K. Yamamoto, "Simplified Corpus with Core Vocabulary," 1603-1, Kamitomioka Nagaoka, Niigata 940-2188, JAPAN.

[22] "NHK の国際放送," NHK WORLD-JAPAN, [Online]. Available: https://www3.nhk.or.jp/nhkworld/ja/. [Accessed 24 April 2023].

[23] "BertJapanese," HuggingFace, [Online]. Available: https://huggingface.co/docs/transformers/v4.28.1/en/model_doc/bert-japanese. [Accessed 24 April 2023].

[24] Tohoku University, "BERT base Japanese (IPA dictionary)," Hugging Face, [Online]. Available: https://huggingface.co/cl-tohoku/bert-base-japanese. [Accessed 12 4 2023].

[25] "tf.data: Build TensorFlow input pipelines," TensorFlow, [Online]. Available: https://www.tensorflow.org/guide/data. [Accessed 24 April 2023].

[26] F. a. o. Chollet, "Keras," 2015. [Online]. Available: https://keras.io/. [Accessed 24 April 2023].